ABSTRACT

Title of dissertation: SPATIO-TEMPORAL REASONING ABOUT AGENT BEHAVIOR

Paulo Shakarian, Doctor of Philosophy, 2011

Dissertation directed by: Professor V.S. Subrahmanian
Department of Computer Science

There are many applications where we wish to reason about spatio-temporal aspects of an agent's behavior. This dissertation examines several facets of this type of reasoning.

First, given a model of past agent behavior, we wish to reason about the probability that an agent takes a given action at a certain time. Previous work combining temporal and probabilistic reasoning has made either independence or Markov assumptions. This work introduces Annotated Probabilistic Temporal (APT) logic which makes neither assumption. Statements in APT logic consist of rules of the form "Formula G becomes true with a probability [L,U] within T time units after formula F becomes true" and can be written by experts or extracted automatically from historical data. In this dissertation, we explore the problem of entailment, specifically what is the probability that an agent performs a given action at a certain time based on a set of such rules. We show this problem to be coNP-hard (in the complexity class coNP under some natural assumptions) and present several sets of linear constraints for solving this problem exactly. We then develop a sound, but incomplete fixpoint operator as a heuristic for such queries. This approach was implemented and tested on 23 different models automatically generated from several datasets. The operator quickly converged to produce tight probability bounds for the queries.

Second, agent behavior often results in "observations" at geospatial locations that imply the existence of other, unobserved, locations we wish to find ("partners"). In this dissertation, we formalize this notion with "geospatial abduction problems" (GAPs). GAPs try to infer a set of partner locations for a set of observations and a model representing the relationship between observations and partners for a given agent. This dissertation presents exact and approximate algorithms for solving GAPs as well as an implemented software package for addressing these problems called SCARE (the Spatio-Cultural Abductive Reasoning Engine). We tested SCARE on counter-insurgency data from Iraq, attempting to locate enemy weapons caches (partners) based on attacks (observations). On average, SCARE was able to locate weapons caches within 690 meters of actual sites. Additionally, we have considered a variant of the problem where the agent wishes to abduce regions that contain partner points. This problem is also NP-hard. To address this issue, we develop and implement a greedy approximation algorithm that finds small regions which contain partner points - on average containing 4 times as many partners as the overall area.

We then provide an adversarial extension to GAPs as follows: given a fixed set of observations, if an adversary has probabilistic knowledge of how an agent were to find a corresponding set of partners, he would place the partners in locations that minimize the expected number of partners found by the agent. In a complementary problem, the agent has probabilistic knowledge of how an adversary locates his partners and wishes to maximize the expected number partners found. We show that both of these problems are NP-hard and design schemes to find approximate solutions - often with theoretical guarantees. With our implementation, we demonstrate that these algorithms often obtain excellent solutions.

We also introduce a class of problems called geospatial optimization problems (GOPs). Here the agent has a set of actions that modify attributes of a geospatial region and he wishes to select a limited number of such actions (with respect to some budget) in a manner that either causes some goal to be true (goal-based GOPs) and/or maximizes a benefit function (benefit-maximizing GOPs). Additionally, there are certain combinations of actions that cannot be combined. We show NP-hardness (membership in NP under reasonable assumptions) as well as provide limits of approximation for these problems. We then develop sets of integer constraints that provide an exact solution and provide an approximation algorithm with a guarantee.

While we look to optimize certain geospatial properties in GOPs, we note that for some real-world applications, such as epidemiology, there is an underlying diffusion process that also affect geospatial proprieties. Assuming the structure of a social network - a directed graph with weighted and labeled vertices and edges - we study optimization with respect to such diffusion processes in social network optimization problems (SNOPs). We show that many well-known social network diffusion process can be embedded into generalized annotated programs [86]. Hence, a SNOP query seeks to find a set of vertices, that if given some initial property, optimize an aggregate with respect to such a diffusion process. We show this class of problems is also NP-hard (NP-complete under certain assumptions). We develop a greedy heuristic that obtains an approximation guarantee for a large class of such queries. We implemented this algorithm and evaluated it on a real-world data-set consisting of a graph of 103,000 edges.

SPATIO-TERMPORAL REASONING ABOUT AGENT BEHAVIOR

by

Paulo Shakarian

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2011

Advisory Committee:
Professor V.S. Subrahmanian, Chair/Advisor
Professor Stuart S. Antman, Dean's Representative
Professor Samir Khuller
Professor Dana Nau
Professor James A. Reggia

Dedication

To my son, Carter.

Acknowledgments

Above all, I would like to thank my wife, Jana, for her love and understanding throughout my graduate experience. I do not think I could have done this without her constant support.

I would like to thank my advisor, Prof. V.S. Subrahmanian, who since 2007, has mentored and guided me throughout this entire process.

I would also like to thank my committee, Prof. Dana Nau, Prof. Samir Khuller, Prof. James Reggia, and Prof. Stuart Antman who have also been supportive.

Additionally, I would like to thank the following people who have all contributed to my success in graduate school (in no particular order): Gerardo Simari, Dan LaRocque, Austin Parker, Patrick Roos, John Dickerson, Geoff Stoker, Prof. Maria Luisa Sapino, and Matthias Broecheler.

Finally, I would like to thank the U.S. Army Advanced Civil Schooling (ACS) program and the U.S. Military Academy (USMA/West Point) instructor's program (Department of Electrical Engineering and Computer Science - EECS) for funding my Ph.D. studies at the University of Maryland. In particular, COL Eugene Ressler, who made it possible for me to earn the degree.

Contents

List of Tables

List of Figures

List of Abbreviations

#P	Sharp-P
α	alpha
β	beta
AI	Artificial Intelligence
ALC-ENT	Entailment using Alternative Linear Constraints
APT Logic	Annotated Probabilistic Temporal
ATS	Associated Thread Subset
BMGOP	Benefit-Maximizing Geospatial Optimization Problem
BMGOP-IP	BMGOP Integer Program
crf	Cutoff Reward Function
CoNP	Complement of Non-deterministic Polynomial Time
DomSet	Dominating Set
efr	Existential Frequency Function
exfd	Explanation Function Distribution
FLOT	Front Line of Trace
FPRAS	Fully Polynomial Randomized Approximation Scheme
FPTAS	Fully Polynomial Time Approximation Scheme
fr	Frequency Function
frf	Fall-off Reward Function
GAP	Geospatial Abduction Problem (spatial chapters) or Generalized Annotated Program (social network chapters)
GBGOP	Goal-Based Geospatial Optimization Problem
GBGOP-IP	GBGOP Integer Program
GCD	Geometric Covering by Discs
GOP	Geospatial Optimization Problem
HSD	Honest Significant Difference
In-#P	Membership in the complexity class #P
In-coNP	Membership in the complexity class coNP
In-NP	Membership in the complexity class NP
IP	Integer Program
IPB	Intelligence Preparation of the Battlefield
I-REP	Induced Region Explanation Problem
I-REP-MCZ	I-REP Minimum Cardinality with a lower distance bound of zero
ISW	Institute for the Understanding of War
JY	Jackson-Yariv model
KEDS	Kansas Event Data System
k-SEP	k-sized Spatial Explanation Problem
lfp	Least Fixed-Point
LP	Logic Program or Linear Program (context-dependent)
MAROB	Minorities as Risk Database
MCA	Maximal Counter-Adversary Strategy
MCA-Exp	Maximal Counter-Adversary Strategy - Explaining
MC	Minimal Cardinality

MCA-LS	Maximal Counter-Adversary Strategy - Local Search
MDP	Markov Decision Process
ME	Maximum Explaining
MILP	Mixed Integer Linear Program
NAI	Named Area of Interest
NP	Non-deterministic Polynomial Time
OAS	Optimal Adversarial Strategy
PCD	Pre-Condition Disjoint
PCTL	Probabilistic Computational Tree Logic
pdf	Probability Distribution Function
pfr	Point Frequency Function
PITF	Political Instability Task Force
PRISM	Probabilistic Symbolic Model Checker
PTIME	Polynomial Time
qfr	Query Frequency Function
REP	Region Explanation Problem
rf	Reward Function
SAT	Satisfiability
SC	Set-Cover Problem
SCARE	Spatio-Cultural Abductive Reasoning Engine
SEC	Securities Exchange C omission
SEP	Spatial Explanation Problem
SLC	Straightforward Linear Constraints
SLC-ENT	Entailment Using SLC
SNOP	Social Network Optimization Problem
SOMA	Stochastic Opponent Modeling Agents
SPM	Sequence Probability Measure
st	Such That
TD-SEP	Total Distance Spatial Explanation Problem
tp	Temporal-Probabilistic
wrf	Weighted Reward Function
wrt	With Respect To
WT-SEP	Weighted Spatial Explanation Problem

Chapter 1

Introduction

There are many applications where we wish to reason about spatio-temporal aspects of an agent's behavior. This dissertation examines several facets of this type of reasoning.

1.1 Temporal Reasoning about an Agent's Actions

Given a model of past agent behavior, we wish to reason about the probability that an agent takes a given action at a certain time. Previous work combining temporal and probabilistic reasoning has made either independence or Markov assumptions. This work introduces Annotated Probabilistic Temporal (APT) logic which makes neither assumption. Statements in APT logic consist of rules of the form "Formula G becomes true with a probability [L,U] within T time units after formula F becomes true" and can be written by experts or extracted automatically from historical data. A set of such statements is referred to as an APT logic program. In Chapter 2, we introduce this framework and explore two key problems: consistency and entailment. The consistency problem for APT logic mirrors the consistency problem of probabilistic logic introduced in [131]. The complementary problem of entailment can be used to determine the probability that an agent performs a given action at a certain time based on an APT program. We study the computational complexity of these two problems and determine that consistency is NP-hard while entailment is coNP-hard. Under some natural assumptions, we are also able to show a matching upper bound on the complexity for both problems (membership in the class NP for consistency and coNP for entailment). We then introduce several sets of linear constraints for solving this problem exactly. In Chapter 3, we develop a sound, but incomplete fixpoint operator as a heuristic for such queries. This operator runs in polynomial time in the size of the APT logic program. This approach was implemented and tested on 23 different models automatically generated from several datasets. The operator quickly converged to produce tight probability bounds for the queries.

1.2 Inferring Geospatial Aspects of an Agents Behavior

Some agent behavior often results in "observations" at geospatial locations that imply the existence of other, unobserved, locations we wish to find ("partners"). In Chapter 4, we formalize this notion with "geospatial abduction problems" (GAPs). GAPs try to infer a set of partner locations for a set of observations and a model representing the relationship between observations and partners for a given agent. We shall refer to a set of partner locations as an "explanation." Given a set of observations and a model of the agent, finding an explanation of a certain size is NP-hard and in-NP under some reasonable assumptions. We provide an enumeration-based algorithm that can find an explanation of size k - if one exists - as well as show reductions to several well-known combinatorial problems - specifically set-cover, dominating-set, and integer programming. These reductions allow us to leverage several known algorithms to find explanations of a cardinality within a certain factor of the minimum. We then develop a new greedy algorithm that achieves the same approximation ratio as the classic greedy approach to set-cover (see [136]) but allows a software designer to use one of a variety of heuristics which do not affect the guarantee. We implement and experimentally evaluate several of these heuristics in a software package called SCARE (the Spatio-Cultural Abductive Reasoning Engine). We tested SCARE on counter-insurgency data from Iraq, attempting to locate enemy weapons caches (partners) based on attacks (observations). On average, SCARE was able to locate weapons caches within 690 meters of actual sites. We then present a variant of the problem in Chapter 5 where the agent wishes to abduce regions that contain partner points. This problem is also NP-hard (NP-complete under some natural assumptions). To address this issue, we develop and implement a greedy approximation algorithm that finds small regions which contain partner points - on average containing 4 times as many partners as the overall area.

1.3 Geospatial Abduction under Adversarial Conditions

In Chapter 6, we provide an adversarial extension to GAPs as follows: given a fixed set of observations, if an adversary has probabilistic knowledge of how an agent were to find a corresponding set of partners, he would place the partners in locations that minimize the expected number of partners found by the agent. In a complementary problem, the agent has probabilistic knowledge of how an adversary locates his partners and wishes to maximize the expected number partners found. We note that the manner in which the explanation of the adversary is compared to that of the agent can differ based on domain. As such, we axiomatically define a "reward function" and prove results for these two problems with respect to this generalization. We show that these problems are both NP-hard, and in-NP under some natural conditions. We also design schemes to find approximate solutions - often with theoretical guarantees. With our implementation, we demonstrate that these algorithms often obtain excellent solutions.

1.4 Optimal Selection of Agent Actions

In Chapter 7, we introduce a class of problems called geospatial optimization problems (GOPs). Here the agent has a set of actions that modify attributes of a geospatial region and he wishes to select a limited number of such actions (with respect to some budget) in a manner that either causes some goal to be true (goal-based GOPs) and/or maximizes a benefit function (benefit-maximizing GOPs). Additionally, there are certain combinations of actions that cannot be performed together. We show NP-hardness (membership in NP under reasonable assumptions) as well as provide limits of approximation for these problems. We then develop sets of integer constraints that provide an exact solution and provide an approximation algorithm with a guarantee.

While we look to optimize certain geospatial properties in GOPs, we note that for some real-world applications, such as some epidemiological phenomena, there is an underlying diffusion process that also affect geospatial proprieties. Assuming the structure of a social network - a directed graph with weighted and labeled vertices and edges - we study optimization with respect to such diffusion processes in Chapter 8 where we introduce social network optimization problems (SNOPs). We show that many well-known social network diffusion process can be embedded into generalized annotated programs [86]. These diffusion processes were previously studied in a variety of different contexts including economics [150][73], epidemiology [5][67], social media [20][167], and business [177]. In a SNOP query, we seek to find a set of vertices, that if given some initial property, optimize an aggregate with respect to such a diffusion process. We show this class of problems is also NP-hard (NP-complete under certain assumptions). We also leverage the results of [46] to provide a limit of the ability to approximate an optimal solution to such problems. For a large class of such queries, we then develop an greedy algorithm that provides the best possible approximation guarantee unless P=NP as well as techniques for scaling it. We implemented this algorithm and evaluated it on a real-world data-set consisting of a graph of 103,000 edges.

1.5 Applications

The various frameworks for reasoning about an agent's behavior presented in this dissertation are sufficiently general to solve difficult problems from a variety of domains. Our discussion of APT logic in Chapters 2-3 include examples illustrating how that framework can be used to reason about power-grids, the stock market, and transportation services. Likewise, we provide examples of geospatial-abduction and its adversarial extension of Chapters 4-6 applied to counter-drug, naturalist, criminology, and paleontology domains. Finally, in Chapters 7 and 8 where we look to optimally select actions for an agent, we provide examples relating to a political campaign, disease-spread, and cell-phone usage.

In addition to the aforementioned problem domains, we note that much of this work can be used to improve military intelligence analysis for counter-insurgency applications. Traditionally, military intelligence practices in the US Army rely on a process known as "Intelligence Preparation of the Battlefield" [170]. In this process, an intelligence analyst studies terrain and cultural factors along with the capabilities of an adversary in order to predict the actions of an enemy combatant on the battlefield. Since the 9/11 attacks, this process has been modified to handle counter-terrorism and counter-insurgency situations as well [171]. However, unlike traditional military situations, these contemporary environments are often more complex for a variety of reasons. Consider the following real-world problems:

1. In a counter-insurgency operation, enemy reconnaissance of a target may not always be indicative of a pending attack on said target (as in a traditional military conflict). Such activity may be designed to elicit a response from local security forces (for evaluation) or to lull security forces into a sense of complacency.

2. There is no "front line" or "FLOT" as in a traditional battlefield. In a conventional conflict, a combatant force conducts logistic operations behind the front line. By contrast, in a counter-insurgency situation, insurgent forces manage logistics through systems of caches used to store weapons, ammunition, and supplies to support their operations.

3. In a traditional military environment, the structure of a combatant is usually well-defined and hierarchical - this is the standard military structure seen throughout the militaries of the world. An insurgent force, by contrast, is often de-centralized and its structure can resemble a social network which can have a variety of different topologies. Such networks are often very survivable - even if the leadership is killed or captured.

The above three aspects of a counter-insurgency can all be addressed with the research presented in this dissertation. For instance, APT logic, introduced in Chapters 2-3 can be used to help determine the probability that a given reconnaissance event implies a pending attack. Using the abductive reasoning of GAPs introduced in Chapter 4, we have created software that has been shown to be useful in locating enemy weapons cache sites. With SNOPs, introduced in Chapter 8, we show that annotated programs can be leveraged to find which members of a social network cause the spread of a certain phenomenon – this can allow an analyst to select targets whose neutralization will have the greatest impact on the insurgent forces. Again, we would also like to point out that these three aspects of counter-insurgency are not the only problems that can be addressed with this research. There are many other applications of this work – both civilian and military – that will be discussed throughout this dissertation.

1.6 Summary of Major Contributions

Chapters 2-3

- Introduced the framework of APT logic.
- Identified the complexity class of consistency and entailment problems for APT logic as NP-complete.
- Introduced three sound and complete algorithms based on linear programming for solving consistency and entailment problems for APT logic.
- Introduced a sound, but incomplete fixed-point operator for approximately solving consistency and entailment problems for ground APT programs.
- Introduced a sound, but incomplete algorithms for approximately solving consistency and entailment problems for non-ground APT programs while avoiding a

full grounding of the program.

- Implemented the ground fixed-point operator and evaluated it using a real-world data set.

Chapters 4-6

- Introduced a framework for studying geospatial abduction problems (GAPs).
- Identified the complexity class of several geospatial abduction problems.
- Developed several exact and approximate approaches to solving GAPs based on reductions to known combinatorial problems.
- Implemented a software package for solving GAPs called SCARE (Spatio-Cultural Abductive Reasoning Engine) and evaluated experimentally showing it to be able to locate weapons cache sites in Baghdad.
- Created a variant of GAPs where we look to abduce regions, proved this problem to be NP-complete under some natural assumptions.
- Developed and implemented an approximation algorithm to abduce regions.
- Extended GAPs to the case where partner locations are place adversarily based on probabilistic knowledge of the agent, as well as the complementary problem. Proved these problems to be NP-complete under natural assumptions.
- Developed approximation algorithms for the adversarial problems - often with guarantees. Showed viability of such algorithms with an implementation.

Chapters 7-8

- Introduced geospatial optimization problems, GOPs, in which the agent attempts to optimally select a set of actions to cause some goal to occur and/or maximize some function of the resulting geospatial properties.

- Proved two variants of GOPs to be NP-complete and established theoretical limits on approximation.

- Developed integer constraints for GOPs as well as an approximation algorithm with a guarantee.

- Introduced social network optimization problems, SNOPs, where we attempt to optimize an agents selection of vertices with respect to an aggregate of the result of some diffusion process.

- Proved SNOPs to be NP-complete, explored the limits of approximation and other properties of these problems.

- Illustrated how many known diffusion processes can be embedded into SNOPs.

- Developed exact and approximate approaches to solving SNOPs. For a large class of SNOPs, our approximation algorithm attains the best guarantee unless P=NP.

- Experimentally evaluated our approach to SNOPs on a real-world data-set.

1.7 Related Work

We now provide a brief overview of work related to this dissertation. Additionally, in each chapter, we also provide a related work section to give a more in-depth look at how specific contributions relate to other work.

APT logic, introduced in Chapters 2-3, is a logic-programming framework for reasoning about time and probability together without making independence assumptions. Perhaps the most well-known method to reason about time and probability together is the Markov Process [140] - a stochastic process where states are labeled with atomic propositions with a transition function that, given two states s_1, s_2, returns the probability that s_1 transitions to s_2. A Markov Process assumes what is known as the "Markov Property" which means that each transition probability only depends on the current state, and no previous state [146]. Hence, the transition probability from state s_1 to s_2 is always the same, regardless of which states preceded s_1. The Markov Property yields independence among transitions. For example, given function p which returns a transition probability for any two states, we know that $p(s_1, s_2)$ is independent of $p(s_2, s_3)$. Hence, with a Markov Process starting in state s_1, we can calculate the probability of sequence s_1, s_2, s_3 as $p(s_1, s_2) \cdot p(s_2, s_3)$. However, in many real-world scenarios, this may not be the case. With APT logic, we can reason about the probability of events that may depend on previous or future events - as there are no independence assumptions among different time points. Further, for a Markov Process where each state has a unique atomic label, we demonstrate that it is possible to create an equivalent

APT program, while proving that the relationship in the opposite direction is not possible.[1]

Geospatial abduction, described in Chapters 4-6 uses a model of an agent, as well as observed geospatial phenomenon, to infer unobserved "partner" locations – a set of which is termed an explanation. Facility location [161] is a related problem where an agent searches for a subset of "supply points" in a plane to service a set of "demand points" in such a manner that optimizes a certain objective function. Most facility location problems reduce to an instance of convex geometric covering - i.e. find a small set of convex shapes centered on supply points that cover all demand points. Geospatial abduction problems, by contrast, reduces to a geometric problem where the shapes are irregular - i.e. they have non-uniform holes.[2] The irregular shape of the covers in geospatial abduction adds another layer of complexity not inherent in a facility location problem. We note that this holds true for the geospatial optimization problems introduced in Chapter 7 as well. To illustrate the difficulty of non-convex covering, [115] shows that for the simple problem of covering by uniformly non-convex shapes in just one dimension is NP-complete and does not admit a fully-polynomial time approximation scheme (FPTAS).

Another problem that resembles geospatial abduction is the k-means clustering problem [116]. In this problem, sets of points on a plane are grouped into k disjoint

[1]We explore these relationships in detail in Chapter 2, Section 2.6.1 on page 79.

[2]Note that this still holds true even for the case of region-based geospatial abduction (Chapter 5) as the covers in such a problem are not the regions, but rather the set of points associated with the region, based on the agent model.

sets such that the mean distance between any two points in a given disjoint set is minimized. Additionally, there is a constrained variant described in [176]. However, this work merely groups points together, and does not make any inference with regard to unobserved phenomenon based on an agent model. For a very simple, restricted agent model, one can naively apply a clustering algorithm as a heuristic for a geospatial abduction problem by returning a central point in each cluster as a partner. However, this heuristic provides no approximation guarantee and in our tests, was outperformed by the algorithms introduced in this dissertation.

Finally, our work on social network optimization problems (SNOPs) introduced in Chapter 8 seeks to find a set of vertices in a social network that optimize an aggregate function with respect to a diffusion process. Some simple approaches to this type of problem use a degree-maximizing or centrality measure to find the set of vertices. It is important to note that these measures do not consider any type of diffusion process - therefore cannot normally provide a guarantee with respect to optimality. For example, the work of [6] describes two diffusions processes and prove that their optimality criteria is proportional to vertex degree in the first diffusion process, while inversely proportional to vertex degree in the second. Further, with these approaches, it is unclear how they apply to graphs with multiple vertex and edge labels as the ones considered in SNOPs.

The classic work of [81] is perhaps the best-known generalized framework for finding the most "influential vertices" in a social network given some diffusion process. However, there are some key differences. With SNOPs, the social network can have weights and labels on the vertices and edges, whereas this is not part of the framework of [81]. Further, [81] does not allow complex aggregate functions as SNOPs does. Finally, the approximation guarantees of [81] are dependent on an approximation guarantee associated with their encoding of the diffusion process. This encoding was shown to be #P-hard in [23] by a reduction from the counting version of S-T connectivity, which has no known approximation algorithm. SNOPs, by contrast, determines the result of a diffusion process by the calculation of the fixed-point operator of [86] - which can be accomplished in polynomial time - which make our conditions for approximation guarantees reasonable.

Chapter 2

Annotated Probabilistic Temporal Logic: Sound and Complete Algorithms for Reasoning

Chapters 2-3 investigate reasoning about an agent's behavior in time. The main contribution of these chapters is Annotated Probabilistic Temporal (APT) logic, a logic-based framework for this type of reasoning that does not make independence or Markovian assumptions. In this chapter, we introduce the framework, present a suite of complexity and algorithmic results for consistency and entailment problems, and perform a detailed comparison with other frameworks for reasoning about time and probability together.[1]

[1]This chapter is based on [155] which was completed in cooperation with Gerardo Simari, Austin Parker, and V.S. Subrahmanian.

2.1 Chapter Introduction

There are numerous applications where we need to make statements of the form "Formula G becomes true with $50-60\%$ probability 5 time units after formula F became true." We now give four examples of how such statements might be applied.

Stock Market Prediction There is ample evidence [53] that reports in newspapers and blogs [33] have an impact on stock market prices. For instance, major investment banks invest a lot of time, effort and money attempting to learn predictors of future stock prices by analyzing a variety of indicators together with historical data about the values of these indicators. As we will show later in Figure 2.1, we may wish to write rules such as "The probability that the stock of company C will drop by 10% at time $(T+2)$ is over 70% if at time T, there is a news report of a rumor of an SEC investigation of the company and (at time T) there is a projected earnings increase of 10%." It is clear that such rules can be learned from historical data using standard machine learning algorithms. Financial companies have the means to derive large sets of such rules and make predictions based on them.

Reasoning about Terror Groups The Laboratory for Computational Cultural Dynamics at the University of Maryland has extensively dealt with historical data on over 40 terrorist groups from the Minorities at Risk project [181] and has published detailed analyses of some of these groups' behaviors (Hezbollah [118] and Hamas [119]). The SOMA Terror Organization Portal [120] has registered users from over 12 US government agencies and contains thousands of (automatically) extracted rules about the behaviors of these groups. For such groups, we might want to say: "Hezbollah targets domestic government security institutions/lives with a probability of 87 to 97% within 3 years (time periods) of years when their major organizational goals were focused on eliminating ethnic discrimination and when representing their interests to government officials was a minor part of their strategy." Figure 2.2 provides a list of such rules associated with Hezbollah. Clearly, analysts all over the world engaged in counter-terrorism efforts need to be able to reason with such rules and make appropriate forecasts; in separate work, we have also done extensive work on making such forecasts [121, 122].

Reasoning about Trains All of us want to reason about train schedules and plane schedules. More importantly, railroad companies, airlines, and shipping companies have an even more urgent need to do such reasoning as it directly impacts their planning process. In such settings, a railroad company may learn rules of the form "If train 1 is at station A at time T, then it will be at station B at time $(T + 4)$ with over 85% probability." Once such rules are learned from historical data, various types of reasoning need to be performed in order for the railroad company to make its plans. Figure 2.3 shows a small toy example of rules associated with trains.

Reasoning about a Power Grid Utility companies need to reason constantly about power grids. Decisions about which lines and transformers should be repaired next are based not only on the costs of these repairs, but also when these components are likely to fail, and many other factors. Thus, for example, a power company may derive rules of the form "if the transformer tr and power line ln are functioning at time T, then there is a probability of over 95% that they will continue to be functioning at time $(T+3)$. Figure 2.4 shows a small toy example of rules associated with power grids.

The examples above illustrate the syntax of an APT-logic program; we will give the formal details as we develop the technical material in this chapter. While it is possible for designers to write such programs manually, we expect that machine learning programs can be used to automatically learn such programs from historical data using standard machine learning algorithms, as done in previous work on ap-programs [83]. Though this is not claimed as a contribution of this dissertation, in order to show that it is possible to automatically learn APT-programs, we have developed a simple algorithm called APT-Extract and used it to learn models of certain behaviors exhibited by several terror groups.

This chapter proceeds as follows. In Section 2.2 we introduce the syntax and semantics of APT-logic programs, including a quick treatment of our notion of a *frequency function*, a structure unique to APT-logic. In Section 2.3 we introduce several methods to check consistency of APT-logic programs, along with appropriate complexity analysis. We introduce several algorithms for consistency checking: one that straightforwardly applies the semantics, one that exploits the relationships between formulas in the heads and bodies of APT-rules, and one that works only on specific sorts of APT-rules but often offers substantial speedup when it is possible. These techniques can also be applied to the problem of entailment, which is covered in Section 2.4. In Section 2.5, we explore some applications of APT-logic programs and finally, we spend a great deal of effort in Section 2.6 distinguishing this work from other frameworks for reasoning about time and probability together. In particular, we examine the relationship between APT-logic programs and Markov Decision Processes (MDPs for short) [140], showing that one can create APT-logic programs "equivalent" to a given MDP and policy, but under natural assumptions, there is no MDP "equivalent" to certain APT-logic programs. We further address the relationship between APT-logic and a well known logic called Probabilistic Computation Tree Logic (PCTL for short) [64] and provide examples demonstrating that PCTL cannot express various things expressible in APT-logic programs.

The entire set of complexity results for APT-logic programs derived in this chapter is summarized in Table 2.1. Consistency of APT-logic programs is determined by solving certain linear programs. In this chapter, we develop successively more sophisticated linear programs that try to use different types of "equivalence classes" to collapse multiple variables in the linear program into one variable; Table 2.2 summarizes the main results related to linear program size reduction for consistency checking. Table 2.2 also provides an analogous summary related to reduction of size of the linear program when considering entailment by APT-logic programs.

APT Complexity Results		
Problem	Complexity	Reference
Consistency of Single Unconstrained Rule	NP-complete	Thm 2
Consistency of Single Constrained Rule	NP-complete	Thm 3
Consistency of a mixed PCD Program with additional restrictions on lower probability bounds	Guaranteed consistent	Thm 4
Entailment of an annotated formula by an program	coNP-hard	Thm 7

Table 2.1: Summary of APT Complexity Results

1. $\mathsf{scandal} \overset{pfr}{\hookrightarrow} \neg\mathsf{scandal} : [1, 0.89, 0.93, 0.8, 1.0]$

   For a given sequence of events, if there is a scandal in the headlines, this will be followed by there not being a scandal in 1 time unit with probability $[0.89, 0.93]$.

2. $\mathsf{sec\_rumor} \wedge \mathsf{earn\_incr(10\%)} \overset{pfr}{\hookrightarrow} \mathsf{stock\_decr(10\%)} : [2, 0.65, 0.97, 0.7, 1.0]$

   For a given sequence of events, if there is a rumor of an SEC investigation and an earnings increase of 10%, then the stock price will decrease by 10% in exactly 2 time units frequency range $[0.7, 1.0]$ and probability $[0.65, 0.97]$.

3. $\mathsf{sec\_rumor} \wedge \mathsf{earn\_incr(10\%)} \overset{pfr}{\hookrightarrow} \mathsf{stock\_decr(10\%)} \wedge \mathsf{cfo\_resigns} : [2, 0.68, 0.95, 0.7, 0.8]$

   For a given sequence of events, if there is a rumor of an SEC investigation and an earnings increase of 10%, this will be followed by a stock price decrease of 10% and the CFO resigning in exactly 2 time units with a frequency range $[0.7, 0.8]$ and probability bounds $[0.68, 0.95]$.

Figure 2.1: $\mathcal{K}_{stock}$, a toy APT-Logic Program modeling the behavior to reactions of stock-related news feeds. As all of these rules are constrained, this is a constrained program. The English translation of each rule is also provided.

1. $(\mathsf{INTERORGCON} = 1) \overset{efr}{\rightsquigarrow} (\mathsf{ARMATTACK} = 1) : [2, 0.85, 0.95]$

   Armed attacks are carried out within two years of inter-organizational conflicts arising, with probability between 0.85 and 0.95.

2. $(\mathsf{DIASUP} = 0) \wedge (\mathsf{MILITIAFORM} = 2) \overset{efr}{\rightsquigarrow} (\mathsf{KIDNAP} = 1) : [3, 0.68, 0.78]$

   Kidnappings are carried out within three years when no support from diaspora is received, and Hezbollah has a standing military wing, with probability between 0.68 and 0.78.

3. $(\mathsf{ORGST2} = 1) \wedge (\mathsf{ORGDOMGOALS} = 1) \overset{efr}{\rightsquigarrow} (\mathsf{DSECGOV} = 1) : [3, 0.87, 0.97]$

   Domestic government/state lives and security are targets of terrorism within three years if Hezbollah represents interests to officials as a minor strategy, and its major organizational goals are focused on eliminating discrimination, with probability between 0.87 and 0.97.

4. $(\mathsf{ORGST4} = 1) \wedge (\mathsf{INTERORGCON} = 1) \wedge (\mathsf{MILITIAFORM} = 1) \overset{efr}{\rightsquigarrow} (\mathsf{BOMB} = 0) : [1, 0.56, 0.66]$

   Hezbollah does *not* carry out bombings within the following year if it solicits external support as a minor strategy, there are inter-organizational conflicts, and its military wing is being created, with probability between 0.56 and 0.66.

Figure 2.2: A real-world set of rules extracted by APT-Extract from the Hezbollah dataset. The atoms in the rules are represented as a variable and its value. A plain English explanation of each rule is also provided.

1. $\mathsf{at\_station(train1, stnA)} \overset{efr}{\leadsto} \mathsf{at\_station(train1, stnB)} : [4, 0.85, 1]$

   If train 1 is at station A, train 1 will be at station B within 4 time units with a probability bounded by $[0.85, 1.00]$

2. $\mathsf{at\_station(train1, stnB)} \overset{pfr}{\leadsto} \mathsf{at\_station(train1, stnC)} : [2, 0.75, 0.9]$

   If train 1 is at station B, train 1 will be at station C in exactly 2 time units with a probability bounded by $[0.75, 0.90]$

3. $\mathsf{at\_station(train1, stnA)} \overset{pfr}{\leadsto} \mathsf{at\_station(train2, stnB)} : [1, 0.95, 1]$

   If train 1 is at station A, train 2 will be at station B in exactly 1 time units with a probability bounded by $[0.95, 1.00]$

4. $\mathsf{at\_station(train1, stnA)} : [1, 0.5, 0.5]$

   For a given sequence of events, train 1 will be at station A at time period 1 with a probability of 0.50.

5. $\mathsf{at\_station(train2, stnA)} : [2, 0.48, 0.52]$

   For a given sequence of events, train 2 will be at station A at time period 2 with a probability bounded by $[0.48, 0.52]$.

Figure 2.3: $\mathcal{K}_{train}$ a toy APT-Logic Program modeling rail transit. Items 1-3 are APT-Rules while items 4-5 are annotated formulas. The English translation of each rule is also provided.

1. $\mathsf{func(ln)} \overset{pfr}{\rightsquigarrow} \neg\mathsf{func(ln)} : [1, 0.05, 0.1]$

   If the power line is functional, in exactly 1 time unit it will be non-functional with a probability bounded by $[0.05, 0.10]$

2. $\neg\mathsf{func(ln)} \overset{efr}{\rightsquigarrow} \mathsf{func(ln)} : [2, 0.99, 1]$

   If the power line is not functional, within 2 time units it will functional with a probability bounded by $[0.99, 1.00]$

3. $\mathsf{func(tr)} \wedge \mathsf{func(ln)} \overset{pfr}{\rightsquigarrow} \neg(\mathsf{func(tr)} \wedge \mathsf{func(ln)}) : [1, 0.025, 0.03]$

   If the transformer is functional and the line is functional, then in exactly 1 time unit, at least one of them is not functional with a probability bounded by $[0.025, 0.030]$

4. $\neg(\mathsf{func(tr)} \wedge \mathsf{func(ln)}) \overset{efr}{\rightsquigarrow} \mathsf{func(tr)} \wedge \mathsf{func(ln)} : [3, 0.95, 1]$

   If the transformer and/or the line is not functional, then within 3 time units, they both are functional with a probability bounded by $[0.95, 1.00]$

5. $\mathsf{func(tr)} \wedge \mathsf{func(ln)} : [1, 0.8, 0.95]$

   For a given sequence of events, the transformer and the power line are functional at the first time point with a probability bounded by $[0.80, 0.95]$.

Figure 2.4: $\mathcal{K}_{power}$ a toy APT-Logic Program modeling a power grid. Items 1-4 are APT-Rules, while item 5 is an annotated formula. The English translation of each rule is also provided.

Type of Linear Constraints	Number of Constraints	Number of Variables	Cost of Identifying Equivalence Classes
SLC (Straightforward Linear Constraints)	$2\lvert\mathcal{K}\rvert + 1$	$2^{\lvert B_{\mathcal{L}}\rvert t_{max}}$	(equivalence classes not used)
WELC (World Equiv. Linear Constraints)	$2\lvert\mathcal{K}\rvert + 1$	$2^{2\lvert\mathcal{K}\rvert t_{max}}$	$O\left(2^{2\lvert\mathcal{K}\rvert + B_{\mathcal{L}}}\right)$
FELC using BFECA to identify classes (Frequency Equiv. Linear Constraints, created via brute-force)	$2\lvert\mathcal{K}\rvert + 1$	$2^{\lvert\mathcal{K}\rvert}$	$O\left(2^{\lvert B_{\mathcal{L}}\rvert t_{max}} \cdot F(t_{max}) \cdot \lvert\mathcal{K}\rvert\right)$
FELC using WEFE to identify classes (Frequency Equiv. Linear Constraints, created via world euqiv.)	$2\lvert\mathcal{K}\rvert + 1$	$2^{\lvert\mathcal{K}\rvert}$	$O\left(2^{2\lvert\mathcal{K}\rvert \cdot t_{max}} \cdot t_{max} \cdot \lvert\mathcal{K}\rvert\right) +$ $O\left(2^{2\lvert\mathcal{K}\rvert + B_{\mathcal{L}}}\right)$
FELC w. PCD restrictions on $\mathcal{K}$ (Pre-Condition Disjoint)	$2\lvert\mathcal{K}\rvert + 1$	$2^{\lvert\mathcal{K}\rvert}$	(equivalence classes guaranteed)

Table 2.2: Comparison of Linear Constraints for APT Consistency Checking

Algorithm	Intuition	Reference
SLC-ENT	Determining both the minimization and maximization of a constraint wrt SLC	Section 2.4
ALC-ENT	Determining both the minimization and maximization of a constraint wrt FELC or WELC	Appendix A.1.2

Table 2.3: Comparison of Linear Constraints for APT Entailment Checking

2.2 APT-Logic Programs

In this section, we first define the syntax of APT-logic programs, and then define the formal semantics.

2.2.1 Syntax

We assume the existence of a first order logical language $\mathcal{L}$, with a finite set $\mathcal{L}_{cons}$ of constant symbols, a finite set $\mathcal{L}_{pred}$ of predicate symbols, and an infinite set $\mathcal{L}_{var}$ of variable symbols. Each predicate symbol $p \in \mathcal{L}_{pred}$ has an *arity* (denoted *arity(p)*). A (ground) *term* is any member of $\mathcal{L}_{cons} \cup \mathcal{L}_{var}$ (resp. $\mathcal{L}_{cons}$); if $t_1, \ldots, t_n$ are (ground) terms, and $p \in \mathcal{L}_{pred}$, then $p(t_1, \ldots, t_n)$ is a (resp. ground) atom. A *formula* is defined recursively as follows.

Definition 1. *A (ground) atom is a (ground) formula. If f_1 and f_2 are (ground) formulas, then $f_1 \wedge f_2$, $f_1 \vee f_2$, and $\neg f_1$ are (ground) formulas.*

We use $B_{\mathcal{L}}$ to denote the Herbrand base (set of all ground atoms) of $\mathcal{L}$. It is easy to see that $B_{\mathcal{L}}$ is finite.

We assume that all applications reason about an arbitrarily large, but fixed size window of time, and that $\tau = \{1, \ldots, t_{max}\}$ denotes the entire set of time points we are interested in. t_{max} can be as large as an application user wants, and the user may choose his granularity of time according to his needs. For instance, in the stock market and power grid examples, the unit of time used might be days, and t_{max} may be arbitrarily set to (say) 1,095 denoting interest in stock market and power grid movements for about 3 years. In the case of the train example, however, the unit of time might be seconds, and the application developer might set t_{max} to 93,600, reflecting that we are only interested in reasoning about one day at a time, but at a temporal resolution of one second. In the case of the terrorism application, on the other hand, our temporal resolution might be one month, and t_{max} might be 360 reflecting an interest in events over a 30-year time span.

Definition 2 (Annotated Formula)**.** *If F is a formula, $t \in \tau$ is a time point, and $[\ell, u]$ is a probability interval, then $F : [t, \ell, u]$ is an* annotated formula*.*

Intuitively, $F : [t, \ell, u]$ says F will be true at time t with probability in $[\ell, u]$.[2]

Example 2.2.1. *Let us reconsider the program $\mathcal{K}_{train}$ from Figure 2.3. The annotated formula* $\mathsf{at\_station}(\mathsf{train1}, \mathsf{stnB}) : [4, 0.85, 1]$ *says that the probability that* train1 *will be at station* stnB *at time point* 4 *is between 85 and 100%.*

Throughout this chapter, we assume the existence of a finite set $\mathcal{F}$ of symbols called *frequency function symbols.* Each of these symbols will denote a specific "frequency function" to be defined later when we define our formal APT semantics. We are now ready to define the syntax of Annotated Probabilistic Temporal (APT for short) rules and logic programs which will form the main topic of study for this chapter.

Definition 3 (APT Rule)**.** *Let F, G be two formulas, Δt be a time interval, ℓ, u be a probability interval, $\mathsf{fr} \in \mathcal{F}$ be a frequency function symbol and $\alpha, \beta \in [0, 1]$.*

[2]**Assumption:** Throughout the chapter we assume, for both annotated formulas and APT-rules, that the numbers ℓ, u can be represented as rationals a/b where a and b are relatively prime and the length of the binary representations of a and b is fixed.

1. $F \overset{\mathsf{fr}}{\rightsquigarrow} G : [\Delta t, \ell, u]$ *is called an* unconstrained APT rule.

2. $F \overset{\mathsf{fr}}{\hookrightarrow} G : [\Delta t, \ell, u, \alpha, \beta]$ *is called a* constrained APT rule.

An APT logic program *is a finite set of APT rules and annotated formulas.*

Note that we use the symbol '$\overset{\mathsf{fr}}{\rightsquigarrow}$' for unconstrained APT rules with frequency function symbol fr, while the symbol '$\overset{\mathsf{fr}}{\hookrightarrow}$' is used for constrained rules with frequency function fr. The formal semantics of these rules is quite complex and will be explained shortly. But informally speaking, both types of rules try to check the probability that a formula F is true Δt units before a formula G becomes true.

Figures 2.1, 2.2, 2.3, and 2.4 respectively show the APT-logic programs associated with our stock market, counter-terrorism, trains, and power grid applications. We now define three types of APT-logic programs.

Definition 4 (Types of APT-Logic Programs)**.**

- *An unconstrained APT-Logic Program consists only of unconstrained APT-rules.*
- *A constrained APT-Logic Program consists only of constrained APT-rules.*
- *A mixed APT-Logic Program consists both of constrained and unconstrained APT-rules.*

Consider the APTprograms from the introduction of this chapter, we see that $\mathcal{K}stock$ is a constrained APT-logic program, $\mathcal{K}_{trains}$, $\mathcal{K}_{power}$, and $\mathcal{K}_{terror}$ are unconstrained APT-logic programs.[3]

[3]Notably absent from the types of APT-Logic Programs described above are annotated formulas.

2.2.2 Semantics of APT-logic programs

In this section, we will provide a formal declarative semantics for APT-logic programs. As the syntax of these programs is quite complex, we will do this one step at a time. We start with the well known definition of a world.

Definition 5. *A* world *is any set of ground atoms.*

The power set of $B_{\mathcal{L}}$ (denoted $2^{B_{\mathcal{L}}}$) is the set of all possible worlds. Intuitively, a world describes a possible state of the (real) world or real world phenomenon being modeled by an APT-logic program. The following are examples of worlds:

Example 2.2.2. *Consider the atoms present in the program $\mathcal{K}train$ from Figure 2.3. A few possible worlds are:* $\{\mathsf{at\_station}(\mathsf{train1}, \mathsf{stnA}), \mathsf{at\_station}(\mathsf{train2}, \mathsf{stnB})\}$, $\{\mathsf{at\_station}(\mathsf{train1}, \mathsf{stnB})\}$, *and* $\{\}$.

As worlds are just ordinary Herbrand interpretations [106], we use $w \models F$ to denote the standard definition of satisfaction of a ground formula F by world w as expressed in [106].

Definition 6 (Satisfaction of a formula by a world)**.** *Let f be a ground formula and w be a world. We say that w satisfies f (denoted $w \models f$) iff:*

- *If $f = a$ for some ground atom a, then $a \in w$.*
- *If $f = \neg f'$ for some ground formula f' then w does not satisfy f'.*

We will show later in Theorem 1 that APT-rules can be used to express annotated formulas and hence there is no loss of expressive power.

- *If $f = f_1 \wedge f_2$ for formulas f_1 and f_2, then w satisfies f_1 and w satisfies f.*

- *If $f = f_1 \vee f_2$ for formulas f_1 and f_2, then w satisfies f_1 or w satisfies f_2.*

We say a formula f is a tautology if for all $w \in 2^{B_{\mathcal{L}}}$, $w \models f$. We say f is a contradiction if for all $w \in 2^{B_{\mathcal{L}}}$, $w \models \neg f$.

A *thread*, defined below, is nothing but a standard temporal interpretation [42, 96] in temporal logic.

Definition 7 (Thread)**.** *A* thread *is a mapping $Th : \{1, \ldots, t_{max}\} \rightarrow 2^{B_{\mathcal{L}}}$.*

$Th(i)$ implicitly says that according to the thread Th, the world at time i will be $Th(i)$. We will use $\mathcal{T}$ to denote the set of all possible threads, and $Th_{\emptyset}$ to denote the "null" thread, *i.e.*, the thread which assigns $\emptyset$ to all time points.

Example 2.2.3. *Consider the train scenario shown in Figure 2.3 and the worlds described in Example 2.2.2. Let $\tau = \{0, \ldots, 9\}$ represent one-hour time periods in a day from 9:00am to 6:00pm,* i.e., *0 represents 9-10am, 1 represents 10-11am, and so forth. Figure 2.5 shows a sample thread for this setting, where only one train is present. According to this thread, the train is at station A at 9 o'clock; at 10 o'clock the thread has an empty world, since the train is still between stations, reaching station B at 12. The thread shows how the train moves throughout the rest of the day.*

A thread represents a possible way the domain being modeled (*e.g.*, where the train is) will evolve over all time points. A *temporal probabilistic (tp) interpretation* gives us a probability distribution over all possible threads.

$Th(1) = \{\mathsf{at\_station(train1, stnA)}\}$, $Th(2) = \{\}$,

$Th(3) = \{\}$, $Th(4) = \{\mathsf{at\_station(train1, stnB)}\}$,

$Th(5) = \{\}$, $Th(6) = \{\mathsf{at\_station(train1, stnC)}\}$,

$Th(7) = \{\}$, $Th(8) = \{\mathsf{at\_station(train1, stnB)}\}$,

$Th(9) = \{\}$, $Th(10) = \{\mathsf{at\_station(train1, stnA)}\}$

Figure 2.5: Example thread for the train scenario from Figure 2.3, where only one train is present.

Definition 8 (Temporal-Probabilistic Interpretation). *A temporal-probabilistic (tp) interpretation I is a probability distribution over the set of all possible threads,* i.e., $\sum_{th \in \mathcal{T}} I(th) = 1$.

Thus, a tp-interpretation I assigns a probability to each thread. This reflects the probability that the world will in fact evolve over time in accordance with what the thread says about the state of the world at various points in time.

Example 2.2.4. *Consider once again the setting of Figure 2.3. A very simple example of a tp-interpretation is the probability distribution that assigns probability 1 to the thread from Figure 2.5 and 0 to every other possible thread. Another example would be a distribution that assigns probability 0.7 to the thread from Figure 2.5 and 0.3 to the thread Th' defined as follows: $\langle Th'(1) = \{\mathsf{at\_station(train1, stnA)}\}$, $Th'(2) = \{\}$, $Th'(3) = \{\}$, $Th'(4) = \{\}$, $Th'(5) = \{\mathsf{at\_station(train1, stnB)}\}$, $Th'(6) = \{\mathsf{at\_station(train1, stnC)}\}$, $Th'(7) = \{\}$, $Th'(8) = \{\mathsf{at\_station(train1, stnB)}\}$, $Th'(9) = \{\}$, $Th'(10) = \{\mathsf{at\_station(train1, stnA)}\}\rangle$; this thread specifies that the train's trip from station A to station B takes one time unit longer than specified by the previous* *thread (Th).*

We now define what it means for a tp-interpretation to satisfy an annotated formula.

Definition 9 (Satisfaction of an Annotated Formula)**.** *Let $F : [t, \ell, u]$ be an annotated formula, and I be a tp-interpretation. We say that I* satisfies *$F : [t, \ell, u]$, written $I \models F : [t, \ell, u]$, iff $\ell \leq \sum_{Th \in \mathcal{T}, Th(t) \models F} I(Th) \leq u$.*

Thus, to check if I satisfies $F : [t, \ell, u]$, we merely sum up the probabilities assigned to those threads $Th \in \mathcal{T}$ which make F true at time t. If this sum is in $[\ell, u]$ then I satisfies $F : [t, \ell, u]$.

2.2.3 Frequency Functions

When defining the syntax of APT-logic programs, we defined frequency function symbols. Each frequency function symbol denotes a frequency function. The basic idea behind a frequency function is to represent temporal relationships *within* a thread. For instance, we are interested in the frequency with which G will be true Δt units after F is true. When we study this w.r.t. a specific thread Th, we need to identify when F was true in thread Th, and whether G really was true Δt units after that. For instance, consider the thread shown in Figure 2.6. Here, F is true at times 1, 3, 6, and 8. G is true at times 2, 4, 5, and 7. F and G should be true at the times indicated above.

- The probability (within the thread of Figure 2.6) that G follows F in *exactly* two units of time is 0.33 *if we ignore the occurrence of F at time 8.* If, on the other

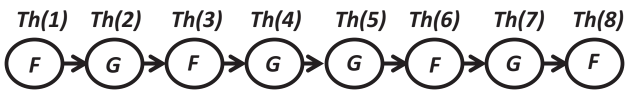


Figure 2.6: Example thread, Th with worlds $Th(1), \ldots, Th(8)$. This figure shows each world that satisfies formula F or formula G.

hand, we do count that occurrence of F at time 8 (even though no times beyond that are possible), then the probability that G follows F in *exactly* two units of time is 0.25.

- The probability that G follows F in *at most* 2 units of time is 100% if we ignore the occurrence of F at time 8; otherwise it is 0.75.

Each of these intuitions leads to different ways to measure the frequency (within a thread) with which G follows F. As we will show shortly, many other possibilities exist as well. *To the best of our knowledge, no past work on reasoning with time and uncertainty deals with frequencies within threads; as a consequence, past works are not able to aggregate frequencies across multiple threads in $\mathcal{T}$ or w.r.t. tp-interpretations.* This capability, we will show, is key for the types of applications described in the Introduction of this chapter.

We see above that there are many different ways to define this frequency from a given body of historical data. Rather than make a commitment to one particular way and in order to allow applications and users to select the frequency function that best meets their application needs, we now define *axioms* that any frequency function must satisfy. Later, we will define some specific frequency functions.[4]

Definition 10 (Frequency Function)**.** *Let Th be a thread, F and G be formulas, and $\Delta t > 0$ be an integer. A* frequency function *fr is one that maps quadruples of the form $(Th, F, G, \Delta t)$ to $[0, 1]$ such that it satisfies the following axioms:*

(FF1) *If G is a tautology, then $\mathsf{fr}(Th, F, G, \Delta t) = 1$.*

(FF2) *If F is a tautology and G is a contradiction, then $\mathsf{fr}(Th, F, G, \Delta t) = 0$.*

(FF3) *If F is a contradiction, $\mathsf{fr}(Th, F, G, \Delta t) = 1$.*

(FF4) *Under the following conditions, there exist threads $Th_1, Th_2 \in \mathcal{T}$ such that $\mathsf{fr}(Th_1, F, G, \Delta t) = 0$ and $\mathsf{fr}(Th_2, F, G, \Delta t) = 1$:*

- *F is not a contradiction*
- *G is not a tautology*
- *F or $\neg G$ is not a tautology*

Axiom FF1 says that if G is a tautology, then $\mathsf{fr}(Th, F, G, \Delta t)$ must behave like material implication and assign 1 to the result. Likewise, if F is a tautology and G is a contradiction, then FF2 says that $\mathsf{fr}(Th, F, G, \Delta t)$ must behave like implication and have a value of 0 ($A \rightarrow B$ is false when A is a tautology and B is a contradiction). Axiom FF3 requires $\mathsf{fr}(Th, F, G, \Delta t)$ to be 1 when F is a contradiction, also mirroring implication. Axiom FF4 ensures that in all cases not covered above, the frequency

[4]**Note:** Throughout this chapter, we will assume that frequency function for a given thread can be computed in polynomial time (i.e. $O(|B_{\mathcal{L}}| \cdot t_{max})$). Additionally, we shall assume that a frequency function will return number that can be represented as a rational number a/b where a and b are relatively prime and the length of the binary represenations of a and b is fixed.

function will be non-trivial by allowing at least one thread that perfectly satisfies (probability 1) and perfectly contradicts (probability 0) the conditional. Note that any function not satisfying Axiom FF4 can be made to do so as long as it returns distinct values: simply map the lowest value returned to 0 and the highest value returned to 1. We now give examples of two frequency functions.

Definition 11 (Point Frequency Function)**.** *Let Th be a thread, F and G be formulas, and $\Delta t \geq 0$ be an integer. A* Point Frequency Function*, denoted $pfr(Th, F, G, \Delta t)$, is defined as:*

$$pfr(Th, F, G, \Delta t) = \frac{|\{t : Th(t) \models F \wedge Th(t + \Delta t) \models G\}|}{|\{t : (t \leq t_{max} - \Delta t) \wedge Th(t) \models F\}|}$$

If there is no $t \in [0, t_{max} - \Delta t]$ such that $Th(t) \models F$ then we define pfr to be 1.

The point frequency function expresses a simple concept: it specifies how frequently G follows F in Δt time points. Mathematically, this is done by finding all time points from $[1, t_{max} - \Delta t]$ at which F is true and of all such time points t, then finding those for which G is true at time $t + \Delta t$. The ratio of the latter to the former is the value of *pfr*. The following lemma says that this is a valid frequency function. Note that the denominator of the point frequency function does not include times where the thread satisfies F after $t_{max} - \Delta t$ because the "end of time" of our finite time model comes before Δt units elapse after F becomes true.

Lemma 1. *pfr satisfies Axioms FF1-FF4.*

Example 2.2.5 (Point Frequency Function)**.** *Consider thread Th from Figure 2.5. Suppose we want to calculate $pfr(Th, \mathsf{at\_station}(\mathsf{train1}, \mathsf{stnB}), \mathsf{at\_station}(\mathsf{train1}, \mathsf{stnC}), 2)$.*

In English, this is the ratio of time $\mathsf{at\_station}(\mathsf{train1}, \mathsf{stnB})$ *is followed by* $\mathsf{at\_station}(\mathsf{train1}, \mathsf{stnC})$ *in two units of time in thread* Th.

We can see that $\mathsf{at\_station}(\mathsf{train1}, \mathsf{stnB})$ *is satisfied by two worlds:* $Th(4)$ *and* $Th(8)$. *We also notice that* $Th(6) \models \mathsf{at\_station}(\mathsf{train1}, \mathsf{stnC})$ *and* $Th(10) \not\models \mathsf{at\_station}(\mathsf{train1}, \mathsf{stnC})$. *Hence, the pfr is simply* 0.5.

Our second type of frequency function, called an *existential* frequency function, does not force G to occur exactly Δt units of time after F is true. It can occur at or before Δt units of time elapse after F becomes true.

Definition 12 (Existential Frequency Function)**.** *Let Th be a thread, F and G be formulas, and $\Delta t \geq 0$ be an integer. An* Existential Frequency Function*, denoted* $\mathit{efr}(Th, F, G, \Delta t)$*, is defined as follows:*[5]

$$\mathit{efr}(Th, F, G, \Delta t) = \frac{\mathit{efn}(Th, F, G, \Delta t, 0, t_{max})}{|\{t : (t \leq t_{max} - \Delta t) \wedge Th(t) \models F\}| + \mathit{efn}(Th, F, G, \Delta t, t_{max} - \Delta t, t_{max})}$$

If the denominator is zero (if there is no $t \in [0, t_{max} - \Delta t]$ such that $Th(t) \models F$ and $\mathit{efn}(Th, F, G, \Delta t, t_{max} - \Delta t, t_{max}) = 0$) then we define efr to be 1.

Note that in the denominator of efr, after time $t_{max} - \Delta t$, we only count satisfaction of F if it is followed by satisfaction of G within $[t_{max} - \Delta t, t_{max}]$.

Lemma 2. *efr satisfies Axioms FF1-FF4.*

The point frequency function expresses what is desired in situations where there is a precise temporal relationship between events (*i.e.*, if one drops an object

[5]Where $efn(Th, F, G, \Delta t, t_1, t_2) = |\{t : (t_1 \leq t \leq t_2)$ *and* $Th(t) \models F$ and there exists $t' \in [t+1, \min(t_2, t + \Delta t)]$ such that $Th(t') \models G\}|$.

from a height of 9.8 meters in a vacuum, it will hit the ground in exactly $\sqrt{2}$ seconds). However, it can be very brittle. Consider mail delivery where one knows a package will arrive in at most 5 business days 95% of the time. The existential frequency function *efr* allows for the implied condition to fall within some specified period of time rather than after exactly $\sqrt{2}$ seconds as before.

Example 2.2.6 (Existential Frequency Function). *Consider thread Th' from Example 2.2.4. Suppose we want to calculate*

$$efr(Th', \mathsf{at\_station}(\mathsf{train1}, \mathsf{stnB}), \neg\mathsf{at\_station}(\mathsf{train1}, \mathsf{stnC}), 2).$$

In English, this is the ratio of times that $\mathsf{at\_station}(\mathsf{train1}, \mathsf{stnB})$ *is followed by* $\neg\mathsf{at\_station}(\mathsf{train1}, \mathsf{stnC})$ *in two units of time in thread Th'.*

We can see that formula $\mathsf{at\_station}(\mathsf{train1}, \mathsf{stnB})$ *is satisfied by two worlds: $Th'(5)$ and $Th'(8)$. Consider world $Th'(6)$, which occurs one time unit after world $Th'(5)$. We can easily see that $Th'(6) \not\models \neg\mathsf{at\_station}(\mathsf{train1}, \mathsf{stnC})$. However, $Th'(7)$, two units later, does satisfy $\neg\mathsf{at\_station}(\mathsf{train1}, \mathsf{stnC})$. As $Th'(9)$ also satisfies $\neg\mathsf{at\_station}(\mathsf{train1}, \mathsf{stnC})$, we have a world within two time units after every world that satisfies* $\mathsf{at\_station}(\mathsf{train1}, \mathsf{stnB})$*. Hence, the efr is 1 in this case.*

Properties of *pfr*: Because of the requirement for F_2 to be satisfied after a specific Δt, *pfr* has several properties (all formulas F_1, F_2 below are assumed to be satisfiable).

1. $pfr(Th, F_1, F_2 \vee F_3, \Delta t) \geq max(pfr(Th, F_1, F_2, \Delta t), pfr(Th, F_1, F_3, \Delta t))$ (valid for *efr* as well)

2. $\mathit{pfr}(Th, F_1, F_2 \wedge \neg F_3, \Delta t) = \mathit{pfr}(Th, F_1, F_2 \wedge F_3, \Delta t) - \mathit{pfr}(Th, F_1, F_3, \Delta t)$

3. $\mathit{pfr}(Th, F_1, F_2, \Delta t) \leq \mathit{pfr}(Th, F_1 \wedge F_3, F_2, \Delta t) \Rightarrow \mathit{pfr}(Th, F_1 \wedge \neg F_3, F_2, \Delta t) \leq \mathit{pfr}(Th, F_1, F_2, \Delta t)$

4. $\mathit{pfr}(Th, F_1, F_2 \wedge F_3, \Delta t) \leq \mathit{min}(\mathit{pfr}(Th, F_1, F_2, \Delta t), \mathit{pfr}(Th, F_1, F_3, \Delta t))$

5. If $\mathit{pfr}(Th, F_1, F_2, \Delta t) = a$ and $\mathit{pfr}(Th, F_1, F_3, \Delta t) = b$ then $\mathit{pfr}(Th, F_1, F_2 \wedge F_3, \Delta t) \geq a + b - 1$.

Properties of *efr*: *efr* satisfies all the properties that *pfr* has above. In addition, *efr* has the property that:

$$\mathit{efr}(Th, F_1, F_2, \Delta t) \geq \mathit{efr}(Th, F_1, F_2, \Delta t - 1)$$

The following result provides some links between *pfr* and *efr*.

Proposition 1. *Let Th be a thread, F and G be formulas,*

1. *Let Δt_1 and Δt_2 be two positive integers. If $\Delta t_1 \leq \Delta t_2$, then:*

$$\mathit{pfr}(Th, F, G, \Delta t_1) \leq \mathit{efr}(Th, F, G, \Delta t_2).$$

2. *Let Δt be a temporal interval. The following inequality always holds:*

$$\mathit{efr}(Th, F, G, \Delta t) \leq \sum_{i=1}^{\Delta t} \mathit{pfr}(Th, F, G, i)$$

2.2.4 Satisfaction of Rules and Programs

We are now ready to define satisfaction of an Annotated Probabilistic Temporal (APT) rule.

Definition 13 (Satisfaction of APT rules)**.** *Let r be an APT rule with frequency function* fr *and I be a tp-interpretation.*

1. *For $r = F \overset{\mathsf{fr}}{\rightsquigarrow} G : [\Delta t, \ell, u]$, we say that I* satisfies *r (denoted $I \models r$) iff*

$$\ell \leq \sum_{Th \in \mathcal{T}} I(Th) \cdot \mathsf{fr}(Th, F, G, \Delta t) \leq u.$$

2. *For $r = F \overset{\mathsf{fr}}{\hookrightarrow} G : [\Delta t, \ell, u, \alpha, \beta]$, we say that I* satisfies *r (denoted $I \models r$), iff*

$$\ell \leq \sum_{\substack{Th \in \mathcal{T}, \\ \alpha \leq \mathsf{fr}(Th, F, G, \Delta t) \leq \beta}} I(Th) \leq u.$$

Intuitively, the unconstrained APT rule $F \overset{\mathsf{fr}}{\rightsquigarrow} G : [\Delta t, \ell, u]$ evaluates the probability that F leads to G in Δt time units as follows: for each thread, it finds the probability of the thread according to I and then multiplies that by the frequency (in terms of fraction of times) with which F is followed by G in Δt time units according to frequency function fr. This product is a little bit like an expected value computation in statistics where a value (frequency) is multiplied by a probability (of the thread). It then sums up these products across all threads in much the same way as an expected value computation.

On the other hand, in the case of constrained rules, the probability is computed by first finding all threads such that the frequency of F leading to G in Δt time units is in the $[\alpha, \beta]$ interval, and then summing up the probabilities of all such threads. This probability is the sum of probabilities assigned to threads where the frequency with which F leads to G in Δt time units is in $[\alpha, \beta]$. To satisfy the constrained APT rule $F \overset{\mathsf{fr}}{\hookrightarrow} G : [\Delta t, \ell, u, \alpha, \beta]$, this probability must be within the probability interval $[\ell, u]$.

Example 2.2.7. *Coming back to the train scenario from Figure 2.3, the following is an example of an unconstrained rule (r_1) and a constrained rule (r_2):*

$$r_1 : \textsf{at\_station(train1,stnC)} \overset{efr}{\rightsquigarrow} \textsf{at\_station(train1,stnB)} : [2, 0.85, 1]$$

$$r_2 : \textsf{at\_station(train1,stnB)} \overset{efr}{\hookrightarrow} \textsf{at\_station(train1,stnC)} : [2, 0.9, 1, 0.5, 1]$$

Consider the second tp-interpretation from Example 2.2.4, which we will call I. By analyzing the two threads considered possible by I, it is clear that $I \models r_1$, since both threads have the property that after being at station C the train reaches station B within two time units, and thus the probability of this event is 1. A similar analysis leads us to confirm that $I \models r_2$, but we must now verify that the constraints placed by the rule on the threads hold; these constraints require that at least half of the times in which the train is at station B, station C be reached within 2 time units. This is indeed the case, since the train stops twice at station B, once going towards C and once going towards A on its way back. As before, the sum of probabilities of reaching the station within 2 time units is 1. Finally, consider the rule:

$$r_3 : \textsf{at\_station(train1,stnA)} \overset{efr}{\rightsquigarrow} \textsf{at\_station(train1,stnC)} : [2, 0.5, 0.6]$$

Clearly, $I \not\models r_3$, since neither of the threads considered possible by the tp-interpretation satisfy the condition that the train reaches station C within two time units of being at station A.

The following proposition says that any tp-interpretation that satisfies certain kinds of constrained or unconstrained APT-logic programs also satisfies a certain APT rule that can be easily constructed from the APT-rules in the original APT-logic program.

Proposition 2. *Let I be a temporal interpretation, F and G be formulas, and Δt be a temporal interval.*

1. *If $I \models \bigcup_{i=1}^{\Delta t} \left\{ F \overset{pfr}{\rightsquigarrow} G : [i, \ell_i, u_i] \right\}$ then $I \models F \overset{efr}{\rightsquigarrow} G : \left[\Delta t, max(\ell_i), min\left(\sum_{i=1}^{\Delta t} u_i, 1\right)\right]$.*
2. *If $I \models F \overset{\mathsf{fr}}{\hookrightarrow} G : [\Delta t, \ell_p, u_p, a, b]$ then $\forall a_\ell, b_\ell, a_u, b_u$ such that $a_\ell \leq a \leq a_u$ and $b_\ell \leq b \leq b_u$ we have $I \models F \overset{\mathsf{fr}}{\hookrightarrow} G : [\Delta t, \ell_p, 1, a_\ell, b_u]$ and $I \models F \overset{\mathsf{fr}}{\hookrightarrow} G : [\Delta t, 0, u_p, a_u, b_\ell]$.*

Note that in unconstrained APT-rules, the ℓ, u probability bounds account for the frequency function as well. In the case of constrained APT-rules, the ℓ, u probability bounds *do not* account for the frequency function. We now show that using a special frequency function called a *query frequency function*, we can use constrained and unconstrained rules to express annotated formulas.

Definition 14 (Query Frequency Function)**.** *Let Th be a thread, F and G be formulas, and $\Delta t \geq 0$ be an integer. A query frequency function, denoted $qfr(Th, F, G, \Delta t)$ is defined as follows:*

1. *If G is a tautology then $qfr(Th, F, G, \Delta t) = 1$*
2. *If F is a tautology and G is a contradiction, then $qfr(Th, F, G, \Delta t) = 0$*
3. *If F is a contradiction then $qfr(Th, F, G, \Delta t) = 1$*
4. *If $Th(1) \models F$ and $Th(\Delta t) \models G$ then $qfr(Th, F, G, \Delta t) = 1$*
5. *Else, $qfr(Th, F, G, \Delta t) = 0$*

The following result shows that *qfr* is a valid frequency function.

Lemma 3. *qfr satisfies Axioms FF1-FF4.*

qfr allows us to construct constrained and unconstrained rules that are equivalent to arbitrary annotated formulas.

Theorem 1. *Let $q = Q : [t, \ell, u]$ be an annotated formula, and I be an interpretation.*

1. *For constrained rule $r = \mathsf{TRUE} \overset{qfr}{\hookrightarrow} Q : [t, \ell, u, 1, 1]$, $I \models q$ iff $I \models r$.*

2. *For unconstrained rule $r = \mathsf{TRUE} \overset{qfr}{\rightsquigarrow} Q : [t, \ell, u]$, $I \models q$ iff $I \models r$.*

The following is an example of how an annotated formula can be expressed as a rule using qfr.

Example 2.2.8. *Consider the train setting from Figure 2.3. One of the annotated formulas given in this example was $\mathsf{at\_station}(\mathsf{train1}, \mathsf{stnA}) : [1, 0.5, 0.5]$. By applying Theorem 1, this formula is equivalent to the constrained rule r_1 and the unconstrained rule r_2:*

$r_1 : \mathsf{TRUE} \overset{qfr}{\hookrightarrow} \mathsf{at\_station}(\mathsf{train1}, \mathsf{stnA}) : [1, 0.5, 0.5, 1, 1]$

$r_2 : \mathsf{TRUE} \overset{qfr}{\rightsquigarrow} \mathsf{at\_station}(\mathsf{train1}, \mathsf{stnA}) : [1, 0.5, 0.5]$

2.3 Consistency

2.3.1 Complexity of Consistency Checking

We are now ready to study the complexity of the problem of checking consistency of APT-logic programs. We say that an APT-logic program $\mathcal{K}$ is *consistent* iff there is a tp-interpretation I such that $I \models \mathcal{K}$. Before stating complexity results, we give results that hold for any frequency function and any APT-rule. The first result follows from axioms FF1-FF4 on frequency functions.

Lemma 4. *Consider the APT-Program $\{r = F \overset{\mathsf{fr}}{\rightsquigarrow} G : [\Delta t, \ell, u]\}$.*

1. *If G is a tautology, then $\{r\}$ is consistent iff $u = 1$.*
2. *If F is a tautology and G is a contradiction, then $\{r\}$ is consistent iff $\ell = 0$.*
3. *If F is a contradiction, then $\{r\}$ is consistent iff $u = 1$.*
4. *If F is not a contradiction, G is not a tautology, and either F is not a tautology or G is not a contradiction then $\{r\}$ is consistent.*

Using this lemma, we can show that for any unconstrained APT-rule, the problem of determining if an APT-logic program consisting of just that APT-rule is consistent *using any frequency function* is NP-complete.

Theorem 2. *Deciding the consistency of an APT-logic program containing a single unconstrained APT-rule is NP-complete in the size of $B_{\mathcal{L}}$.*

The proof of hardness above is by reduction from the SAT problem, while membership in NP relies on manipulating Lemma 4.

In deciding the consistency of a single constrained rule, we take a slightly different approach. The intuition is that if the lower probability bound is not zero, we *must* have a thread whose frequency function value falls within $[\alpha, \beta]$. Otherwise, there is no thread available that would ensure a non-zero probability mass as per the definition of satisfaction. The idea of classifying threads in this manner for constrained rules comes into play later when we present consistency-checking algorithms in Section 2.3.4.

Lemma 5. *Let* $\mathcal{K} = \{r = F \overset{\mathsf{fr}}{\hookrightarrow} G : [\Delta t, \ell, u, \alpha, \beta]\}$ *be a constrained* APT*-logic program consisting of a single rule.* $\mathcal{K}$ *is consistent iff at least one of the following conditions hold.*

- $u = 1$ ***and*** *there exists a thread* Th_{in} *such that* $\alpha \leq \mathsf{fr}(Th_{in}, F, G, \Delta t) \leq \beta$.
- $\ell = 0$ ***and*** *there exists a thread* Th_{out} *such that either* $\alpha > \mathsf{fr}(Th_{out}, F, G, \Delta t)$ *or* $\beta < \mathsf{fr}(Th_{out}, F, G, \Delta t)$.
- *There exists a thread* Th_{in} *such that* $\alpha \leq \mathsf{fr}(Th_{in}, F, G, \Delta t) \leq \beta$ *and a thread* Th_{out} *such that either* $\alpha > \mathsf{fr}(Th_{out}, F, G, \Delta t)$ *or* $\beta < \mathsf{fr}(Th_{out}, F, G, \Delta t)$.

Lemma 5, used in conjunction with the frequency function axioms, allow us to prove that deciding the consistency of a single constrained rule is also NP-complete.

Theorem 3. *Deciding the consistency of an* APT*-logic program containing a single constrained* APT*-rule is NP-complete in the size of* $B_{\mathcal{L}}$.

The NP-hardness of consistency checking for APT programs (whether constrained, unconstrained, or mixed) with more than one rule follows trivially from Theorems 2 and 3. In the next chapter, we show that the consistency-checking problem is in the complexity class NP for general APTprograms (under some natural assumptions).

However, if we assume that certain conditions hold, we can show that consistency for an APT-logic program containing multiple APT-rules can be guaranteed. These restrictions are termed Pre-Condition Disjoint, or PCD; intuitively, they refer to an APT-Program such that there exists a unique world that satisfies exactly one of the rule pre-conditions (the F formulas). Hence, we say that the pre-conditions are "disjoint" from each other. Perhaps such conditions could be specified by a a tool used to learn the rules from the data-set.

Definition 15 (Pre-Condition Disjoint (PCD) APT-Logic Program)**.** *Let $\mathcal{K}$ be an APT-Logic Program such that $\mathcal{K} = \{r_1, \ldots, r_n\}$, where $r_i = F_i \overset{fr}{\rightsquigarrow} G_i : [\Delta t_i, \ell_i, u_i]$ or $r_i = F_i \overset{fr}{\hookrightarrow} G_i : [\Delta t_i, \ell_i, u_i, \alpha_i, \beta_i]$. $\mathcal{K}$ is Pre-Condition Disjoint (PCD) if the following conditions hold true.*

1. *$\forall i$, if r_i is constrained, then $\beta_i = 1$.*

2. *$\forall i$, $\Delta t_i \geq 1$.*

3. *$\forall i$ there exists a world w_i such that $w_i \models F_i$ and $\forall j$ where $j \neq i, w_i \not\models F_j$.*

4. *$\forall i$, fr_i is equal to either pfr, or efr.*

5. *$t_{max} \geq |\mathcal{K}| \cdot max(\Delta t_i)$ (where t_{max} is the length of each thread).*

6. *$\exists$ world $w_\emptyset$ such that $\forall i$ $w_\emptyset \not\models F_i$ and $w_\emptyset \not\models G_i$.*

7. *$\forall r_i \in \mathcal{K}$, $u_i = 1$.*

While somewhat limiting, this restriction still allows APT-Logic Programs that are useful. Consider the following example.

Example 2.3.1. *Consider the set of rules shown in Figure 2.3. These rules do not constitute a PCD program for various reasons. For instance, the upper bound on the probability of the second rule is not 1. Likewise, condition 3 is not satisfied since the first and third rule have the same antecedent. However, the following set of rules satisfies all of the conditions for being a PCD program:*

$$\mathsf{at\_stn}(\mathsf{trn1}, \mathsf{stnA}) \wedge \neg \mathsf{at\_stn}(\mathsf{trn1}, \mathsf{stnB}) \wedge \neg \mathsf{at\_stn}(\mathsf{trn1}, \mathsf{stnC}) \overset{efr}{\rightsquigarrow}$$
$$\mathsf{at\_stn}(\mathsf{trn1}, \mathsf{stnB}) : [4, 0.85, 1]$$
$$\mathsf{at\_stn}(\mathsf{trn1}, \mathsf{stnB}) \wedge \neg \mathsf{at\_stn}(\mathsf{trn1}, \mathsf{stnA}) \wedge \neg \mathsf{at\_stn}(\mathsf{trn1}, \mathsf{stnC}) \overset{pfr}{\rightsquigarrow}$$
$$\mathsf{at\_stn}(\mathsf{trn1}, \mathsf{stnC}) : [2, 0.75, 1]$$
$$\mathsf{at\_stn}(\mathsf{trn1}, \mathsf{stnC}) \wedge \neg \mathsf{at\_stn}(\mathsf{trn1}, \mathsf{stnA}) \wedge \neg \mathsf{at\_stn}(\mathsf{trn1}, \mathsf{stnB}) \overset{efr}{\rightsquigarrow}$$
$$\mathsf{at\_stn}(\mathsf{trn1}, \mathsf{stnB}) : [3, 0.9, 1]$$

Conditions 1, 2, 4, and 7 are trivially satisfied, and t_{max} can be easily chosen to satisfy condition 5. Condition 3 can be seen to hold by noting that no two antecedents of rules can be satisfied at once. Finally, condition 6 holds since the empty world does not satisfy any of the formulas involved in the rules.

The useful feature in a PCD program is that (based on the axioms) we are guaranteed threads with certain frequency function values for each rule. Consider Lemma 6 below, where for any subset of a given APT-program, we are guaranteed the existence of a thread whose frequency is 1 according to the rules in the subset and is 0 according to the other rules.

Lemma 6. *Consider APT-Program $\mathcal{K} = \{r_1, \ldots, r_i, \ldots, r_n\}$ where $r_i = F_i \overset{fr_i}{\hookrightarrow} G_i[\Delta t_i, \ell_i, u_i, \alpha_i, \beta_i]$ or $r_i = F_i \overset{fr_i}{\rightsquigarrow} G_i : [\Delta t_i, \ell_i, u_i]$, depending on whether r_i is* *a constrained or unconstrained rule. If $\mathcal{K}$ is PCD, then for any disjoint partition of rules, $\mathcal{K}_1$, $\mathcal{K}_2$, there exists a thread Th such that for all rules $r_i \in \mathcal{K}_1$, $\mathsf{fr}_i(Th, F_i, G_i, \Delta t_i) = 1$ and for all rules $r_i \in \mathcal{K}_2$, $\mathsf{fr}_i(Th, F_i, G_i, \Delta t_i) = 0$.*

The PCD conditions add a "one-tailed" requirement (the first requirement of Definition 15) to the constrained rules so that β is always one. This allows us to be guaranteed the existence of threads in the $[\alpha, \beta]$ bounds. As it turns out, if the lower bounds on the probabilities are less than a certain amount, we can create an interpretation to guarantee the consistency of the PCD program.

Theorem 4. *For a mixed PCD APT-Program $\mathcal{K} = \{r_1, \ldots, r_i, \ldots, r_n\}$, if for all r_i, $\ell_i \leq \dfrac{|\mathcal{K}| - 1}{|\mathcal{K}|}$ then $\mathcal{K}$ is consistent.*

In the appendix, we show how PCD assumptions can be leveraged for a significant reduction in complexity for constrained APT-programs.

2.3.2 Linear Constraints for Consistency Checking

A *straightforward* algorithm to find a satisfying interpretation given an APT-logic program $\mathcal{K}$ is a brute-force approach that considers each thread. Given k atoms and t_{max} timepoints, there are 2^k possible worlds at each timepoint, and $2^{k \cdot t_{max}}$ possible threads. For ease of notation, we shall refer to the number of threads as n. Hence, note that a function that is linear in the number of threads is exponential in the number of atoms.

Let $\mathcal{T} = \{Th_1, \ldots, Th_i, \ldots, Th_n\}$ be the set of threads. In our linear program, we will use the variables $V = \{v_1, \ldots, v_j, \ldots, v_n\}$. Each v_i represents the (as yet unknown) probability of thread Th_i. We will design the linear program so that solutions of the linear program are in a one to one correspondence with interpretations that satisfy the APT-logic program. Thus, if θ is a solution of the linear program, we want to be sure that the tp-interpretation I_θ such that $I_\theta(Th_i) = \theta(v_i)$ is an interpretation that satisfies $\mathcal{K}$.

Hence, given an APT-logic program $\mathcal{K}$, we will construct a set of "straightforward" linear constraints $\mathsf{SLC}(\mathcal{K})$ over variables $V = \{v_1, \ldots, v_j, \ldots, v_n\}$, such that the interpretation I_θ associated as above with any solution θ satisfies $\mathcal{K}$. The set of constraints are as follows:

Definition 16 (Straightforward Linear Constraints (SLC))**.** *Let $\mathcal{K}$ be an APT-logic program; the set of* straightforward linear constraints *contains exactly the following:*

1. $\sum_{j=1}^{n} v_j = 1$

2. *For each unconstrained rule* $F_i \overset{\mathsf{fr}_i}{\rightsquigarrow} G_i : [\Delta t_i, \ell_i, u_i] \in \mathcal{K}$

   (a) $\ell_i \leq \sum_{j=1}^{n} \mathsf{fr}_i(Th_j, F_i, G_i, \Delta t_i) \cdot v_j$

   (b) $u_i \geq \sum_{j=1}^{n} \mathsf{fr}_i(Th_j, F_i, G_i, \Delta t_i) \cdot v_j$

3. *For each constrained rule* $F_i \overset{\mathsf{fr}_i}{\hookrightarrow} G_i : [\Delta t_i, \ell_i, u_i, \alpha_i, \beta_i] \in \mathcal{K}$

   (a) $\ell_i \leq \sum_{Th_j \in \mathcal{T}_{\alpha_i \leq \mathsf{fr}_i(Th_j, F_i, G_i, \Delta t_i) \leq \beta_i}} v_j$

   (b) $u_i \geq \sum_{Th_j \in \mathcal{T}_{\alpha_i \leq \mathsf{fr}_i(Th_j, F_i, G_i, \Delta t_i) \leq \beta_i}} v_j$

We refer to this set as SLC($\mathcal{K}$).

The first constraint above says that the threads are exhaustive. The second constraint is derived from the formula for satisfaction of an unconstrained rule, while the third constraint is derived from the formula for satisfaction of a constrained rule. Note that the coefficient of v_j in constraints (2) and (3) above are both constants (after the calculations are performed), so these constraints are all linear.

Example 2.3.2. *Recall the program $\mathcal{K}_{power}$ from Figure 2.4. In this simple example, we supposed the power plant delivers power to a transformer (named tr), which is in turn connected via a power line (named ln) to a home. Hence, the atoms* $\mathsf{func}(\mathsf{tr})$ *and* $\mathsf{func}(\mathsf{ln})$ *denote that the various components are functioning, and the home receives power only if both tr and ln are func. Therefore, we have four possible worlds:* $w_0 = \{\mathsf{func}(\mathsf{tr}), \mathsf{func}(\mathsf{ln})\}$, $w_1 = \{\mathsf{func}(\mathsf{tr})\}$, $w_2 = \{\mathsf{func}(\mathsf{ln})\}$, *and* $w_3 = \emptyset$. *If we set the time limit to 4 days, then there are* $4^4 = 256$ *possible threads (each world may occur at each time point). We name these threads* $Th_0, ..., Th_{255}$ *so that the world at time point t of thread* Th_i *is* $((i/4^t) \mod 4)$ *(i.e.* Th_{25} *is* $\langle w_1, w_2, w_1, w_0 \rangle$*) and associate the variable* v_i *with* $I(Th_i)$*. We now show the constraints in* $SLC(\mathcal{K}_{power})$*:*

1. $\sum_{i=0}^{i<256} v_i = 1$

2. $0.025 \leq \sum_{i=0}^{i<256} pfr(Th_i, \mathsf{func}(\mathsf{tr}) \wedge \mathsf{func}(\mathsf{ln}), \neg(\mathsf{func}(\mathsf{tr}) \wedge \mathsf{func}(\mathsf{ln})), 1) \cdot v_i \leq 0.03$

3. $0.95 \leq \sum_{i=0}^{i<256} efr(Th_i, \neg(\mathsf{func}(\mathsf{tr}) \wedge \mathsf{func}(\mathsf{ln})), \mathsf{func}(\mathsf{tr}) \wedge \mathsf{func}(\mathsf{ln}), 3) \cdot v_i \leq 1$

4. $0.05 \leq \sum_{i=0}^{i<256} pfr(Th_i, \mathsf{func}(\mathsf{ln}), \neg\mathsf{func}(\mathsf{ln}), 1) \cdot v_i \leq 0.1$

5. $0.99 \leq \sum_{i=0}^{i<256} efr(Th_i, \neg\mathsf{func}(\mathsf{ln}), \mathsf{func}(\mathsf{ln}), 2) \cdot v_i \leq 1$

Given a solution θ of these constraints, we can see immediately that I_θ satisfies $\mathcal{K}$.

Algorithm 1 Compute consistency of $\mathcal{K}$ using SLC.

```
SLC-CONSISTENT(APT-Program $\mathcal{K}$)

1. Construct $\mathsf{SLC}(\mathcal{K})$.

2. Attempt to solve $\mathsf{SLC}(\mathcal{K})$.

3. If solvable, return consistent, otherwise, inconsistent.
```

We provide the following proposition about correctness of the above procedure for mixed programs.

Proposition 3. *For mixed* APT*-Logic Program* $\mathcal{K}$*,* $\mathcal{K}$ *is consistent iff* $\mathsf{SLC}(\mathcal{K})$ *has a solution.*

The size of the linear program for SLC follows immediately from the definition. As each rule requires two linear constraints, and one linear constraint is required to ensure the variables sum to 1, we have $2|\mathcal{K}|+1$ constraints. The number of variables is equal to the number of threads.

Remark 1. SLC *contains* $2|\mathcal{K}|+1$ *constraints and* $2^{|B_{\mathcal{L}}|\cdot t_{max}}$ *variables.*

Using SLC we can create Algorithm 1, which is guaranteed to give a correct answer to the question of consistency for any APT-Logic Program. However, the linear program's size is exponential in terms of $|B_{\mathcal{L}}| \cdot t_{max}$, making it a very expensive operation in many situations. There are several obvious ways to reduce this cost. One such way would be to consider the set of atoms to be *only* the atoms present in the rules. An obvious method to reduce the other factor in the exponent, t_{max}, would be to adjust the granularity of time used. For example, convert all time to hours instead of minutes. However, this would only provide a correct result in terms of the new granularity. This is an issue we intend to explore in future research.

It turns out that for arbitrary sets of rules and annotated formulas, one need not use one variable for each of the $2^{|B_{\mathcal{L}}| \cdot t_{max}}$ threads. Some threads are equivalent, and may in fact be considered together. We provide two such methods that consider equivalent threads. One that reduces the number of worlds based on *world equivalence* and one that reduces the number of threads based on *frequency equivalence*.

2.3.3 World Equivalence

World equivalence uses the following intuition: when two worlds satisfy exactly the same formulas from the APT-program, they are identical from the APT-program's point of view. By partitioning the set of worlds into classes of identical worlds, and working with the classes instead of the individual worlds, we can create smaller linear programs by associating just one variable with each equivalence class (rather than one variable with each world as is the case of SLC).

Consider the rule $F \overset{\mathsf{fr}}{\hookrightarrow} G : [\Delta t, \ell, u, \alpha, \beta]$. The four world-based equivalence classes resulting from this rule would be the sets of worlds that satisfy $F \wedge G$, $F \wedge \neg G$, $\neg F \wedge G$, and $\neg F \wedge \neg G$. We apply this concept to APT-Logic Programs and divide the set of worlds accordingly. We can treat these resulting equivalence classes as worlds and create world-based thread equivalence classes, and use them instead of threads. This reduces the number of linear constraints for an algorithm similar to SLC. One must note, however, that the equivalence classes must be computed first, which we will show to be NP-complete.

As world equivalence for APT-Logic is based on the formulas found in APT-Rules and annotated formulas, we will formalize the set of formulas associated with a program. We introduce the notation *formula*($\mathcal{K}$) to denote the set of all formulas present in an APT-logic program:

$$\begin{aligned} \mathit{formula}(\mathcal{K}) = {} & \{F, G \mid F \overset{\mathsf{fr}}{\hookrightarrow} G : [\Delta t, \ell, u, \alpha, \beta] \in \mathcal{K}\} \cup \\ & \{F, G \mid F \overset{\mathsf{fr}}{\rightsquigarrow} G : [\Delta t, \ell, u] \in \mathcal{K}\} \end{aligned}$$

Example 2.3.3. *Recall the program $\mathcal{K}_{power}$ from Figure 2.4. The set formula($\mathcal{K}_{power}$) is then*

$$\{\mathsf{func(ln)}, \neg\mathsf{func(ln)}, \mathsf{func(tr)} \wedge \mathsf{func(ln)}, \neg(\mathsf{func(tr)} \wedge \mathsf{func(ln)})\},$$

since these are the only formula appearing in $\mathcal{K}_{power}$.

The cardinality of *formula*($\mathcal{K}$) for a given APT-Logic Program is bounded by $2|\mathcal{K}|$ since APT-Rules have two formulas, F and G. We notice that for each world w in $2^{B_{\mathcal{L}}}$ there is a subset of *formula*($\mathcal{K}$) that w satisfies and a disjoint subset of *formula*($\mathcal{K}$) that w does not satisfy. Hence, with respect to a given set of formulas, certain worlds are indistinguishable: that is, they satisfy exactly the same formulas from the set. We call such worlds $\mathcal{K}$-equivalent.

Definition 17 (World Equivalence)**.** *For APT-logic program $\mathcal{K}$, a world w is $\mathcal{K}$-equivalent to a world w' (denoted $w \equiv_{\mathcal{K}} w'$) iff for all $F \in$ formula($\mathcal{K}$), $w \models F$ iff $w' \models F$.*

Example 2.3.4. *Continuing with $\mathcal{K}_{power}$ from Figure 2.4, recall the 4 worlds:* $w_0 = \{\mathsf{func(tr)}, \mathsf{func(ln)}\}$*,* $w_1 = \{\mathsf{func(tr)}\}$*,* $w_2 = \{\mathsf{func(ln)}\}$*, and* $w_3 = \emptyset$ *and the formula from $\mathcal{K}_{power}$:*

$$formula(\mathcal{K}_{power}) = \{\mathsf{func(ln)}, \neg\mathsf{func(ln)}, \mathsf{func(tr)} \wedge \mathsf{func(ln)}, \neg(\mathsf{func(tr)} \wedge \mathsf{func(ln)})\}.$$

Here w_1 is $\mathcal{K}_{power}$-equivalent to w_3, since both w_1 and w_3 do not satisfy the first formula, do satisfy the second formula, do not satisfy the third formula, and do satisfy the fourth formula. However, w_1 is not $\mathcal{K}_{power}$-equivalent to w_2 since w_1 satisfies $\neg\mathsf{func(ln)}$ *(the second formula), while w_2 does not.*

The relation $\equiv_{\mathcal{K}}$ can be extended to threads in the obvious way.

Definition 18 (Thread Equivalence)**.** *For APT-logic program $\mathcal{K}$, a thread Th_1 is $\mathcal{K}$-equivalent to a thread Th_2 (denoted $Th_1 \equiv_{\mathcal{K}} Th_2$) iff for all time points t, the world $Th_1(t)$ is $\mathcal{K}$-equivalent to world $Th_2(t)$.*

Example 2.3.5. *In Example 2.3.4, we saw that w_1 is $\mathcal{K}_{power}$-equivalent to w_3. Assuming four time points, then the thread $Th = \langle w_3, w_1, w_1, w_0\rangle$ will be equivalent to $Th' = \langle w_1, w_3, w_3, w_0\rangle$, since at every time point t, $Th(t)$ is a world that is $\mathcal{K}$-equivalent to world $Th'(t)$.*

The relation $\equiv_{\mathcal{K}}$ is an equivalence relation (*i.e.*, it is transitive, reflexive, and symmetric) both for threads and for worlds; therefore, it can be used to construct a partitioning of threads into equivalence classes. Let $\mathcal{T}[\equiv_{\mathcal{K}}] = \{P_1, \cdots, P_m\}$ be that partitioning. All threads in each P_i are $\mathcal{K}$-equivalent. The following result states that these partitions have the useful property that all threads in any partition P_i have the same value for *pfr*, *efr*, or *qfr* for formulas in *formula*($\mathcal{K}$):

Lemma 7. *For APT-logic program $\mathcal{K}$, partitioning $P_1, \ldots, P_m$ of $\mathcal{T}$ induced by $\equiv_{\mathcal{K}}$, for all threads $Th, Th' \in P_i$, all $F, G \in formula(\mathcal{K})$, and all Δt;*

1. $pfr(Th, F, G, \Delta t) = pfr(Th', F, G, \Delta t)$
2. $efr(Th, F, G, \Delta t) = efr(Th', F, G, \Delta t)$
3. $qfr(Th, F, G, \Delta t) = qfr(Th', F, G, \Delta t)$

Lemma 7 tells us that each partition P_i has a unique value for *pfr*, *efr*, and *qfr* (for each F, G, and Δt). We introduce the notation $pfr(P_i, F, G, \Delta t)$, $efr(P_i, F, G, \Delta t)$, and $qfr(P_i, F, G, \Delta t)$ to denote these values. For technical reasons, we associate a *label* with each thread Th such that all threads in the same partition P_i have the same label. To define the label, we first order the set $formula(\mathcal{K}) = \{F_1, \cdots, F_n\}$. Then, for a thread Th, we assign $label(Th)$ to be a length $t_{max} \cdot n$ bitstring where bit $t' \cdot i$ ($1 \leq t' \leq t_{max}$ and $1 \leq i \leq n$) is 1 if $Th(t') \models F_i$ and 0 if $Th(t') \not\models F_i$.

Clearly, all Th, Th' in the same partition P_i have the same label. Also, all partitions P_i have a unique label equivalent to the labels of the contained threads and denoted $label(P_i)$. There are at most as many partitions as there are length $t_{max} \cdot n$ bitstrings, and determining if there is a partition associated with a given bitstring b can be done by checking if there is thread whose label is b.

Example 2.3.6. *Using $\mathcal{K}_{power}$ from Figure 2.4, we number $formula(\mathcal{K}_{power})$ as follows:*

$\{F_1 = \mathsf{func(ln)}, F_2 = \neg\mathsf{func(ln)}, F_3 = \mathsf{func(tr)} \wedge \mathsf{func(ln)}, F_4 = \neg(\mathsf{func(tr)} \wedge \mathsf{func(ln)})\}$.

Here, the label for $Th = \langle w_3, w_1, w_1, w_0 \rangle$ (worlds w_i defined in Example 2.3.4) is

$$\underbrace{0101}_{w_3}\underbrace{0101}_{w_1}\underbrace{0101}_{w_1}\underbrace{1010}_{w_0}.$$

To see this, consider the first four digits 0101 *for world w_3. World w_3 does not satisfy F_1, hence the first* 0. *It does, however, satisfy F_2 and F_4 causing the second and fourth digits to be 1.*

The thread $Th' = \langle w_1, w_3, w_1, w_0 \rangle$ has the same label: 0101010101011010*; any two threads which are $\mathcal{K}_{power}$-equivalent will have the same labels.*

We immediately notice that the number of thread partitions is potentially smaller than the number of threads. While there are $2^{B_{\mathcal{L}} \cdot t_{max}}$ threads, there are only $2^{|formula(\mathcal{K})| \cdot t_{max}} \leq 2^{2|\mathcal{K}| \cdot t_{max}}$ partitions. Therefore, using these partitions, rather than threads, is preferable in designing linear constraints. We can use Lemma 7 to construct smaller sets of linear constraints than SLC. For these constraints, we introduce the variable $\hat{v}_{lbl}$, where lbl is a length $t_{max} \cdot |formula(\mathcal{K})|$ bitstring ($lbl \in \{0,1\}^{|formula(\mathcal{K})| t_{max}}$) representing the probability mass assigned to the set of threads in the partition labeled lbl ($\hat{v}_{lbl} = \sum_{Th \in P_i, label(P_i) = lbl} I(Th)$). We can now define the world-equivalence linear constraints.

Definition 19 (World Equivalence Linear Constraints (WELC))**.** *Let $\mathcal{K}$ be an APT-logic program that uses only the frequency functions pfr and efr; the set of* World Equivalence Linear Constraints, *$\mathsf{WELC}(\mathcal{K})$, contains exactly the following:*

1. $\sum_i \hat{v}_i = 1$.

2. *For* $F \overset{fr}{\hookrightarrow} G : [\Delta t, \ell, u, \alpha, \beta]$

(a) $\sum_{lbl \in \{l | \alpha \leq fr(P_i,F,G,\Delta t) \leq \beta \wedge l = label(P_i)\}} \hat{v}_{lbl} \geq \ell$

(b) $\sum_{lbl \in \{l | \alpha \leq fr(P_i,F,G,\Delta t) \leq \beta \wedge l = label(P_i)\}} \hat{v}_{lbl} \leq u$

3. For $F \overset{fr}{\rightsquigarrow} G : [\Delta t, \ell, u]$

   (a) $\sum_{P_i} fr(P_i, F, G, \Delta t) \hat{v}_{label(P_i)} \geq \ell$

   (b) $\sum_{P_i} fr(P_i, F, G, \Delta t) \hat{v}_{label(P_i)} \leq u$

4. For all $lbl \in \{0,1\}^{|formula(\mathcal{K})| \cdot t_{max}}$ for which there is no P_i such that $lbl = label(P_i)$, $\hat{v}_{lbl} = 0$.

Example 2.3.7. *$\mathsf{WELC}(\mathcal{K}_{power})$ (based on program $\mathcal{K}_{power}$ from Figure 2.4) is constructed using variables $\hat{v}_{lbl}$ for each of the $2^{4 \cdot 4} = 65,536$ possible labels. Due to constraint 4, at most 256 of these variables will be non-zero, since there are 256 worlds to populate these 65,536 possible equivalence classes. We will therefore be able to eliminate all but at most 256 of the variables from the representation altogether, since they will be known to be zero in every possible solution. As such, we only need to use the variables not eliminated via constraint 4 when constructing $\mathsf{WELC}(\mathcal{K}_{power})$, and we will do so in this example. The only labels that will have associated threads are those that are combinations of the labels for the worlds w_0, w_1, w_2, and w_3 (defined in Example 2.3.2). With formula($\mathcal{K}_{power}$) being:*

$$\{F_1 = \mathsf{func(ln)}, F_2 = \neg\mathsf{func(ln)}, F_3 = \mathsf{func(tr)} \wedge \mathsf{func(ln)}, F_4 = \neg(\mathsf{func(tr)} \wedge \mathsf{func(ln)})\}$$

these labels are $lbl(w_0) = 1010$, $lbl(w_1) = 0101$, $lbl(w_2) = 1001$ and $lbl(w_3) = 0101$. So, for any label lbl, each four digit sequence must be 1010, 0101, or 1001. Otherwise there cannot possibly be a thread Th such that label(Th) = lbl. In fact, since there *are only* 3 *labels for the worlds* (w_1 *and* w_2*, being* $\mathcal{K}_{power}$*-equivalent, share a label), we know that when there are four time points, there are only* $3^4 = 81$ *variables that can be non-zero in our linear program (one label at each time point). So, leaving out the zeroing constraints and supposing each sum* $\sum_{lbl}$ *sums over those 81 variables not known to be zero via the zeroing constraints, the set of linear constraints is:* $\mathsf{WELC}(\mathcal{K}_{power}) =$

1. $\sum_{lbl} \hat{v}_{lbl} = 1$

2. $0.025 \leq \sum_{lbl} pfr(Th_i, \mathsf{func(tr)} \wedge \mathsf{func(ln)}, \neg(\mathsf{func(tr)} \wedge \mathsf{func(ln)}), 1) \cdot \hat{v}_{lbl} \leq 0.03$

3. $0.95 \leq \sum_{lbl} efr(Th_i, \neg(\mathsf{func(tr)} \wedge \mathsf{func(ln)}), \mathsf{func(tr)} \wedge \mathsf{func(ln)}, 3) \cdot \hat{v}_{lbl} \leq 1$

4. $0.05 \leq \sum_{lbl} pfr(Th_i, \mathsf{func(ln)}, \neg\mathsf{func(ln)}, 1) \cdot \hat{v}_{lbl} \leq 0.1$

5. $0.99 \leq \sum_{lbl} efr(Th_i, \neg\mathsf{func(ln)}, \mathsf{func(ln)}, 2) \cdot \hat{v}_{lbl} \leq 1$

Note that this set of linear constraints is substantially smaller than $\mathsf{SLC}(\mathcal{K}_{power})$*, which used 256 variables where* $\mathsf{WELC}(\mathcal{K}_{power})$ *uses only 81 variables and exactly the same number of constraints (after removal of trivial zeroing constraints).*

Proposition 4. *For any* APT*-program* $\mathcal{K}$*,* $\mathsf{WELC}(\mathcal{K})$ *is solvable iff* $\mathcal{K}$ *is consistent.*

This approach can provide a substantial speedup. As we noted earlier, the number of partitions is bounded by $2^{2|\mathcal{K}|\cdot t_{max}}$ which will often be much smaller than the number of threads, $2^{|B_{\mathcal{L}}|\cdot t_{max}}$. Further, the number of partitions is bound by the number of threads, regardless of the size of $\mathcal{K}$.

Proposition 5. WELC *requires* $2|\mathcal{K}| + 1$ *constraints and at most* $2^{2|\mathcal{K}|t_{max}}$ *variables.*

Algorithm 2 Compute consistency of $\mathcal{K}$ using WELC.

```
WELC-CONSISTENT(APT-Program $\mathcal{K}$)
    1. Construct $\mathsf{WELC}(\mathcal{K})$.
    2. Attempt to solve $\mathsf{WELC}(\mathcal{K})$.
    3. If solvable, return consistent, otherwise, inconsistent.
```

This suggests Algorithm 2 for checking consistency of $\mathcal{K}$. The complexity of Algorithm 2 comes from both creating and solving WELC. Proposition 5 gives the number of constraints required of a linear program to implement $\mathsf{WELC\text{-}CONSISTENT}$. Building WELC is also difficult: we have constraint 4, which requires the inclusion of the constraint $\hat{v}_{lbl} = 0$ if there is no non-empty partition in $\mathcal{T}[\equiv_{\mathcal{K}}]$ with label lbl. Unfortunately, this is an NP-complete operation.

Theorem 5. *For APT-Logic Program, $\mathcal{K}$, and label lbl, determining if there is non-empty $P_i \in \mathcal{T}[\equiv_{\mathcal{K}}]$ such that $label(P_i) = lbl$ is NP-complete.*

To properly construct WELC, we must solve SAT for every subset of $formula(\mathcal{K})$. As $formula(\mathcal{K}) \leq 2|\mathcal{K}|$, this amounts to $O(2^{2|\mathcal{K}|})$ calls to a SAT solver. Assuming $O(2^{|B_{\mathcal{L}}|})$ operations per SAT solution procedure, this operation will take time $O(2^{2|\mathcal{K}|+|B_{\mathcal{L}}|})$. However, as for most linear program implementations, the running time for $\mathsf{WELC\text{-}CONSISTENT}$ will be exponential in terms of $7|\mathcal{K}|t_{max}$ [79], the generation of world equivalence classes will be dominated by WELC itself. Therefore, in most cases, Algorithm 2 will have a better big-O run time than solving the set of straightforward linear constraints.

2.3.4 Frequency Equivalence

For constrained rules it is possible to develop a different set of linear constraints. Rather than considering equivalent worlds, we develop a partition of the set of threads based on the value of the frequency function with respect to each rule in the program. We will then create a new set of linear constraints based on this equivalence, as with WELC, in order to improve performance.

Therefore, the partitions will depend on the thread's relationship to the probability interval $[\alpha, \beta]$, which we shall refer to as the *frequency bounds* for a given rule. Due to the requirement of considering the frequency bounds, this type of thread equivalence will be referred to as *frequency equivalence* and apply only to constrained rules, though there are manipulations one can apply to include annotated formulas; we first define an equivalence relation over threads.

Definition 20 (Frequency Equivalence)**.** *For threads Th_1 and Th_2, and constrained rule $r = F \overset{\mathsf{fr}}{\hookrightarrow} G : [\Delta t, \ell, u, \alpha, \beta]$, we say Th_1 is r-frequency-equivalent to Th_2 (denoted $Th_1 \sim^r Th_2$) iff $(\alpha \leq \mathsf{fr}(Th_1, F, G, \Delta t) \leq \beta \Leftrightarrow \alpha \leq \mathsf{fr}(Th_2, F, G, \Delta t) \leq \beta)$. For APT-Logic Program $\mathcal{K}$ containing only constrained conditionals, we say Th_1 is $\mathcal{K}$-frequency-equivalent to Th_2 (denoted $Th_1 \sim^{\mathcal{K}} Th_2$) iff for all rules $r \in \mathcal{K}$, $Th_1 \sim^r Th_2$.*

Example 2.3.8. *Consider rule $\mathsf{scandal} \overset{pfr}{\hookrightarrow} \neg\mathsf{scandal} : [1, 0.89, 0.93, 0.8, 1.0]$ from Figure 2.1, where we used APT-Rules to represent the behavior of stock price based on news reports. Let $\mathcal{K}_{\textit{fr-ex}}$ be an APT-program containing exactly this rule. We will consider the set of atoms to consist only of $\mathsf{scandal}$ and t_{max} to be 3. In Figure 2.7*

Thread	$pfr(Th, \mathsf{scandal}, \neg\mathsf{scandal}, 1)$
$\langle \mathsf{scandal}, \mathsf{scandal}, \mathsf{scandal} \rangle$	0
$\langle \mathsf{scandal}, \mathsf{scandal}, \neg\mathsf{scandal} \rangle$	1/2
$\langle \mathsf{scandal}, \neg\mathsf{scandal}, \mathsf{scandal} \rangle$	1
$\langle \mathsf{scandal}, \neg\mathsf{scandal}, \neg\mathsf{scandal} \rangle$	1
$\langle \neg\mathsf{scandal}, \mathsf{scandal}, \mathsf{scandal} \rangle$	0
$\langle \neg\mathsf{scandal}, \mathsf{scandal}, \neg\mathsf{scandal} \rangle$	1
$\langle \neg\mathsf{scandal}, \neg\mathsf{scandal}, \mathsf{scandal} \rangle$	1
$\langle \neg\mathsf{scandal}, \neg\mathsf{scandal}, \neg\mathsf{scandal} \rangle$	1

Figure 2.7: For a set of atoms consisting of $\mathsf{scandal}$, and t_{max} of 3 time points, the above chart shows the *pfr* for all possible threads based on a program consisting only of rule $\mathsf{scandal} \overset{pfr}{\hookrightarrow} \neg\mathsf{scandal} : [1, 0.89, 0.93, 0.8, 1.0]$ from Figure 2.1. Figure 2.8 groups these threads in frequency equivalence classes based on *pfr*.

$$\mathcal{K}_{fr\text{-}ex} = \{\mathsf{scandal} \overset{pfr}{\hookrightarrow} \neg\mathsf{scandal} : [1, 0.89, 0.93, 0.8, 1.0]\}$$

$$\mathcal{T}[\sim^{\mathcal{K}_{fr\text{-}ex}}] =$$

$$\left\{ \begin{array}{l} E_1 = \left\{ \begin{array}{l} \langle \mathsf{scandal}, \neg\mathsf{scandal}, \mathsf{scandal} \rangle, \\ \langle \mathsf{scandal}, \neg\mathsf{scandal}, \neg\mathsf{scandal} \rangle, \\ \langle \neg\mathsf{scandal}, \mathsf{scandal}, \neg\mathsf{scandal} \rangle, \\ \langle \neg\mathsf{scandal}, \neg\mathsf{scandal}, \mathsf{scandal} \rangle, \\ \langle \neg\mathsf{scandal}, \neg\mathsf{scandal}, \neg\mathsf{scandal} \rangle \end{array} \right\}, \\ E_2 = \left\{ \begin{array}{l} \langle \mathsf{scandal}, \mathsf{scandal}, \mathsf{scandal} \rangle, \\ \langle \mathsf{scandal}, \mathsf{scandal}, \neg\mathsf{scandal} \rangle, \\ \langle \neg\mathsf{scandal}, \mathsf{scandal}, \mathsf{scandal} \rangle \end{array} \right\} \end{array} \right\}$$

Figure 2.8: For a program consisting only of rule $\mathsf{scandal} \overset{pfr}{\hookrightarrow} \neg\mathsf{scandal} : [1, 0.89, 0.93, 0.8, 1.0]$ from Figure 2.1, we have frequency equivalence classes E_1 and E_2 based on the *pfr* for all possible threads seen in Figure 2.7.

we compute the pfr based on this single rule for all possible threads. In Figure 2.8 we can then group these threads into two equivalence classes, those whose pfr is within $[0.8, 1]$ *and those whose frequency is outside this range.*

For instance, threads $\langle \mathsf{scandal}, \mathsf{scandal}, \mathsf{scandal} \rangle$ *and* $\langle \mathsf{scandal}, \mathsf{scandal}, \neg\mathsf{scandal} \rangle$ *both have a pfr less than* 0.8*. Therefore, we have that* $\langle \mathsf{scandal}, \mathsf{scandal}, \mathsf{scandal} \rangle \sim^{\mathcal{K}_{fr\text{-}ex}} \langle \neg\mathsf{scandal}, \mathsf{scandal}, \mathsf{scandal} \rangle$*.*

The relation $\sim^{\mathcal{K}}$ satisfies several common properties of relations.

Proposition 6. *For any constrained* APT*-logic program* $\mathcal{K}$*,* $\sim^{\mathcal{K}}$ *is reflexive, symmetric, and transitive.*

Therefore $\sim^{\mathcal{K}}$ is an equivalence relation, and we can partition $\mathcal{T}$ (the set of all possible threads) into equivalence classes according to a given $\sim^{\mathcal{K}}$. We let $\mathcal{T}[\sim^{\mathcal{K}}]$ be this partitioning, where each set $E \in \mathcal{T}[\sim^{\mathcal{K}}]$ contains only $\mathcal{K}$-frequency-equivalent threads. We then assign each set E a binary string $str(E)$ of length m (the number of constrained formulas in $\mathcal{K}$) where digit i is 1 if for all $Th \in E$, $\alpha_i \leq \mathsf{fr}(Th, F_i, G_i, \Delta t_i) \leq \beta_i$, and 0 otherwise.

Example 2.3.9. *In Figure 2.8 we see a partitioning of the threads* $\mathcal{T}[\sim^{\mathcal{K}_{fr\text{-}ex}}]$ *with two partitions:* E_1 *and* E_2*. The associated binary strings are:* $str(E_1) = 1$ *and* $str(E_2) = 0$*. Notice that we only have two frequency equivalence classes of threads, which is only* 25% *of the* 8 *threads we had originally.*

In the following linear program, we introduce variables $\bar{v}_b$ for each binary string b of length $|\mathcal{K}|$.

Definition 21 (Frequency-Equivalence Linear Constraints)**.** *For constrained APT-Logic Program $\mathcal{K}$, the set of Frequency-Equivalence Linear Constraints FELC$(\mathcal{K})$ contains only the following:*

1. $\sum_{E \in \mathcal{T}[\sim^{\mathcal{K}}]} \bar{v}_{str(E)} = 1$ *(where $str(E)$ is the binary number that labels frequency equivalence class E)*
2. *For all length $|\mathcal{K}|$ binary strings b if there is no $E \in \mathcal{T}[\sim^{\mathcal{K}}]$ such that $str(E) = b$ then $\bar{v}_b = 0$*
3. *For all $F_i \overset{\textit{fr}}{\hookrightarrow} G_i : [\Delta t_i, \ell_i, u_i, \alpha_i, \beta_i] \in \mathcal{K}$, $\ell_i \leq \sum_{s \in [0,1]^m, s_i = 1} \bar{v}_s \leq u_i$*

Theorem 6. *For constrained APT-Logic Program $\mathcal{K}$, $\mathcal{K}$ is consistent iff there is a solution to FELC$(\mathcal{K})$.*

As FELC provides a correct result for consistency, we can use it to develop the consistency-checking algorithm FELC-CONSISTENT shown below.

Algorithm 3 Compute consistency of $\mathcal{K}$ using FELC.

```
FELC-CONSISTENT(APT-Program $\mathcal{K}$)

1. Construct FELC($\mathcal{K}$).

2. Attempt to solve FELC($\mathcal{K}$).

3. If solvable, return consistent, otherwise, inconsistent.
```

If the frequency equivalence classes of threads for a given program are known, FELC also offers an improvement in complexity over SLC.

Proposition 7. *FELC requires $2|\mathcal{K}| + 1$ constraints and $2^{|\mathcal{K}|}$ variables.*

Example 2.3.10. *Consider the* APT*-Program* $\mathcal{K}_{stock}$ *from Figure 2.1. Let* $B_{\mathcal{L}}$ *be the set of atoms seen in that program (hence* $|B_{\mathcal{L}}| = 5$*). We consider a* t_{max} *of* 4*. From Proposition 1, we know that using* SLC *to determine the consistency of* $\mathcal{K}_{stock}$ *would require* 7 *constraints and* $2^{20} = 1,048,576$ *variables. We show below a set of linear constraints based on* FELC *below that requires* 7 *constraints and only* $2^3 = 8$ *variables. For the program* $\mathcal{K}_{stock}$*, we have the following linear constraints:*

- *For rule* $\mathsf{scandal} \overset{pfr}{\hookrightarrow} \neg\mathsf{scandal} : [1, 0.89, 0.93, 0.8, 1.0]$

  $0.89 \leq \bar{v}_{001} + \bar{v}_{011} + \bar{v}_{101} + \bar{v}_{111} \leq 0.93$

- *For rule* $\mathsf{sec\_rumor} \wedge \mathsf{earn\_incr(10\%)} \overset{pfr}{\hookrightarrow} \mathsf{stock\_decr(10\%)} : [2, 0.65, 0.97, 0.7, 1.0]$

  $0.65 \leq \bar{v}_{010} + \bar{v}_{011} + \bar{v}_{110} + \bar{v}_{111} \leq 0.97$

- *For rule*

  $\mathsf{sec\_rumor} \wedge \mathsf{earn\_incr(10\%)} \overset{pfr}{\hookrightarrow} \mathsf{stock\_decr(10\%)} \wedge \mathsf{cfo\_resigns} : [2, 0.68, 0.95, 0.7, 0.8]$

  $0.68 \leq \bar{v}_{100} + \bar{v}_{101} + \bar{v}_{110} + \bar{v}_{111} \leq 0.95$

- $\bar{v}_{000} + \bar{v}_{001} + \bar{v}_{010} + \bar{v}_{011} + \bar{v}_{100} + \bar{v}_{101} + \bar{v}_{110} + \bar{v}_{111} = 1$

The running time of consistency checking via FELC is *independent* of the number of atoms or time points or number of worlds. Thus, even though it runs in time exponential in $|\mathcal{K}|$, it will in many cases run faster than SLC, which runs in time linear in $|\mathcal{K}|$ and exponential in the number of worlds or the number of time points. Further, since the size of $\mathcal{K}$, the number of worlds, and the number of time points are all known in advance, one can tell which approach will be faster dynamically, and dispatch the smaller, faster linear program.

However, as with WELC, significant computation cost is required to construct the linear constraints, specifically in identifying the frequency equivalence classes that are empty. We refer to the obvious, exhaustive, and exact method for identifying empty frequency equivalence classes as the Brute Force Frequency Equivalence Class Algorithm or BFECA.

Algorithm 4 Find Frequency Equivalence Classes of Constrained Program $\mathcal{K}$

```
BFECA(APT-Program $\mathcal{K}$)

1. Generate all possible threads.
2. For each thread, $Th$, for all $i$, compute $\mathsf{fr}_i(Th, F_i, G_i, \Delta t_i)$.
3. Determine for each thread, $Th$, for each rule, $r_i$, if the associated frequency
   function, $\mathsf{fr}_i$ for $Th$ falls within the range $[\alpha_i, \beta_i]$.
4. Based on the result of step 3, determine which frequency equivalence class $Th$
   belongs to.
5. After all threads are generated, return EMPTY if there are no threads found
   for a given frequency equivalence class is empty and OK otherwise.
```

As BFECA exhaustively considers all threads, we have the following trivial proposition concerning correctness.

Proposition 8. *For each frequency equivalence class C, if C is empty BFECA returns EMPTY; otherwise, if C contains at least one thread, BFECA returns OK.*

For each thread, BFECA calculates the frequency function with regard to each rule. Hence, for each of the $2^{|B_{\mathcal{L}}| t_{max}}$ threads, it calculates $|\mathcal{K}|$ frequency functions. This leads us to the complexity result below.

Proposition 9. *The complexity of* BFECA *is:*

$$O\left(2^{|B_{\mathcal{L}}| t_{max}} \cdot F(t_{max}) \cdot |\mathcal{K}|\right)$$

where $F(t_{max})$ is defined as follows. Suppose $time_i$ is the time required to compute $\mathsf{fr}_i(Th, F_i, G_i, \Delta t_i)$. Then $F(t_{max})$ equals $\max_i(time_i)$.

Note that if $F(t_{max})$ is linear, then the complexity of finding the frequency equivalence classes and then performing FELC is still better than SLC. The dominating term in the complexity of FELC has an exponent of $|B_{\mathcal{L}}| \cdot t_{max}$ when BFECA is used. SLC, on the other hand, will have an exponent of $3.5 \cdot |B_{\mathcal{L}}| \cdot t_{max}$ for most linear program solvers [79]. The following example shows how BFECA works.

Example 2.3.11. *Consider the FELC constraints set up for $\mathcal{K}_{stock}$ in Example 2.3.10. Look at rules $\mathsf{sec\_rumor} \wedge \mathsf{earn\_incr(10\%)} \overset{pfr}{\hookrightarrow} \mathsf{stock\_decr(10\%)} : [2, 0.65, 0.97, 0.7, 1.0]$ and $\mathsf{sec\_rumor} \wedge \mathsf{earn\_incr(10\%)} \overset{pfr}{\hookrightarrow} \mathsf{stock\_decr(10\%)} \wedge \mathsf{cfo\_resigns} : [2, 0.68, 0.95, 0.7, 0.8]$. For a given thread, Th, consider the pfr's associated with those rules. Let $p_1 = pfr(Th, \mathsf{sec\_rumor} \wedge \mathsf{earn\_incr}(10\%), \mathsf{stock\_decr}(10\%), 2)$ and $p_2 = pfr(Th, \mathsf{sec\_rumor} \wedge \mathsf{earn\_incr}(10\%), \mathsf{stock\_decr}(10\%) \wedge \mathsf{cfo\_resigns}, 2)$.*

*We note that p_2 **must** be less than or equal to p_1 as the G formula for both rules differs only by one conjuncted atom. Therefore, there is no possible Th such that $p_2 > p_1$. Hence, variables $\bar{v}_{100}$ and $\bar{v}_{101}$ from the FELC constraints in Example 2.3.10 **must** be set to zero.*

To find such variables, BFECA calculates the frequency function for all possible threads. However, with SLC-CONSISTENT, the dominating term in this example requires 2^{70} operations, where BFECA requires only 2^{20} operations. Note that the complexity of BFECA often will dominate the complexity of FELC-CONSISTENT.

As suggested earlier, FELC can be used on programs that consist of both constrained rules and annotated formulas. We can include annotated formulas in our constrained program by writing rules that are essentially equivalent to annotated formulas, as described earlier through use of the Query Frequency Function in Definition 14.

Note that if the PCD conditions are met (Page 46), we can often be guaranteed that all FELC equivalence classes will be non-empty, making the BFECA algorithm unnecessary. See the Appendix for a complete discussion of this special case.

2.3.5 Combining World and Frequency Equivalence

We have introduced two improved methods for computing consistency: FELC-CONSISTENT/BFECA and WELC-CONSISTENT. We now introduce a hybrid approach that uses the world-equivalence classes of WELC to ease the computation necessary to compute the frequency-equivalence classes needed in FELC. World-equivalence can be used to determine if a frequency equivalence class is empty or not. The intuition is simple: we follow the approach of BFECA, generating the set of threads and finding the frequency function for each one. However, rather than generating the set of threads, we generate the set of world-based thread partitions and find their frequency functions. As shown in the discussion of WELC, the number of world-based thread partitions can be considerably less than the number of threads. Hence, we present world equivalence for finding frequency equivalence, or WEFE.

Algorithm 5 World Equivalence for finding Frequency Equivalence Classes of Constrained Program $\mathcal{K}$

```
WEFE(APT-Program $\mathcal{K}$)

1. Find the world equivalence classes based on $formula(\mathcal{K})$.

2. Generate all world-equivalence based thread partitions for $\mathcal{K}$.

3. For each world-equivalence thread partition, $P$, for all $i$, compute
   $\mathsf{fr}_i(P, F_i, G_i, \Delta t_i)$.

4. For each rule, $r_i$ let $IN_i$ be the set of thread partitions such that $\alpha_i \leq$
   $\mathsf{fr}_i(P, F_i, G_i, \Delta t_i) \leq \beta_i$. For each rule, let $OUT_i$ be all partitions not in $IN_i$.

5. For string $s \in [0,1]^{|\mathcal{K}|}$ let the set $PCLASS_s$ be defined as $\left\{\bigcap_{s_i=1} IN_i\right\} \cap$
   $\left\{\bigcap_{s_i=0} OUT_i\right\}$.

6. For each class $cl_s$ return EMPTY if $PCLASS_s \equiv \emptyset$ and OK otherwise.
```

As WEFE exhaustively considers all world equivalence based thread partitions, and each thread belongs to exactly one partition, WEFE provides a correct answer.

Proposition 10. *If a given frequency equivalence class is empty, WEFE returns EMPTY. If there is a thread in a given frequency equivalence class, WEFE returns*

OK.

The computational complexity of this algorithm is dependent upon the number of thread-partitions resulting from world-equivalence. As stated before, this is $2^{2|\mathcal{K}|\cdot t_{max}}$. Further, the cost of calculating the frequency function for each thread is only $O(t_{max})$ as checking the satisfiability of the F and G formulas in a rule by a world equivalence class is a trivial operation, since the satisfaction is pre-determined when the world-equivalence classes are generated.

Proposition 11. *The complexity of* WEFE *is*

$$O\left(2^{2|\mathcal{K}|\cdot t_{max}} \cdot t_{max} \cdot |\mathcal{K}|\right)$$

when the set of world-equivalence classes for $\mathcal{K}$ is known.

WEFE/FELC-CONSISTENT is generally preferable for checking the consistency of constrained programs: because it considers threads on a world-equivalence basis rather than individually, it should generally have a shorter run time than BFECA even taking into account the costs of constructing world-equivalence classes. We illustrate this in the following example:

Example 2.3.12. *Suppose we want to build* FELC *constraints for $\mathcal{K}_{stock}$ as we did in Example 2.3.10 where $t_{max} = 4$. We note that formula($\mathcal{K}_{stock}$) consists of the following:*

1. scandal

2. $\neg$scandal

3. $\mathsf{sec\_rumor} \wedge \mathsf{earn\_incr}(10\%)$

4. $\mathsf{stock\_decr}(10\%)$

5. $\mathsf{stock\_decr}(10\%) \wedge \mathsf{cfo\_resigns}$

Although the number of world equivalence classes, based on formula($\mathcal{K}_{stock}$) would be 2^5, which is also the number of worlds due to there only being 5 atoms referenced in the program, we note that many of the world equivalence classes are empty. For example, we know that there can be no world that satisfies both of the first two formulas, which immediately reduces our number of world equivalence classes by a factor of two. Further, there can be no world that does not satisfy $\mathsf{stock\_decr}(10\%)$ *but satisfies* $\mathsf{stock\_decr}(10\%) \wedge \mathsf{cfo\_resigns}$. *Hence, the number of world equivalence classes is 12 in this case, a significant reduction from the 32 worlds originally considered.*

Therefore, WEFE only considers $12^4 = 20,736$ world-equivalent threads, as opposed to BFECA, which considers $32^4 = 1,048,576$ threads. Note that if the world equivalence classes are known, this cost of WEFE may still dominate FELC-CONSISTENT. This is a vast improvement over the 2^{70} operations required by SLC-CONSISTENT.

2.4 Entailment by APT-logic programs

Now that we have dealt with consistency, we can explore the issue of entailment, which is defined in the usual way.

Definition 22 (Entailment)**.** *Let $\mathcal{K}$ be an APT-logic program, r be a rule, and af be* *an annotated formula. We say that $\mathcal{K}$ entails af iff for all models I of $\mathcal{K}$, $I \models af$, and that $\mathcal{K}$ entails r iff for all models I of $\mathcal{K}$, $I \models r$.*

Example 2.4.1 (Entailment)**.** *Recall that in Example 2.3.8 we presented the following* APT*-Program:*

$$\mathcal{K}_{\textit{fr-ex}} = \{\textsf{scandal} \overset{pfr}{\hookrightarrow} \neg\textsf{scandal} : [1, 0.89, 0.93, 0.8, 1.0]\}$$

Suppose we form the following rule as a hypothesis.

$$r_{hyp} = \textsf{scandal} \overset{pfr}{\hookrightarrow} \neg\textsf{scandal} : [1, 0.88, 0.94, 0.8, 1.0]$$

Does $\mathcal{K}_{\textit{fr-ex}}$ entail r_{hyp}? A quick examination of the only rule in the program and the hypothesis tells us that except for the probability bounds, they are the same. Notice that the rule in $\mathcal{K}_{\textit{fr-ex}}$ has probability bounds $[0.89, 0.93]$ and the probability bounds of r_{hyp} are a superset, $[0.88, 0.94]$. Therefore, we know that any interpretation in which the sums of the probabilities of threads with a frequency ratio between $[0.8, 1.0]$ sum to a quantity in $[0.89, 0.93]$, are also in $[0.88, 0.94]$. So, by the definitions of satisfaction and entailment, we can say that $\mathcal{K}_{\textit{fr-ex}}$ entails r_{hyp}.

The following result shows that checking entailment of an annotated formula by an APT-logic program is coNP-hard.

Theorem 7. *Given an* APT*-logic program $\mathcal{K}$ and an annotated formula, af, deciding if $\mathcal{K}$ entails af is coNP-hard in $|B_{\mathcal{L}}|$ (the number of atoms).*

In the next chapter, we prove a matching upper bound for the complexity of this problem.

2.4.1 Linear Constraints for Entailment

We shall now provide algorithms for computing entailment based on the linear constraints SLC, WELC, and FELC. In all cases, the method is straightforward: we determine the minimal and maximal probability for the annotated formula in interpretations satisfying the original knowledgebase by minimizing and maximizing the appropriate sum subject to some set of linear constraints. Due to the fact that any annotated formula can be viewed as a constrained rule, we will not describe the entailment of annotated formulas in this section.

Algorithm 6 Entailment of Rule r by Program $\mathcal{K}$ with SLC

```
SLC-ENT(APT-Program $\mathcal{K}$)

1. If $r$ is unconstrained, ($r = F \overset{\mathsf{fr}}{\rightsquigarrow} G : [\Delta t, \ell, u]$), create rule $r' = F \overset{\mathsf{fr}}{\rightsquigarrow} G : [\Delta t, \ell', u']$ where $\ell', u'$ are variables.

2. If $r$ is constrained, ($r = F \overset{\mathsf{fr}}{\hookrightarrow} G : [\Delta t, \ell, u, \alpha, \beta]$) create rule $r' = F \overset{\mathsf{fr}}{\hookrightarrow} G : [\Delta t, \ell', u', \alpha, \beta]$ where $\ell', u'$ are variables.

3. Create set of linear constraints $\mathsf{SLC}(\mathcal{K} \cup \{r'\})$.

4. Let $\bar{\ell'}$ be the minimization of $\ell'$ subject to $SLC(\mathcal{K} \cup \{r'\})$.

5. Let $\bar{u'}$ be the maximization of $u'$ subject to $SLC(\mathcal{K} \cup \{r'\})$.

6. If $[\bar{\ell'}, \bar{u}'] \subseteq [\ell, u]$ return ENTAILS otherwise return NOT ENTAILS.
```

We can show Algorithm 6 to be correct and to take time exponential in $|B_\ell|$ (as expected due to Theorem 7).

Proposition 12 (Checking Entailment using *SLC*)**.** *For unconstrained rule* $r = F \overset{fr}{\rightsquigarrow} G : [\Delta t, \ell, u]$ *or constrained rule* $r = F \overset{fr}{\hookrightarrow} G : [\Delta t, \ell, u, \alpha, \beta]$ *and program* $\mathcal{K}$, *SLC-ENT returns ENTAILS iff* $\mathcal{K}$ *entails* r *and returns NOT ENTAILS iff* $\mathcal{K}$ *does not entail* r

Proposition 13. *SLC-ENT requires solving at most two linear programs. Each linear program has* $2|\mathcal{K}|+1$ *constraints and* $2^{|B_{\mathcal{L}}\cdot|t_{max}}$ *variables.*

We now give an example of how Algorithm 6 will run in practice.

Example 2.4.2. *Consider APT-Program* $\mathcal{K}_{stock}$ *introduced in Figure 2.1 with* $t_{max} = 4$. *Suppose we want to see if* $\mathcal{K}_{stock}$ *entails the annotated formula query* $= \mathsf{earn\_decr}(10\%) : [3, 0.50, 0.80]$.

First, we re-write the query as a rule using qfr. *Hence,* $query_{rule} = \mathsf{TRUE} \overset{qfr}{\hookrightarrow} \mathsf{earn\_decr}(10\%) : [3, 0.50, 0.80, 1, 1]$. *From this rule, we create* $query'_{rule} = \mathsf{TRUE} \overset{qfr}{\hookrightarrow} \mathsf{earn\_decr}(10\%) : [3, \ell', u', 1, 1]$.

We now consider all possible threads given $\mathcal{K}_{stock} \cup \{query'_{rule}\}$ *and* $t_{max} = 4$. *As there are 6 atoms in the union of the program and query, we have* $2^{24} = 16,777,216$ *possible threads* ($|\mathcal{T}| = 2^{24}$)*. Hence, we set up the following linear constraints:*

- *For rule* $\textsf{scandal} \overset{pfr}{\hookrightarrow} \neg\textsf{scandal} : [1, 0.89, 0.93, 0.8, 1.0]$

  $0.89 \leq \sum_{Th_j \in \mathcal{T}_{0.8 \leq pfr(Th_j, \mathsf{scandal}, \neg\mathsf{scandal}, 1) \leq 1.0}} v_j$

  $0.93 \geq \sum_{Th_j \in \mathcal{T}_{0.8 \leq pfr(Th_j, \mathsf{scandal}, \neg\mathsf{scandal}, 1) \leq 1.0}} v_j$

- *For rule* $\textsf{sec\_rumor} \wedge \textsf{earn\_incr(10\%)} \overset{pfr}{\hookrightarrow} \textsf{stock\_decr(10\%)} : [2, 0.65, 0.97, 0.7, 1.0]$

  $0.65 \leq \sum_{Th_j \in \mathcal{T}_{0.7 \leq pfr(Th_j, \mathsf{sec\_rumor} \wedge \mathsf{earn\_incr(10\%)}, \mathsf{stock\_decr(10\%)}, 2) \leq 1.0}} v_j$

  $0.97 \geq \sum_{Th_j \in \mathcal{T}_{0.7 \leq pfr(Th_j, \mathsf{sec\_rumor} \wedge \mathsf{earn\_incr(10\%)}, \mathsf{stock\_decr(10\%)}, 2) \leq 1.0}} v_j$

- *For rule*

  $\textsf{\textit{sec\_rumor}} \wedge \textsf{\textit{earn\_incr(10\%)}} \overset{pfr}{\hookrightarrow} \textsf{\textit{stock\_decr(10\%)}} \wedge \textsf{\textit{cfo\_resigns}} : [2, 0.68, 0.95, 0.7, 0.8]$

  $0.68 \leq \sum_{Th_j \in \mathcal{T}_{0.7 \leq pfr(Th_j, \textsf{sec\_rumor} \wedge \textsf{earn\_incr(10\%)}, \textsf{stock\_decr(10\%)} \wedge \textsf{cfo\_resigns}, 2) \leq 0.8}} v_j$

  $0.95 \geq \sum_{Th_j \in \mathcal{T}_{0.7 \leq pfr(Th_j, \textsf{sec\_rumor} \wedge \textsf{earn\_incr(10\%)}, \textsf{stock\_decr(10\%)} \wedge \textsf{cfo\_resigns}, 2) \leq 0.8}} v_j$

- *For rule* $query'_{rule} = \textsf{TRUE} \overset{qfr}{\hookrightarrow} \textsf{earn\_decr}(10\%) : [3, \ell', u', 1, 1]$

  $\ell' \leq \sum_{Th_j \in \mathcal{T}_{1 \leq qfr(Th_j, \textsf{TRUE}, \textsf{earn\_decr}(10\%), 3) \leq 1.0}} v_j$

  $u' \geq \sum_{Th_j \in \mathcal{T}_{1 \leq qfr(Th_j, \textsf{TRUE}, \textsf{earn\_decr}(10\%), 3) \leq 1.0}} v_j$

- $\sum_{j=0}^{j<2^{24}} v_j = 1.$

As it turns out, the minimization of ℓ' *is* 0 *and the maximization of* u' *is* 1. *Since* $[0,1] \not\subseteq [0.5, 0.8]$, *we can say that* $\mathcal{K}_{stock}$ *does* ***not*** *entail query.*

SLC-ENT uses the SLC set of linear constraints. However, one could easily substitute WELC or FELC for SLC in SLC-ENT. We present an algorithm for alternate linear constraints, ALC-ENT, that mirrors SLC-ENT and leverages these other constraints in the appendix.

There is a further improvement that can be made in practice: if we solve the linear program once, and find that the minimization of ℓ' is less than ℓ, we have determined that the rule is not entailed by the program, and solving the linear program again is not necessary to decide entailment.

2.5 Applications of APT Logic

APT-logic programs have many possible applications; in this section we will

Algorithm 7 The APT-Extract Algorithm.

APT-Extract(T, $ActCond$, $MaxBody$, Δ, $SuppLB$, σ,STAT-Test)

```
1.  $Rules := \emptyset$;
2.  for each combination ($environment\ variable, value$) choose $1, \ldots, MaxBody$ {
3.      let $Body$ be the current combination; $supportBody$:= 0; $supportBoth$:= 0;
4.      for $t = 1$ to $maxTime(T)$ {
5.          $bodyHappened$:= false;
6.          if $Body$ is true at time $t$ then
7.              $bodyHappened$:= true; $actHappened$:= false;
8.          for $d = 1$ to $\Delta$ {
9.              if $ActCond$ is true at time $t + d$ then $actHappened$:= true;
10.                 break for;
11.         }
12.         if $bodyHappened$ then $supportBody$:= $supportBody$ + 1;
13.         if $bodyHappened$ and $actHappened$ then $supportBoth$:= $supportBoth$ + 1;
14.     }
15.     if $supportBody$ <> 0 then $confidence$:= $supportBoth$ / $supportBody$;
16.     else $confidence$:= 0;
17.     if $(supportBoth > suppLB) \wedge STAT\_TEST(Body, ActCond)$ then
18.         add $Body \overset{pfr}{\rightsquigarrow} ActCond : [\Delta, confidence - \sigma, confidence + \sigma]$ to $Rules$;
19. }
20. return $Rules$;
```

briefly describe an effort to learn conditions under which various terror groups took various actions, in the form of APT-programs. We assume that the data is given in the form of a table that contains two kinds of attributes: *action* and *environment*, and that each tuple represents the values of each of these attributes for a certain time point. A good example of this kind of data is the “Minorities at Risk Organizational Behavior” (MAROB) data set [181]. This data set has identified around 150 parameters to monitor for about 300 groups around the world that are either involved in terrorism or are at risk of becoming full-fledged terrorist organizations. The 150 attributes describe aspects of these groups, such as whether or not the group engaged in violent attacks, if financial or military support was received from foreign governments, and the type of leadership the group has. It was a simple task to divide the attributes into actions that could be taken by the group (*i.e.*, bombings, kidnappings, armed attacks, etc.) and environmental conditions (*i.e.*, the type of leadership, the kind and amount of foreign support, whether the group has a military wing, etc.). Values for these 150 parameters are available for up to 24 years per group, though it is less for some groups (*e.g.*, groups that have been around for a shorter duration). For each group, MAROB provides a table whose columns correspond to the 150 parameters and the rows correspond to the years. There are many social science data sets that use such data. These include the KEDS data set from the University of Kansas that tracks country stability data (rather than terror group data) [151] and the Political Instability Task Force (PITF) data [57].

The APT-Extract algorithm provides a basic approach to extracting APT-rules [6]. The inputs are: a table of historic data, a condition on an action variable (variable name and value), a maximum size for the body, a value for Δ, a lower bound for the *support* of the rule, and a real number $\sigma \in [0, 1]$ that will determine the width of the probability annotations for the extracted rules, and an arbitrary statistical test (*e.g.*, a t-test or something based on p-values in statistics) selected by the user that measures the correlation between the values of the body of a possible rule and the head. We use the standard measurements of support and confidence from the literature on association rules: given table T, the *support* of a condition C in T is the number of tuples for which C is true; given conditions C_1 and C_2, the confidence in the fact that C_1 is accompanied by C_2 is the ratio of the support of $C_1 \wedge C_2$ to the support of C_1. As an example of the kind of rules that can be extracted by this algorithm, some of the rules extracted from the data for Hezbollah are given in Figure 2.2.

2.6 Chapter 2 Related Work

In addition to past work on probabilistic logic programming [130, 129], probabilistic logic programs were studied in [84], [86], and [93, 94, 95], who showed how to introduce various probabilistic dependencies into probabilistic LPs. [111, 112] made major contributions to bottom up computations of probabilistic LPs.

[98] and [66] were among the first to provide a logic to integrate time and prob-

[6]Note that this algorithm is not a novel one, and simply performs calculations to capture interesting relationships present in the data in order to build rules. More complex algorithms for rule extraction are outside the scope of this dissertation.

ability. [78] also studied the integration of time and probability in order to facilitate efficient planning. He was primarily interested in how the probability of facts and events change over time. [62] developed a logic for reasoning about actions, probability and time using an interval time model. [30] developed methods to extend possibilistic logic to handle temporal information. This logic associates, with each formula of possibilistic logic, a set of time points describing when the formula has a possibilistic truth value. [63] studied the semantics of reasoning about distributed systems where uncertainty is present using a logic where a process has knowledge about the probability of events for decision making by the process. [44, 43] developed logics of time and belief to model the behavior of distributed systems, while [169] developed a framework that integrates beliefs, time, commitment, desires, and multiple agents. [13] developed a language to reason about actions in a probabilistic setting; their models use static and dynamic causal laws together with background (unknown) variables whose values are determined by factors not in the model. Building on top of past work by [34], [36] introduce heterogeneous temporal probabilistic agents to model agent behavior and develop a model theory and fixpoint semantics focusing on agents built using legacy code.

Though there has been extensive work on temporal reasoning, the key difference between APT logic programs and past works in verification [96, 42, 173, 25, 56, 97] is the use of frequency functions in our work to define the frequency with which a given formula G holds (some given time) after a given formula F holds. We show that such a definition can be given in many different ways and, rather than committing to one such definition, we provide axioms that any frequency function should satisfy. A result of our introduction of the frequency function is that the probability an event occurs at time t is dependent on the events that occur in interval $[1, t]$ *and* interval $[t, t_{max}]$.

APT-Logic distinguishes itself from other temporal logics in the following ways:

1. It provides for reasoning about probability of events within a sequence of events *and* probabilistic comparison between sequences of events.

2. Future worlds can depend on more than just the current world.

3. It provides bounds on probabilities rather than just a point probability.

4. It does not make any independence assumptions.

2.6.1 Markov Decision Processes

Many temporal logics, whether probabilistic or not, make use of some sort of state transition system as an underlying structure. A state-transition system is said to conform to the *Markov Property* if each transition probability only depends on the current state [146]. We demonstrate that while APT-Logic Programs maintain much of the expressiveness of most state-transition systems, they also have the ability of expressing non-Markovian sequences of events. Specifically, the semantic structures used in APT-Logic (worlds, threads, interpretations) can be represented by state transition systems when the following restrictions are applied:

1. As APT-Logic only deals with finite temporal sequences, only the first t_{max} states generated by an MDP will be considered.

2. By definition, each world represents a unique set of atoms. Therefore, a corresponding state transition system must have the restriction that each state is uniquely labeled; *i.e.*, each state in the MDP represents exactly one world.

3. Each transition in the MDP takes one unit of time.

Our notation for an MDP most resembles the *reactive probabilistic labeled transition system* (RPLTS) [25, 56, 97]. Below, we will formally define an MDP with respect to a set of actions Act, and a set of atomic propositions, $B_{\mathcal{L}}$. When comparing MDPs to APT-Programs, we will assume that the APT-Program uses the same set of ground atoms, and that each state in an MDP has a unique atomic label. In this manner, we can equate MDP states with worlds in tp-interpretations. Hence, an MDP is defined as follows:

Definition 23 (MDP). *A Markov Decision Process (MDP) consists of a 4-tuple* $L = (S, \delta, P, lbl, s_1)$ *where:*

- S *is a finite set of states*
- $\delta \subseteq S \times Act \times S$ *is the transition relation*
- $P : \delta \to [0, 1]$ *is the transition probability distribution, which satisfies:*
  - $\forall s \in S, \forall a \in Act\ \sum_{s':(s,a,s') \in \delta} P(s, a, s') \in [0, 1]$
  - $\forall s \in S, \forall a \in Act\ (\exists s'(s, a, s') \in \delta) \Rightarrow \sum_{s':(s,a,s') \in \delta} P(s, a, s') = 1$
- $lbl : S \to 2^{B_{\mathcal{L}}}$ *is the labeling of each state that specifies the set of propositions that are true in a state. Each state has a unique set of propositions.*

- *$s_1 \in S$ is the initial state.*

When an MDP is employed with policy π, it means that in state s_i, action $\pi(s_i)$ is taken. An MDP that uses only a single policy is often referred to as a *Stochastic Process*, or *Markov Process*. With the definition of an MDP and notion of a policy, we can now state what it means for a tp-interpretation to satisfy an MDP.

Definition 24. *Let L be an MDP, π be a policy, I be a tp-interpretation, and t_{max} be the maximum value of time. We say that I **satisfies** the pair (L, π) iff: for all sequences of $n = t_{max}$ states, $seq \equiv s_1 \rightarrow \ldots \rightarrow s_i \rightarrow \ldots \rightarrow s_n$, there exists a thread Th such that:*

- *For every s_i in seq, $a \in lbl(s_i)$ iff $a \in Th(i)$*
- *$\prod_{i=1}^{n-1} P(s_i, \pi(s_i), s_{i+1}) = I(Th)$*

Further, we say that an interpretation I satisfies an MDP L and set of policies POL iff there exists a policy $\pi \in POL$ such that $I \models (L, \pi)$.

We can extend the notion of *entailment* described earlier to MDP's and describe entailment relationships between MDP's and APT-Programs. Based on this idea, we now can define a notion of *equivalence* between an MDP and an APT-Program as follows.

Definition 25 (Equivalence/Entailment)**.** *An MDP L and set of policies POL is **equivalent** to APT-Program $\mathcal{K}$ when tp-interpretation $I \models (L, POL)$ iff $I \models \mathcal{K}$. (L, POL) is said to entail $\mathcal{K}$ if for all tp-interpretations I, if $I \models (L, POL)$ then*

$I \models \mathcal{K}$. Finally, $\mathcal{K}$ is said to entail (L, POL) if for all tp-interpretations I, if $I \models \mathcal{K}$ then $I \models (L, POL)$.

With this notion, given an MDP and policy, we can now create an APT-Logic Program such that the set of satisfying interpretations for the MDP and policy is the same as the set of satisfying interpretations for the APT-Logic Program. We use these notions of entailment and equivalence to specify the semantic relationship between APT-Logic: if for any APT-Program there is an equivalent MDP and a set of policies, then we will consider APT-Logic to be no more expressive than MDPs. Soon we will see this is not the case, and that APT-Logic is in fact more expressive than MDPs.

First however, we provide the following formula notation. F is a mapping of states to formulas such that $F(s) \equiv (\bigwedge_{a \in lbl(s)} a) \wedge (\bigwedge_{b \notin lbl(s)} \neg b)$. Second, we provide the following probability measurement of a t-length sequence starting with state s_1 and ending with state s_t. We use the notation $s \rightarrow^t s'$ to denote the set of sequences of t transitions from s to s'.

Definition 26 (Sequence Probability Measure)**.** *Let L be an MDP, π be a policy, s_1, s_t be states, and t be a positive integer. The* sequence probability measure*, SPM is defined as follows:*

$$SPM_{L,\pi}(s_t, t) = \sum_{s_1 \rightarrow^{t-1} s_t} \left(\prod_{i=1}^{t-1} P(s_i, \pi(s_i), s_{i+1}) \right)$$

So, the SPM totals the probabilities of all sequences from the initial state to s_t in $t-1$ transitions.

Next, we will present Algorithm 8 that, given an MDP and set of policies (L, POL), creates an APT-Program $\mathcal{K}$ such that (L, POL) entails $\mathcal{K}$. This construction is guaranteed to be correct by the following theorem.

Algorithm 8 Generate APT-Program that is entailed by a given MDP and set of policies.

```
MAKE-APT($MDP\ L, PolicySet\ POL$)

1. Create annotated formula $F(s_1) : [1, 1, 1]$.

2. For each state $s$, and each time point $t$, there are $|POL|$ SPM's, one for each
   policy. Let $min(SPM_{L,\pi}(s, t))$ be the minimum such SPM.

3. For each state $s$, and each time point $t$, let $max(SPM_{L,\pi}(s, t))$ be the maxi-
   mum SPM.

4. For each time point $t \in [1, t_{max}]$, and each state $s_i$, create the following anno-
   tated formula: $F(s_i) : [t, min(SPM_{L,\pi}(s_i, t)), max(SPM_{L,\pi}(s_i, t))]$.
```

Theorem 8. *If an interpretation I satisfies MDP L with set of policies L, then it satisfies APT-Program $\mathcal{K}$ generated from MAKE-APT.*

Clearly, if we restrict the MDP to a single policy, then we can create an APT-Program using MAKE-APT that is equivalent to the MDP and single policy.

Corollary 1. *An interpretation I satisfies MDP L with policy π, iff it satisfies APT-Program $\mathcal{K}$ generated from MAKE-APT.*

It is interesting to note, however, that although we can create an APT-Logic Program that is entailed by a given MDP and set of policies, we cannot always create an APT-Logic Program that *entails* an MDP and a set of policies. The intuition is that, in certain circumstances we are guaranteed that an APT-Logic Program has an infinite number of satisfying interpretations. If an MDP and set of policies are created such that these circumstances hold, then creating an APT-Program that entails the given MDP and set of policies is impossible. Hence, we first make the claim of the special circumstance that guarantees an infinite number of satisfying interpretations. The claim is that for APT-Program $\mathcal{K}$, if there exists satisfying tp-interpretations for $\mathcal{K}$, I_1, I_2, such that for threads Th_1, Th_2, $I_1(Th_1) = 1$ and $I_2(Th_2) = 1$, then there is an infinite number of satisfying interpretations for $\mathcal{K}$. We describe why this is true in the following paragraph.

Let $c \in (0,1)$ and $b \in (c,1)$. Let I_3 represent an infinite number of interpretations such that $I_3(Th_1) = b$ and $I_3(Th_2) = (1-b)$. $\mathcal{K}$ is then satisfied by an infinite number of interpretations if all possible I_3 interpretations satisfy $\mathcal{K}$. Suppose by way of contradiction that some I_3 does not satisfy $\mathcal{K}$. We have two cases:

Case 1: There exists an unconstrained rule, r such that $I_3 \not\models r$.

Let $r = F \overset{\mathsf{fr}}{\rightsquigarrow} G : [\Delta t, \ell, u]$. Let $a_1 = \mathsf{fr}(Th_1, F, G, \Delta t)$ and $a_2 = \mathsf{fr}(Th_2, F, G, \Delta t)$. Let $a_1 \leq a_2$. By the definition of satisfaction, we know that $[a_1, a_2] \subseteq [\ell, u]$. By the definition of satisfaction, we know that $\sum_{Th \in \mathcal{T}} I_3(Th)\mathsf{fr}(Th, F, G, \Delta t) < \ell$ or $\sum_{Th \in \mathcal{T}} I_3(Th)\mathsf{fr}(Th, F, G, \Delta t) > u$ as $I_3 \not\models r$. Therefore, $b \cdot a_1 + (1-b) \cdot a_2 < \ell$ or $b \cdot a_1 + (1-b) \cdot a_2 > u$. However, clearly, $b \cdot a_1 + (1-b) \cdot a_2 \subseteq (a_1, a_2)$ which implies $b \cdot a_1 + (1-b) \cdot a_2 \subseteq [\ell, u]$. Hence, we have a contradiction.

Case 2: There exists a constrained rule, r such that $I_3 \not\models r$.

Let $r_i = F \overset{\mathsf{fr}}{\hookrightarrow} G : [\Delta t, \ell, u, \alpha, \beta]$. We have three cases:

Case 2.1: $Th_1, Th_2 \in \mathsf{ATS}_i$

Then, $\ell \leq 1 \leq u$ and the probabilities of both threads summed together must fall in this probability bounds. As $I_3(Th_1) + I_3(Th_2) = 1$, I_3 then must satisfy r_i, so we have a contradiction.

Case 2.2: Either $Th_1 \in \mathsf{ATS}_i$ or $Th_2 \in \mathsf{ATS}_i$

If $\ell \neq 1$, then there exists $c \in (0,1)$ such that there is an infinite number of interpretations as per the definition of I_3 such that $I_3 \models r_i$. If $\ell = 1$, then either I_1 or I_2 does not satisfy r_i. Hence, we have a contradiction.

Case 2.3: $Th_1, Th_2 \notin \mathsf{ATS}_i$

In this case, any interpretation that assigns probabilities only to Th_1 and Th_2 satisfies r_i. Therefore, I_3 must satisfy r_i.

Now we consider a very simple MDP with only two policies. We see that this MDP causes the above mentioned circumstances to occur. Hence, we cannot construct an APT-Program that entails the MDP and set of policies.

Let L be an MDP, the set of atoms, $B_{\mathcal{L}}$, be $\{a\}$, $S = \{s_1, s_2\}$ be such that $lbl(s_1) \equiv \{a\}$ and $lbl(s_2) \equiv \emptyset$, $Act = \{x, y\}$, $P(s_1, x, s_1) = 1$ and $P(s_1, x, s_2) = 0$, $P(s_1, y, s_1) = 0$, and $P(s_1, y, s_2) = 1$. We define the set of policies, $POL = \{\pi_1, \pi_2\}$

such that $\pi_1(s_1) = x$ and $\pi_2(s_1) = y$. Let $t_{max} = 2$. We claim that it is impossible to construct an APT-Program that entails (L, POL).

So, we can see why there does not exist an APT-Program that entails the MDP described above. Assume by way of contradiction that we can create an APT-Logic Program $\mathcal{K}$ such that all interpretations that satisfy (L, π_1) or (L, π_2) satisfy $\mathcal{K}$. As each MDP-policy tuple is satisfied by exactly one interpretation, we have the following threads and interpretations based on the set of worlds $W = \{w_1, w_2\}$ where $w_1 \equiv lbl(s_1)$ and $w_2 \equiv lbl(s_2)$.

- Thread $Th_1 \equiv \langle w_1, w_2 \rangle$. Let I_1 be an interpretation such that $I_1(Th_1) = 1$ and sets the probability of all other threads to zero.
- Thread $Th_2 \equiv \langle w_1, w_1 \rangle$. Let I_2 be an interpretation such that $I_2(Th_2) = 1$ and sets the probability of all other threads to zero.

Hence, APT-Logic Program $\mathcal{K}$ must be satisfied by exactly I_1 and I_2. However, by the claim above, any program satisfied by these two interpretations is also satisfied by an infinite number of interpretations, so we have a contradiction.

So, based on the earlier definition of *equivalence*, while we can construct an equivalent APT-Program for an MDP and a single policy, we cannot do so for an MDP and set of policies. However, is the opposite true? It is: it would be trivial to construct an MDP that entails an APT-Program, since the null MDP can accomplish this. This highlights a difference between MDPs and APT-Logic Programs: we cannot have rules that say *this relationship holds with probability* p_1 ***or*** *probability* p_2. However, we can express ranges of probabilities.

While we cannot create an APT-Program that entails a given MDP and set of policies, APT-Programs can be satisfied by tp-interpretations that cannot satisfy *any* MDP. In other words, there are APT-programs and tp-interpretations that satisfy those APT-programs where there is no MDP that is satisfied by that tp-interpretation. Consider the set of ground atoms $B_{\mathcal{L}} = \{a\}$ and $t_{max} = 4$ and the below APT-Logic Program, $\mathcal{K}$:

- $\mathsf{a} : [1, 1, 1]$

- $\mathsf{a} \overset{pfr}{\rightsquigarrow} \neg\mathsf{a} : [1, 0.5, 0.5]$ (or $\mathsf{a} \overset{pfr}{\hookrightarrow} \neg\mathsf{a} : [1, 0.5, 0.5, 1, 1]$)

We included an alternate second rule to illustrate that this type of expressiveness result is true about both constrained and unconstrained programs. Consider worlds $w_1 \equiv \{a\}$ and $w_2 \equiv \emptyset$. Let I be an interpretation that assigns probabilities to the threads below:

- $Th_1 \equiv \langle w_1, w_2, w_1, w_2, \rangle$, $I(Th_1) = 0.5$

- $Th_2 \equiv \langle w_1, w_1, w_1, w_1, \rangle$, $I(Th_2) = 0.5$

It is trivial to show that $I \models \mathcal{K}$. We claim that it is impossible to build an MDP L with set of policies POL such that tp-interpretation $I \models (L, POL)$.

Let $S = \{s_1, s_2\}$ such that $lbl(s_1) \equiv w_1$ and $lbl(s_2) \equiv w_2$. Suppose by way of contradiction that $I \models (L, POL)$. Therefore, there exists a policy, $\pi \in POL$ such that I satisfies (L, π). Hence, the following must be true:

- $P(s_1, \pi(s_1), s_2) \cdot P(s_2, \pi(s_2), s_1) \cdot (s_1, \pi(s_1), s_2) = I_1(th_1) = 0.5$

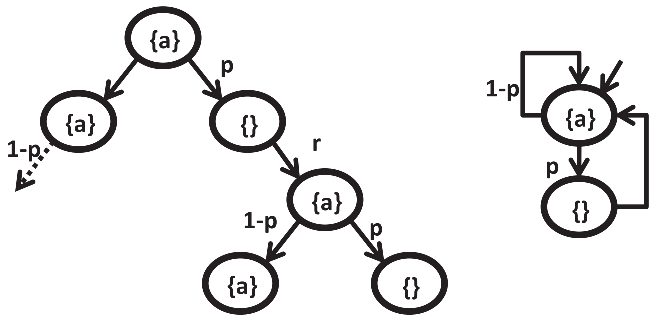


Figure 2.9: Left: Unrolled MDP in an attempt to create an MDP that satisfies interpretation I in the text. Notice how the sequence $\langle \{a\}, \{\}, \{a\}, \{a\} \rangle$ must be assigned a non-zero probability. Right: A standard representation of the MDP on the left. Notice that the MDP must allow for non-zero probability of threads that are given a zero probability in interpretation I.

- $P(s_1, \pi(s_1), s_1) \cdot P(s_1, \pi(s_1), s_1) \cdot (s_1, \pi(s_1), s_1) = I_1(th_2) = 0.5$

Refer to the left side of Figure 2.9 for a graphical representation of what follows. Let $P(s_1, \pi(s_1), s_2) = p$. Then, by the definition of an MDP, $P(s_1, \pi(s_1), s_1) = 1-p$. By the above equalities, $1-p > 0$. Let $P(s_2, \pi(s_2), s_1) = r$. Therefore, $p^2 \cdot r = 0.5$. Now consider the sequence $seq \equiv s_1 \to s_2 \to s_1 \to s_1$. The probability of this sequence must be set to zero, by the definition of I. Then, $P(seq) = p \cdot r \cdot (1-p) = 0$. However, we know that $p \cdot r$ cannot be zero and we know that $1-p > 0$. Hence, we have a contradiction.

The above discussion illustrates the differences between MDPs and APT-Logic. One could argue that the use of policies is overly restrictive for an MDP, *i.e.*, that perhaps the action should be decided based on time, or a combination of time and the current state. However, we can easily modify the above claim based on time or actions based on time and current state and obtain the same result. We suspect that it is not possible to have an MDP that replicates an APT-Logic Program without breaking the Markov Property, or causing a massive increase in the number of states, which also would change the assumption about the relationship between worlds and states.

2.6.2 Comparison with Probabilistic Computation Tree Logic (PCTL)

In this section, we show that APT-Logic rules differ significantly in meaning from similar structures presented in PCTL [12, 64], a well-known probabilistic temporal logic.

A derived operator in LTL with an intuition similar to that of our APT-Rules was introduced by Susan Owicki and Leslie Lamport in [132]. The operator, known as *leads-to* and an equivalent LTL formula are shown below (p and q are state formulas).

$$(p \curvearrowright q) \equiv \mathsf{G}(p \Rightarrow \mathsf{F}(q))$$

This formula intuitively says that if p is true in a state, then q must be true in the same (or future) state. As Owicki and Lamport's operator is based on LTL, it does not describe the correlation between p and q with probabilities or with reference to a specific time interval; q merely must happen sometime after (or with) p. A probabilistic version of CTL, known as PCTL [12, 64] introduces another operator based on a similar intuition; the authors refer to this operator as "leads-to" as well.

This derived operator, and the equivalent PCTL formula, are shown below (f_1 and f_2 are state formulas).

$$f_1 \curvearrowright^{\leq t}_{\geq p} f_2 \equiv \left[\mathsf{G}\left[(f_1 \Rightarrow \mathsf{F}^{\leq t}_{\geq p} f_2)\right]\right]_{>1}$$

Intuitively, this operator reads as "f_2 follows f_1 within t periods of time with a probability of p or greater". As PCTL formulas are satisfied by a Markov Process (an MDP with a single policy), satisfaction is determined by the transition probabilities. So, to determine if a Markov Process satisfies the above leads-to formula, we must compute the minimum probability of all sequences that start in a state satisfying f_1 and satisfy f_2 in t units of time or less. Note that this is determined by the transition probabilities of the Markov Process; hence, whether a Markov Process satisfies the lead-to operator depends on the interval between f_1 and f_2, but *not* on the total length of the sequence of states. So, if we limit the number of states being considered, using an operator such as $G^{\leq t_{max}}_{\geq 1}$ which PCTL provides to limit consideration to only the first t_{max} states, the Markov Process will satisfy the formula *regardless* of the value of t_{max}. Note that $G^{\leq t_{max}}_{\geq 1}$ placed at the head of the PCTL has no effect on the satisfaction of the formula as there is already a G path-quantifier included at the beginning of the leads-to operator.

As previously described, the frequency function is often highly sensitive to t_{max}. Our two primary examples of frequency functions, *pfr* and *efr*, are based on ratios of numbers of worlds in a given thread. For example, if we create a thread *Th* on a single atom a, we can see that for thread $\langle\{a\},\{a\},\{\}\rangle$, the value of $pfr(Th, a, \neg a, 1)$ is much greater than if *Th* were $\langle\{a\},\{a\},\{\},\{a\},\{a\},\{a\},\{a\}\rangle$. The fact that the length of the thread has an effect on the frequency function further illustrates how APT-Logic allows for reasoning beyond the restrictions of the Markov Property. The limited thread length forces us to consider worlds before and after a time-point we wish to reason about. If our probabilities were fixed, based on transition probabilities, they would not, and we would conform to the Markov Property.

Even though there are syntactic similarities, in the Appendix we provide a short example illustrating semantic differences between APT-rules and PCTL.

2.7 Chapter Summary

Statements of the form "Formula G is/was/will be true with a probability in the range $[\ell, u]$ in/within Δt units of time after formula F became true" are common. In this chapter, we have provided examples from four domains (stock markets, counter-terrorism, reasoning about trains, and power grids), but many more examples exist. Further, the counter-terrorism logic program (described in further detail in the next chapter) are more than mere examples – they are created using an extraction algorithm and a real-world data-set. They could be used, for instance, to describe when the health or environmental effects of industrial pollution may arise after a polluting event occurred, to the time taken for a medication to produce (with some probability) some effects. In the same way, they can be used in domains as widely divergent as industrial control systems to effects of educational investment on improved grades or graduation rates.

In this chapter, we have provided the concept of Annotated Probabilistic Temporal (APT) logic programs within which such statements can be expressed. APT-logic programs consist of two kinds of rules: *unconstrained* and *constrained* rules with an expected value style semantics and a more ordinary semantics. Both types of rules are parameterized by the novel concept of a *frequency function.* Frequency functions capture the probability that G follows F in exactly (or within) T time units *within a thread* (temporal interpretation). We show that this notion of "follows" can intuitively mean many different things, each leading to a different meaning. We propose an *axiomatic definition* of frequency functions which is rich enough to capture these differing intuitions and then provide a formal semantics for APT-logic programs.

We then study the problems of consistency and entailment for APT-logic programs. We show that the consistency problem is computationally intractable and is naturally solved via linear programming. We develop three successively more sophisticated linear programs for consistency checking and show that they lead to smaller linear programs (though not always). We also develop a suite of complexity results characterizing the entailment problem and provide algorithms to solve the entailment problem.

A natural question that arises in any probabilistic logic framework is "Where do the probabilities come from?" In order to answer this question, we develop the (straightforward) APT-Extract algorithm that shows how APT-logic programs can be derived from certain types of databases. We have applied APT-Extract to extract APT-rules about several terror groups (further details on these programs are provided in the next chapter).

Last, but not least, we have developed a detailed comparison between our APT-framework and two well known frameworks: Markov decision processes [140] and probabilistic computation tree logic [64]. We show the former can be captured within APT-logic program framework (but not vice versa). The latter has a more complex relationship with APT-logic programs, but cannot express intra-thread properties of the type expressed via APT-logic programs.

We note that the algorithms of this chapter all rely on the solution to a linear program with an exponential number of variables, which is not practical for a real-world implementation. Additionally, our complexity results of NP and coNP hardness for consistency and entailment checking suggest that this is an intractable problem under the assumption that P$\neq$NP. In the next chapter, we take a more practical approach, resorting to approximation algorithms that provide sound, but incomplete solutions to consistency and entailment problems for APT-logic.

Chapter 3

Annotated Probabilistic Temporal Logic: Approximate Algorithms

In the previous chapter, we explored reasoning about an agent's behavior with respect to time by introducing APT logic. This framework allows us to reason about the probability that an agent takes a certain action at a given time based on a model consisting of probabilistic rules. In that chapter, we showed that the consistency and entailment in APT logic are NP and coNP hard respectively. In that chapter, we provided several sound and complete algorithms for these problems, but due to the complexity of the problem, these approaches are not viable for a real-world system. In this chapter, we take a more practical approach, creating sound, but incomplete algorithms for the consistency and entailment problems.[1]

[1]This chapter is based on [156] which was completed in cooperation with Gerardo Simari and V.S. Subrahmanian.

3.1 Chapter Introduction

In the previous chapter, we have shown that there are numerous applications where we need to make statements of the form "Formula G becomes true with $50-60\%$ probability 5 time units after formula F became true." Statements of this kind arise in a wide variety of application domains.

This chapter takes a more practical approach to the problems associated with Annotated Probabilistic Temporal (APT) logic already present in this dissertation. Although the previous chapter presented algorithms for consistency and entailment problems that are sound and complete, they are not practical for general problems. This chapter takes a more practical approach. We develop a fixpoint operator for APT-logic that we prove to be sound. We can use this operator to correctly identify many inconsistent APT-programs – although we cannot guarantee a program to be consistent by this means. Additionally, this operator can infer probability ranges for queries, but we cannot guarantee that they are the tightest possible bounds. Most importantly, finding the fixpoint of this operator is efficient to compute. We also show that some of the techniques can also be adopted in a sound algorithm for non-ground APT-programs, where we only require a partial grounding.

We also implement an algorithm for the ground case and perform experiments on two data sets — the well known Minorities at Risk Organization Behavior (MAROB) data set [10] that tracks behaviors of numerous terror groups, and another real-world data counter-insurgency data from the Institute for the Study of War [72] (ISW). We used the algorithm APT-EXTRACT from the previous chapter to automatically learn 23 APT-logic programs — *no bias exists in these APT-logic programs as no one manually wrote them.* We then conducted experiments using those APT-logic programs and entailment problems were solved on an average in under 0.1 seconds per ground rule, while in the other, it took up to 1.3 seconds per ground rule. Consistency was also checked in a reasonable amount of time. *To the best of our knowledge, ours is the first implementation of a system for reasoning simultaneously about logic, time, and probabilities without making independence or Markovian assumptions.*

The chapter is organized as follows. Section 3.2 extends the syntax and semantics of APT LPs from the last chapter to add integrity constraints (ICs) as well as probabilistic time formulas (ptf's) – a generalization of the "annotated formulas" from the previous chapter (also seen in [34]). Section 3.3 shows that consistency and entailment in APT-logic are in-NP and in-coNP, respectively, matching the hardness results from the previous chapter (identifying these respective problems as NP-complete and coNP-complete). Section 3.4 describes our approximate fixpoint algorithm which is based on a sound (but not complete) fixpoint operator. The operator works by syntactically manipulating the rules in the APT-program to iteratively tighten the probability bounds of the formula whose entailment is being checked. We adapt the techniques for a consistency-checking and entailment algorithms for non-ground APT-programs in Section 3.5 (note that these algorithms do not require a full grounding of a program). In Section 3.6 we present our implementation of the fixpoint approach to solving consistency and entailment problems for ground programs. Finally, in Section 3.7, we provide an overview of related work.

Before continuing, we note that applications such as those above use automated rule learning (e.g. using the **APT-Extract** algorithm of the previous chapter) to automatically learn relationships and correlations between atoms. In particular, the existence of specific such relationships make independence and Markovian assumptions invalid for these types of applications.

1. $\mathsf{sec\_rumor} \wedge \mathsf{earn\_incr(10\%)} \overset{pfr}{\hookrightarrow} \mathsf{stock\_decr(10\%)} : [2, 0.65, 0.97, 0.7, 1.0]$

   An SEC rumor and a rumor of an earnings increase leads to a stock price decrease of 10% in 2 days with probability $[0.65, 0.97]$.

2. $\mathsf{sec\_rumor} \wedge \mathsf{earn\_incr(10\%)} \overset{pfr}{\hookrightarrow} \mathsf{stock\_decr(10\%)} \wedge \mathsf{cfo\_resigns} : [2, 0.68, 0.95, 0.7, 0.8]$

   An SEC rumor and a rumor of an earnings increase of 10% leads to the CFO resigning in exactly 1 days with a probability $[0.5, 0.95]$.

3. $\mathsf{OCC(cfo\_resigns)} : [0, 1]$

   The CFO resigns between 0 and 1 times (*i.e.*, $[\mathsf{lo}, \mathsf{up}] = [0, 1]$).

4. $\mathsf{BLK(sec\_rumor)} :< 4$

   An SEC rumor cannot be reported more than 3 days in a row (*i.e.*, $\mathsf{blk} = 4$).

5. $(\neg\mathsf{sec\_rumor} \wedge \neg\mathsf{rum\_incr(10\%)} \wedge \neg\mathsf{stock\_decr(10\%)} \wedge \neg\mathsf{cfo\_resigns}) : 1\wedge$

   $(\mathsf{sec\_rumor} \wedge \mathsf{rum\_incr(10\%)} \wedge \neg\mathsf{stock\_decr(10\%)} \wedge \neg\mathsf{cfo\_resigns}) : 2\wedge$

   $(\mathsf{sec\_rumor} \wedge \neg\mathsf{rum\_incr(10\%)} \wedge \mathsf{stock\_decr(10\%)} \wedge \neg\mathsf{cfo\_resigns}) : 3\wedge$

   $(\mathsf{sec\_rumor} \wedge \mathsf{rum\_incr(10\%)} \wedge \neg\mathsf{stock\_decr(10\%)} \wedge \mathsf{cfo\_resigns}) : 4 : [1, 1]$

   Based on events that have already occured, we can state things such as "at day 1 there was no SEC rumor, there is no rumor of a stock increase, the stock price did not decrease, and the CFO did not resign."

Figure 3.1: $\mathcal{K}_{stock}$, a toy APT-Logic Program about stocks.

1. $\mathsf{detainment\_distr(2)} \wedge \mathsf{detainment\_relig(1)} \overset{efr}{\rightsquigarrow} \mathsf{attack\_relig(1)}$:[2, 0.0906, 0.1906]

   A detainment in district 2 and detainment in an area where religion 1 dominates is followed by an attack in an area where religion 1 dominates within 2 days with a probablilty $[0.0906, 0.1906]$.

2. $\mathsf{attack\_neigh(28)} \wedge \mathsf{attack\_relig(1)} \overset{efr}{\rightsquigarrow} \mathsf{cache\_relig(1)}$:[7, 0.6833, 0.7833]

   An attack in neighborhood 28 and an attack in an area where religion 1 dominates is followed by a cache being found in an area where religion 1 dominates within 7 days with a probablilty $[0.6833, 0.7833]$.

3. $\mathsf{cache\_distr(2)} \overset{efr}{\rightsquigarrow} \mathsf{detainment\_relig(2)}$:[10, 0.6559, 0.7558]

   Cache being found in district 2 is followed by a a detainment in an area where religion 2 dominates within 10 days with a probablilty $[0.6559, 0.7558]$.

4. $\mathsf{detainment\_distr(2)} \overset{efr}{\rightsquigarrow} \mathsf{attack\_distr(7)}$:[10, 0.1346, 0.2346]

   A detainment in district 2 is followed by a an attack in district 7 within 10 days with a probablilty $[0.1346, 0.2346]$.

5. $\mathsf{attack\_neigh(28)} \overset{efr}{\rightsquigarrow} \mathsf{detainment\_distr(2)}$:[9, 0.5410, 0.6500]

   An attack in neighborhood 28 is followed by a detainment in district 2 within 9 days with a probablilty $[0.5410, 0.6500]$.

6. $\mathsf{cache\_distr(5)} \overset{efr}{\rightsquigarrow} \mathsf{strike\_relig(1)}$:[8, 0.2833, 0.3833]

   A cache found in disrict 5 is followed by a precision strike conudcted in an area domnated by religion 1 within 8 days with a probablilty $[0.2833, 0.3833]$.

Figure 3.2: $\mathcal{K}_{ISW}$ a real-world APT-Logic Program extracted from counterinsurgency data.

1. $\mathsf{orgst1(1)} \wedge \mathsf{orgst11(2)} \wedge \mathsf{domorgviolence(2)} \overset{efr}{\leadsto} \mathsf{armattack(1)}$:[2, 0.95, 1]

   Whenever education and propaganda are used as a minor strategy, coalition building is used as a major strategy, and the group is using domestic violence regularly by targeting security personnel (but not government non-security personnel or civilians), the group carries out armed attacks within two time periods with probability at least 0.95.

2. $\mathsf{orgst1(1)} \wedge \mathsf{orgst11(2)} \wedge \mathsf{domorgviolence(2)} \overset{efr}{\leadsto} \mathsf{dsecgov(1)}$:[3, 0.95, 1]

   This rule has the same antecedent as the previous one, but the consequent stands for the group targeting people working for the government in security, or in non-state armed militias.

3. $\mathsf{violrhetrans(0)} \wedge \mathsf{orgst5(0)} \wedge \mathsf{drug(0)} \overset{efr}{\leadsto} \mathsf{armattack(1)}$:[2, 0.58, 0.68]

   Whenever the group does not justify targeting transnational entities in public statements, uses non-coercive methods to collect local support (as a minor strategy), and does not engage in drug production/traficking, armed attacks are carried out within two time periods with probability between 0.58 and 0.68.

4. $\mathsf{orgst1(1)} \wedge \mathsf{orgst11(2)} \wedge \mathsf{orgst8(2)} \overset{efr}{\leadsto} \mathsf{dsecgov(1)}$:[3, 0.9500, 1]

   Whenever education and propaganda are used as a minor strategy, coalition building is used as a major strategy, and insurgencies are used as a major strategy, the group targets people working for the government in security, or in non-state armed militias, within 3 time periods with probability at least 0.95.

Figure 3.3: $\mathcal{K}_{MAROB}$ a real-world APT-Logic Program extracted from Minorities at Risk Organizational Behavior data.

3.2 Technical Background

This section extends the syntax and semantics of APT LPs from the previous chapter to include integrity constraints, probabilistic time formulas, and non-ground semantics for all previously introduced constructs.

3.2.1 Syntax

We assume the existence of a logical language $\mathcal{L}$ as specified in the previous chapter (see page 26). We also assume the existence of a finite set $\mathcal{F}$ whose members are called *frequency function* symbols (see the previous chapter, page 32). A (ground) term, atom, and formula are defined as per the previous chapter.

Also as in the last chapter, we assume that all applications are interested in reasoning about an arbitrarily large, but fixed size window of time, and that $\tau = \{1, \ldots, t_{max}\}$ denotes the entire set of time points we are interested in. t_{max} can be as large as an application user wants, and the user may choose his granularity of time according to his needs.

We now extend the syntax with the definition of a "time formula."

Definition 27 (Time Formula)**.** *A **time formula** is defined as follows:*

- *If F is a (ground) formula and $t \in [1, t_{max}]$ then $F : t$ is an (ground) **elementary** time formula.*

- *If ϕ, ρ are (ground) time formulas, then $\neg\phi$, $\phi \wedge \rho$, and $\phi \vee \rho$ are (resp. ground) time formulas.*

Example 3.2.1. *Consider the ground atoms in the* APT*-program from Figure 3.1. The expression* $(\neg \mathit{sec\_rumor} \wedge \neg \mathit{rum\_incr(10\%)} \wedge \neg \mathit{stock\_decr(10\%)} \wedge \neg \mathit{cfo\_resigns}) : 1$ *is an elementary time formula.*

Throughout, we will use Greek letters ϕ, ρ for time formulas and capital letters F, G for regular formulas. We now extend a time formula to include a probability annotation.

Definition 28. *If* ϕ *is a (ground) time formula and* $[\ell, u] \subseteq [0, 1]$*, then* $\phi : [\ell, u]$ *is a (resp. ground)* ***probabilistic time formula****, or* ***ptf*** *for short.*

Note that when considering ptf's of the form $F : t : [\ell, u]$*, we will sometimes abuse notation and write* $F : [t, \ell, u]$*.*

Example 3.2.2. *Item 5 in the* APT*-program from Figure 3.1 is a ptf.*

Intuitively, $\phi : [\ell, u]$ says time formula ϕ is true with a probability in $[\ell, u]$.[2] Our next extension to the syntax of the previous chapter are *integrity constraints*.

Definition 29 (Integrity constraint)**.** *Suppose* $A_i \in B_{\mathcal{L}}$ *and* $[\mathsf{lo}_i, \mathsf{up}_i] \subseteq [0, t_{max}]$*. Then* $\mathsf{OCC}(A_i) : [\mathsf{lo}_i, \mathsf{up}_i]$ *is called an* occurrence IC*. If* $\mathsf{blk}_i \in [2, t_{max} + 1]$ *is an integer, then* $\mathsf{BLK}(A_i) :< \mathsf{blk}_i$ *is called a block-size IC. If* A_i *is ground then the occurrence (resp. block-size) IC is ground – otherwise it is non-ground.*

An occurrence IC $\mathsf{OCC}(A_i) : [\mathsf{lo}_i, \mathsf{up}_i]$ says that A must be true at least lo_i times and at most up_i times. Likewise, the block-size IC says that A cannot be

[2]**Assumption:** Throughout the chapter we assume, for both ptf's and APT-rules, that the numbers ℓ, u can be represented as rationals a/b where a and b are relatively prime integers and the length of the binary representations of a and b is fixed.

consecutively true for blk_i or more time points. Figure 3.1 also contains an example occurrence IC and an example block-size IC.

Example 3.2.3 (Integrity Constraints)**.** *Consider the ground atoms in the* APT*-program from Figure 3.1 and* $t_{max} = 6$*. Suppose historical data indicates that for a sequence of* 6 *days, there is never more than* 1 *day where the CFO resigns. Hence, we should add the constraint* $\mathit{OCC}(\mathit{cfo\_resigns}) : [0, 1]$ *to the program. There are other types of integrity constraints that could be useful in this domain. For example, a drastic stock price decrease may never occur more than a few times a quarter.*

To see why block-size constraints are natural, consider the ground atom $\mathit{sec\_rumor}$*. Suppose there is never more than* 3 *days historically where an SEC rumor is reported. This would make the constraint* $\mathit{BLK}(\mathit{sec\_rumor}) :< 4$ *appropriate. Other examples of such constraints in this domain would be reports of profits, which only occur once per quarter (*i.e.*, we would have* $\mathsf{blk} = 2$ *for such a case).*

We have automatically extracted APT-programs from the ISW and MAROB data sets mentioned earlier. In the case of the ISW data set, occurrence and block-size constraints are needed because militant groups have constrained resources, *i.e.*, a limited amount of personnel and munitions to carry out an attack. Hence, an occurrence integrity constraint can limit the amount of attacks we believe they are capable of in a given time period. Likewise, such groups often limit the amount of consecutive attacks, as police and military often respond with heightened security. Block-size constraints allow us to easily encode this into our formalism.

We now extend the definition of APT rules and programs from the previous chapter to include non-ground versions of these syntactic elements.

Definition 30 (APT Rules and Programs)**.** *(i) Suppose F, G are (ground) formulas, Δt is a time interval, $[\ell, u]$ is a probability interval, and* $\mathsf{fr} \in \mathcal{F}$ *is a frequency function symbol. Then $F \overset{\mathsf{fr}}{\rightsquigarrow} G : [\Delta t, \ell, u]$ is an (ground)* APT rule*.*
(ii) An (ground) APT logic program *is a finite set of (ground) APT rules, ptf's, and integrity constraints.*
(iii) Given a non-ground APT-logic program $\mathcal{K}^{(ng)}$, the set of ground instances of all rules, ptf's, and IC's in $\mathcal{K}^{(ng)}$ is called the grounding of $\mathcal{K}^{(ng)}$.

Note: Unless specified otherwise, throughout this chapter, APT-logic programs, rules, IC's, and ptf's are ground.

Example 3.2.4. *Figure 3.1 shows a small APT LP dealing with the stock market, together with an intuitive explanation of each rule.*

3.2.2 Semantics

We now extend the semantics of APT LPs from the previous chapter to account for the extended syntax and the non-ground case. The structures of *worlds* and *threads* are defined exactly as in the previous chapter (see page 29). However, here we define a notion of a thread satisfying a time formula or integrity constraint as follows:

Definition 31. *(i) Given thread Th and ground time formula ϕ, we say Th* ***satisfies*** *ϕ (written $Th \models \phi$) iff:*

1. $\mathsf{at\_station}(\mathsf{T}, \mathsf{S_1}) \wedge \mathsf{adjEast}(\mathsf{S_1}, \mathsf{S_2}) \overset{efr}{\leadsto} \mathsf{at\_station}(\mathsf{T}, \mathsf{S_2}) : [4, 0.85, 1]$

   If train T is at station S_1 and the station adjacent to it to the East is S_2, T will be at station S_2 within 4 time units with a probability bounded by $[0.85, 1]$.

2. $\mathsf{at\_station}(\mathsf{T}, \mathsf{S_1}) \wedge \mathsf{adjWest}(\mathsf{S_1}, \mathsf{S_2}) \overset{efr}{\leadsto} \mathsf{at\_station}(\mathsf{T}, \mathsf{S_2}) : [2, 0.6, 0.7]$

   If train T is at station S_1 and the station adjacent to it to the West is S_2, T will be at station S_2 within 2 time units with a probability in the interval $[0.6, 7]$.

3. $\bigwedge_{t=1}^{t_{max}} \mathsf{adjEast(stnA,\ stnB)} : t : [1,1]$, $\bigwedge_{t=1}^{t_{max}} \mathsf{adjEast(stnB,\ stnC)} : t : [1,1]$, $\bigwedge_{t=1}^{t_{max}} \mathsf{adjWest(stnB,\ stnA)} : t : [1,1]$, $\bigwedge_{t=1}^{t_{max}} \mathsf{adjWest(stnC,\ stnB)} : t : [1,1]$

   Probabilistic time formulas specifying that Station B is (always) adjacent to the East of A, and C is adjacent to the East of B.

4. $\mathsf{at\_station}(\mathsf{train1}, \mathsf{stnA}) : 1 : [0.5, 0.5]$

   For a given sequence of events, train 1 will be at station A at time period 1 with a probability of 0.50.

5. $\mathsf{at\_station}(\mathsf{train2}, \mathsf{stnA}) : 2 : [0.48, 0.52]$

   For a given sequence of events, train 2 will be at station A at time period 2 with a probability bounded by $[0.48, 0.52]$.

Figure 3.4: $\mathcal{K}_{train}$, a toy APT-Logic Program modeling rail transit. Items 1-2 are non-ground APT-Rules, the formulas in 3 are probabilistic temporal formulas, and items 4-5 are annotated formulas. The English translation of each rule is also provided.

- $\phi \equiv F : t$: *Th* $\models \phi$ *iff* $Th(t) \models F$

- $\phi \equiv \neg\rho$: *Th* $\models \phi$ *iff* *Th* $\not\models \rho$

- $\phi \equiv \rho \wedge \rho'$: *Th* $\models \phi$ *iff* *Th* $\models \rho$ *and* *Th* $\models \rho'$

- $\phi \equiv \rho \vee \rho'$: *Th* $\models \phi$ *iff* *Th* $\models \rho$ *or* *Th* $\models \rho'$

(ii) Given thread Th and ground occurrence IC $\mathsf{OCC}(A_i) : [\mathsf{lo}_i, \mathsf{up}_i]$*, we say Th* ***satisfies*** $\mathsf{OCC}(A_i) : [\mathsf{lo}_i, \mathsf{up}_i]$ *iff* $|\{i \mid Th(i) \models A_i\}| \in [\mathsf{lo}_i, \mathsf{up}_i]$.

(iii) Given thread Th and block-size IC $\mathsf{BLK}(A_i) :< \mathsf{blk}_i$*, we say Th* ***satisfies*** $\mathsf{BLK}(A_i) :< \mathsf{blk}_i$ *iff there does not exist an interval* $[i, i + \mathsf{blk}_i - 1]$ *such that for all* $j \in [i, i + \mathsf{blk}_i - 1]$, $Th(j) \models A_i$.

(iv) Th satisfies a non-ground formula or IC iff it satisfies all ground instances of it.

Given a set $\mathcal{T}$ of threads and a set IC of integrity constraints, we use $\mathcal{T}(IC)$ to refer to the set $\{Th \in \mathcal{T} | Th \models IC\}$.

We use the symbol '$\models$' to denote entailment between two time formulas.

Definition 32. *Given time formulas* ϕ, ρ*, we say:* $\phi \models \rho$ *iff* $\forall Th \in \mathcal{T}$ *s.t.* $Th \models \phi$*, it is the case that* $Th \models \rho$.

If we view time formulas as sets of threads, we can think of $\phi \models \rho$, as equivalent to $\phi \subseteq \rho$.

As in the previous chapter, a *temporal probabilistic (tp) interpretation* gives us a probability distribution over all possible threads. Thus, a tp-interpretation I assigns a probability to each thread. This reflects the probability that the world will in fact evolve over time in accordance with what the thread says. We now define what it means for a tp-interpretation to satisfy a ptf or integrity constraint.

Definition 33. *(i) Given interpretation I and ptf $\phi : [\ell, u]$, we say I* ***satisfies*** *$\phi : [\ell, u]$ (written $I \models \phi : [\ell, u]$) iff:*

$$\ell \leq \sum_{\substack{Th \in \mathcal{T} \\ Th \models \phi}} I(Th) \leq u$$

(ii) Given interpretation I and occurrence IC $\mathsf{OCC}(A_i) : [\mathsf{lo}_i, \mathsf{up}_i]$, we say I ***satisfies*** *$\mathsf{OCC}(A_i) : [\mathsf{lo}_i, \mathsf{up}_i]$ (written $I \models \mathsf{OCC}(A_i) : [\mathsf{lo}_i, \mathsf{up}_i]$) iff $\forall Th \in \mathcal{T}$ s.t. $Th \not\models \mathsf{OCC}(A_i) : [\mathsf{lo}_i, \mathsf{up}_i]$, it is the case that $I(Th) = 0$.*
(iii) Given interpretation I and block-size IC $\mathsf{BLK}(A_i) :< \mathsf{blk}_i$, we say I ***satisfies*** *$\mathsf{BLK}(A_i) :< \mathsf{blk}_i$ (written $I \models \mathsf{BLK}(A_i) :< \mathsf{blk}_i$) iff $\forall Th \in \mathcal{T}$ s.t. $Th \not\models \mathsf{BLK}(A_i) :< \mathsf{blk}_i$, it is the case that $I(Th) = 0$.*
(iv) Interpretation I ***satisfies*** *a non-ground formula or IC iff it satisfies all ground instances of it.*

With the above definition, we now define a special type of ptf that can be used to specify a set of threads that start with the same worlds – the intuition is based on the idea of a *prefix* in [25].

Definition 34. *For $n \leq t_{max}$, let $F_1, \dots, F_i, \dots, F_n$ be formulas s.t. each F_i is satisfied by exactly one world. Then, the following ptf:*

$$F_1 : 1 \wedge \dots \wedge F_i : i \wedge \dots \wedge F_n : n : [1, 1]$$

is called a ***prefix****.*

Example 3.2.5. *Item 5 in the APT-program from Figure 3.1 is a prefix.*

Intuitively, including a prefix in an APT-program forces the first n worlds of every thread assigned a non-zero probability to satisfy certain formulas. Further, we can use a prefix to force the first n worlds of every thread with a non-zero probability to be the same. For example, if we want the i'th world of thread Th to be set to world w, we would simply use the following formula as F_i in the prefix: $(\bigwedge_{a \in w} a) \wedge (\bigwedge_{a \notin w} \neg a)$.

The definition of a frequency function is also exactly the same as in the previous chapter. For the sake of simplicity, in this chapter we only use the *existential* frequency function (also defined in the previous chapter). Most techniques in this chapter can be easily extended for use with other frequency functions. Now we extend the definition of satisifaction of APT rules to account for the non-ground case.

Definition 35 (Satisfaction of APT rules)**.** *Let $r = F \overset{fr}{\rightsquigarrow} G : [\Delta t, \ell, u]$ be an APT rule and I be a tp-interpretation.*

*(i) If r is a **ground rule**, interpretation I satisfies r (denoted $I \models r$) iff*

$$\ell \leq \sum_{Th \in \mathcal{T}} I(Th) \cdot \mathsf{fr}(Th, F, G, \Delta t) \leq u.$$

(ii) Interpretation I satisfies a non-ground rule r iff I satisfies all ground instances of r.

Interpretation I satisfies an APT-program iff it satisfies all rules, ptf's, and IC's in that program. Given an APT-program $\mathcal{K}$, we will often refer to the set of integrity constraints in $\mathcal{K}$ as simply IC.

Intuitively, the APT rule $F \overset{\mathsf{fr}}{\rightsquigarrow} G : [\Delta t, \ell, u]$ evaluates the probability that F leads to G in Δt time units as follows: for each thread, it finds the probability of the thread according to I and then multiplies it by the frequency (in terms of fraction of times) with which F is followed by G in Δt time units according to frequency function fr. This product is like an expected value in statistics where a value (frequency) is multiplied by a probability (of the thread). It then sums up these products across all threads.

3.3 Complexity

In the previous chapter, we showed that consistency and entailment in APT-logic are NP-hard (consistency) and coNP-hard (entailment). In this section, we prove that consistency is in the complexity class NP and entailment is in the complexity class coNP. The result is somewhat surprising, because the exact algorithms presented in the previous chapter relied on the solution to linear programs with an exponential number of variables. For example, consider the following linear program.

Definition 36 (CONS)**.** *Given an APT-logic program, $\mathcal{K}$, where $IC \subset \mathcal{K}$ is the set of integrity constraints in $\mathcal{K}$, we can create the linear constraints* $\mathsf{CONS}(\mathcal{K})$ *as follows:*

For each $Th_j \in \mathcal{T}(IC)$, variable v_j denotes the probability of thread Th_j.

(1) $\sum_{j=1}^{|\mathcal{T}(IC)|} v_j = 1$

(2) $\forall F_i \overset{\mathsf{fr}_i}{\rightsquigarrow} G_i : [\Delta t_i, \ell_i, u_i] \in \mathcal{K}$ *(a)* $\ell_i \leq \sum_{j=1}^{|\mathcal{T}(IC)|} \mathsf{fr}_i(Th_j, F_i, G_i, \Delta t_i) \cdot v_j$

(b) $u_i \geq \sum_{j=1}^{|\mathcal{T}(IC)|} \mathsf{fr}_i(Th_j, F_i, G_i, \Delta t_i) \cdot v_j$

(3) $\forall \phi_i : [\ell_i, u_i] \in \mathcal{K}$ *(a)* $\ell_i \leq \sum_{Th_j \in \mathcal{T}(IC)\ Th_j \models \phi_i} v_j$

(b) $u_i \geq \sum_{Th_j \in \mathcal{T}(IC)\ Th_j \models \phi_i} v_j$

We proved in the previous chapter that there is a solution to $\mathsf{CONS}(\mathcal{K})$ iff $\mathcal{K}$ is consistent and that, given ptf $\phi : [\ell, u]$, let L be the minimization and U be the maximization of $\sum_{Th_j \in \mathcal{T}(IC)\ Th_j \models \phi} v_j$ subject to $\mathsf{CONS}(\mathcal{K})$. Then $\phi : [\ell, u]$ is entailed by $\mathcal{K}$ iff $[L, U] \subseteq [\ell, u]$. See Proposoiton 3 (page 50) and Proposition 12 (page 73) respectiely.

However, it turns out that we can be guaranteed a solution to the linear program where only a polynomial number of the variables are set to a value other than 0. Consider the following theorem from [24] and later used in [44] to show that deciding the validity of a formula in the logic of [44] is NP-Complete.

Theorem 9 ([24, 44])**.** *If a system of m linear equalities and/or inequalities has a nonnegative solution, then it has a nonnegative solution with at most m positive variables.*

We can leverage the previous two results to guarantee the existence of an interpretation that assigns a zero probability to all but a polynomial number of threads, thus giving us a "small model" theorem.

Theorem 10. *Deciding if* APT*-program* $\mathcal{K}$ *is consistent is NP-complete if* $|\mathcal{K}|$ *is a polynomial in terms of* $|B_{\mathcal{L}}|$*.*

Theorem 11. *Deciding if* APT*-rule* r *is entailed by* APT*-program* $\mathcal{K}$ *is coNP-complete if* $|\mathcal{K}|$ *is a polynomial in terms of* $|B_{\mathcal{L}}|$*.*

One may wonder if APT-programs can be made more tractable if we assume a single probability distribution over threads, that is a single tp-interpretation. Unfortunately, even if we assume a uniform probability distribution, this special case is still not tractable.

Theorem 12. *Given* APT*-program* $\mathcal{K}$*, interpretation* I*, and ptf* ϕ*, determining the maximum* ℓ *and minimum* u *such that* $\phi : [\ell, u]$ *is entailed by* $\mathcal{K}$ ***and*** *is satisfied by* I *is #P-hard. Furthermore, for constant* $\epsilon > 0$*, approximating either the maximum* ℓ *and/or minimum* u *within* $2^{|B_{\mathcal{L}}|^{1-\epsilon}}$ *is NP-Hard.*

The above theorem is proved using an interpretation that assigns a uniform probability across all threads. The negative approximation result follows from a result of [145].

Although it remains an open question if the APT-entailment problem (without the single-interpretation requirement) can be approximated within a reasonable factor, the above result is not encouraging.[3] Further, Definition 36 illustrates several challenges relating the intractability of this problem. (i) First, we need to compute $\mathcal{T}(IC)$, which is a challenge because $\mathcal{T}$ contains $2^{t_{max} \cdot card(B_{\mathcal{L}})}$ possible threads and

[3]As an aside, as the construction in the proof of Theorem 12 does not depend on multiple time-points, this result holds for the probabilistic logic of [131] as well.

each must be examined to see if it satisfies IC; (ii) Second, the constraints in items (1-2) may contain up to $O\left(2^{t_{max}\cdot card(B_{\mathcal{L}})}\right)$ variables (this bound can be tightened), so even though linear programming is polynomial [79], the input is exponential in the size of t_{max} and $B_{\mathcal{L}}$. In practice, even if we consider $t_{max} = 10$ and $B_{\mathcal{L}}$ to consist of just 100 ground atoms, we are looking at the possibility of examining $2^{1,000}$ threads to find $\mathcal{T}(IC)$ and writing constraints containing exponentially large numbers of variables. In practice, we will not be able to even write these constraints. With these intractability results in mind, we proceed to develop heuristics in the next two sections.

3.4 A Sound but Incomplete Fixpoint-Computation Algorithm: The Ground Case

This section presents a heuristic algorithm based on a fixpoint operator Γ which maps APT-programs to APT-programs and iteratively tightens the probability bounds on rules and ptf's in the program. To find probability bounds on some time formula ϕ, we simply add the ptf $\phi : [0, 1]$ to the program, iteratively apply Γ until a fixed point is reached, and then examine the bounds on the ptf formed with ϕ in the resulting program. Our approach is sound – so, if the interval $[\ell, u]$ is assigned to ϕ, then $\mathcal{K}$ entails $\phi : [\ell, u]$ (provided, of course, that $\mathcal{K}$ is consistent). However, there may exist some $[\ell', u'] \subset [\ell, u]$ such that $\phi : [\ell', u']$ is also entailed.

Our algorithm requires that $\mathcal{K}$ contain at least one APT-rule of the form

$F : [\ell, u]$. This is not really a restriction in most applications where a prefix would exist (cf. Definition 34, Page 107). The rest of the section is organized as follows. Section 3.4.1 describes how to find bounds on a frequency function given ptf's. Section 3.4.2 describes how to use frequency bounds to syntactically manipulate rules and ptf's in APT-programs – which in turn allow us to tighten the probability bounds. Section 3.4.3 performs various syntactic manipulations in the Γ operator and shows that the operator has a least fixed point. Finally, Section 3.4.4 demonstrates how Γ can also be used to check the consistency of an APT logic program. Again, such a consistency check is sound but not complete – Γ correctly identifies inconsistent programs but does not guarantee consistency.

3.4.1 Bounding Frequency Function Values

In this chapter, we only use the *efr* frequency function. However, our techniques can be easily adapted to other frequency functions such as *pfr* from the previous chapter. Our first definition is a function, EFR, which returns tight bounds on *efr* given F, G, and Δt.

Definition 37. *Suppose F, G are formulas, Δt is a time point, and ϕ is a time formula. We define $EFR(F, G, \Delta t, \phi) = [\alpha_{tight}, \beta_{tight}]$ where*

$$\begin{aligned}\alpha_{tight} &= \mathbf{inf}\{efr(Th, F, G, \Delta t) \mid Th \in \mathcal{T} \wedge Th \models \phi\}.\\ \beta_{tight} &= \mathbf{sup}\{efr(Th, F, G, \Delta t) \mid Th \in \mathcal{T} \wedge Th \models \phi\}.\end{aligned}$$

The intuition in the above definition is that α_{tight} is the least value of efr (w.r.t. formulas F, G and time interval Δt) for all threads satisfying ϕ. Likewise,

β_{tight} is the greatest value of efr (w.r.t. formulas F, G and time interval Δt) for all threads satisfying ϕ. We can easily approximate $[\alpha_{tight}, \beta_{tight}]$ when we make certain assumptions on ϕ. Consider the following special case of a ptf:

Definition 38. *Suppose $ETF \equiv \{F_1 : t_1, \ldots, F_n : t_n\}$ is a set of elementary time formulas, where $n \leq t_{max}$ and for any two such formulas, $F_i : t_i, F_j : t_j \in ETF$, $t_i \neq t_j$. Then $F_1 : t_1 \wedge \ldots \wedge F_n : t_n$ is a* ***time conjunction****.*

Example 3.4.1. *Item 5 in the APT-program from Figure 2.1 is a time-conjunction. We shall refer to this time-conjunction as ϕ_{stock} in later examples.*

We notice right away that a prefix (Definition 34, Page 107) is simply a special case of time conjunction annotated with probability $[1, 1]$. One useful property of time conjunctions that we leverage in our operator is the following.

Observation 3.4.1. *If $F_1 : t_1 \wedge \ldots \wedge F_n : t_n \wedge F_{n+1} : t'_1 \wedge \ldots \wedge F_{n+m} : t'_m$ and $G_1 : t_1 \wedge \ldots \wedge G_n : t_n \wedge G_{n+1} : t''_1 \wedge \ldots \wedge G_{n+k} : t''_k$ are time conjunctions, then*

$$(F_1 \wedge G_1) : t_1 \wedge \ldots \wedge (F_n \wedge G_n) : t_n \wedge F_{n+1} : t'_1 \wedge \ldots \wedge F_{n+m} : t'_m \wedge G_{n+1} : t''_1 \wedge \ldots \wedge G_{n+k} : t''_k$$

is also a time conjunction.

We leverage the above property in the following way: if we know a bound for $EFR(F, G, \Delta t, \phi)$ and $EFR(F, G, \Delta t, \phi \wedge \rho)$, we may be able to use this information to find probability bounds on ρ. We will describe this further when we discuss syntactic manipulation. Next, with a **time conjunction** in mind, we will show how to find a tight bound on EFR. In this case, we introduce the following notation and obtain a bound on EFR in Proposition 14.

Definition 39. *For formulas F, G, time Δt, and time conjunction ϕ, we define the following:*

- $cnt(\phi, F, G, \Delta t) = |\{t \in [1, t_{max} - \Delta t] | \exists t' \in (t, t + \Delta t] \ s.t. \ (\phi \models F : t \wedge G : t')\}|$
- $end(\phi, F, G, \Delta t) = |\{t \in (t_{max} - \Delta t, t_{max}) | \exists t' \in (t, t_{max}] \ s.t. \ (\phi \models F : t \wedge G : t')\}|$
- $denom(\phi, F, \Delta t) = |\{t \in [1, t_{max} - \Delta t] | \exists Th \ s.t. \ (Th \models \phi) \wedge (Th \models F : t)\}|$
- $poss(\phi, F, G, \Delta t) = |\{t \in [1, t_{max} - \Delta t] | \exists t' \in (t, t + \Delta t] \ s.t. \ \exists Th \ s.t. \ (Th \models \phi) \wedge (Th \models F : t \wedge G : t')\}|$
- $endposs(\phi, F, G, \Delta t) = |\{t \in (t_{max} - \Delta t, t_{max}) | \exists t' \in (t, t_{max}] \ s.t. \ \exists Th \ s.t. \ (Th \models \phi) \wedge (Th \models F : t \wedge G : t')\}|$

The intuitions behind the components of Definition 39 are as follows. For a given $F, G, \Delta t$, *cnt* is simply the number of times in the first $t_{max} - \Delta t$ timesteps (of all threads satisfying some ptf ϕ) where a world satisfying F is followed by a world satisfying G within Δt time units. Likewise, *end* performs this count for the last Δt time units. Similarly, *poss* and *endposs* perform similar calculations, but rather than considering all threads that satisfy ϕ, there must only exist a thread satisfying ϕ where a world satisfying F is followed by a world satisfying G in Δt time units. The definition of *denom* captures the total number of times F is satisfied in the first $t_{max} - \Delta t$ worlds (for all threads satisfying ϕ). Due to the boundary condition of efr (refer to Section 3.2 for details), we use *end* and *endposs* to perform this count in the last $t_{max} - \Delta t$ worlds of the threads. Hence, in the below proposition, we are able to show that $EFR(F, G, \Delta t, \phi)$ is a subset of two fractions created from the components we defined.

Proposition 14. *For formulas F, G, time Δt, and time conjunction ϕ,*

$EFR(F, G, \Delta t, \phi) \subseteq$

$$\left[\frac{cnt(\phi, F, G, \Delta t) + end(\phi, F, G, \Delta t)}{denom(\phi, F, \Delta t) + end(\phi, F, G, \Delta t)}, \frac{poss(\phi, F, G, \Delta t) + endposs(\phi, F, G, \Delta t)}{denom(\phi, F, \Delta t) + endposs(\phi, F, G, \Delta t)}\right]$$

Example 3.4.2. *Consider the APT-program from Figure 2.1 that includes time conjunction ϕ_{stock} (see Example 3.4.1). Consider the pre and post conditions of rules 1-2; we shall refer to them as follows (in this and later examples):*

$$F_1 \equiv \quad \textit{sec\_rumor} \wedge \textit{rum\_incr(10\%)}$$

$$G_1 \equiv \quad \textit{stock\_decr(10\%)}$$

$$F_2 \equiv \quad \textit{sec\_rumor} \wedge \textit{rum\_incr(10\%)}$$

$$G_2 \equiv \quad \textit{stock\_decr(10\%)} \wedge \textit{cfo\_resigns}$$

Using Definition 39, we can determine that:

$$EFR(\phi_{stock}, F_1, G_1, 2) \subseteq [0.5, 1.0]$$

and

$$EFR(\phi_{stock}, F_2, G_2, 1) \subseteq [0.0, 0.667]$$

3.4.2 Theorems for Syntactic Manipulation

In the last section, we bounded the values that efr can have for a thread given some time formula ϕ. This section leverages that information to obtain tighter bounds on ptf's and APT-rules. First, we introduce a simple result that allows for syntactic manipulation of ptf's without these bounds.

Lemma 8. *Let $\rho : [\ell', u']$ be a ptf and I be an interpretation; then:*

1. *If $I \models \phi : [\ell, u]$, then $I \models \phi \wedge \rho : [\max(0, \ell + \ell' - 1), \min(u, u')]$*

2. *If $I \models \phi : [\ell, u]$, then $I \models \phi \vee \rho : [\max(\ell, \ell'), \min(1, u + u')]$*

3. *If $I \models \phi : [\ell, u]$ and $\phi \models \rho$ then $I \models \rho : [\ell, 1]$*

4. *If $I \models \phi : [\ell, u]$ and $\rho \models \phi$ then $I \models \rho : [0, u]$*

5. *If $I \models \phi : [\ell, u]$ then $I \models \neg\phi : [1 - u, 1 - \ell]$*

Example 3.4.3. *Suppose program $\mathcal{K}_{stock}$ entails ptf $\mathsf{sec\_rumor} : 6 : [0.3, 0.6]$. Then, it also entails $\neg\mathsf{sec\_rumor} : 6 : [0.4, 0.7]$.*

We notice right away that syntactic manipulation sometimes identifies inconsistent APT-programs. For example, if $\phi : [0.7, 0.6]$ is entailed via Lemma 8, then we know that $\mathcal{K}$ is not consistent. We explore this issue further in Section 3.4.4. Next, we use the bounds on EFR to syntactically manipulate APT-rules, yielding rules with tighter probability bounds – perhaps uncovering an inconsistent program. Theorem 13 tightens the bound when the APT-program includes a ptf that happens with probability 1. Its corollary tightens the lower bound given any ptf.

Theorem 13. *Suppose I is an interpretation and ϕ is a time formula such that $I \models \phi : [1, 1]$ and $EFR(F, G, \Delta t, \phi) \subseteq [\alpha, \beta]$. Then $I \models F \overset{efr}{\rightsquigarrow} G : [\Delta t, \alpha, \beta]$.*

Corollary 2. *Suppose I is an interpretation and ϕ is a time formula such that $I \models \phi : [\ell, u]$ and $EFR(F, G, \Delta t, \phi) \subseteq [\alpha, \beta]$. Then $I \models F \overset{efr}{\rightsquigarrow} G : [\Delta t, \alpha \cdot \ell, 1]$.*

The above theorem and corollary are proved by showing that the lower probability bound of an APT-rule has to be at least as much as the lower bound on the associated EFR for all threads.

Example 3.4.4. *Consider the scenario from Example 3.4.2. By the result of that example and Corollary 2, we know that $\mathcal{K}_{stock}$ must entail:*

$$\textsf{\textit{sec\_rumor}} \wedge \textsf{\textit{rum\_incr(10\%)}} \overset{efr}{\rightsquigarrow} \textsf{\textit{stock\_decr(10\%)}} : [2, 0.5, 1.0] \text{ and}$$

$$\textsf{\textit{sec\_rumor}} \wedge \textsf{\textit{rum\_incr(10\%)}} \overset{efr}{\rightsquigarrow} \textsf{\textit{stock\_decr(10\%)}} \wedge \textsf{\textit{cfo\_resigns}} : [1, 0.0, 0.667]$$

Note that we can now find a tighter bound on rule 2, obtaining a probability bound of $[0.5, 0.667]$, that is substantially tighter than $[0.5, 1]$ from the original rule using just one syntactic manipulation.

We can use APT-rules, EFR, and Theorem 8 to further tighten the bounds on ptf's with the following theorem.

Theorem 14. *Suppose F, G are formulas, ϕ, ρ are time formulas, I is an interpretation, and $[\alpha_1, \beta_1], [\alpha_2, \beta_2]$ are intervals such that $EFR(F, G, \Delta t, \phi) \subseteq [\alpha_1, \beta_1]$ and $EFR(F, G, \Delta t, \phi \wedge \rho) \subseteq [\alpha_2, \beta_2]$, $I \models \phi : [1, 1]$ (see note[4]) and $I \models F \overset{efr}{\rightsquigarrow} G : [\Delta t, \ell, u]$. Then:*

1. *If $\beta_2 < \beta_1$, then $I \models \rho : \left[0, \min\left(\frac{\ell - \beta_1}{\beta_2 - \beta_1}, 1\right)\right]$*

[4]Note that Theorem 13 requires $\ell \leq \beta_1$ and $\alpha_1 \leq u$

2. *If $\alpha_2 > \alpha_1$, then $I \models \rho : \left[0, \min\left(\frac{u-\alpha_1}{\alpha_2-\alpha_1}, 1\right)\right]$*

From the above theorem, we can easily obtain the following corollary that can be used with just one time formula (*i.e.*, only ρ). Simply consider the case where ϕ is $\mathsf{TRUE} : t_{max}$ and $[\alpha_1, \beta_1] = [0, 1]$.

Corollary 3. *Suppose F, G are formulas, ρ is a time formula, I is an interpretation, and $[\alpha, \beta]$ is an interval such that $EFR(F, G, \Delta t, \rho) \subseteq [\alpha, \beta]$ and $I \models F \overset{efr}{\rightsquigarrow} G : [\Delta t, \ell, u]$. Then:*

1. *If $\beta < 1$ then $I \models \rho : [0, \min(\frac{\ell-1}{\beta-1}, 1)]$*

2. *If $\alpha > 0$ then $I \models \rho : [0, \min(\frac{u}{\alpha}, 1)]$*

Example 3.4.5. *Following from Example 3.4.4, consider the time-formula $\textsf{stock\_decr(10\%)}$: 5. Using Definition 39, we find that $EFR(\phi_{stock} \wedge \textsf{stock\_decr(10\%)} : 5, F_1, G_1, 2) \subseteq [1, 1]$. Previously, we saw that $EFR(\phi_{stock}, F_1, G_1, 2) \subseteq [0.5, 1]$. As the lower bound on frequency increases (by conjuncting the new time formula), that is $1 > 0.5$, we apply part 2 of Theorem 14 (and/or Corollary 3) to obtain an upper probability bound on $\textsf{stock\_decr(10\%)} : 5$. Hence, this formula is no more probable than 0.94.*

Finally, we show that we can also use integrity constraints to aid in syntactic manipulation. For certain ptf's with probability 1, a given IC may cause another ptf (or multiple ptf's) to be entailed with a probability of 0, which can also contribute to bounding EFR.

Proposition 15. *For atom A_i and program $\mathcal{K}$ where $\mathsf{BLK}(A_i) :< \mathsf{blk}_i \in \mathcal{K}$, if there exists a ptf $\phi : [1, 1] \in \mathcal{K}$ such that $\phi \models A_i : t - \mathsf{blk}_i + 1 \wedge A_i : t - \mathsf{blk}_i + 2 \wedge \ldots \wedge A_i : t - 1$,* *then $\mathcal{K}$ entails $A_i : t : [0,0]$.*

Proposition 16. *For atom A_i and program $\mathcal{K}$ where $\mathsf{OCC}(A_i) : [\mathsf{lo}_i, \mathsf{up}_i] \in \mathcal{K}$, if there exists a ptf $\phi : [1,1] \in \mathcal{K}$ such that there are numbers $t_1, \ldots, t_{\mathsf{up}_i} \in \{1, \ldots, t_{max}\}$ where $\phi \models A_i : t_1 \wedge \ldots \wedge A_i : t_{\mathsf{up}_i}$ then for any $t \notin \{t_1, \ldots, t_{\mathsf{up}_i}\}$ $\mathcal{K}$ entails $A_i : t : [0,0]$.*

Example 3.4.6. *Consider $\mathcal{K}_{stock}$ from the previous examples. As this program includes $\mathsf{OCC}(\mathsf{cfo\_resigns}) : [0,1]$ and entails $\mathsf{cfo\_resigns} : 4 : [1,1]$ (by the included prefix), we can conclude that $\mathsf{cfo\_resigns} : 5 : [0,0]$ and $\mathsf{cfo\_resigns} : 6 : [0,0]$ are entailed by this program.*

3.4.3 The Fixpoint-Based Heuristic

We are now ready to use the results of the last section to create the Γ operator. First, we present some preliminary definitions to tighten probability bounds for ptf's and rules. Note that the correctness of these bounds follows directly from the results of the previous section. First we show how, given an APT-program, we can tighten the lower and upper bound of a ptf.

Definition 40. *Suppose $\mathcal{K}$ is an APT-program and ϕ is a time formula. We define:*

$$\mathbf{l\_bnd}(\phi, \mathcal{K}) = \mathbf{sup}\left(\{\,0\,\} \cup \{\,\ell \mid \rho : [\ell, u] \in \mathcal{K} \wedge (\rho \models \phi)\,\}\right).$$

$\mathbf{u\_bnd}(\phi, \mathcal{K})$ *is the **inf** of the set:*

$$
\begin{array}{lll}
\{ & 1 & \} \cup \\
\{ & u & \mid \rho : [\ell, u] \in \mathcal{K} \wedge (\phi \models \rho) \} \cup \\
\{ & \min(\frac{\ell - \beta_1}{\beta_2 - \beta_1}, 1) & \mid (F \overset{efr}{\rightsquigarrow} G : [\Delta t, \ell, u], \rho : [1,1] \in \mathcal{K} \cup \{\textsf{true} : t_{max} : [1,1]\}) \wedge \\
 & & (EFR(F, G, \Delta t, \rho) \subseteq [\alpha_1, \beta_1]) \wedge \\
 & & (EFR(F, G, \Delta t, \rho \wedge \phi) \subseteq [\alpha_2, \beta_2]) \wedge (\beta_2 < \beta_1) \} \cup \\
\{ & \min(\frac{u - \alpha_1}{\alpha_2 - \alpha_1}, 1) & \mid (F \overset{efr}{\rightsquigarrow} G : [\Delta t, \ell, u], \rho : [1,1] \in \mathcal{K} \cup \{\textsf{true} : t_{max} : [1,1]\}) \wedge \\
 & & (EFR(F, G, \Delta t, \rho) \subseteq [\alpha_1, \beta_1]) \wedge \\
 & & (EFR(F, G, \Delta t, \rho \wedge \phi) \subseteq [\alpha_2, \beta_2]_{\wedge} (\alpha_2 > \alpha_1) \}
\end{array}
$$

This bound on a time formula is derived from its relationship with other time formulas (by Lemma 8) or it relationship with rules (by Theorem 14 and/or Corollary 3). Below we show an example.

Example 3.4.7. *Following from Example 3.4.5, consider, once again, the time-formula* stock_decr(10%) $: 5$. *For program* $\mathcal{K}_{stock}$*, we know that* $\mathbf{l\_bnd}(\textsf{stock\_decr(10\%)} : 5, \mathcal{K}_{stock}) = 0.0$. *This is due to the simple fact that there is no lower probability bound assigned to the time formula* stock_decr(10%) $: 5$ *by* $\mathcal{K}_{stock}$ *that is greater than* 0.0. *Examining the upper bound, we consider the* **inf** *of set* $\{1, 0.94\}$ *as* 1 *is the trivial upper bound, there are no other upper probability bounds for* stock_decr(10%) $: 5$ *seen directly in* $\mathcal{K}_{stock}$ *and we have already used Example 3.4.5 to derive the upper bound of* 0.94 *based on syntatic manipulation of rules in* $\mathcal{K}_{stock}$ *(which reflects the last two* *parts of the* **u_bnd** *definition). Hence,* **u_bnd**(*stock_decr(10%)* $: 5, \mathcal{K}_{stock}) = 0.94$.

Note that for ptf's we do not include any manipulation that relies on the bounds of the negated time formula in the above definitions. We handle this type of manipulation in the definition of the operator. The following are versions of **l_bnd**, **u_bnd** for rules.

Definition 41. *Suppose $\mathcal{K}$ is an APT-program, F, G are formulas, and $\Delta t > 0$ is an integer.*

- *The quantity* **l_bnd**$(F, G, \Delta t, \mathcal{K})$ *is the* ***sup*** *of the following set:*

$$\begin{array}{lcl}
\{ & 0 & \} \cup \\
\{ & \ell & \mid F \overset{efr}{\rightsquigarrow} G : [\Delta t, \ell, u] \in \mathcal{K} \} \cup \\
\{ & \alpha \cdot \ell & \mid (\phi : [\ell, u], \rho : [1,1] \in \mathcal{K} \cup \{\textit{true} : t_{max} : [1,1]\}) \wedge \\
 & & (EFR(F, G, \Delta t, \rho \wedge \phi) \subseteq [\alpha, \beta]) \} \cup \\
\{ & \alpha \cdot (1 - u) & \mid (\phi : [\ell, u], \rho : [1,1] \in \mathcal{K} \cup \{\textit{true} : t_{max} : [1,1]\}) \wedge \\
 & & (EFR(F, G, \Delta t, \rho \wedge \neg\phi) \subseteq [\alpha, \beta]) \}
\end{array}$$

- *The quantity* **u_bnd**$(F, G, \Delta t, \mathcal{K})$ *is the* ***inf*** *of the following set:*

$$\begin{array}{lcl}
\{ & 1 & \} \cup \\
\{ & u & \mid F \overset{efr}{\rightsquigarrow} G : [\Delta t, \ell, u] \in \mathcal{K} \} \cup \\
\{ & \beta & \mid (\rho : [1,1] \in \mathcal{K}) \wedge (EFR(F, G, \Delta t, \rho) \subseteq [\alpha, \beta]) \}
\end{array}$$

Hence, the new probability bound assigned to a rule is based on how the bounds on the frequency function are tightened given the ptf's present in the program. Given a ptf, we use a bound on EFR, which allows us to leverage Theorem 13 and Corollary 2 to obtain a tighter bound on the rule. Tighter bounds on rules are useful for two reasons: (1) subsequent applications of the fixpoint operator will in turn use these new bounds to tighten bounds on ptf's and (2) they can be used to identify inconsistent program (as we discuss in Section 3.4.4).

We now define set *formula*($\mathcal{K}$) which intuitively means "all time formulas that appear in $\mathcal{K}$". These are the formulas upon which Definition 40 will act, and also through syntactic manipulation, affect other ptf's in $\mathcal{K}$. As stated earlier, we can find bounds for any time formula ρ by adding $\rho : [0, 1]$ to the initial APT program.

Definition 42. *Given program $\mathcal{K}$ consisting of ptf's and constrained rules, formula($\mathcal{K}$) is the following set:*

$$\begin{aligned}
&\{ \quad \phi \quad | \, \phi : [\ell, u] \in \mathcal{K} \, \} \cup \\
&\{ \quad F : t \quad | \, (t \in [1, t_{max}]) \wedge (F \overset{efr}{\leadsto} G : [\Delta t, \ell, u] \in \mathcal{K}) \, \} \cup \\
&\{ \quad G : t \quad | \, (t \in [1, t_{max}]) \wedge (F \overset{efr}{\leadsto} G : [\Delta t, \ell, u] \in \mathcal{K}) \, \}
\end{aligned}$$

We now have all the pieces we need to define our operator Γ.

Definition 43. *Given program $\mathcal{K}$, $\Gamma(\mathcal{K})$ is defined as the following set:*

$$
\begin{array}{lcl}
\{ & F \overset{efr}{\leadsto} G : [\Delta t, \textsf{\textbf{l\_bnd}}(F,G,\Delta t,\mathcal{K}), & \\
 & \qquad \textsf{\textbf{u\_bnd}}(F,G,\Delta t,\mathcal{K})] & \mid F \overset{efr}{\leadsto} G : [\Delta t, \ell, u] \in \mathcal{K} \,\} \cup \\
\{ & \phi : [\textsf{\textbf{l\_bnd}}(\phi,\mathcal{K}), \textsf{\textbf{u\_bnd}}(\phi,\mathcal{K})] \cap & \\
 & [1 - \textsf{\textbf{u\_bnd}}(\neg\phi,\mathcal{K}), 1 - \textsf{\textbf{l\_bnd}}(\neg\phi,\mathcal{K})] & \mid \phi \in \mathit{formula}(\mathcal{K}) \,\} \cup \\
\{ & A_i : t : [0,0] & \mid (\mathit{BLK}(A_i) :< \mathit{blk}_i \in \mathcal{K}) \wedge (\phi : [1,1] \in \mathcal{K}) \wedge \\
 & & (\phi \models A_i : t - \mathit{blk}_i + 1 \wedge \ldots \wedge A_i : t-1)\} \cup \\
\{ & A_i : t : [0,0] & \mid (\mathit{OCC}(A_i) : [\mathit{lo}_i, \mathsf{up}_i] \in \mathcal{K}) \wedge (\phi : [1,1] \in \mathcal{K}) \wedge \\
 & & (\exists t_1, \ldots, t_{\mathsf{up}_i} \in \{1, \ldots, t_{max}\}) \wedge \\
 & & (\phi \models A_i : t_1 \wedge \ldots \wedge A_i : t_{\mathsf{up}_i}) \wedge \\
 & & (t \notin \{t_1, \ldots, t_{\mathsf{up}_i}\})\} \cup \\
\{ & \mathit{BLK}(A_i) :< \mathit{blk}_i & \mid \mathit{BLK}(A_i) :< \mathit{blk}_i \in \mathcal{K}\} \cup \\
\{ & \mathit{OCC}(A_i) : [\mathit{lo}_i, \mathsf{up}_i] & \mid \mathit{OCC}(A_i) : [\mathit{lo}_i, \mathsf{up}_i] \in \mathcal{K}\}
\end{array}
$$

Intuitively, Γ tightens the probability bounds on rules by leveraging probabilistic time formulas using the results we proved in Theorem 13 and Corollary 2. It tightens the probability bounds on time formulas based other time formulas, rules, and integrity constraints. This uses the results proved in Lemma 8, Theorem 14 (and/or Corollary 3), and Propositions 15-16 respectively.

Example 3.4.8. *Consider the program $\mathcal{K}_{stock}$ from the previous examples. By Definition 42, we know that a ptf time-formula* stock_decr(10%) $: 5$ *will be included in* $\Gamma(\mathcal{K}_{stock})$. *We saw in Example 3.4.7 that* $\textsf{\textbf{l\_bnd}}(\textit{stock\_decr(10\%)} : 5, \mathcal{K}_{stock}) = 0.0$

and $\textbf{u\_bnd}(\textsf{\textit{stock\_decr(10\%)}} : 5, \mathcal{K}_{stock}) = 0.94$*. In the same manner, we can compute that* $\textbf{l\_bnd}(\neg\textsf{\textit{stock\_decr(10\%)}} : 5, \mathcal{K}_{stock}) = 0.0$ *and* $\textbf{u\_bnd}(\neg\textsf{\textit{stock\_decr(10\%)}} : 5, \mathcal{K}_{stock}) = 0.667$ *(this follows from the fact that* $EFR(\phi_{stock} \wedge \neg\textsf{\textit{stock\_decr(10\%)}} : 5, F_1, G_1, 2) \subseteq [0.5, 0.667]$*). Hence, we know that the ptf* $\textsf{\textit{stock\_decr(10\%)}} : 5 : [0.333, 0.94]$ *is included in* $\Gamma(\mathcal{K}_{stock})$*.*

Note that Γ returns an APT-program that is satisfied by the exact same set of interpretations as the original program; this follows directly from the results in the previous section.

Proposition 17. *Suppose I is an interpretation and $\mathcal{K}$ is an APT-program. Then: $I \models \mathcal{K}$ iff $I \models \Gamma(\mathcal{K})$.*

We can also make the following statement about the complexity of the operator.

Proposition 18. *One iteration of Γ can be performed in time complexity $O(|\mathcal{K}|^2 \cdot CHK)$ where CHK is the bound on the time it takes to check (for arbitrary time formulas ϕ, ρ) if $\phi \models \rho$ is true.*

One source of complexity is comparing ptf's with other ptf's. If a ptf is formed with an elementary time formula, then it only needs to be compared to other ptf's that share the same time point – this could reduce complexity. As is usual in logic programming, Γ can be iteratively applied as follows.

Definition 44. *We define multiple applications of Γ as follows.*

- $\Gamma(\mathcal{K}) \uparrow 0 = \mathcal{K}$

- $\Gamma(\mathcal{K}) \uparrow (i+1) = \Gamma(\Gamma(\mathcal{K}) \uparrow i)$

Now, we will show that Γ has a least fixed point. First, we define a partial ordering of APT-programs.

Definition 45 (Preorder over APT-Programs)**.** *Given $\mathcal{K}_1, \mathcal{K}_2$, we say $\mathcal{K}_1 \sqsubseteq^{pre} \mathcal{K}_2$ if and only if:*

- $\forall \phi : [\ell, u] \in \mathcal{K}_1,\ \exists \phi : [\ell', u'] \in \mathcal{K}_2$ *s.t.* $[\ell', u'] \subseteq [\ell, u]$
- $\forall F \overset{efr}{\rightsquigarrow} G : [\Delta t, \ell, u] \in \mathcal{K}_1,\ \exists F \overset{efr}{\rightsquigarrow} G : [\Delta t, \ell', u'] \in \mathcal{K}_2$ *s.t.* $[\ell', u'] \subseteq [\ell, u]$
- *If* $\mathit{BLK}(A_i) :< \mathit{blk}_i \in \mathcal{K}_1$*, then* $\mathit{BLK}(A_i) :< \mathit{blk}_i \in \mathcal{K}_2$
- *If* $\mathit{OCC}(A_i) : [\mathit{lo}_i, \mathsf{up}_i] \in \mathcal{K}_1$*, then* $\mathit{OCC}(A_i) : [\mathit{lo}_i, \mathsf{up}_i] \in \mathcal{K}_2$

The intuition behind the above definition is that program $\mathcal{K}_1$ is below $\mathcal{K}_2$ if it has less rules or ptf's – or rules/ptf's with tighter probability bounds. Note that if $\mathcal{K}_2$ is above $\mathcal{K}_1$, then $\mathcal{K}_1$ has at least as many satisfying interpretations, and possibly more, than $\mathcal{K}_2$. Let $PROG_{B_{\mathcal{L}}, t_{max}}$ be the set of all APT-programs given Herbrand base $B_{\mathcal{L}}$ and time t_{max}. It is easy to see that $\langle PROG_{B_{\mathcal{L}}, t_{max}}, \sqsubseteq^{pre} \rangle$ is a reflexive and transitive, and therefore a preorder. In the following, we will say that $\mathcal{K}_1 \sim \mathcal{K}_2$, read "$\mathcal{K}_1$ is equivalent to $\mathcal{K}_2$" if and only if $\mathcal{K}_1 \sqsubseteq^{pre} \mathcal{K}_2$ and $\mathcal{K}_2 \sqsubseteq^{pre} \mathcal{K}_1$. The "$\sim$" relation is clearly an equivalence relation; we will use $[\mathcal{K}]$ to denote the equivalence class corresponding to $\mathcal{K}$ w.r.t. this relation.

Definition 46 (Partial Ordering of APT-Programs)**.** *Given two equivalence classes $[\mathcal{K}_1], [\mathcal{K}_2]$ w.r.t. relation $\sim$, we say $[\mathcal{K}_1] \sqsubseteq [\mathcal{K}_2]$ if and only if $\mathcal{K}_1 \sqsubseteq^{pre} \mathcal{K}_2$.*

The "$\sqsubseteq$" relation is clearly reflexive, antisymmetric, and transitive, and therefore a partial order over sets of APT-programs. Note that when we use the symbol $\sqsubseteq$, we will often write $\mathcal{K}_1 \sqsubseteq \mathcal{K}_2$ as shorthand for $[\mathcal{K}_1] \sqsubseteq [\mathcal{K}_2]$. We will also overload the symbol $PROG_{B_{\mathcal{L}},t_{max}}$ to mean "all equivalence classes of APT-programs" (for a given t_{max} and $B_{\mathcal{L}}$) where appropriate. Therefore, we can now define a complete lattice, where the top element is a set containing all inconsistent programs, and the bottom element is set containing the empty program.

Lemma 9. *Given $\bot = \{\emptyset\}$ and $\top = \{\mathcal{K} \mid \mathcal{K}$ is inconsistent$\}$, then the partial order $\langle PROG_{B_{\mathcal{L}},t_{max}}, \sqsubseteq\rangle$ defines a complete lattice.*

What remains to be shown is that Γ is monotonic; if this holds, we can state it has a least fixed point.

Lemma 10. $\mathcal{K} \sqsubseteq \Gamma(\mathcal{K})$.

Lemma 11. Γ *is monotonic.*

By the Tarski-Knaster theorem, Γ has a least fixed point.

Theorem 15. Γ *has a least fixed point.*

3.4.4 Using Γ for Consistency Checking

As noted earlier, the Γ operator can be used to find "loose" entailment bounds by simply adding an entailment time formula (ϕ) with probability bounds $[0, 1]$ to the logic program, and then examining the tightened bounds after one or more applications of the operator. In this section, we examine how to use Γ for consistency checking. First, we have a straightforward lemma on consistency.

Lemma 12. *Let $\mathcal{K}$ be an* APT*-logic program that entails rule $F \overset{efr}{\rightsquigarrow} G : [\Delta t, \ell, u]$ or $\phi : [\ell, u]$ such that one of the following is true:*

- $\ell > u$
- $\ell < 0$ *or* $\ell > 1$
- $u < 0$ *or* $u > 1$.

*Under this circumstance, $\mathcal{K}$ is **inconsistent**,* i.e.*, there is no interpretation I such that $I \models \mathcal{K}$.*

The following result follows immediately.

Corollary 4. *Let $\mathcal{K}$ be an* APT*-logic program whetre there exists natural number i such that $\Gamma(\mathcal{K}) \uparrow i$ entails rule $F \overset{efr}{\rightsquigarrow} G : [\Delta t, \ell, u]$ or $\phi : [\ell, u]$ such that one of the following is true:*

- $\ell > u$
- $\ell < 0$ *or* $\ell > 1$
- $u < 0$ *or* $u > 1$.

*Under this circumstance, $\mathcal{K}$ is **inconsistent**.*

We note that the Γ adds time formulas whose probaiblity bounds is determined by an intersection operation. We observe that an empty intersection of the probability bounds is equivalent to the case where $\ell > u$, which allows us to apply the above corollary to correctly state that the program is not consistent. We illustrate this in the below example.

Example 3.4.9. *Consider $\mathcal{K}_{stock}$ from the previous examples. By the definition of Γ, the ptf $\mathsf{stock\_decr(10\%)} \wedge \mathsf{cfo\_resigns} : 5 : [0.499, 1]$ is in $\Gamma(\mathcal{K}_{stock})$. By Example 3.4.6, we know that $\mathsf{cfo\_resigns} : 5 : [0, 0]$ is also in $\Gamma(\mathcal{K}_{stock})$. However, another application of Γ entails $\mathsf{cfo\_resigns} : 5 : [0.499, 0]$ (equivalently, $\mathsf{cfo\_resigns} : 5 : \emptyset$). As $0.499 > 0$, we know that $\mathcal{K}_{stock}$ is not consistent.*

In addition to checking consistency with the Γ operator, we can check for inconsistencies based on the block and occurence ICs via the following result.

Proposition 19. *If there does not exist at least one thread that satisfies all integrity constraints in an APT-logic program, then that program is inconsistent.*

The **Thread Existence Problem (ThEX)** problem is that of checking existence of a thread that satisfies all block and integrity constraints. Here we show that **ThEX** can be solved in constant time – this can allow us to quickly identify certain inconsistencies in an APT-program. First, we define a partial thread.

Definition 47. *A* partial thread *PTh is a thread such that for all $1 \leq i \leq t_{max}$, $PTh(i)$ is a singleton set.*

For any ground atom A_i with a single associated block-size and occurrence constraint[5] if more than $\left\lceil \frac{(\mathsf{blk}_i - 1) \cdot t_{max}}{\mathsf{blk}_i} \right\rceil$ worlds must satisfy A_i in each partial thread, then all partial threads will have a block of size blk_i. This allows us to derive the following results.

[5]There is no loss of generality looking at just one block-size IC per ground atom as multiple such ICs can be coalesced into one by taking the minimum; likewise, there is no loss of generality in considering just one occurrence per ground atom as they can be merged into one by intersecting the $[\mathsf{lo}, \mathsf{up}]$ intervals for that atom.

Proposition 20. *If* $\mathsf{lo}_i > \left\lceil \frac{(\mathsf{blk}_i - 1)\cdot t_{max}}{\mathsf{blk}_i} \right\rceil$ *then there does not exist a partial thread for ground atom* A_i *such that the single block-size and occurrence IC associated with* A_i *hold.*

Proposition 21. *For ground atom* A_i *(with associated ICs), if* $\mathsf{up}_i > \left\lceil \frac{(\mathsf{blk}_i - 1)\cdot t_{max}}{\mathsf{blk}_i} \right\rceil$ *we know that the number of worlds satisfying* A_i *cannot be in the range* $\left[\left\lceil \frac{(\mathsf{blk}_i - 1)\cdot t_{max}}{\mathsf{blk}_i} \right\rceil, up_i\right]$.

The reason for this is simple: it would force the partial thread to have a sequence of blk_i consecutive worlds satisfying A_i. We also notice that these checks can be performed based solely on the values of $\mathsf{lo}_i, \mathsf{up}_i, \mathsf{blk}_i$, and t_{max}. Hence, we have the following proposition.

Proposition 22. ***ThEX*** *can be solved in constant time.*

In the next section we extend these results for non-ground APT-programs.

3.5 Consistency and Entailment Algorithms for Non-Ground Programs

The fixpoint procedure described via the Γ operator works in the ground case. In this section, we study how we may avoid grounding. We start (Section 3.5.1) with a sampling based approach for consistency checking of non-ground programs. Section 3.5.2 defines a non-ground fixpoint operator for entailment queries. This operator avoids grounding the entire program, but guaranteed to provide entailment bounds for a query that are as tight as our ground operator. We remind the reader that both our consistency-checking algorithm and our fixpoint operator presented in this section are sound, but not complete.

3.5.1 Consistency Checking for Non-Ground Programs

In this section, we present a sound algorithm for consistency checking of non-ground programs. We avoid complete grounding of the rules, while still maintaining soundness of the algorithm through random sampling of ground instances of rules. The larger the sample, the more potential inconsistencies can be found.

For a non-ground time formula, ϕ_{ng}, we shall use the notation $gnd(\phi_{ng})$ to refer to the ground formula $\bigwedge\{\phi \mid \textit{is a ground instance of } \phi_{ng}\}$. We are now ready to describe a non-ground analog to the bounds EFR described in the previous section.

Definition 48. *For non-ground formulas F_{ng}, G_{ng}, time Δt, and non-ground time formula ϕ_{ng}, we define*

1.

$$EFR\_SET(F_{ng}, G_{ng}, \Delta t, \phi_{ng}) = \{EFR(F, G, \Delta t, gnd(\phi_{ng}))| \\ F, G \textit{ are ground instances of } F_{ng}, G_{ng}\}$$

2.

$$EFR\_IN(F_{ng}, G_{ng}, \Delta t, \phi_{ng}) = (\alpha_{in}, \beta_{in})$$

Where $\exists[\alpha_{in}, \beta'], [\alpha', \beta_{in}] \in EFR\_SET(F_{ng}, G_{ng}, \Delta t, \phi_{ng})$, *and* $\nexists[\alpha^*, \beta''], [\alpha'', \beta^*] \in EFR\_SET(F_{ng}, G_{ng}, \Delta t, \phi_{ng})$ *s.t.* $\alpha^* > \alpha_{in}$ *and* $\beta^* < \beta_{in}$

3.

$$EFR\_OUT(F_{ng}, G_{ng}, \Delta t, \phi_{ng}) = [\alpha_{out}, \beta_{out}]$$

Where $\exists [\alpha_{out}, \beta'], [\alpha', \beta_{out}] \in EFR\_SET(F_{ng}, G_{ng}, \Delta t, \phi_{ng})$, *and* $\nexists [\alpha^*, \beta''], [\alpha'', \beta^*] \in EFR\_SET(F_{ng}, G_{ng}, \Delta t, \phi_{ng})$ *s.t.* $[\alpha^*, \beta^*] \supset [\alpha_{out}, \beta_{out}]$

The intuition behind Definition 48 is as follows. EFR_SET is the set of all frequency bounds for the different ground instances of F_{ng}, G_{ng}. EFR_IN is a pair consisting of the greatest lower bound of efr (α_{in}) and the least upper bound of efr (β_{in}) of all the elements of EFR_SET. $(\alpha_{in}, \beta_{in})$ is a tuple, not a bound. It is possible for $\alpha_{in} > \beta_{in}$. EFR_OUT represents the tight bound of efr for any ground instance of F_{ng}, G_{ng}. We now prove these bounds to be tight.

Lemma 13. *Suppose* F_{ng}, G_{ng} *are non-ground formulas, time* $\Delta t > 0$ *is an integer, and* ϕ_{ng} *is a non-ground time formula. Let* $(\alpha_{in}, \beta_{in}) = EFR\_IN(F_{ng}, G_{ng}, \Delta t, \phi_{ng})$ *and* $[\alpha_{out}, \beta_{out}] = EFR\_OUT(F_{ng}, G_{ng}, \Delta t, \phi_{ng})$. *If* $Th \models \phi_{ng}$, *then:*

1. *for all ground instances* F, G *of* F_{ng}, G_{ng} *we have* $efr(F, G, \Delta t, Th) \in [\alpha_{out}, \beta_{out}]$
2. *there exist ground instances* F, G *of* F_{ng}, G_{ng}, *and we have* $efr(F, G, \Delta t, Th) \geq \alpha_{in}$
3. *there exist ground instances* F, G *of* F_{ng}, G_{ng}, *and we have* $efr(F, G, \Delta t, Th) \leq \beta_{in}$

Note that if we were to use the techniques of Section 3.4 for entailment, we would most likely need to find tight bounds on the elements in the tuple returned by $EFR\_OUT(F_{ng}, G_{ng}, \Delta t, \phi_{ng})$ (specifically a tight lower bound on EFR – as we can be sure that for all ground instances F, G of F_{ng}, G_{ng} that $\mathit{EFR}(F, G, \Delta t, \mathit{gnd}(\phi_{ng}))$ will fall within these bounds). However, there are a few difficulties with this. First, we conjecture that to find a good bound on EFR_OUT, we would most likely have to examine all combinations of ground instances of F_{ng}, G_{ng} – which is most likely equivalent to grounding out the logic program and using Γ. Second, even if we could efficiently find tight bounds on EFR_OUT, they would most likely be trivial - i.e. $[0, 1]$.

Conversely, consider the tuple $(\alpha_{in}, \beta_{in}) = EFR\_IN(F_{ng}, G_{ng}, \Delta t, \phi_{ng})$. We know that for all ground instances F, G of F_{ng}, G_{ng} such that for

$$[\alpha, \beta] = EFR(F, G, \Delta t, \mathit{gnd}(\phi_{ng}))$$

we have $\alpha_{in} \geq \alpha$ and $\beta_{in} \leq \beta'$. We also know that finding a lower bound on α_{in} and an upper bound on β_{in} can be done by simply considering any subset of combinations of ground instances of F_{ng} and G_{ng}.

Example 3.5.1. *Consider $\mathcal{K}_{train}$ from Figure 2.3 with $t_{max} = 4$. Suppose we add the following ptf (called ϕ) to the program.*

$$\begin{aligned} &\mathsf{at\_station}(\mathsf{train1}, \mathsf{stnA}) : 1 \wedge \quad \mathsf{adjEast}(stnA, stnB) : 1 \wedge \\ \neg(&\mathsf{at\_station}(\mathsf{train1}, \mathsf{stnA}) : 2) \wedge \quad \mathsf{at\_station}(\mathsf{train1}, \mathsf{stnA}) : 3 \quad : [1, 1] \end{aligned}$$

Clearly, as

$EFR(\mathsf{at\_station}(\mathsf{train1}, \mathsf{stnA}) \wedge \mathsf{adjEast}(stnA, stnB), \mathsf{at\_station}(\mathsf{train1}, \mathsf{stnA}), 2, \phi) = [1, 1]$

we know for

$$(\alpha_{in}, \beta_{in}) = EFR\_IN(\mathsf{at\_station}(\mathsf{T}, \mathsf{S}_1) \wedge \mathsf{adjWest}(\mathsf{S}_1, \mathsf{S}_2), \mathsf{at\_station}(\mathsf{T}, \mathsf{S}_2), 2, \phi)$$

that $\alpha_{in} = 1$ *and* $\beta_{in} \leq 1$.

Algorithm 9 Finds bounds on EFR_IN

```
FIND-EFR-IN($\mathbf{F}_{sam}, \mathbf{G}_{sam}$ subsets of ground instances of non-ground
 formulas $F_{ng}, G_{ng}, \Delta t$ natural number , $\phi_{ng}$ non-ground time formula),
returns natural numbers $\alpha_{in}^{-}, \beta_{in}^{+}$

  1. Compute $gnd(\phi_{ng})$
  2. Set $\alpha_{in}^{-} = 0$ and $\beta_{in}^{+} = 1$
  3. For each $F \in \mathbf{F}_{sam}$
     (a) For each $G \in \mathbf{G}_{sam}$
           i. Let $(\alpha, \beta) = EFR(F, G, \Delta t, gnd(\phi_{ng}))$
          ii. $\alpha_{in}^{-} = \max(\alpha, \alpha_{in}^{-})$
         iii. $\beta_{in}^{+} = \min(\beta, \beta_{in}^{+})$
```

Algorithm 9 leverages this technique – if ϕ_{ng} is already ground, algorithm FIND-EFR-IN runs in time quadratic in the size of the sample of ground instances of F_{ng}, G_{ng}. Clearly, this simple algorithm is guaranteed to return a lower bound on α_{in} and an upper bound on β_{in}.

This information can be leveraged in order to perform consistency checks similar to those described in Section 3.4.4 without resorting to fully grounding out F_{ng}, G_{ng} and considering all combinations of those ground instances. The intuition is simple – if there is just one ground instance of a non-ground rule where $\ell > u$, then the program is inconsistent. The theorem and corollary below mirror Theorem 13 and Corollary 2 (Page 118) that we described in Section 3.4.2 for the ground case.

Theorem 16. *Let $\mathcal{K}^{(ng)}$ be a non-ground* APT*-program that contains the following:*

$$\textbf{\textit{Non-ground rule:}} \quad F_{ng} \overset{efr}{\rightsquigarrow} G_{ng} : [\Delta t, \ell, u]$$

$$\textbf{\textit{Non-ground ptf:}} \qquad \phi_{ng} : [1, 1]$$

and $(\alpha_{in}, \beta_{in}) = EFR\_IN(F_{ng}, G_{ng}, \Delta t, \phi_{ng})$*. If we are given* $\alpha_{in}^{-} \leq \alpha_{in}$ *and* $\beta_{in}^{+} \geq \beta_{in}$*, then,* $\mathcal{K}^{(ng)}$ *is not consistent if either* $\alpha_{in}^{-} > u$ *or* $\beta_{in}^{+} < \ell$*.*

Corollary 5. *Let $\mathcal{K}^{(ng)}$ be a non-ground* APT*-program that contains the following:*

$$\textbf{\textit{Non-ground rule:}} \quad F_{ng} \overset{efr}{\rightsquigarrow} G_{ng} : [\Delta t, \ell, u]$$

$$\textbf{\textit{Non-ground ptf:}} \qquad \phi_{ng} : [\ell', u']$$

and $(\alpha_{in}, \beta_{in}) = EFR\_IN(F_{ng}, G_{ng}, \Delta t, \phi_{ng})$*. If we are given* $\alpha_{in}^{-} \leq \alpha_{in}$ *and* $\beta_{in}^{+} \geq \beta_{in}$*, then,* $\mathcal{K}^{(ng)}$ *is not consistent if* $\alpha_{in}^{-} \cdot \ell' > u$*.*

Algorithm 10 is a sound (but not complete) method to quickly check for inconsistency in the non-ground case.

Proposition 23. *If the list returned by* NG-INCONSIST-CHK *contains any elements, then $\mathcal{K}^{(ng)}$ is not consistent.*

Note that the algorithm performs only a quadratic number of comparisons.

Proposition 24. NG-INCONSIST-CHK *performs $O(|\mathcal{K}^{(ng)}|^2)$ comparisons.*[6]

[6]**Note:** each comparison requires generating samples of ground instances of two formulas in a rule and running FIND-EFR-IN.

Algorithm 10 Checks for inconsistencies in a non-ground program

```
NG-INCONSIST-CHK($\mathcal{K}^{(ng)}$ *non-ground program*)

returns list of rules that cause inconsistencies

1. Let $L$ be a list of rules initialized to $\emptyset$
2. For each ptf $\phi_{ng} : [\ell', u'] \in \mathcal{K}^{(ng)}$ where $u' = 1$, do the following.
   (a) For each rule $F_{ng} \overset{efr}{\rightsquigarrow} G_{ng} : [\Delta t, \ell, u] \in \mathcal{K}^{(ng)}$, do the following.
       i. Generate sample sets $\mathbf{F}_{sam}, \mathbf{G}_{sam}$ of ground instances of $F_{ng}, G_{ng}$.
       ii. Let $(\alpha^{-}_{in}, \beta^{+}_{in}) = \text{FIND-EFR-IN}(\mathbf{F}_{sam}, \mathbf{G}_{sam}, \Delta t, \phi_{ng})$
       iii. If $\alpha^{-}_{in} \cdot \ell' > u$, then add $F_{ng} \overset{efr}{\rightsquigarrow} G_{ng} : [\Delta t, \ell, u] \in \mathcal{K}^{(ng)}$ to $L$
       iv. Elseif $\ell' = 1$ and $\beta^{+}_{in} < \ell$, then add $F_{ng} \overset{efr}{\rightsquigarrow} G_{ng} : [\Delta t, \ell, u] \in \mathcal{K}^{(ng)}$ to $L$
3. Return list $L$
```

3.5.2 Entailment for the Non-Ground Case

In this section, we introduce a non-ground operator, $\Lambda_{\mathcal{K}^{(ng)}}$, that maps ground programs to ground programs. Using the same lattice of APT-programs we used in Section 3.4.3, we show that $\Lambda_{\mathcal{K}^{(ng)}}$ also has a least fixed point. Our intuition is as follows. Suppose we want to find the tightest entailment bounds on some ptf ϕ; if we compute $lfp(\Lambda_{\mathcal{K}^{(ng)}}(\phi : [0, 1]))$, the result will be an APT-program (let us call this program $\mathcal{K}_\phi$) s.t. $lfp(\Gamma(\mathcal{K}_\phi))$ will provide the same entailment bounds on ϕ as if we had computed the least fixed point of Γ on the grounding of $\mathcal{K}^{(ng)}$. However, in most cases, $\mathcal{K}_\phi$ will be much smaller than the grounding of $\mathcal{K}^{(ng)}$.

Definition 49. *For non-ground program $\mathcal{K}^{(ng)}$ and ground program $\mathcal{K}$ (note that $formula(\mathcal{K})$ is a set of ground formulas, as defined in Definition 42), $\Lambda_{\mathcal{K}^{(ng)}}$ maps ground programs to ground programs and is defined as follows. $\Lambda_{\mathcal{K}^{(ng)}}(\mathcal{K}) =$*

$$
\begin{array}{ll}
\mathcal{K} & \cup \\
\{F \overset{efr}{\rightsquigarrow} G : [\Delta t, \ell, u] | & F \overset{efr}{\rightsquigarrow} G : [\Delta t, \ell, u] \textit{ is a ground instance of a rule in } \mathcal{K}^{(ng)} \textit{ s.t.} \\
 & \exists \phi \in formula(\mathcal{K}) \textit{ where } \phi \textit{ is ground and} \\
 & \exists t \in [1, t_{max}] \textit{ s.t. } \phi \models F : t \textit{ or } \phi \models G : t \\
 & \textit{ or } \phi \models \neg F : t \textit{ or } \phi \models \neg G : t\} \cup \\
\{\rho : [\ell, u] | & \rho : [\ell, u] \textit{ is a ground instance of a ptf in } \mathcal{K}^{(ng)} \textit{ s.t.} \\
 & \exists \phi \in formula(\mathcal{K}) \textit{ where } \phi \textit{ is ground and } \phi \models \rho \\
 & \textit{ or } \phi \models \neg \rho\} \cup \\
\{\mathsf{BLK}(A) :< \mathsf{blk} | & \mathsf{BLK}(A) :< \mathsf{blk} \textit{ is a ground instance of a constraint in } \mathcal{K}^{(ng)} \textit{s.t.} \\
 & \exists \phi \in formula(\mathcal{K}) \textit{ where } \phi \textit{ is ground and} \\
 & \exists t \in [1, t_{max}] \textit{ s.t. } \phi \models A : t \textit{ or } \phi \models \neg A : t\} \cup \\
\{\mathsf{OCC}(A) : [\mathsf{lo}, \mathsf{up}] | & \mathsf{OCC}(A) : [\mathsf{lo}, \mathsf{up}] \textit{ is a ground instance of a constraint in } \mathcal{K}^{(ng)} \textit{s.t.} \\
 & \exists \phi \in formula(\mathcal{K}) \textit{ where } \phi \textit{ is ground and } \} \\
 & \exists t \in [1, t_{max}] \textit{ s.t. } \phi \models A : t \textit{ or } \phi \models \neg A : t\}
\end{array}
$$

We will now present an example for this operator.

Example 3.5.2. *Recall $\mathcal{K}_{train}$ from Figure 2.3 with $t_{max} = 4$. The following rules* *comprise the set* $\Lambda_{\mathcal{K}_{train}}(\{\mathsf{at\_station}(\mathsf{train1}, \mathsf{stnB}) : 4\})$*:*

$$\mathsf{at\_station}(\mathsf{train1}, \mathsf{stnB}) : 4$$

$$\mathsf{at\_station}(\mathsf{train1}, \mathsf{stnA}) \wedge \mathsf{adjEast}(\mathsf{stnA}, \mathsf{stnB}) \overset{efr}{\rightsquigarrow} \mathsf{at\_station}(\mathsf{train1}, \mathsf{stnB}) : [4, 0.85, 1.0]$$

$$\mathsf{at\_station}(\mathsf{train1}, \mathsf{stnB}) \wedge \mathsf{adjEast}(\mathsf{stnB}, \mathsf{stnB}) \overset{efr}{\rightsquigarrow} \mathsf{at\_station}(\mathsf{train1}, \mathsf{stnB}) : [4, 0.85, 1.0]$$

$$\mathsf{at\_station}(\mathsf{train1}, \mathsf{stnC}) \wedge \mathsf{adjEast}(\mathsf{stnC}, \mathsf{stnB}) \overset{efr}{\rightsquigarrow} \mathsf{at\_station}(\mathsf{train1}, \mathsf{stnB}) : [2, 0.85, 1.0]$$

$$\mathsf{at\_station}(\mathsf{train1}, \mathsf{stnA}) \wedge \mathsf{adjWest}(\mathsf{stnA}, \mathsf{stnB}) \overset{efr}{\rightsquigarrow} \mathsf{at\_station}(\mathsf{train1}, \mathsf{stnB}) : [2, 0.6, 0.7]$$

$$\mathsf{at\_station}(\mathsf{train1}, \mathsf{stnB}) \wedge \mathsf{adjWest}(\mathsf{stnB}, \mathsf{stnB}) \overset{efr}{\rightsquigarrow} \mathsf{at\_station}(\mathsf{train1}, \mathsf{stnB}) : [2, 0.6, 0.7]$$

$$\mathsf{at\_station}(\mathsf{train1}, \mathsf{stnC}) \wedge \mathsf{adjWest}(\mathsf{stnC}, \mathsf{stnB}) \overset{efr}{\rightsquigarrow} \mathsf{at\_station}(\mathsf{train1}, \mathsf{stnB}) : [2, 0.6, 0.7]$$

We use the same partial ordering and lattice from Section 3.4.3, and show the monotonicity of $\Lambda_{\mathcal{K}^{(ng)}}$ as follows.

Lemma 14. $\mathcal{K} \sqsubseteq \Lambda_{\mathcal{K}^{(ng)}}(\mathcal{K})$ *wrt* $\langle PROG_{B_{\mathcal{L}}, t_{max}}, \sqsubseteq \rangle$

Lemma 15. $\Lambda_{\mathcal{K}^{(ng)}}$ *is monotonic.*

Now, we show that $\Lambda_{\mathcal{K}^{(ng)}}$ has a least fixed point.

Definition 50. *We define multiple applications of* Λ *as follows.*

- $\Lambda_{\mathcal{K}^{(ng)}}(\mathcal{K}) \uparrow 0 = \mathcal{K}$
- $\Lambda_{\mathcal{K}^{(ng)}}(\mathcal{K}) \uparrow (i+1) = \Lambda_{\mathcal{K}^{(ng)}}(\Lambda_{\mathcal{K}^{(ng)}}(\mathcal{K}) \uparrow i)$

Theorem 17. $\Lambda_{\mathcal{K}^{(ng)}}$ *has a least fixed point.*

The next two results demonstrate the soundness of Λ. Given non-ground program $\mathcal{K}^{(ng)}$, let $ground(\mathcal{K}^{(ng)})$ be the grounding of this program. The lemma below follows directly from the definition of the operator. It states that the least fixed point of the operator is a subset of the grounding of $\mathcal{K}^{(ng)}$.

Lemma 16. *Given non-ground program $\mathcal{K}^{(ng)}$, and ground program $\mathcal{K}$, $lfp(\Lambda_{\mathcal{K}^{(ng)}}(\mathcal{K})) \subseteq ground(\mathcal{K}^{(ng)}) \cup \mathcal{K}$.*

Additionally, the following result states that, for a given entailment query, we obtain the same result whether we use $\Lambda_{\mathcal{K}^{(ng)}}$ or simply ground out $\mathcal{K}^{(ng)}$.

Theorem 18. *Given non-ground program $\mathcal{K}^{(ng)}$*

$$\phi : [\ell, u] \in lfp(\Gamma(lfp(\Lambda_{\mathcal{K}^{(ng)}}(\{\phi : [0, 1]\}))))$$

iff

$$\phi : [\ell, u] \in lfp(\Gamma(ground(\mathcal{K}^{(ng)}) \cup \{\phi : [0, 1]\}))$$

3.6 Experimental Results

This section reports on experiments carried out in the ground case with our fixpoint algorithm. We demonstrate the Γ operator on 23 different ground APT-programs automatically extracted from two different data sets using a slight improvement of the APT-EXTRACT algorithm from the previous chapter. We were able to compute fixpoints of APT-programs consisting of over 1,000 ground rules in about 20 minutes (see the left-hand side of Figure 3.5). Note that this is the time to compute the fixpoint, not to perform a deduction (i.e., via the Λ operator), which can be done for specific entailment queries, and would be faster.

This section is organized as follows. Section 3.6.1 describes our experimental setup, data set, and how we extracted rules, integrity constraints, and ptf's while Section 3.6.2 examines the runtime of the fixpoint operator.

3.6.1 Experimental Setup

All experiments were run on multiple multi-core Intel Xeon E5345 processors at 2.33GHz, 8GB of memory, running the Scientific Linux distribution of the GNU/Linux OS, kernel version 2.6.9-55.0.2.ELsmp.[7] Our implementation consists of approximately 4,000 lines of Java code (JDK 1.6.0).

Iraq Special Groups (ISW) This data-set contains daily counterinsurgency events from Baghdad in 2007-2008. The event data was provided by the Institute for the Study of War (ISW) and augmented with neighborhood data from the International Medical Corps. The historical data was represented with 187 ground atoms over 567 days – which is the time granularity we used. Using the APT-Extract algorithm (presented in the previous chapter), we extracted 3,563 ground rules using the efr frequency function.

We considered 13 logic programs from this dataset; each smaller program is a subset of any of the larger ones, so we have $\mathcal{K}_1 \subseteq \mathcal{K}_2 \subseteq \ldots \subseteq \mathcal{K}_{12} \subseteq \mathcal{K}_{13}$. In each program, we included a prefix consisting of 50 worlds (for more on prefixes, refer to

[7]We note that this implementation makes use of only one processor and one core for a single run, though different runs were distributed across the cluster.

Definition 34 on Page 107). The same prefix was used for each ISW program. We set $t_{max} = 60$ for all ISW programs. Additionally, for all ground atoms appearing in a given program, we added the appropriate block and occurrence integrity constraints. Later we will present our extraction algorithms for these constraints.

Minorities at Risk Organizational Behavior (MAROB) This data set contains yearly attributes for a variety of political and violent groups over a period of 25 years [181]. Overall, we have extracted over 21.4 million APT-rules from this data set. These rules were also extracted using APT-EXTRACT with the efr frequency function.

We considered 10 APT-logic programs from this dataset, each corresponding to a different group. As each of these logic programs is associated with actions for a specific group, all 10 of the MAROB programs are pairwise disjoint. In each MAROB program, we included a unique prefix of 10 worlds specific to the group in the program. We set $t_{max} = 13$ for each MAROB program. Block-size and occurrence constraints were also included in each program. Tables 3.1-3.2 provides some information on these APT-programs.

While integrity constraints (as with rules) could come from an expert, we decided to extract our ICs from the data. We have included the straightforward algorithms OC-EXTRACT and BLOCK-EXTRACT to show how we extracted occurrence and block-size IC's (respectively) for each of the 187 atoms in the data set.

Proposition 25. *OC-EXTRACT runs in time $O((n - t_{max}) \cdot t_{max})$.*

Proposition 26. *There are no historical threads such that atom a_i is satisfied by*

Algorithm 11 Extracts occurrence constraints

```
OC-EXTRACT($a_i$ ground atom, $W_1, \ldots, W_n$ historical worlds, $t_{max}$ maximum time),
returns natural numbers $\mathsf{lo}_i, \mathsf{up}_i$

1. Set $\mathsf{up}_i = 0$ and $\mathsf{lo}_i = t_{max}$
2. For $i = 1$, $i \leq n - t_{max} + 1$, loop
    (a) Set $cur = 0$
    (b) For $j = i$, $j < i + t_{max}$ loop
        i. If $W_j \models a_i$, then $cur = cur + 1$
    (c) If $cur < \mathsf{lo}_i$ then set $\mathsf{lo}_i = cur$
    (d) If $cur > \mathsf{up}_i$ then set $\mathsf{up}_i = cur$
3. Return $\mathsf{lo}_i, \mathsf{up}_i$
```

less than lo_i or more than up_i worlds when $\mathsf{lo}_i, \mathsf{up}_i$ are produced by OC-EXTRACT.

Proposition 27. *BLOCK-EXTRACT runs in time $O(n)$.*

Proposition 28. *Given blk_i as returned by BLOCK-EXTRACT, there is no sequence of blk_i or more consecutive historical worlds that satisfy atom a_i.*

3.6.2 Run Time Evaluation

To evaluate performance, for each logic program, we clocked 10 trials until Γ reached a fixpoint. In all our trials, a fixpoint was reached after only two or three applications (see Tables 3.1-3.2). We also note that the experimental relationship

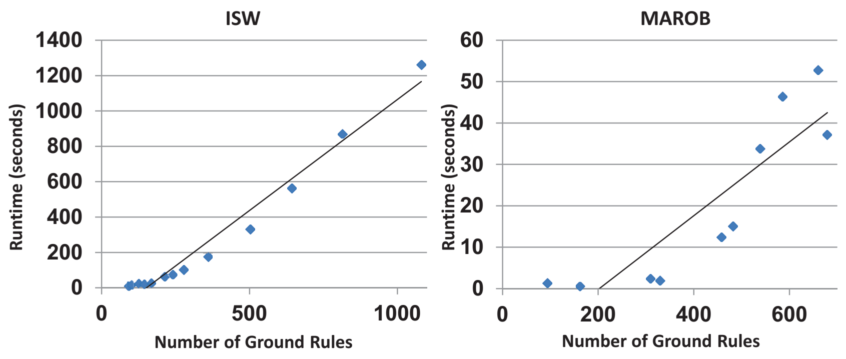


Figure 3.5: Number of ground rules vs. run time (Left: ISW, Right: MAROB). Note these run-times include the full computation of the fixed point of the Γ operator.

between run time and the number of rules was linear – we conducted a statistical R^2-test for this and came up with an R^2 value of 0.97 for ISW programs and 0.77 for MAROB programs (refer to Figure 3.5). We must point out that the disjoint relationship among MAROB programs may account for why the run time relationship is not as linear as that for the ISW programs. This graceful degradation in performance is most likely due to the fact that the number of rules/ptfs that can tighten the bound of a given rule or ptf is much smaller than the set of entire rules, which makes the running time of the inner loop very small. Hence, for practical purposes, the $O(|\mathcal{K}|^2)$ is a loose bound; this worst case is likely to occur only in very rare circumstances.

We checked entailment by looking at the probability bounds of formulas in $formula(\mathcal{K})$ (see Definition 42), which is obtained by finding the fixpoint for the Γ operator on a consistent APT-program. After our initial runs of Γ on the 23 logic programs, we found that 21 of them were inconsistent. As inconsistencies are

Program	Gr. Rules	Post. Gr. Atoms	Range of Δt	t_{max}	Time Points	Γ App.
$\mathcal{K}_1$	92	76	[2,10]	60	567	2
$\mathcal{K}_2$	102	76	[2,10]	60	567	3
$\mathcal{K}_3$	126	76	[2,10]	60	567	3
$\mathcal{K}_4$	144	76	[2,10]	60	567	2
$\mathcal{K}_5$	169	76	[2,10]	60	567	2
$\mathcal{K}_6$	214	76	[2,10]	60	567	3
$\mathcal{K}_7$	241	76	[2,10]	60	567	3
$\mathcal{K}_8$	278	76	[2,10]	60	567	3
$\mathcal{K}_9$	360	79	[2,10]	60	567	3
$\mathcal{K}_{10}$	503	80	[2,10]	60	567	3
$\mathcal{K}_{11}$	644	80	[2,10]	60	567	3
$\mathcal{K}_{12}$	816	80	[2,10]	60	567	3
$\mathcal{K}_{13}$	1081	84	[2,10]	60	567	3

Table 3.1: APT-logic programs used in the run time evaluations. Programs $\mathcal{K}_1 - \mathcal{K}_{13}$ are based on the ISW data-set.

Program	Gr. Rules	Post. Gr. Atoms	Range of Δt	t_{max}	Time Points	Γ App.
$\mathcal{K}_H$	586	189	[2,3]	13	23	3
$\mathcal{K}_J$	679	192	[3,3]	13	25	2
$\mathcal{K}_A$	661	162	[2,3]	13	25	2
$\mathcal{K}_B$	163	175	[3,3]	13	24	2
$\mathcal{K}_D$	539	176	[3,3]	13	25	2
$\mathcal{K}_{FT}$	482	188	[2,3]	13	22	3
$\mathcal{K}_{FR}$	310	177	[3,3]	13	25	2
$\mathcal{K}_{HA}$	458	168	[3,3]	13	13	2
$\mathcal{K}_{HI}$	330	182	[2,3]	13	25	2
$\mathcal{K}_K$	94	181	[1,3]	13	25	3

Table 3.2: APT-logic programs used in the run time evaluations. The programs in this table are based on the MAROB data-set.

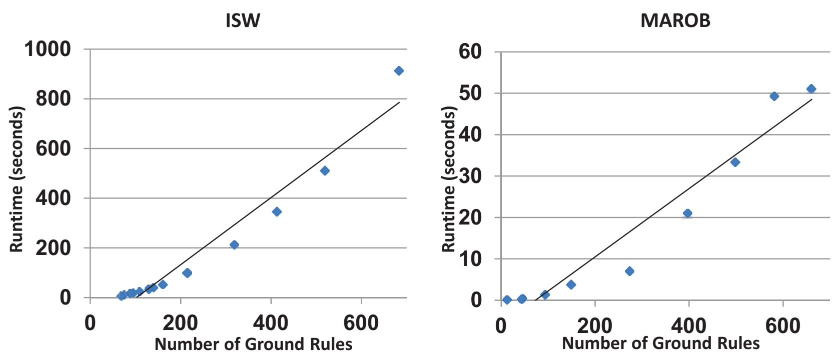


Figure 3.6: Number of ground rules vs. run time for entailment checking (Left: ISW, Right: MAROB).

found in a constructive way (refer to Section 3.4.4 on Page 127), we could eliminate rules that caused inconsistencies (we designate the "consistent" subset of a program with a tick mark, i.e., $\mathcal{K}_2'$ is $\mathcal{K}_2$ with inconsistency-causing rules removed). Using these "consistent" APT-programs, we first looked to revalue the performance of the Γ operator for entailment. Unsurprisingly, as with the run time evaluation we performed for consistency checking, we found that the run time was related linearly to the number of ground rules considered. We obtained R^2 values of 0.95 for ISW programs and 0.94 for MAROB programs. See Figure 3.6 for details; run times are based on the average of 10 trials for each logic program.

As a consequence of Definition 42 (Page 123), the logic program returned by multiple applications of Γ includes several ptf's not in the original program. These ptf's were either based on formulas seen in the rules, or atoms seen in the rules where an integrity constraint forces the associated atomic ptf to be assigned probability 0. Many of these ptf's have probability bounds tighter than $[0, 1]$ – some extremely

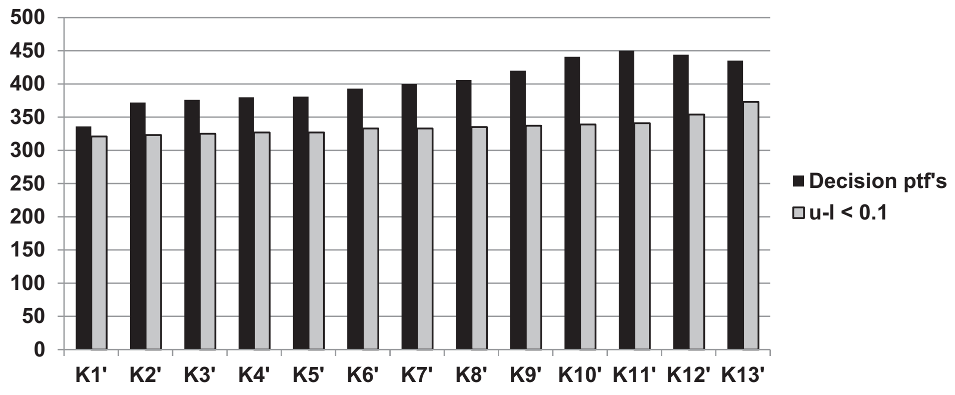


Figure 3.7: Attributes of ptf's entailed by the different logic programs (ISW dataset)

tight. We note, as shown in Figure 3.7, that all of our ISW logic program produce over 300 ptfs where the difference between ℓ and u is less than 0.1 (the number steadily increases with larger ISW programs[8]). We also looked at "decision ptf's"; these are ptf's where either $\ell \leq 0.5$ or $u \leq 0.5$ – the intuition is that the probability mass is either above or below 0.5, allowing a user to make a decision. The Γ operator also was successful in producing many ptf's of this type, producing well over 400 in over half of the logic programs we considered from the ISW dataset.

[8]It is important to point out that all numbers of ptf's with tight bounds are associated with a world outside the range of the prefix.

Algorithm 12 Extracts block-size constraints

```
BLOCK-EXTRACT($a_i$ ground atom, $W_1, \ldots, W_n$ historical worlds),

returns natural number $\mathsf{blk}_i$

1. Set $cur = 0$
2. Set $best = 0$
3. For $i = 1$, $i \leq n$, loop
   (a) If $W_i \models a_i$
       i. $cur = cur + 1$
   (b) Else
       i. If $cur > best$ then set $best = cur$
       ii. Set $cur = 0$
4. If $cur > best$ set $best = cur$
5. Set $\mathsf{blk}_i = best + 1$
6. Return $\mathsf{blk}_i$
```

3.7 Chapter 3 Related Work

In the previous chapter, we showed that APT-Logic distinguishes itself from other temporal logics in the following ways: (i) It supports reasoning about probability of events over time, (ii) Future worlds can depend on more than just the current world (i.e., it does not assume the Markov property). (iii) It provides probability bounds instead of a point probability. (iv) No independence assumptions are made.

[34] was the first effort to provide a declarative semantics for temporal probabilistic LPs. We compared this work with APT-Logic in the previous chapter. No implementation was proposed and thus no experimental results were studied.

[124] introduce an extension to the Situation Calculus for handling actions with uncertain effects. The semantics of their logical language is given in terms of a "Randomly Reactive Automaton", which allows for probabilistic effects of actions but has no notion of the current time apart from that implied by the sequence of actions. They examine next move determination where the results of a move are dependent on the move chosen as well as on draws from single or from multiple distributions.

Santos and Young [148] propose the Probabilistic Temporal Network model (PTNs), which allows to represent temporal (and atemporal) information in combination with probabilistic semantics. PTNs are suitable for representing causality constrained by time, conditions for the occurrence of events (and at what time they occur), and periodic and recurrent processes. This model is based on Bayesian networks (for the probabilistic aspect) and on work by Allen [3] on temporal interval algebra for the temporal aspect. Even though this work's goals overlap to some extent with those of our own, the fundamental difference lies in the initial assumptions made. In order to build a PTN, one must have available information regarding dependencies, prior probabilities for all random variables, temporal causal relationships between random variables in temporal aggregates, etc. The focus of our work is to reason about events making no independence assumptions, and only based on limited information relating small subsets of events. The PTN framework is, however, very useful for scenarios in which the required information is available, as is the case in probabilistic reasoning with traditional Bayesian Networks. The key aspect that separates APT-logic from PTN's, is the fact that **APT-logic makes no assumptions about independence.** For example, consider item 1 of Theorem 8, one of the key building blocks of our fixpoint heuristic. In this case, if $I \models \phi : [p, p]$ and $\rho : [p', p']$, then $I \models \phi \wedge \rho : [\max(0, p + p' - 1), \min(p, p')]$. If we had assumed independence, then $I \models \phi \vee \rho : [p^2, p^2]$ – clearly a different answer and not appropriate for domains where we do not wish to make assumptions about dependence/independence (i.e., the counter-insurgency data that we used for our experiments). This also is our motivation for the use of probability intervals – rather than point probabilities.

3.7.1 Work in Verification and PRISM

Logics merging time and probabilities have been studied quite a bit in the area of verification. [173] was one of the pioneers in this, followed by many including probabilistic CTL [65], and others [25]. Building on this work, Kwiatkowska et. al. developed a tool known as PRISM [91, 92] to perform this type of model checking. PRISM has the following characteristics:

1. The user specifies a **model** - a discrete-time Markov chain (DTMC), continuous-time Markov chain (CTMC) or Markov decision processes (MDP)
2. The user also specifies a **property** - which is normally a CTL formula
3. PRISM returns a **value** (normally a probability or expected value) associated with the property

One can view our implementation in the same light - taking an APT-program as a model, time formula as a property, and returning entailment bounds as the value. However, PRISM operates under some very different assumptions than APT-logic which are appropriate for some applications but not for all.

1. The **model** specified by the user in PRISM is a stochastic process that assumes the Markov property - that is the probability of being in the next state only depends on the current state and action. Conversely, an APT-program **does not assume the Markov property**. Further, we demonstrated translations from stochastic processes to APT-programs in Chapter 2. Also, in that chapter, we showed how it is easy to construct a very simple APT-program where there is no analogous MDP (using a natural construction).
2. Based on the **model** specified by the user, PRISM also makes an independence assumption. Suppose we are in initial state S_1 and consider the following

sequence of states, actions, and probabilities in an MDP: $S_1 \xrightarrow{a}_{p_1} S_2 \xrightarrow{b}_{p_2} S_3$ which states that "state 1 transitions to state 2 on action a with probability p_1 and state 2 transitions to state on action b with probability p_2." PRISM would calculate the probability of such a sequence - $p_1 \cdot p_2$ - hence it has assumed independence between the two transitions. Likewise, consider the formulas $F(S_1), F(S_2), F(S_3)$ – formulas satisfied exactly by states S_1, S_2, S_3. Using the natural translation described in Chapter 2, we can create an analogous APT-program as follows:

- $(F(S_1) \wedge a \wedge \neg b) : 1 \wedge F(S_2) : 2 : [p_1, p_1]$
- $(F(S_2) \wedge b \wedge \neg a) : 2 \wedge F(S_3) : 3 : [p_2, p_2]$

By item 1 of Theorem 8, the following ptf is tightly entailed:

$(F(S_1) \wedge a \wedge \neg b) : 1 \wedge (F(S_2) \wedge b \wedge \neg a) : 2 \wedge F(S_3) : 3 : [\max(0, p_1 + p_2 - 1), \min(p_1, p_2)]$

With APT-logic, we allow for uncertainty - **all** we can say about the sequence is it has a probability in $[\max(0, p_1 + p_2 - 1), \min(p_1, p_2)]$ – which is clearly different than $p_1 \cdot p_2$.

3. The **property** specified by the user in PRISM is based on PCTL [12, 65]. Although there are constructs in PCTL that appear similar to the syntax of APT-logic, as our semantics differ substantially, the statements have different meanings. Even if an MDP is encoded in an APT-program, a "leads-to" PCTL operator (which has a strikingly similar intuition to an APT-rule) has a very

different meaning. We explored the specifics of these differences in the previous chapter.

Basically, PRISM is best suited for situations where the underlying model can be represented as a stochastic process. Popular applications have included software verification and certain biology problems that can be easily represented as stochastic processes. APT-logic is best suited for situations where there are no independence or Markov assumptions made about the model - which is often the case when we are working with extracted rules. We have shown APT-logic to be viable for studying the actions of militia groups in a counter-insurgency environment. Other applications where APT-logic is well suited include policy analysis and stock price movement.

3.8 Chapter Summary

Logical reasoning with time and probabilities is essential in any application where the occurrence of certain conditions at time t may cause or imply that other phenomena may occur δ units in the future. There are numerous such applications including ones relating to how stock markets will move in the future based on current or past conditions, medicine where the condition of a patient in the future depends on various things true now, behavior modeling where the behavior of an individual or group in the future may depend on his current/past situation. In addition, most applications where we reason about the future are fraught with uncertainty. Annotated Probabilistic Temporal Logic (APT-logic for short) was introduced in the previous chapter as a paradigm for reasoning about sentences of the form "If formula

F is true at time t, then formula G will be true at time Δt with a probability in the range $[L, U]$." More importantly, APT-logic programs were introduced in a manner that did not require independence or Markovian assumptions, many of which are inapplicable for several applications.

To date, no implementation of probabilistic temporal logic exists that does not make use of Markovian or independence assumptions. To our knowledge, this chapter represnt the first attempt at any implementation of such logics. However, due to the high complexity of such reasoning (which may also explain why implementations may not exist), practical temporal probabilistic reasoning systems may not always be complete.

In this chapter, we developed, implemented, and evaluated a fixpoint-based heuristic for consistency and entailment problems in APT-logic programs. This chapter makes the following contributions:

1. We show NP-completeness of the APT-logic consistency problem, and coNP-completeness of the APT-logic entailment problem, extending hardness results of the previous chapter.

2. We developed a **fixpoint based** heuristic from the following observations:

   - The presence of ptf's with the probability of 1 in an APT-program allows us to tightly bound values for frequency functions.

   - The bound on frequency functions, in turn, allows us to tighten the bounds of elements in an APT-program

   - The above two characteristics can be employed in an operator that maps APT-programs to APT-programs and has a least fixed point

3. We developed consistency and entailment algorithms for the non-ground case.

4. We implemented our fixpoint heuristic and applied it to 23 *real world* APT-logic programs derived *automatically* from two different real world data sets. This suite of test programs was not written by us. Our experiments show that our fixpoint based heuristical can calculate fixpoints in time roughly linear w.r.t. the number of ground rules

5. We also show that using our implementation, we can solve the "tight entailment problem" where the goal is to find the tightest interval $[\ell, u]$ such that $F : [t, \ell, u]$ is entailed by an APT-logic program for a given time t and formula F.

Chapter 4

Geospatial Abduction

In the previous two chapter, we explored temporal aspects of an agent's behavior with APT logic. The next three chapters deal with spatial aspects of an agent's behavior. These chapters are primarily concerned with variants of *geospatial abduction* problems - inferring unobserved geospatial locations associated with agent behavior. In this chapter, we formalize the idea of geopspatial abduction and study some natural problems associated with this framework.[1]

4.1 Chapter Introduction

There are numerous applications where we wish to draw geospatial inferences from observations. For example, criminologists [144, 15] have found that there are spatial relationships between a serial killer's house (the geospatial inference we wish to make), and locations where the crimes were committed (the observations).

[1]This chapter is based on [157] and [158] which were completed in cooperation with Maria Luisa Sapino and V.S. Subrahmanian.

A marine archaeologist who finds parts of a wrecked ship or its cargo at various locations (the observations) is interested in determining where the main portion of the wreck lies (the geospatial inference). Wildlife experts might find droppings of an endangered species such as the Malayan sun bear (observations) and might want to determine where the bear's den is (the geospatial inference to be made). *In all these cases, we are trying to find a single location that best explains the observations (or the k locations that best explain the observations).* There are two common elements in such applications.

First, there is a set $\mathcal{O}$ of *observations* of the phenomena under study. For the sake of simplicity, we assume that these observations are points where the phenomenon being studied was known to have been present. Second, there is some *domain knowledge* $\mathcal{D}$ specifying known relationships between the geospatial location we are trying to find and the observations. For instance, in the serial killer application, the domain knowledge might tell us that serial killers usually select locations for their crimes that are at least 1.2 km from their homes and at most 3 km from their homes. In the case of the sun bear, the domain knowledge might state that the sun bear usually prefers to have a den in a cave, while in the case of the wreck, it might be usually within a radius of 10 miles of the artifacts that have been found.

The *geospatial abduction problem* (GAP for short) is the problem of finding the most likely *set of locations* that is compatible with the domain knowledge $\mathcal{D}$ and that best "explains" the observations in $\mathcal{O}$. To see why we look for a *set* of locations, we note that the serial killer might be using both his home and his office as launching pads for his attacks. In this case, no single location may best account for the observations. In this chapter, we show that many natural problems associated with geospatial abduction are NP-Complete, which cause us to resort to approximation techniques. We then show that certain geospatial abduction problems reduce to several well-studied combinatorial problems that have viable approximation algorithms. We implement some of the more viable approaches with heuristics suitable for geospatial abduction, and test them on a real-world data-set. The organization and main contributions of this chapter are as follows.

- Section 4.1.1 formally defines geospatial abduction problems (GAPs for short) and Section 4.2 analyzes their complexity.
- Section 4.3 develops a "naive" algorithm for a basic geospatial abduction problem called k-**SEP** and shows reductions to set-covering, dominating set, and linear-integer programming that allow well-known algorithms for these problems to be applied to GAPs.
- Section 4.4 describes two greedy algorithms for k-**SEP** and compares them to a reduction to the set-covering problem.
- Section 4.5 describes our implementation and shows that our greedy algorithms outperform the set-covering reduction in a real-world application on identifying weapons caches associated with Improvised Explosive Device (IED) attacks on US troops in Iraq. We show that even if we simplify k-**SEP** to only cases where k-means classification algorithms work, our algorithms outperform those. We also note that k-means can only be applied to geospatial abduction in certain,

restricted cases as a heuristic with no approximation guarantee. Such cases are quite limited as the sociol-culutral variables encoded is a feasibility overlay cannot be incorporated into the input of a k-means algorithm.

- Section 4.6 compares our approach with related work.

4.1.1 Geospatial Abduction Problem (GAP) Definition

Throughout this chapter, we assume the existence of a finite, 2-dimensional $M \times N$ space $\mathcal{S}^2$ for some integers $M, N \geq 1$ called the geospatial universe (or just universe). Each point $p \in \mathcal{S}$ is of the form (x, y) where x, y are integers and $0 \leq x \leq M$ and $0 \leq y \leq N$. We assume that all observations we make occur within space $\mathcal{S}$. We use the space shown in Figure 4.1 throughout this chapter to illustrate the concepts we introduce. We assume that $\mathcal{S}$ has an associated distance function d which assigns a non-negative distance to any two points and satisfies the usual distance axioms.[3]

Definition 51 (observation)**.** *An* observation $\mathcal{O}$ *is any finite subset of* $\mathcal{S}$*.*

Consider the geospatial universe shown in Figure 4.1. In the serial killer application, the red dots would indicate the locations of the murders, while in the ship-wreck example, they would indicate the locations where artifacts were found. We wish to identify the killer's location (or the sunken ship or the sun bear's den).

[2]We use integer coordinates as most real world geospatial information systems (GIS) systems use discrete spatial representations.

[3]$d(x, x) = 0; d(x, y) = d(y, x); d(x, y) + d(y, z) \geq d(x, z)$.

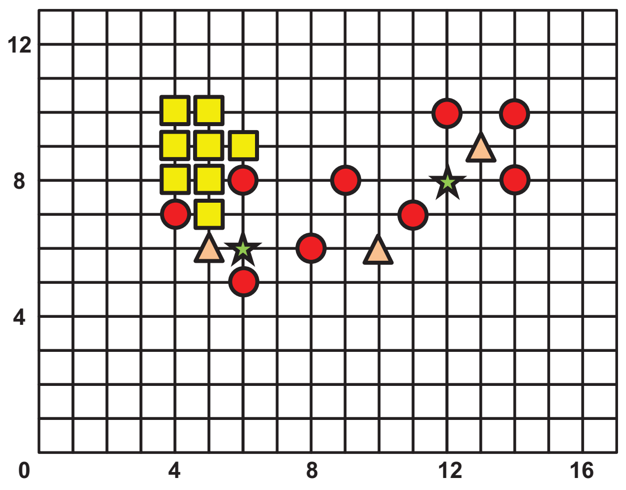


Figure 4.1: A space. Red dots denote observations. Yellow squares denote infeasible locations. Green stars show one (0,3) explanation, while pink triangles show another (0,3) explanation.

As mentioned earlier, there are many constraints that govern where such locations might be. For instance, it is unlikely that the sun-bear's den (or the killer's house or office) is in the water, while the sunken ship is unlikely to be on land.

Definition 52 (feasibility predicate)**.** *A feasibility predicate* feas *is a function from* $\mathcal{S}$ *to* $\{\mathit{TRUE}, \mathit{FALSE}\}$.

Thus, $\mathsf{feas}(p) = \mathsf{TRUE}$ means that point p is feasible and must be considered in the search. Figure 4.1, denotes infeasible places via a yellow square. Throughout this chapter, we assume that feas is an arbitrary, but fixed predicate.[4] Further, as feas is defined as a function over $\{\mathsf{TRUE}, \mathsf{FALSE}\}$, it can allow for user input based

[4]We also assume throughout the chapter that feas is computable in constant time. This is a realistic assumption, as for most applications, we assume feas to be user-defined. Hence, we can leverage a data-structure indexed with the coordinates of $\mathcal{S}$ to allow for constant-time computation.

on analytical processes currently in place. For instance, in the military, analysts often create "MCOO" overlays where "restricted terrain" is deemed infeasible [170]. We can also easily express feasibility predicates in a Prolog-style language – we can easily state (in the serial killer example) that point p is considered feasible if p is within R units of distance from some observation and p is not in the water. Likewise, in the case of the sun bear example, the same language might state that p is considered feasible if p is within R_1 units of distance from marks on trees, within R_2 units of scat, and if p has some landcover that would allow the bear to hide. A Prolog-style language that can express such notions of feasibility is the hybrid knowledge base paradigm [108] in which Prolog style rules can directly invoke a GIS system.

Definition 53 ((α, β) explanation)**.** *Suppose $\mathcal{O}$ is a finite set of observations, $\mathcal{E}$ is a finite set of points in $\mathcal{S}$, and $\alpha \geq 0$, $\beta > 0$ are some real numbers. $\mathcal{E}$ is said to be an* (α, β) explanation *of $\mathcal{O}$ iff:*

- $p \in \mathcal{E}$ *implies that* $\mathsf{feas}(p) = \textit{TRUE}$, *i.e. all points in $\mathcal{E}$ are feasible and*
- $(\forall o \in \mathcal{O})(\exists p \in \mathcal{E})\ \alpha \leq d(p, o) \leq \beta$, *i.e. every observation is neither too close nor too far from some point in $\mathcal{E}$.*

Thus, an (α, β) explanation is a set of points (e.g. denoting the possible locations of the home/office of the serial killer or the possible locations of the bear's den). Each point must be feasible and every observation must have an analogous point in the explanation which is neither too close nor too far.

Given an (α, β) explanation $\mathcal{E}$, there may be an observation $o \in \mathcal{O}$ such that there are two (or more) points $p_1, p_2 \in \mathcal{E}$ satisfying the conditions of the second bullet above. If $\mathcal{E}$ is an explanation for $\mathcal{O}$, a *partnering function* $\wp_{\mathcal{E}}$ is a function from $\mathcal{O}$ to $\mathcal{E}$ such that for all $o \in \mathcal{O}$, $\alpha \leq d(\wp_{\mathcal{E}}(o), o) \leq \beta$. $\wp_{\mathcal{E}}(o)$ is said to be o's *partner* according to the partnering function $\wp_{\mathcal{E}}$. We now present a simple example of (α, β) explanations.

Example 4.1.1. *Consider the observations in Figure 4.1 and suppose* $\alpha = 0, \beta = 3$. *Then the two green stars denote an* (α, β) *explanation, i.e. the set* $\{(6, 6), (12, 8)\}$ *is a* $(0, 3)$ *explanation. So is the set of three pink triangles, i.e. the set* $\{(5, 6), (10, 6), (13, 9)\}$ *is also an* $(0, 3)$ *explanation.*

The basic problem that we wish to solve in this chapter is the following.

The Simple (α, β) **Explanation Problem (SEP).**

INPUT: Space $\mathcal{S}$, a set $\mathcal{O}$ of observations, a feasibility predicate feas, and numbers $\alpha \geq 0$, $\beta > 0$.

OUTPUT: "Yes" if there exists an (α, β) explanation for $\mathcal{O}$ — "no" otherwise.

A variant of this problem is the k-**SEP** problem which requires, in addition, that $\mathcal{E}$ contains k elements or less, for $k < |\mathcal{O}|$. Yet another variant of the problem tries to find an explanation $\mathcal{E}$ that is "best" according to some cost function.

Definition 54 (cost function χ). *A cost function* χ *is a mapping from explanations to non-negative reals.*

We will assume that cost functions are designed so that the smaller the value they return, the more desirable an explanation is. Some example cost functions are given below. The simple one below merely looks at the mean distances between observations and their partners.

Example 4.1.2 (Mean-distance)**.** *Suppose $\mathcal{S}, \mathcal{O}, \mathsf{feas}, \alpha, \beta$ are all given and suppose $\mathcal{E}$ is an (α, β) explanation for $\mathcal{O}$ and $\wp_{\mathcal{E}}$ is a partnering function. We could initially set the cost of an explanation $\mathcal{E}$ (with respect to this partnering function) to be:*

$$\chi_{\wp_{\mathcal{E}}}(\mathcal{E}) \quad = \quad \frac{\Sigma_{o \in \mathcal{O}}\, d(o, \wp_{\mathcal{E}}(o))}{|\mathcal{O}|}.$$

Suppose $ptn(\mathcal{E})$ is the set of all partner functions for $\mathcal{E}$ in the above setting. Then we can set the cost of $\mathcal{E}$ as:

$$\chi_{mean}(\mathcal{E}) \quad = \quad inf\{\chi_{\wp_{\mathcal{E}}}(\mathcal{E}) \mid \wp_{\mathcal{E}} \in ptn(\mathcal{E})\}.$$

The above definition removes reliance on a single partnering function as there may be several partnering functions associated with a single explanation. We illustrate this definition using our sun bear example.

Example 4.1.3. *Wildlife experts have found droppings and other evidence of the Malayan sun bear in a given space, $\mathcal{S}$, depicted in Figure 4.2. Points $\{o_1, o_2, o_3\}$ indicate locations of evidence of the Malayan sun bear (we shall refer to these as set $\mathcal{O}$). Points $\{p_1, p_2, \ldots, p_8\}$ indicate feasible dwellings for the bear. The concentric rings around each element of $\mathcal{O}$ indicate the distance $\alpha = 1.7km$ and $\beta = 3.7km$. The set $\{p_3, p_6\}$ is a valid $(1.7, 3.7)$ explanation for the set of evidence, $\mathcal{O}$. However, we note that observation o_2 can be partnered with either point. If we are looking to*

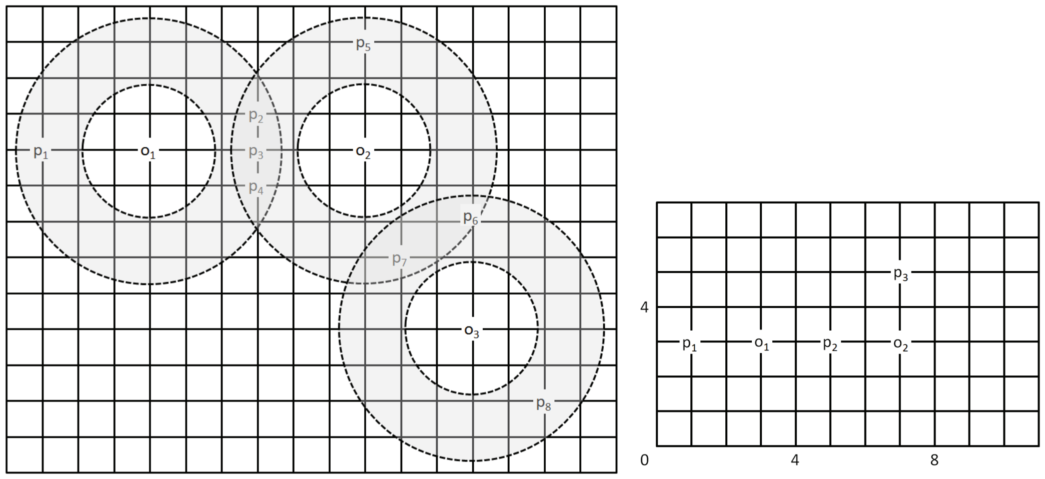


Figure 4.2: **Left:** Points $\{o_1, o_2, o_3\}$ indicate locations of evidence of the Malayan sun bear (we shall refer to these as set $\mathcal{O}$). Points $\{p_1, p_2, \ldots, p_8\}$ indicate feasible dwellings for the bear. The concentric rings around each element of $\mathcal{O}$ indicate the distance $\alpha = 1.7km$ and $\beta = 3.7km$. **Right:** Points $\{p_1, p_2, p_3\}$ are feasible for crime-scenes $\{o_1, o_2\}$. $\{p_1, p_2\}$ are safe-houses within a distance of $[1, 2]$ km. from crime scene o_1 and $\{p_2, p_3\}$ are safe-houses within a distance of $[1, 2]$ km. from crime scene o_2.

minimize distance, we notice that $d(o_2, p_3) = 3km$ and $d(o_2, p_6) = 3.6km$, hence p_3 is the partner for o_2 such that the distance is minimized.

We now define an "optimal" explanation as one that minimizes cost.

Definition 55. *Suppose $\mathcal{O}$ is a finite set of observations, $\mathcal{E}$ is a finite set of points in $\mathcal{S}$, $\alpha \geq 0$, $\beta > 0$ are some real numbers, and χ is a cost function. $\mathcal{E}$ is said to be an* optimal *(α, β) explanation iff $\mathcal{E}$ is an (α, β) explanation for $\mathcal{O}$ and there is no other (α, β) explanation $\mathcal{E}'$ for $\mathcal{O}$ such that $\chi(\mathcal{E}') < \chi(\mathcal{E})$.*

We present an example of optimal (α, β) explanations below.

Example 4.1.4. *Consider the sun bear from Example 4.1.3 whose behavior is depicted in Figure 4.2 (left). While* $\{p_3, p_6\}$ *is a valid solution for the k-**SEP** problem* $(k = 2)$*, it does not optimize mean distance. In this case the mean distance would be* $3km$*. However, the solution* $\{p_3, p_7\}$ *provides a mean-distance of* $2.8km$*.*

Suppose we are tracking a serial killer who has struck at locations $\mathcal{O} = \{o_1, o_2\}$*. The points* $\{p_1, p_2, p_3\}$ *are feasible locations as safe-houses for the killer (partners). This is depicted in Figure 4.2 (right). Based on historical data, we know that serial killers strikes are at least* $1km$ *away from a safe-house and at most* $2km$ *from the safe house* $(\alpha = 1,\ \beta = 2)$*. Thus, for* $k = 2$*, any valid explanation of size* 2 *provides an optimal solution wrt mean-distance as every feasible location for a safe-house is within* $2km$ *of a crime scene.*

We are now ready to define the cost-based explanation problem.

The Cost-based (α, β) Explanation Problem.

INPUT: Space $\mathcal{S}$, a set $\mathcal{O}$ of observations, a feasibility predicate feas, numbers $\alpha \geq 0$, $\beta > 0$, a cost function χ and a real number $v > 0$.

OUTPUT: "Yes" if there exists an (α, β) explanation $\mathcal{E}$ for $\mathcal{O}$ such that $\chi(\mathcal{E}) \leq v$ — "no" otherwise.

It is easy to see that standard classification problems like k-means [5] can be captured within our framework by simply assuming that $\alpha = 0$, $\beta > max(M, N)^2$

[5] See [4] for a survey on classification work.

and that all points are feasible. In contrast, standard classification algorithms cannot take feasibility into account - and this is essential for the above types of applications.

4.2 Complexity of GAP Problems

SEP can be easily solved in PTIME. Given a set $\mathcal{O}$ of observations, for each $o \in \mathcal{O}$, let $P_o = \{p \in \mathcal{S} \mid feas(p) = \textsf{TRUE} \wedge\ \alpha \leq d(p,o) \leq \beta\}$. If $P_o \neq \emptyset$ for each o, we return "yes". We call this algorithm STRAIGHTFORWARD-SEP. Another algorithm would merely find the set F of all feasible points and return "yes" iff for every observation o, there is at least one point $p \in F$ such that $\alpha \leq d(p,o) \leq \beta$. In this case, F is the explanation produced - but it is a very poor explanation. In the serial killer example, F merely tells the police to search all feasible locations without trying to do anything intelligent. k-SEP allows the user to constrain the size of the explanation so that "short and sweet" explanations that are truly meaningful are produced. The following result states that k-**SEP** is NP-Complete - the proof is a reduction from *Geometric Covering by Discs* (GCD) [76].

Theorem 19. *k-**SEP** is NP-Complete.*

In the associated optimization problem with k-**SEP**, we wish to produce an explanation of minimum cardinality. Note that minimum cardinality is a common criterion for parsimony in abduction problems [141]. We shall refer to this problem as MINSEP. This problem is obviously NP-hard by Theorem 19. We can adjust STRAIGHTFORWARD-SEP to find a solution to MINSEP by finding the minimum hitting set of the P_o's.

Example 4.2.1. *Consider the serial killer scenario in Example 4.1.4 and Figure 4.2 (right). Crime scene (observation) o_1 can be partnered with two possible safe-houses $\{p_1, p_2\}$ and crime scene o_2 can be partnered with $\{p_2, p_3\}$. We immediately see that the potential safe house located at p_2 is in both sets. Therefore, p_2 is an explanation for both crime scenes. As this is the only such point, we conclude that $\{p_2\}$ is the minimum-sized solution for the* ***SEP*** *problem. However, while it is possible for STRAIGHTFORWARD-SEP to return this set, there are no assurances it does. As we saw in Example 4.1.4, $\mathcal{E} = \{p_1, p_2\}$ is a solution to* ***SEP****, although a solution with lower cardinality ($\{p_2\}$) exists. This is why we introduce the MINSEP problem.*

With the complexity of k-**SEP**, the following corollary tells us the complexity class of the Cost-based Explanation problem. We show this reduction by simply setting the cost function $\chi(\mathcal{E}) = |\mathcal{E}|$.

Corollary 6. *Cost-based Explanation is NP-Complete.*

As described earlier, MINSEP has the feel of a set-covering problem. Although the generalized cost-based explanation cannot be directly viewed with a similar intuition (as the cost maps explanations to reals – not elements of $\mathcal{S}$), there is an important variant of the Cost-based problem that does. We introduce weighted **SEP**, or **WT-SEP** below.

Weighted Spatial Explanation. (**WT-SEP**)

INPUT: A space $\mathcal{S}$, a set $\mathcal{O}$ of observations, a feasibility predicate feas, numbers $\alpha \geq 0$, $\beta > 0$, a weight function $c : \mathcal{S} \to \Re$, and a real number $v > 0$.

OUTPUT: "Yes" if there exists an (α, β) explanation $\mathcal{E}$ for $\mathcal{O}$ such that $\sum_{p \in \mathcal{E}} c(p) \leq v$ — "no" otherwise.

In this case, we can easily show NP-Completeness by reduction from k-**SEP**, we simply set the weight for each element of $\mathcal{S}$ to be one, causing $\sum_{p \in \mathcal{E}} c(p)$ to equal the cardinality of $\mathcal{E}$.

Corollary 7. ***WT-SEP*** *is NP-Complete.*

Cost-based explanation problems presented in this section are very general. While the complexity results hold for an arbitrary function in a general case, we also consider specific functions as well. Below we present the total-distance minimization explanation problem (**TD-SEP**). This is a problem where we seek to minimize the sum of distances between observations and their closest partners while imposing a restriction on cardinality.

Total Distance Minimization Explanation Problem. (**TD-SEP**)
For space $\mathcal{S}$, let $d : \mathcal{S} \times \mathcal{S} \rightarrow \Re$ be the Euclidean distance between two points in $\mathcal{S}$.
INPUT: A space $\mathcal{S}$, a set $\mathcal{O}$ of observations, a feasibility predicate feas, numbers $\alpha \geq 0$, $\beta > 0$, positive integer $k < |\mathcal{O}|$, and real number $v > 0$.
OUTPUT: "Yes" if there exists an (α, β) explanation $\mathcal{E}$ for $\mathcal{O}$ such that $|\mathcal{E}| = k$ and $\sum_{o_i \in \mathcal{O}} \min_{p_j \in \mathcal{E}} d(o_i, p_j) \leq v$ — "no" otherwise.

Theorem 20. ***TD-SEP*** *is NP-Complete.*

The NP-hardness of the **TD-SEP** is based on a reduction from the k-Median

Problem [134]. This particular reduction (details in the appendix) also illustrates how the k-median problem is a special case of GAPs, but k-median problems cannot handle arbitrary feasibility predicates of the kind that occur in real-life geospatial reasoning. The same argument applies to k-means classifiers as well.

4.3 Exact Algorithm for GAP Problems

This section presents four exact approaches to solve k-**SEP** and **WT-SEP**. First, we provide an enumerative approach that exhaustively searches for an explanation. Then, we show that the problem reduces to set-cover, dominating set, and linear-integer programming. Existing algorithms for these problems can hence be used directly. Throughout this section, we shall use the symbols Δ to represent the bound on the number of partners that can be associated with a single observation and f to represent the bound on the number of observations supported by a single partner. Note that both values are bounded by $\pi(\beta^2 - \alpha^2)$, however they can be much less in practice – specifically f is normally much smaller than Δ.

4.3.1 Naive Exact Algorithm

We now show correctness of NAIVE-KSEP-EXACT. This algorithm provides an exact solution to k-**SEP** but takes exponential time (in k). The algorithm first identifies a set L of all elements of $\mathcal{S}$ that could be possible partners for $\mathcal{O}$. Then, it considers all subsets of L of size less than or equal to k. It does this until it identifies one such subset as an explanation.

Algorithm 13 (NAIVE-KSEP-EXACT)

INPUT: Space $\mathcal{S}$, a set $\mathcal{O}$ of observations, a feasibility predicate feas, real numbers $\alpha \geq 0$, $\beta > 0$, and natural number $k > 0$

OUTPUT: Set $\mathcal{E} \subseteq \mathcal{S}$ of size k (or less) that explains $\mathcal{O}$

1. Let M be a matrix array of pointers to binary string $\{0,1\}^{|\mathcal{O}|}$. M is of the same dimensions as $\mathcal{S}$. Each element in M is initialized to NULL. For a given $p \in \mathcal{S}$, $M[p]$ is the place in the array.

2. Let L be a list of pointers to binary strings. L is initialized as null.

3. For each $o_i \in \mathcal{O}$ do the following

   (a) Determine all points $p \in S$ such that $\alpha \leq d(o,p) \leq \beta$ such that $\mathsf{feas}(p) = \mathsf{TRUE}$.

   (b) For each of these points, p, if $M[p] = \mathsf{NULL}$ then initialize a new array where only bit i is set to 1. Then add a pointer to $M[p]$ in L.

   (c) Otherwise, set bit i of the existing array to 1.

4. For any k elements of L (actually the k elements pointed to by elements of L), we shall designate $\ell_1, \ldots, \ell_j, \ldots \ell_k$ as the elements. We will refer to the ith bit of element ℓ_j as $\ell_j(i)$.

5. Exhaustively generate all possible combinations of k elements of L until one such combination is found where $\forall i \in [1, |\mathcal{O}|]$, $\sum_{j=1}^{k}(\ell_j(i)) > 0$

6. If no such combination is found, return NO. Otherwise, return the first combination that was found.

Proposition 29. *If there is a k-sized simple (α, β) explanation for $\mathcal{O}$, then NAIVE-KSEP-EXACT returns an explanation. Otherwise, it returns NO.*

Finally, we have the complexity of the algorithm.

Proposition 30. *The complexity of NAIVE-KSEP-EXACT is $O(\frac{1}{(k-1)!}(\pi(\beta^2-\alpha^2)|\mathcal{O}|)^{(k+1)})$.*

An exact algorithm for the cost-based explanation problems follows trivially from the NAIVE-KSEP-EXACT algorithm by adding the step of computing the value for χ for each combination. Provided this computation takes constant time, this does not affect the $O(\frac{1}{(k-1)!}(\pi(\beta^2 - \alpha^2)|\mathcal{O}|)^{(k+1)})$ run time of that algorithm.

4.3.2 An Exact Set-Cover Based Approach

We now show that k-**SEP** polynomially reduces to an instance of the popular set-covering problem [80] which allows us to directly apply the well-known greedy algorithm reviewed in [136]. SET_COVER is defined as follows.

The Set-Cover Problem. (SET_COVER)
INPUT: Set of elements, E and a family of subsets of E, $F \equiv \{S_1, \ldots, S_{max}\}$, and positive integer k.
OUTPUT: "Yes" if there exists a k-sized subset of F, F_k, such that $\bigcup_{i=1}^{k}\{S_i \in F_k\} \equiv E$.

Through a simple modification of NAIVE-KSEP-EXACT, we can take an instance of k-**SEP** and produce an instance of SET_COVER. We run the first four steps, which only takes $O(\Delta \cdot |\mathcal{O}|)$ time by the proof of Proposition 30.

Theorem 21. *k-**SEP** polynomially reduces to SET_COVER.*

Example 4.3.1. *Consider the serial killer scenario in Example 4.1.4 and Figure 4.2 (right). Suppose we want to solve this problem as an instance of k-**SEP** by a reduction to set-cover. We consider the set of crime-scene locations, $\mathcal{O} \equiv \{o_1, o_2\}$ as the set we wish to cover. We obtain our covers from the first four steps of NAIVE-KSEP-EXACT. Let us call the result list L. Hence, we can view the values of the elements in L as the following sets $S_1 \equiv \{o_1\}, S_2 \equiv \{o_1, o_2\}, S_3 \equiv \{o_2\}$. These correspond with points p_1, p_2, p_3 respectively. As S_2 covers $\mathcal{O}$, p_2 is an explanation.*

The traditional approach for approximation of set-cover has a time complexity of $O(|E| \cdot |F| \cdot size)$, where $size$ is the cardinality of the largest set in the family F (i.e. $size = \max_{i \leq |F|} |S_i|$). This approach obtains an approximation ratio of $1 + \ln(size)$ [136]. As f is the quantity of the largest number of observations supported by a single partner, the approximation ratio for k-**SEP** using a greedy-scheme after a reduction from set-cover is $1+\ln(f)$. The NAIVE-KSEP-SC algorithm below leverages the above reduction to solve the k-SEP problem.

Proposition 31. *NAIVE-KSEP-SC has a complexity of $O(\Delta \cdot f \cdot |\mathcal{O}|^2)$ and an approximation ratio of $1 + \ln(f)$.*

Proposition 32. *A solution $\mathcal{E}$ to NAIVE-KSEP-SC provides a partner to every observation in $\mathcal{O}$ if a partner exists – otherwise, it returns IMPOSSIBLE.*

The algorithm NAIVE-KSEP-SC is a naive, straight-forward application of the $O(|E| \cdot |F| \cdot size)$ greedy approach for set-cover as presented in [136]. We note that it is

Algorithm 14 (NAIVE-KSEP-SC)

INPUT: Space $\mathcal{S}$, a set $\mathcal{O}$ of observations, a feasibility predicate feas, and real numbers $\alpha \geq 0$, $\beta > 0$

OUTPUT: Set $\mathcal{E} \subseteq \mathcal{S}$ that explains $\mathcal{O}$

```
1. Initialize list $\mathcal{E}$ to null. Let $M$ be a matrix array of the same dimensions as $\mathcal{S}$ of lists of pointers initialized to null. For a given $p \in \mathcal{S}$, $M[p]$ is the place in the array. Let $L$ be a list of pointers to lists in $M$, $L$ is initialized to null.
2. Let $\mathcal{O}'$ be an array of Booleans of length $|\mathcal{O}|$. $\forall i \in [1, |\mathcal{O}|]$, initialize $\mathcal{O}'[i] = \mathsf{TRUE}$. For some element $o \in \mathcal{O}$, $\mathcal{O}'[o]$ is the corresponding space in the array. Let numObs $= |\mathcal{O}|$
3. For each element $o \in \mathcal{O}$, do the following.
    (a) Determine all elements $p \in \mathcal{S}$ such that feas$(p) = \mathsf{TRUE}$ and $d(o,p) \in [\alpha, \beta]$
    (b) If there does not exist a $p \in \mathcal{S}$ meeting the above criteria, then terminate the program and return IMPOSSIBLE.
    (c) If $M[p] =$ null then add a pointer to $M[p]$ to $L$
    (d) Add a pointer to $o$ to the list $M[p]$.
4. While numObs $> 0$ loop
    (a) Initialize pointer $cur\_ptr$ to null, integer $cur\_size$ to 0
    (b) For each $ptr \in L$, do the following:
        i. Initialize integer $this\_size$ to 0, let $M[p]$ be the element of $M$ pointed to by $ptr$
        ii. For each $obs\_ptr$ in the list $M[p]$, do the following
            A. Let $i$ be the corresponding location in array $\mathcal{O}'$ to $obs\_ptr$
            B. If $\mathcal{O}'[i] = \mathsf{TRUE}$, increment $this\_size$ by 1
        iii. If $this\_size > cur\_size$, set $cur\_size = this\_size$ and have $cur\_ptr$ point to $M[p]$
    (c) Add $p$ to $\mathcal{E}$
    (d) For every $obs\_ptr$ in the list pointed to by $cur\_ptr$, do the following:
        i. Let $i$ be the corresponding location in array $\mathcal{O}'$ to $obs\_ptr$
        ii. If $\mathcal{O}'[i]$, then set it to FALSE and decrement numObs by 1
    (e) Add the location in space $\mathcal{S}$ pointed to by $cur\_ptr$ to $\mathcal{E}$
5. Return $\mathcal{E}$
```

possible to implement a heap to reduce the time-complexity to $O(\Delta \cdot f \cdot |\mathcal{O}| \cdot \lg(\Delta \cdot |\mathcal{O}|))$ - avoiding the cost of iterating through all possible partners in the inner-loop.

In addition to the straightforward greedy algorithm for set-covering, there are several other algorithms that provide different time complexity/approximation ratio combinations. However, with a reduction to the set-covering problem we must consider the result of [113] which states that set-cover cannot be approximated within a ratio $c \cdot log(n)$ for any $c < 0.25$ (where n is the number of subsets in the family F) unless $NP \subseteq DTIME[n^{\texttt{poly log } n}]$.

A reduction to set-covering has the advantage of being straightforward. It also allows us to leverage the wealth of approaches developed for this well-known problem. In the next section, we show that k-**SEP** reduces to the dominating set problem as well. We then explore alternate approximation techniques based on this reduction.

4.3.3 An Exact Dominating Set Based Approach

We show below that k-**SEP** also reduces to the well known dominating set problem (**DomSet**) [54] allowing us to potentially leverage fast algorithms such as the randomized-distributed approximation scheme in [75]. **DomSet** is defined as follows.

Dominating Set. (**DomSet**)

INPUT: Graph $G = (V, E)$ and positive integer $K \leq |V|$.

OUTPUT: "Yes" if there is a subset $V' \subset V$ such that $|V'| \leq K$ and such that every vertex $v \in V - V'$ is joined to at least one member of V' by an edge in E.

As the dominating set problem relies on finding a certain set of nodes in a graph, then, unsurprisingly, our reduction algorithm, Algorithm 15, takes space $\mathcal{S}$, an observation set $\mathcal{O}$, feasibility predicate feas, and numbers α, β and returns graph $G_{\mathcal{O}}$ based on these arguments.

We now present an example to illustrate the relationship between a dominating set of size k in $G_{\mathcal{O}}$ and a k-sized simple (α, β) explanation for $\mathcal{O}$. The following example illustrates the relationship between a k-**SEP** problem and **DomSet**.

Example 4.3.2. *Consider the serial killer scenario in Example 4.1.4, pictured in Figure 4.2 (right). Suppose we want to solve this problem as an instance of k-**SEP** by a reduction to **DomSet**. We want to find a 1-sized simple (α, β) explanation (safe-house) for $\mathcal{O}$ (the set of crime scenes, $\{o_1, o_2\}$). Suppose that after running an algorithm such as STRAIGHFORWARD-SEP, we find that $\{p_1, p_2, p_3\}$ are elements of $\mathcal{S}$ that are feasible. $\{p_1, p_2\}$ are all within a distance of α, β from o_1 and $\{p_2, p_3\}$ are all within a distance of α, β from o_2. We run KSEP-TO-DOMSET which creates graph, $G_{\mathcal{O}}$. Refer to Figure 4.3 for the graph. We can see that $\{p_2\}$ is a 1-sized dominating sets for $G_{\mathcal{O}}$, hence a 1-sized explanation for $\mathcal{O}$.*

We notice that the inner loop of KSEP-TO-DOMSET is bounded by $O(\Delta)$ operations and the outer loop will iterate $|\mathcal{O}|$ times. Thus, the complexity of KSEP-TO-DOMSET is $O(\Delta \cdot |\mathcal{O}|)$.

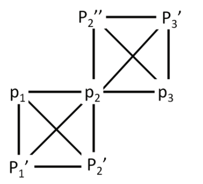


Figure 4.3: Results of KSEP-TO-DOMSET based on data seen in Figure 4.2 (right). Note that $\{p_1, p_2, p'_1, p'_2\}$ form a complete graph and $\{p_2, p_3, p''_2, p'_3\}$ also form a complete graph. Note that $\{p_2\}$ is a dominating set of size 1. Hence, $\{p_2\}$ is a 1-sized simple (α, β) explanation for $\mathcal{O}$, as depicted in Figure 4.2 (right).

Proposition 33. *The complexity of KSEP-TO-DOMSET is $O(\Delta \cdot |\mathcal{O}|)$.*

Example 4.3.2 should give us some intuition into why the reduction to **DomSet** works. We provide the formal proof in the Appendix.

Theorem 22. *k-**SEP** is polynomially reducible to **DomSet**.*

The straightforward approximation scheme for **DomSet** is to view the problem as an instance of SET_COVER and apply a greedy algorithm. The reduction would view the set of vertices in $G_{\mathcal{O}}$ as the elements, and the family of sets as each vertex and its neighbors. This results in both a greater complexity and a worse approximation ratio when compared with the reduction directly to SET_COVER.

Proposition 34. *Solving k-**SEP** by a reduction to **DomSet** using a straightforward greedy approach has time-complexity $O(\Delta^3 \cdot f \cdot |\mathcal{O}|^2)$ and an approximation ratio bounded by $O(1 + \ln(2 \cdot f \cdot \Delta))$.*

There are other algorithms to approximate **DomSet** [75, 89]. By leveraging [75], we can obtain an improved complexity while retaining the same approximation ratio as the greedy approach.

Proposition 35. *Solving k-**SEP** by a reduction to **DomSet** using the distributed, randomized algorithm presented in [75] has a time complexity $O(\Delta \cdot |\mathcal{O}| + \ln(2 \cdot \Delta \cdot |\mathcal{O}|) \cdot \ln(2 \cdot \Delta \cdot f))$ with high probability and approximation ratio of $O(1 + \ln(2 \cdot f \cdot \Delta))$.*

Hence, although a reduction to dominating set generally gives us a worse approximation guarantee, we can (theoretically) outperform set-cover with the randomized algorithm for dominating set in terms of complexity.

4.3.4 An Exact Integer Linear Programming based Approach

Given an instance of k-**SEP**, we show how to create a set of integer constraints that if solved, will yield a solution to the problem.

Definition 56 (OPT-KSEP-IPC)**.** *The k-**SEP** integer programming constraints (OPT-KSEP-IPC) require the following information, obtained in $O(|\mathcal{O}| \cdot \pi(\beta^2 - \alpha^2)$ time:*

- *Let L be the set of all possible partners generated in the first four steps of NAIVE-KSEP-EXACT.*
- *For each $p \in L$, let $str(p)$ be the string of $|\mathcal{O}|$ bits, where bit $str(p)_i$ is 1 if p is a partner of the ith observation (this is also generated in the first four steps of NAIVE-KSEP-EXACT).*

For each $p_j \in L$, let $x_j \in \{0, 1\}$. $x_j = 1$ iff p_j is in $\mathcal{E}$.

Then KSEP-IPC consists of the following:

Minimize** $\sum_{p_j \in L} x_j$ **subject to

1. $\forall o_i \in \mathcal{O}$, $\sum_{p_j \in L} x_j \cdot str(p_j)_i \geq 1$

2. $\forall p_j \in L$, $x_j \in \{0,1\}$ *(for the relaxed linear program:* $x_j \leq 1$*)*

Proposition 36. *OPT-KSEP-IPC consists of* $O(|\mathcal{O}|\pi(\beta^2 - \alpha^2))$ *variables and* $O(|\mathcal{O}| \cdot \pi(\beta^2 - \alpha^2))$ *constraints.*

Proposition 37. *For a given instance of the optimization version k-**SEP**, if OPT-KSEP-IPC is solved, then* $\bigcup_{p_j \in L_{x_j=1}} p_j$ *is an optimal solution to k-**SEP**.*

Example 4.3.3. *Consider the serial killer scenario in Example 4.1.4, pictured in Figure 4.2 (right). Suppose we want to solve this problem as an instance of MINSEP. We would set up the constraints as follows:*

Minimize $x_1 + x_2 + x_3$ ***subject to*** $1 \cdot x_1 + 1 \cdot x_2 + 0 \cdot x_3 \geq 1$ *and* $0 \cdot x_1 + 1 \cdot x_2 + 1 \cdot x_3 \geq 1$, *where* $x_1, x_2, x_3 \in \{0,1\}$

Obviously, setting $x_1 = 0, x_2 = 1, x_3 = 0$ *provides an optimal solution. Hence, as* x_2 *is the only non-zero variable,* p_2 *is the explanation for the crime-scenes.*

A solution to the constraints OPT-KSEP-IPC can be approximated using the well-known "rounding" technique [68, 174] that relaxes constraints. We present an OPT-KSEP-IPC using rounding.

Proposition 38. *NAIVE-KSEP-ROUND returns an explanation for* $\mathcal{O}$ *that is within a factor* Δ *from optimal, where* Δ *is the maximum number of possible partners associated with any observation.*

There are several things to note about this approach. First, it can be easily adapted to many of the weighted variants - such as **WT-SEP**. Second, we note that the rounding algorithm is not a randomized rounding algorithm – which often produces a solution that satisfies all of the constraints in the linear-integer program. The above algorithm guarantees that all of the observations will be covered (if an explanation exists). Finally, this approach allows us to leverage numerous software packages for solving linear and linear-integer programs.

Algorithm 15 (KSEP-TO-DOMSET)

INPUT: Space $\mathcal{S}$, a set $\mathcal{O}$ of observations, a feasibility predicate feas, and real numbers $\alpha \geq 0$, $\beta > 0$

OUTPUT: Graph $G_{\mathcal{O}}$ for use in an instance of a **DomSet** problem

1. Let $G_{\mathcal{O}} = (V_{\mathcal{O}}, E_{\mathcal{O}})$ be a graph. Set $V_{\mathcal{O}} = \mathcal{S}$ and $E_{\mathcal{O}} = \emptyset$.
2. Let S be a mapping defined as $S : \mathcal{S} \rightarrow V_{\mathcal{O}}$. In words, S takes elements of the space and returns nodes from $G_{\mathcal{O}}$ as defined in the first step. This mapping does not change during the course of the algorithm.
3. For each $o_i \in \mathcal{O}$ do the following
   (a) Determine all points $p \in S$ that are such that $\alpha \leq d(o, p) \leq \beta$. Call this set P_i
   (b) For all $p \in P_i$ calculate $\mathsf{feas}(p)$. If $\mathsf{feas}(p) = \mathsf{FALSE}$, remove p from P_i.
   (c) Let $V_i = \{v \in V_{\mathcal{O}} | \exists p \in P_i \textit{ such that } S(p) = v\}$.
   (d) Add $|P_i|$ *new* nodes to $V_{\mathcal{O}}$. Add these nodes to V_i as well.
   (e) For every pair of nodes $v_1, v_2 \in V_i$, add edge (v_1, v_2) to $E_{\mathcal{O}}$.
4. Remove all $v \in V_{\mathcal{O}}$ where there does not exist an v' such that $(v, v') \in E_{\mathcal{O}}$
5. If any $P_i \equiv \emptyset$ return IMPOSSIBLE. Otherwise return $G_{\mathcal{O}}$.

Algorithm 16 (NAIVE-KSEP-ROUND)

INPUT: Space $\mathcal{S}$, a set $\mathcal{O}$ of observations, a feasibility predicate feas, and real numbers $\alpha \geq 0$, $\beta > 0$

OUTPUT: Set $\mathcal{E} \subseteq \mathcal{S}$ that explains $\mathcal{O}$

```
1. Run the first four steps of NAIVE-KSEP-EXACT
2. Solve the relaxation of OPT-KSEP-IPC
3. For the $o \in \mathcal{O}$ with the most possible partners, let $\Delta$ be the number of possible
   partners associated with $o$. This can be done in line 1
4. Return all $p_j \in L$ where $x_j \geq \frac{1}{\Delta}$
```

4.4 Greedy Heuristics for **GAP** Problems

4.4.1 A Linear Time Greedy Approximation Scheme

In this section, we introduce a greedy approximation scheme for the optimization version of k-**SEP** that has a lower time-complexity than NAIVE-KSEP-SC but still maintains the same approximation ratio. Our GREEDY-KSEP-OPT1 algorithm runs in linear time w.r.t. $\mathcal{O}$. The key intuition is that NAIVE-KSEP-SC iterates through $O(\Delta \cdot |\mathcal{O}|)$ possible partners in line 4. Our algorithm first randomly picks an observation and then greedily selects a partner for it. This results in the greedy step iterating through only $O(\Delta)$ partners.

Example 4.4.1. *Consider the sun bear from Example 4.1.3 and Figure 4.2. After initializing the necessary data structures in lines 1-3, GREEDY-KSEP-OPT1 iterates through the observations in $\mathcal{O}$ where the associated position in $\mathcal{O}'$ is TRUE. Suppose the algorithm picks o_1 first. It now accesses the list pointed to from OBS$[o_1]$. This gives us a set of pointers to the following elements of $\mathcal{S}$: $\{p_1, p_2, p_3, p_4\}$. Following the greedy selection outlined in line 4 of NAIVE-KSEP-SC, the algorithm iterates through these points, visiting the list of observations associated with each one in the matrix array M.*

First, the algorithm accesses the list pointed to by $M[p_1]$. Figure 4.4 (left) shows the observations considered when p_1 is selected. As there is only one observation in list $M[p_1]$ whose associated Boolean in $\mathcal{O}'$ is TRUE, the variable cur_size is set to 1 (see line 4(b)iii of NAIVE-KSEP-SC). cur_ptr is then set to $M[p_1]$.

Algorithm 17 (GREEDY-KSEP-OPT1)

```
INPUT: Space $\mathcal{S}$, a set $\mathcal{O}$ of observations, a feasibility predicate feas, and real numbers $\alpha \geq 0$, $\beta > 0$
OUTPUT: Set $\mathcal{E} \subseteq \mathcal{S}$ that explains $\mathcal{O}$

1. Run lines 1-2 of NAIVE-KSEP-SC
2. Let OBS be an array, size $|\mathcal{O}|$ of lists to pointers in $M$. For some observation $o$, let OBS$[o]$ be the corresponding list in the array.
3. Run the loop in line 3 of NAIVE-KSEP-SC but when partner $p$ of observation $o$ is considered, add a pointer to $M[p]$ in the list OBS$[o]$. The list $L$ need not be maintained.
4. While numObs $> 0$ loop
   (a) Randomly select an element $o \in \mathcal{O}$ such that $\mathcal{O}'[o] =$ TRUE
   (b) Run the greedy-selection loop of line 4 of NAIVE-KSEP-SC, but consider the list OBS$[o]$ instead of $L$
5. Return $\mathcal{E}$
```

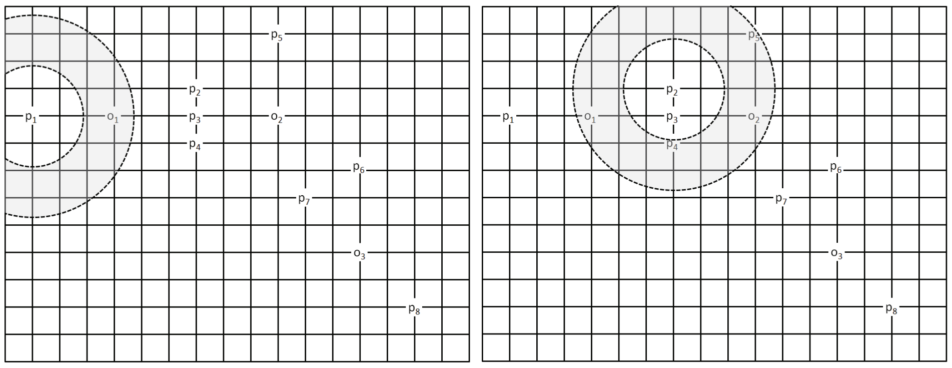


Figure 4.4: **Left:** GREEDY-KSEP-OPT1 accesses the list pointed to by $M[p_1]$ thus considering all observations available to p_1. **Right:** GREEDY-KSEP-OPT1 accesses the list pointed to by $M[p_2]$ and finds it has more active observations than it found in the list pointed to by $M[p_1]$.

Now we consider the next element, p_2. Figure 4.4 (right) shows the list pointed to by $M[p_2]$. As $M[p_2]$ points to more observations whose associated $\mathcal{O}'$ Boolean is TRUE, we update cur_size to 2 and cur_ptr to $M[p_2]$.

The algorithm then iterates through p_3 and p_4, but finds they do not offer more observations than p_2. Hence, p_2 is added to the solution set ($\mathcal{E}$). The algorithm updates the array of Booleans, $\mathcal{O}'$ and sets $\mathcal{O}'[o_1]$ and $\mathcal{O}'[o_2]$ to FALSE (depicted by X's over those observations in subsequent figures). numObs is decremented by 2.

Now, we enter the second iteration of line 4. The only element for the algorithm to pick at this point is o_3, as only $\mathcal{O}'[o_3]$ is TRUE. The list OBS$[o_3]$ points to the positions $\{p_6, p_7, p_8\}$. In Figure 4.5 we look at what happens as the algorithm considers the p_7. As OBS$[o_2]$ = FALSE, it only considers o_3 when computing this_size.

When the algorithm finishes its consideration of all the elements pointed to

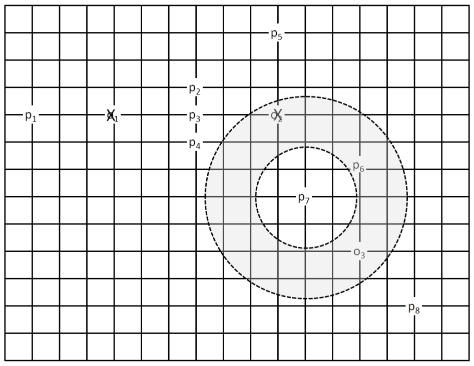


Figure 4.5: GREEDY-KSEP-OPT1 considers the observations available to p_7. The X's on o_1 and o_2 signify that OBS$[o_1]$ and OBS$[o_2]$ are set to FALSE.

by OBS$[o_3]$, it will return the first element of that set (p_6) as neither p_7 nor p_8 were partners to more available observations than p_6 (in our implementation of this algorithm, we use a coin-flip to break ties among partners with the same number of observations). GREEDY-KSEP-OPT1 then adds p_6 to $\mathcal{E}$ and terminates. The final solution returned, $\{p_2, p_6\}$, is a valid (and in this case, optimal) explanation.

Proposition 39 (Complexity of GREEDY-KSEP-OPT1)**.** *GREEDY-KSEP-OPT1 has a complexity of $O(\Delta \cdot f \cdot |\mathcal{O}|)$ and an approximation ratio of $1 + \ln(f)$.*

Proposition 40. *GREEDY-KSEP-OPT1 returns a $|\mathcal{E}|$-sized (α, β) explanation for $\mathcal{O}$.*

GREEDY-KSEP-OPT1 returns IMPOSSIBLE if there is no explanation for $\mathcal{O}$.

We can bound the approximation ratio for GREEDY-KSEP-OPT1 by $O(1 + \ln(f))$, as it is still essentially a greedy algorithm for a covering problem. The main difference between GREEDY-KSEP-OPT1 is the way it greedily chooses covers (partners). This algorithm randomly picks an uncovered observation in each loop and then greedily chooses a cover that covers that observation. Improving the accuracy of this algorithm (in practice) is tied directly to the selection criteria used to pick observations, which is random in GREEDY-KSEP-OPT1. In Section 4.4.2 we develop an algorithm that "smartly" picks observations with a dynamic ranking scheme while maintaining a time complexity lower than the standard set-covering approach.

4.4.2 Greedy Observation Selection

GREEDY-KSEP-OPT1 randomly selects observations although subsequent partner selection was greedy. It is easy to implement an *a-priori* ranking of observations based on something like the maximum number of other observations which share a partner with it. Such a ranking could be implemented at the start of GREEDY-KSEP-OPT1 with no effect on complexity, but the ranking would be static and may lose its meaning after several iterations of the algorithm. We could also implement a dynamic ranking. We present a version of GREEDY-KSEP-OPT1 that we call GREEDY-KSEP-OPT2 that picks the observations based on dynamic ranking, runs in time $O(\Delta \cdot f^2 \cdot |\mathcal{O}| + |\mathcal{O}| \cdot \ln(|\mathcal{O}|))$, and maintains the usual approximation ratio of $1 + \ln(f)$ for greedy algorithms. Our key intuition was to use a Fibonacci heap [49]. With such a data structure, we can update the rankings of observations at constant amortized cost per observation being updated. The most expensive operation is to remove an observation from the heap - which costs an amortized $O(\ln(|\mathcal{O}|))$, however as we can never remove more than $|\mathcal{O}|$ items from the heap, this cost is most likely dominated by the cost of the rest of the algorithm, which is more expensive than GREEDY-KSEP-OPT1 by a factor of f. Recall that f is the bound on the number of observations supported by a single partner - and is often very small in practice.

In order to leverage the Fibonacci heap, there are some restrictions on how the ranking can be implemented. First, the heap puts an element with the minimal key on top, and can only decrease the key of elements - an element in the heap can never have its key increased. Additionally, there is a need for some auxiliary data structures as searching for an element in the heap is very expensive. Fortunately, the k-**SEP** problem is amenable to these type of data structures.

We based the key (ranking) on a simple heuristic for each observation. The key for a given observation o is the number of unique observations that share a partner with o. As we are extracting the minimum-keyed observation, we are taking the observation that has the "least in common" with the other observations. The intuition of choosing an observation with "less in common" with other observations ensures that outliers get covered with larger covers. Meanwhile, elements with a higher rank in this scheme are covered last, which may lead to a more efficient cover. In Section 4.5 we show experimentally that this heuristic was viable for the data-set we considered - providing more accurate results than the reduction from set-covering.

Example 4.4.2. *The basic intuition behind GREEDY-KSEP-OPT2 is similar to GREEDY-KSEP-OPT1 in that it iterates through the observations and greedily chooses a partner. The main difference is that it ranks the observations instead of just ran-*

Algorithm 18 GREEDY-KSEP-OPT2

INPUT: Space $\mathcal{S}$, a set $\mathcal{O}$ of observations, a feasibility predicate feas, and real numbers $\alpha \geq 0$, $\beta > 0$

OUTPUT: Set $\mathcal{E} \subseteq \mathcal{S}$ that explains $\mathcal{O}$

```
1. Run lines 1-3 of GREEDY-KSEP-OPT1.
2. Let $key_1, \dots key_{|\mathcal{O}|}$ be natural numbers associated with each observation. Initially, they are set to 0. For
   some $o \in \mathcal{O}$ let $key_o$ be the associated number.
3. Let REL_OBS be an array of lists of pointers to elements of $\mathcal{O}$. The size of the array is $\mathcal{O}$. For element
   $o \in \mathcal{O}$, let REL_OBS[$o$] be the corresponding space in the array.
4. For each $o \in \mathcal{O}$, do the following:
   (a) For each element $p \in$ OBS[$o$], do the following.
       i. For each element $obs\_ptr$ of the list pointed to by $M[p]$, do the following
          A. If $obs\_ptr$ points to an element of $\mathcal{O}$ not pointed to in the list REL_OBS[$o$], then add
             $obs\_ptr$ to REL_OBS[$o$] and increment $key_o$ by 1.
5. Let OBS_HEAP be a Fibonacci heap. Let QUICK_LOOK be an array (size $\mathcal{O}$) of pointers to elements of
   the heap. For each $o \in \mathcal{O}$, add the tuple $\langle o, key_o \rangle$ to the heap, along with a pointer to the tuple to
   QUICK_LOOK[$o$]. Note we are using $key_o$ as the key for each element in the heap.
6. While OBS_HEAP is not empty, loop
   (a) Take the minimum element of OBS_HEAP, let $o$ be the associated observation with this element.
   (b) Greedily select an element of OBS[$o$] as done in the loop at line 4 of GREEDY-KSEP-OPT1. We shall
       call this element $p$.
   (c) For every $o' \in \mathcal{O}$ pointed to by a pointer in $M[p]$, such that $\mathcal{O}'[o'] =$ TRUE, do the following.
       i. Set $\mathcal{O}'[o'] =$ FALSE
       ii. Remove the element pointed to by QUICK_LOOK[$o'$] from OBS_HEAP
       iii. For every element $o'' \in \mathcal{O}$ pointed to by an element of REL_OBS[$o'$] where $\mathcal{O}'[o''] =$ TRUE do
            the following.
            A. Decrease the $key_{o''}$ by 1.
7. Return $\mathcal{E}$
```

Observation	key_i	$\mathsf{REL\_OBS}[o_i]$
o_1	2	$\{o_1, o_2\}$
o_2	2	$\{o_1, o_2\}$
o_3	2	$\{o_2, o_3\}$

Table 4.1: *key* values and related observations for observations in the sun bear scenario introduced in Example 4.1.3.

*domly selecting them. Consider the sun bear from Example 4.1.3 whose behavior is depicted in Figure 4.2. In Example 4.4.1, we used GREEDY-KSEP-OPT1 to solve the associated k-**SEP** problem for this situation. We shall discuss how GREEDY-KSEP-OPT2 differs.*

The first main difference is that the algorithm assigns a rank to each observation o_i, called key_i, which is also the key used in the Fibonacci heap. This is done in the loop at line 4. It not only calculates key_i for each observation, but it also records the elements "related" to it in the array REL_OBS. Note that a "related" observation needs only to share a partner with a given observation. Not all related observations need to have the same partner. For the sun bear scenario, we show the keys and related observations in Table 4.1.

As the key values are the same for all elements of $\mathcal{O}$, let's assume the algorithm first considers o_1 as in Example 4.4.1. As written, we would take the minimum element in the Fibonacci heap (a constant time operation). We would then consider the partners for o_1 which would result in the greedy selection of p_2, (just as in GREEDY-

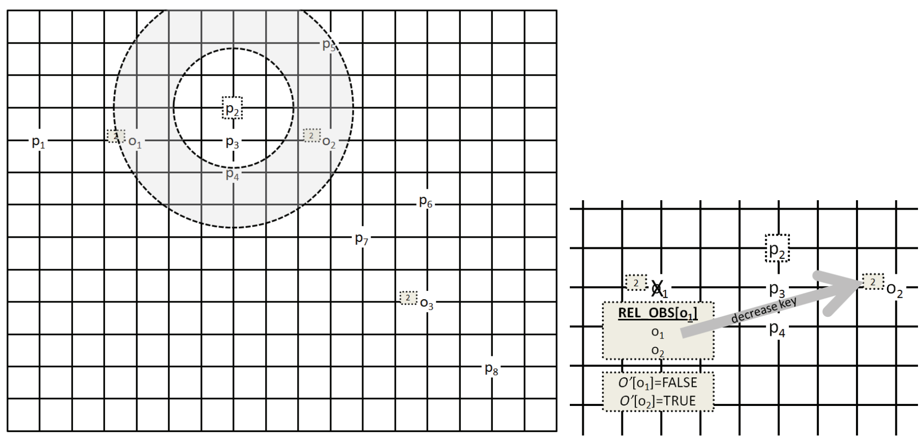


Figure 4.6: **Left:** GREEDY-KESP-OPT2 considers all observations that can be partnered with p_2. Notice that in this figure by each observation we show a box that represents the key of the observation in the Fibonacci heap. **Right:** GREEDY-KSEP-OPT2 removes o_1 from the heap, and iterates through the elements in REL_OBS$[o_1]$, causing it to decrease the key of o_2.

KSEP-OPT1 and NAIVE-KSEP-SC. Also notice we retain the array of Booleans, $\mathcal{O}'$ as well as the array of lists, M to help us with these operations.).

Now the issue arises that we must update the keys for the remaining observations, as well as remove observations covered by p_2. As we maintain REL_OBS and $\mathcal{O}'$, the procedure quickly iterates through the elements covered by p_2: o_1 and o_2. Figure 4.6 shows the status of the observations at this point.

We remove o_1 from the heap, and set $\mathcal{O}'[o_1]$ to FALSE. This prevents us from considering it in the future. We now iterate through each o'' in the list pointed to by REL_OBS$[o_1]$ where $\mathcal{O}'[o'']$ is TRUE and decrease the key of each by one. As per table 4.1, REL_OBS$[o_1] = \{o_1, o_2\}$. As $\mathcal{O}'[o_1] =$ FALSE we do nothing. As $\mathcal{O}'[o_2] =$ TRUE, we decrease the key of the associated node in the Fibonacci heap. The array

QUICK_LOOK ensures we can access that element in constant time. Figure 4.6 (left) graphically depicts this action.

Next, we consider the other element covered by partner p_2: o_2. After removing this element from the heap and setting $\mathcal{O}'[o_2]$ to FALSE, we can easily see that there does not exist any $o'' \in$ REL_OBS$[o_2]$ where $\mathcal{O}'[o''] =$ TRUE. Hence, we can proceed to pick a new minimum observation from the heap - which is o_3 in this case. The greedy selection proceeds (resulting in the choice of p_6), followed by the update procedure (which simply removes the node associated with o_3 from the heap and sets $\mathcal{O}'[o_3] =$ FALSE). As there are no more elements in the heap, GREEDY-KSEP-OPT2 returns the solution $\{p_2, p_6\}$.

Theorem 23 (Complexity of GREEDY-KSEP-OPT2)**.** *GREEDY-KSEP-OPT2 has a complexity of $O(\Delta \cdot f^2 \cdot |\mathcal{O}| + |\mathcal{O}| \cdot \ln(|\mathcal{O}|))$ and an approximation ratio of $1 + \ln(f)$.*

Proposition 41. *GREEDY-KSEP-OPT2 returns a $|\mathcal{E}|$-sized (α, β) explanation for $\mathcal{O}$.*

GREEDY-KSEP-OPT2 returns IMPOSSIBLE if there is no explanation for $\mathcal{O}$.

4.5 Implementation and Experiments

In this section, we show that our geospatial abduction framework and algorithms are viable in solving real-world geospatial abduction problems. Using a real-world data set consisting of counter-insurgency information from Iraq, we were able to accurately locate insurgent weapons cache sites (partners) given previous attacks (observations) and some additional data (used for feas and α, β). This validates our primary research goal for the experiments - to show that geospatial abduction can be used to solve problems in the real-world.

We considered the naive set-covering approach along with GREEDY-KSEP-OPT1 and GREEDY-KSEP-OPT2, which according to our analytical results, had the best approximation ratios and time-complexities. We implemented these algorithms in 4000 lines of Java code, running on a Lenovo T400 ThinkPad laptop running Vista with an Intel Core 2 Duo T9400 2.53 GHz processor and 4.0 GB of RAM. Our SCARE (Social-Cultural Abductive Reasoning Engine) system [157] enabled us to carry out tests on real-world data. This data includes 21 months of Improvised Explosive Device or IED attacks in Baghdad[6] (a 25x27 km region) – these constitute our observations. It also included information on locations of caches associated with those attacks discovered by US forces. The locations of the caches constitute the (α, β) explanation we want to learn. We used data from the International Medical Corps to define feasibility predicates which took the following factors into account: (i) the ethnic makeup of neighborhoods in Baghdad - specifically, Sunni locations were deemed infeasible for cache locations, (ii) the locations of US bases in Baghdad were also considered infeasible and (iii) bodies of water were also deemed infeasible. We also separately ran tests on that part of the above data focused on Sadr City (a 7x7 km district in Baghdad) alone. On both these regions, we overlaid a grid whose cells were 100m x 100m each — about the size of a standard US city block. All timings were averaged over 100 runs.

We split the data into 2 parts — the first 7 months of data was used as a

[6]Attack and cache location data was provided by the Institute for the Study of War

Algorithm 19 (FIND-BOUNDS)

```
INPUT: Historical, time-stamped observations $\mathcal{O}_h$, historical, time-stamped partners, $\mathcal{E}_h$, real number (distance threshold) $\beta_{max}$
OUTPUT: Real numbers $\alpha, \beta$

1. Set $\alpha = 0$ and $\beta = \beta_{max}$
2. Set Boolean variable $flag$ to TRUE
3. For each $o \in \mathcal{O}_h$, do the following:
    (a) For each $p \in \mathcal{E}_h$ that occurs after $o$, do the following.
        i. Let $d$ be the Euclidean distance function.
        ii. If $flag$, and $d(o,p) \leq \beta_{max}$ then set $\alpha = d(o,p)$ and $\beta = d(o,p)$
        iii. If not $flag$, then do the following:
            A. If $d(o,p) < \alpha$ then set $\alpha = d(o,p)$
            B. If $d(o,p) > \beta$ and $d(o,p) \leq \beta_{max}$ then set $\beta = d(o,p)$
4. Return reals $\alpha, \beta$
```

"training" set and the next 14 months of data was used for experimental evaluation. We used the following simple algorithm, FIND-BOUNDS, to determine the α, β values. We set β_{max} to 2.5 km. We leave more advanced procedures for learning these parameters to future work. Such parameters could also come from an expert.

Accuracy. Our primary goal in the experiments was to determine if the geospatial abduction framework and algorithms could provide viable results in a real-world setting. "Accuracy" in this section refers to two aspects - size of the solution, and

Area	Algorithm	Sample Mean Solution Size	Sample Mean Number of Partners ≤ 0.5 km to actual cache
Baghdad	NAIVE-KSEP-SC	14.53	8.13
	GREEDY-KSEP-OPT1	15.02	7.89
	GREEDY-KSEP-OPT2	14.00	7.49
Sadr City	NAIVE-KSEP-SC	8.00	3.00
	GREEDY-KSEP-OPT1	6.61	4.44
	GREEDY-KSEP-OPT2	6.00	5.28

Table 4.2: k-**SEP** Algorithm Results - Solution Size

the distance to the nearest actual cache site. The distance to nearest cache site was measured by taking the straight-line Euclidean distance to the nearest cache site that was found after the first attack supported by the projected cache site. We used the raw coordinate for the actual cache in the data set - not the position closest to the nearest point in the 100 m resolution grid that we overlaid on the areas. The accuracy results are summarized in Tables 4.2-4.3.

Overall, GREEDY-KSEP-OPT2 consistently found the smallest solution - of cardinality 14 for Baghdad and 6 for Sadr City - on all 100 trials. For Baghdad, the other two algorithms both found a solution of size 14, but both averaged a higher solution. For Sadr City, GREEDY-KSEP-OPT1 often did find a solution of 6 caches while NAIVE-KSEP-SC only found solutions of size 8. Additionally, in both tests, the solution sizes for GREEDY-KSEP-OPT1 varied more than the other two algorithms.

Area	Algorithm	Sample Mean Avg Dist to actual cache	Sample Std Dev of Avg Dist to actual cache	Sample Mean Std Dev of Dist to actual cache
Baghdad	NAIVE-KSEP-SC	0.79 km	0.02	0.64
	GREEDY-KSEP-OPT1	0.76 km	0.07	0.60
	GREEDY-KSEP-OPT2	0.72 km	0.03	0.63
Sadr City	NAIVE-KSEP-SC	0.72 km	0.03	0.46
	GREEDY-KSEP-OPT1	0.45 km	0.03	0.46
	GREEDY-KSEP-OPT2	0.35 km	0.03	0.47

Table 4.3: k-**SEP** Algorithm Results - Distances to Actual Cache Sites

Moreover, the HSD for both Baghdad and Sadr City indicated significant difference between all pairs of algorithms *wrt* solution size.

Of the partners in a given solution, we also recorded the number of partners less than 0.5 km away from an actual cache. For Baghdad, NAIVE-KSEP-SC performed best in this regard - averaging 8.13 partners less than 0.5 km from an actual cache site. Although this result for Baghdad is significant based on an analysis of variance (ANOVA) and honest significant differences (HSD) (p-value of $2.3 \cdot 10^{-9}$), we also note that the greatest difference among averages was still less than one partner. This same result for Sadr City, however, tells a different story. For this test, NAIVE-KSEP-SC performed poorly with regard to the other two algorithms - only finding 3 partners meeting these criteria for each of the 100 trials. GREEDY-KSEP-OPT2 performed very well in this aspect (for Sadr City). It averaged over 5 partners less than 0.5 km from an actual cache. Further, for Sadr City, *all* partners found by GREEDY-KSEP-OPT2 were within 600 m of an actual cache site. The ANOVA (p-value of $2.2 \cdot 10^{-16}$) and HSD of partners less than 0.5 km from an actual cache for the Sadr City trials indicate that these results are significant.

Our primary metric of accuracy was average distance to actual cache. In this regard, GREEDY-KSEP-OPT2 performed the best. It obtained an average distance of 0.72 km for Baghdad and 0.35 km for Sadr City. This number was 40 m less for Baghdad and 100 m less for Sadr City when compared to GREEDY-KSEP-OPT1, whose average distance varied widely among the trials. With regard to this metric, NAIVE-KSEP-SC performed the worst - particularly in Sadr City, where it predicted caches over twice as far from actual caches as GREEDY-KSEP-OPT2 (on average). For both Baghdad and Sadr City, the simple ANOVA yielded a p-value of $2.2 \cdot 10^{-16}$, which suggests with a 99% probability that there is a difference among the algorithms. Also, for both areas, Tukey's HSD indicates significant difference between each pair-wise comparison of algorithms.

Algorithm run times. Table 4.4 shows the run-times of our algorithms. In order to validate the findings suggested by Table 4.4 statistically, we ran analysis of variance (ANOVA) and Tukey's Honest Significant Difference test (HSD) for pair-wise comparisons [50]. An ANOVA for the Baghdad run-times gave a p-value of $2.2 \cdot 10^{-16}$, which suggests with well over 99% probability that GREEDY-KSEP-OPT1 is statistically faster than GREEDY-KSEP-OPT2. The HSD for Baghdad indicates that, with regard to run-times, all pair-wise-comparison of the three algorithms are significantly different. For Sadr City, the ANOVA gave a p-value of $4.9 \cdot 10^{-3}$, which

Area	Algorithm	Sample Mean Run-Time	Sample Run-Time Standard Deviation
Baghdad	NAIVE-KSEP-SC	354.75 ms	12.86
	GREEDY-KSEP-OPT1	162.08 ms	40.83
	GREEDY-KSEP-OPT2	201.40 ms	36.44
Sadr City	NAIVE-KSEP-SC	28.85 ms	10.52
	GREEDY-KSEP-OPT1	25.44 ms	9.33
	GREEDY-KSEP-OPT2	24.64 ms	8.95

Table 4.4: k-**SEP** Algorithm Performance Results

suggests with a 99% probability that the algorithms differ in run-times. However, the HSD indicates, with an 82% probability, that there is no difference among GREEDY-KSEP-OPT1 and GREEDY-KSEP-OPT2, while both differ significantly from NAIVE-KSEP-SC.

4.5.1 A Simple Heuristic to Improve Accuracy

In our implementation of all three algorithms, "ties" in greedy selection of partners were determined by a "coin toss." Specifically, we are considering the case where $this\_size = cur\_size$ in line 4(b)iii of NAIVE-KSEP-SC in Section 4.3.2. Let us re-phrase the situation as follows. Let $\mathcal{O}$ be the entire set of observations and $\mathcal{O}' \subseteq \mathcal{O}$ be the set of observations currently not assigned a partner. Let p be the current partner that best meets the criteria for greedy selection and p' be the partner we are considering. We define P and P' as subsets of $\mathcal{O}$ that are the observations associated with p and p' respectively. Hence, if $|P' \cap \mathcal{O}'| > |P \cap \mathcal{O}'|$, we pick p'. As implemented, if $|P' \cap \mathcal{O}'| = |P \cap \mathcal{O}'|$, we flip a coin. We add a simple heuristic that simply states that "partners that cover more observations are preferred." We change the criteria as follows:

- If $|P' \cap \mathcal{O}'| = |P \cap \mathcal{O}'|$, then do the following:
  - If $|P'| > |P|$, pick p'
  - If $|P| > |P'|$, pick p
  - If $|P| = |P'|$, flip a coin

We shall refer to this as the "tie-breaker" heuristic. The result is that the solution set of partners covers more observations and hence provides a more "dense" solution.

We added this heuristic to our existing code for all three algorithms and ran each one 100 times for both the Baghdad and Sadr City areas. Unsurprisingly, as this is a constant-time operation, run-times were not affected. However, accuracy improved in all cases. As GREEDY-KSEP-OPT2 still provided the most accurate results, the following exposition shall focus on how the heuristics affected the solution size and accuracy for this algorithm.

Because the tie-breaker heuristic only adjusts how two partners are chosen - both of which can be paired with the same uncovered observations - the size of the solution was unaffected in both the Baghdad and Sadr City trials. However, the number of predicted cache sites less than 500 m from an actual site increased for both the Baghdad and Sadr City tests. For Baghdad, more trials returned solutions

Area	Tie-Breaker Heuristic	Sample Mean Solution Size	Sample Mean Number of Partners ≤ 0.5 km to actual cache
Baghdad	No	14.00	7.49
	Yes	14.00	7.87
Sadr City	No	6.00	5.28
	Yes	6.00	6.00

Table 4.5: The Tie-Breaker heuristic on GREEDY-KSEP-OPT2 - Solution Size

Area	Tie-Breaker Heuristic	Sample Mean Avg Dist to actual cache	Sample Std Dev of Avg Dist to actual cache	Sample Mean Std Dev of Dist to actual cache
Baghdad	No	0.72 km	0.03	0.63
	Yes	0.69 km	0.02	0.64
Sadr City	No	0.35 km	0.03	0.47
	Yes	0.28 km	0.02	0.11

Table 4.6: The Tie-Breaker heuristic on GREEDY-KSEP-OPT2 - Distances to Actual Cache Sites

with 8 predictions less than 500 m from an actual site than returned 7 - the opposite being the case without the tie-breaker heuristic. For Sadr City, all elements of every solution set returned was less than 500 m from an actual cache site. Using the well known T-Test [50], we showed that these results are statistically significant as this test returned a p-value of $6.2 \cdot 10^{-8}$ for Baghdad and $2.2 \cdot 10^{-16}$ for Sadr City.

Summary. The above experiments demonstrate statistically that GREEDY-KSEP-OPT2 provides a viable solution - consistently producing the smaller solution sets which were closer to actual cache sites faster than the basic set-covering approach, at times approaching the faster, although less-accurate GREEDY-KSEP-OPT1. The proximity of the elements of the solution set to actual cache sites is encouraging for real-world use. The results are strong enough that two US Army units used SCARE to aide in locating IED caches.

4.6 Chapter 4 Related Work

In this section we present related work of three different varieties. We compare GAPs to other forms of abduction, facility location, *k-means* clustering, and constrained clustering. As an aside, readers interested in a discussion of the SCARE software from the perspective of military analysis or social science can refer to [157] where the software package was introduced. However, that work does not include any formal technical details on the framework of geospatial abduction, complexity results, or algorithm analysis.

GAPs and other forms of Abduction. Abduction [137] has been extensively studied in medicine [141, 138], fault diagnosis [26], belief revision [133], database updates [77, 27] and AI planning [37]. Two major existing theories of abduction include logic-based abduction [41] and set-covering abduction [19]. Though none of the above papers deals with spatial inference, [160] presents a logical formalism dealing with objects' spatial occupancy, while [149] describe the construction of a qualitative spatial reasoning system based on sensor data from a mobile robot. In [149], sensor data are explained by hypothesizing the existence of physical objects along with the dynamic relationships that hold between them, all with respect to a (possibly moving) viewpoint. This approach combines both space and time. [90] describes the *Spatial Semantic Hierarchy* which formalizes, the spatial context in which a robot moves. In the hierarchy, the topological level defines a map which describes the environment as a collection of places, paths, and regions, linked by topological relations such as connectivity, order, containment, boundary, and abstraction. Places (i.e., zero-dimensional points), paths (i.e., one dimensional subspaces, denoting for example a street in a city, possibly represented as an ordering relation on the places they contain), and boundary regions (i.e., two-dimensional subspaces of the robot environment) are created from experience represented as a sequence of views and actions. They are created by abduction, positing the minimal additional set of places, paths, and regions required to explain the sequence of observed views and actions.

Set-covering abduction [19] assumes the existence of a function determining the observable effects of a set of hypotheses, and is based on inverting such function. Given a set of hypotheses H and a set of observations O, the domain knowledge is represented by a function e that takes as an argument a set of hypotheses and gives as a result the corresponding set of observations. Thus, for every subset of the hypotheses $H' \subseteq H$, their effects are known to be $e(H')$. In this case, abduction finds a set $H' \subseteq H$ such that $O \subseteq e(H')$, that is, it finds a set of hypotheses H' whose effects $e(H')$ include all observations in O. A common assumption is that the effects of the hypotheses are independent, that is, for every $H' \subseteq H$, it holds that $e(H') = \bigcup_{h \in H'} e(\{h\})$. If this condition is met, abduction can be seen as a form of set-covering. No spatial reasoning is done here.

Comparison with facility location. There are several important ways in which GAPs differ from facility location problems.

- Although it is possible to specify a distance-based cost function, in a GAP problem, the distances between observations and partners are constraints (α and β in this chapter) whereas facility location problems usually attempt to minimize the distance between producers and consumers.

- In this chapter, GAP problems have a minimum distance between observations and partners that must be exceeded. In many respects, this requirement makes GAP problems more difficult than facility location and other computational geometry problems as the set of possible partners that cover a given observation is a non-convex ring. Further, the feasibility function (feas) adds non-uniform holes to such a ring. [115] addresses the complexity of non-convex covering and highlights issues with problems such as this.

- The feasibility predicate, feas is not part of a facility location problem. This gives us the ability to restrict certain locations that can be partners.

- In general, the relation between observations and partners can be viewed to be a set of constraints. In this chapter, we only used α, β,and feas. However, in the future, we could add additional constraints. Further, as our formalism represents space as a set of discrete points (also not typically done with facility location), we can easily specify certain properties of these points to apply such constraints (such as feas).

Comparsion with *k*-means clustering. A well-known and studied problem in clustering location is the *k-means* problem [116]. This problem can be expressed as follows:

k-means:

INPUT: Coordinates on a plane C and natural number k

OUTPUT: k disjoint sets of C, $C'_1, \ldots, C'_k$ such that for each C_i, all the mean Euclidean distance among all $c \in C_i$ is minimized.

Clustering problems group points into clusters, associating each cluster with a center. At first glance, one may think that the points are equivalent to observations and the "centers are equivalent to partners. However, this is not so. Most versions of the clustering problem seek only to arrange points in groups – with "centers" being a side-effect of the algorithm. *Geospatial abduction problem seeks to find partners that support observations and places constraints on the location of the partners -*

this is a key difference from "centers" in clustering problems. Clustering algorithms cannot handle the generality of our feasibility predicate or the (α, β) constraints.

In addition to these obvious differences, we experimentally compared an implementation of *k-means* with GREEDY-KSEP-OPT2 on the Sadr City data. Even when we ignore the obvious value of α, β and the feasibility predicate, GREEDY-KSEP-OPT2 outperforms the SimpleKMeans solver in WEKA version 3.7.0 [180]. Note that the exclusion of these parameters makes GREEDY-KSEP-OPT2 perform worse than it performs with these parameters – yet, it performed better than k-means in terms of accuracy. Our experiment was set-up as follows:

- We used the same area for the Sadr City tests, as the α value was 0 in these tests and there were virtually no non-feasible points near the observations. This allowed us to use WEKA's k-means implementation "out-of-the-box" as we did not have to implement any extra infrastructure to deal with feasibility and $\alpha = 0$.

- We set $k = 6$, the number of partners consistently found by GREEDY-KSEP-OPT2. Normally, we would rather have the algorithm determine this size. Note that supplying the algorithm with a size already determined by GREEDY-KSEP-OPT2 (and, also the smallest size of any explanation for Sadr City we found in our trials) gives an advantage to *k-means*. Hence, we did not compare solution sizes.

- We clustered the observations with *k-means* and considered the "center" of each cluster the cache location for the cluster.

- We did not compare timing results, as we ran WEKA in its GUI environment.

We ran 500 iterations of the SimpleKMeans and worked with the average centers for the clusters as reported by WEKA. Multiple runs of the 500 iterations yielded the same centers.

Average Distance Using WEKA, we obtained an average accuracy of 0.38 km, which is worse than GREEDY-KSEP-OPT2 (average over 100 trials, 0.28 km).

Worst-Case Distance WEKA's SimpleKMeans returned 2 of the 6 points with a distance of greater than 600 meters from a cache site. Without the "tie-breaking" heuristic, GREEDY-KSEP-OPT2 never reported a prediction over 600 meters from a cache site (all reported partners over 100 trials). With the heuristic, GREEDY-KSEP-OPT2 never reported a prediction over 500 meters from a cache site.

Best-Case Distance The closest partners ever returned by GREEDY-KSEP-OPT2 (either with our without the heuristic) were 200 m away from an actual cache site (on average, the closest partner per explanation was 220 m away). WEKA's SimpleKMeans did return two partners less than 200 m - each only 100 m away from an actual cache site.

These results suggest that *k-means* may not be the optimal method for GAP problems. Further, it does not support feasibility and α. The results do hold some promise for some variants of cost-based spatial explanation problems that require a k input from one of our greedy-approaches. However, even in this case, there would be modification required of the *k-means* algorithm to support feasibility and α.

Comparison with Constrained clustering. *Constrained clustering* [176] studies clustering where, in addition to the points to be clustered, there are constraints that either force two points in the same cluster (must-link) or force two points to be in different clusters (cannot-link). Later work on constrained clustering has focused on distance constraints between elements of C or distance constraints between clusters [32]. Much of the work in this area is summarized in [14].

At first glance, it may appear that spatial abduction can be expressed as a cannot-link constrained clustering problem as follows: For each $o, o' \in \mathcal{O}$ if $\nexists p \in \mathcal{S}$ s.t. $d(o,p) \in [\alpha, \beta]$, $d(o',p) \in [\alpha, \beta]$, and $\mathsf{feas}(p)$, then create a cannot-link constraint for o, o'.

However, such a mapping cannot be guaranteed to provide a correct result. For example, take o_1, o_2, o_3 and p_{12}, p_{23}, p_{13}. Suppose o_1 and o_2 share just partner p_{12}, o_2 and o_3 share just partner p_{23} and o_1, o_3 share just partner p_{13}. This is entirely possible given the generality of feas. In such a case, all three observations could be incorrectly grouped into a single cluster - although it is obvious there is no single partner that supports all of them. Hence, such a mapping would not be trivial. Further, most clustering algorithms are not seeking to constructively find centers that are constrained. We leave the study of constrained clustering to solve GAP problems (i.e. an adaption of the k-means algorithm) to future work. However, it is also worth noting that solving constrained clustering problems given cannot-link constraints is NP-complete, so the application of clustering techniques to this problem does not imply a more tractable version of geospatial abduction, but rather an alternative heuristic.

4.7 Chapter Summary

There are a wide variety of problems where we can make geo-located observations "on the ground" and where we want to infer a partner location. In this chapter, we have presented four examples of such problems — one dealing with serial killers, another dealing with wildlife studies, and a third (perhaps more fun) application dealing with finding sunken ships. A fourth real world application we have looked at is that of finding weapons caches associated with Improvised Explosive Device (IED) attacks in Iraq where we were able to use real world, open source data. It is clear that many other applications exist as well. For example, a bizarre (but real world) combination of two of our examples involves frequent attacks by man-eating leopards on children in certain parts of greater Bombay in India. This situation is analogous to the serial killer scenario where the leopard is the serial killer. We want to find the leopard's favorite "hang outs", capture it, and solve the problem.

In this chapter, we have made an attempt to formally define a class of *geospatial abduction problems* (GAPs for short). We specifically made the following contributions.

- We developed formal mathematical definitions of geospatial abduction problems, including several variants of the above problems. We conducted a detailed analysis of the complexity of these problems.

- We developed exact algorithms for many of the problems, including a straightforward enumeration approach (NAIVE-KSEP-EXACT), by showing and leveraging reductions to both the set-covering and dominating set problems, and by articulating these geospatial abduction problems via integer linear programs.

- As the complexity of most of the problems we have studied is NP-hard, we developed two greedy approximation schemes for the k-**SEP** problem (other than set-covering) and illustrated a scheme to quickly find a solution using randomized approaches to the dominating set problem.

- We have implemented these algorithms and conducted experimental comparisons of the reduction to set-covering and two other greedy approaches - GREEDY-KSEP-OPT1 and GREEDY-KSEP-OPT2. Both of these algorithms outperformed the set-covering reduction in an experiment on the *Understanding War Special Groups* data set. We also implemented a "tie-breaker" heuristic that further improved the accuracy of the algorithms.

- We have also developed approximation schemes using relaxations of the linear-integer program for k-**SEP** and the cost-based variant **WT-SEP**.

There are many interesting directions for future work. For example, spatial abduction in dimensions greater than two might be explored. A probabilistic variant might replace the feasibility predicate with a probability distribution function, or express the relationship between observations and partners as a PDF based on distance rather than rely on α, β. Also, the use of randomization in the approximation algorithms may improve results for both the greedy and linear programming approaches presented in this chapter.

One aspect to explore in future work is the relationship between observations and partners. k-**SEP** and its cost based variants only rely on α, β. However, many applications may have other constraints. Perhaps there is a direction associated with each observation (as in identifying where an artillery round originated from), which would limit the locations of the partner. Another possibility is to add geographic constraints. Perhaps the observation cannot have a partner across a body of water, or beyond the edge of a cliff.

Another important question is: where do we look for partners if they are placed they are placed by an adversary? We can think of scenarios, such as in counterinsurgency, where an enemy obtains a copy of our software and wants to plan his cache sites in a place where an agent would be unlikely to search for them. We study this particular problem in Chapter 6. Another natural question is: what if we want to abduce regions rather than point locations for partners? There are many real-world applications where a user may wish to find an area to search rather than a point - in fields varying from paleontology to intelligence. We describe this extension to the geospatial abduction framework in chapter 5.

Chapter 5

Abducing Regions

In the previous chapter, we studied a variety of geospatial problems where the space is represented as a plane that used discrete integer coordinates. In this chapter, we modify the framework to use a continuous space instead. Additionally, rather than abducing points, we assume the space is divided into a set of regions, and we wish to abduce a set of regions that explains the agent's behavior.[1]

5.1 Chapter Introduction

In this chapter, we introduce a variant GAPs called *region-based geospatial abduction problems* (RGAPs). In RGAPs, we are given a map, a set $\mathcal{O}$ of observations, and a set of subregions of the map (this could include all subregions of the map in the worst case or can be defined by some logical condition). We want to find a set of regions that best "explain" the observations and includes, for each observation, at least one partner.

[1]This chapter is based on [153] which was completed in cooperation with V.S. Subrahmanian.

In this chapter, we make several contributions. In Section 5.2 we introduce multiple possible formal definitions of RGAPs - including cases where the regions are determined by a given radius from each observation, regions are non-convex, and when regions are of irregular shape due to terrain restrictions. We then perform a detailed complexity analysis in Section 5.3, proving that most of these problems are NP-complete. This leads us to use approximation techniques in Section 5.4. We also describe some practical implementation issues. Section 5.5 describes our implementation and includes an experimental evaluation on a real-world data-set consisting of IED attacks in Baghdad, Iraq and related weapons cache sites. In our evaluation, regions outputted by the algorithm contained, on average, 1.7 *cache sites, with an average cache density of over* 8 *caches per square kilometer – significantly higher than the city-wide average of* 0.4. Further, the algorithm ran quickly, performing computation in just over 2 seconds on commodity desktop hardware. Finally, we survey related work in Section 5.6.

5.2 Technical Preliminaries

To address the problem of region-based geospatial abduction, we introduce a framework that resembles that of Chapter 4 - but differs in several important aspects. These include the use of a continuous space and multiple types of explanations. In Chapter 6, we return to the original framework of Chapter 4.

We assume the existence of a real-valued $M \times N$ space $\mathcal{S}$ whose elements are pairs of real numbers from the set $[0, M] \times [0, N]$. An observation is any member of $\mathcal{S}$. We use $\mathcal{O}$ to denote an arbitrary, but fixed, finite set of *observations*. We assume there are real numbers $\alpha \leq \beta$ such that for each observation o , there exists a partner p_o (to be found) whose distance from o is in the interval $[\alpha, \beta]$.[2] Without loss of generality, we also assume that all elements of $\mathcal{O}$ are over β distance away from the edge of $\mathcal{S}$. Example 5.2.1 presents a neighborhood as a space and locations of illegal drug sales as observations.

Example 5.2.1 (Illegal Drug Sales)**.** *A criminal gang is selling illegal drugs. Consider the space $\mathcal{S}$ depicted in Figure 5.1. Drug dealers were arrested by police at points $\mathcal{O} \equiv \{o_1, \ldots, o_{13}\}$. Historical data suggests that safe houses are located within $5km$ of such transactions (i.e. $\alpha = 0$ and $\beta = 5km$). Note that in Figure 5.1, circles of radius $5km$ are drawn around the observation points. Police are interested in locating such safe-houses.*

Throughout this chapter, we assume the notion of a *distance function* d on $\mathcal{S}$ satisfying the usual properties of such distance functions.[3] We now define a region and how they relate to the set of observations. Our intuition is simple - a region *explains* an observation if that region contains a partner point for that observation.

Definition 57 (Region / Super-Explanation / Sub-Explanation)**.** *A region r is a subset of $\mathcal{S}$ such that for any two points $(x, y), (x', y') \in r$, there is sequence a of line segments from (x, y) to (x', y') s.t. no line segment lies outside r.*

[2]Chapter 4 describes methods to learn α, β automatically from historical data.

[3]$d(x, x) = 0; d(x, y) = d(y, x); d(x, y) + d(y, z) \geq d(x, z)$.

1. *A region r **super-explains** point o in $\mathcal{S}$ iff there exists a point $p \in r$ such that $d(o,p) \in [\alpha, \beta]$.*

2. *A region r **sub-explains** some point o in $\mathcal{S}$ iff $(\forall p \in r)$ $d(o,p) \in [\alpha, \beta]$.*

First, note that regions can have any shape and may overlap. Throughout this chapter, we assume that checking if some point o is sub-(super-) explained by region r can be performed in constant (i.e. $O(1)$) time. This is a reasonable assumption for most regular shaped regions like circles, ellipses and polygons. The following result follows immediately from Definition 57.

Observation 5.2.1. *If region $r \neq \emptyset$ sub-explains point o, then r super-explains point o.*

We would like to explain observations by finding regions containing a partner. In some applications, the user may be able to easily search the entire region – hence a super-explaining region would suffice. In other applications, we may want to be sure that any point within the region can be a partner as not to waste resources - so only a sub-explanation would make sense in such a case. Often, these situations may depend on the size of the regions. We shall discuss the issue of restricting region size later in this section. For now, we shall consider regions any shape or size. Example 5.2.2 shows regions that super- or sub-explain various observations.

Example 5.2.2. *Consider the scenario from Example 5.2.1 and the regions $R = \{r_a, r_b, r_c, r_d, r_e, r_f, r_g\}$ shown in Figure 5.1. Suppose these regions correspond with "support zones" for the drug sales – i.e. places that may contain a safe-house.*

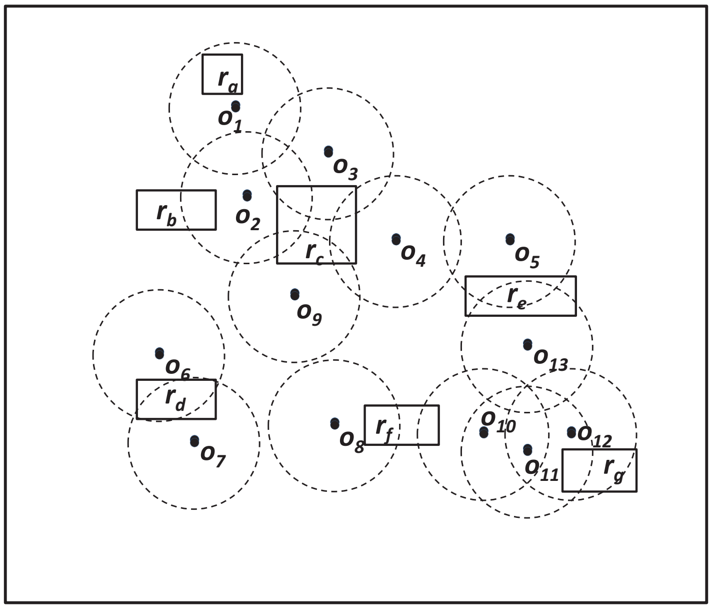


Figure 5.1: Locations of illegal drug sales and suspected support zones $\{r_a, r_b, r_c, r_d, r_e, r_f, r_g\}$. The β distance for each observation is shown with a dashed circle.

Consider region r_a. As it totally lies within the α, β distance of o_1, it sub- and super-explains this observation. Conversely, region r_d super-explains both o_6 and o_7 but sub-explains neither.

This chapter studies following decision problems.

Sub-(Super-)Region Explanation Problem (Sub/Sup-REP)

INPUT: Given a space $\mathcal{S}$, distance interval $[\alpha, \beta]$, set $\mathcal{O}$ of observations, set R of regions, and natural number $k \in [1, |\mathcal{O}|]$.

OUTPUT: Set $R' \subseteq R$, where $|R'| \leq k$ and for each $o \in \mathcal{O}$, there is an $r \in R$ s.t. r sub-(super-) explains o.

The fact that a set R of regions is part of the input is not an assumption, but a feature. A user might set R to be all the regions associated with $\mathcal{S}$; alternatively, he might use a logical condition to define regions, taking into account, the terrain and/or known aspects of the population living in the area of interest. For instance, when trying to identify regions containing IED caches in Baghdad used for attacks by Shi'ite groups, regions were defined to be places that were not predominantly Sunni and that did not contain US bases or bodies of water. Other kinds of logical conditions may be used when dealing with burglaries or drug trafficking. Thus, the set R of regions allows an analyst to specify any knowledge he has, and allows the system to benefit from that knowledge. If no such knowledge is available, R can be taken to be the set of all regions associated with $\mathcal{S}$. R can also be used to restrict the size of the region (e.g. only considering regions whose area is less than 5 sq. km.).

There are two different associated optimization problems associated with both Sub-REP and Sup-REP. The first deals with finding a subset of regions of minimal cardinality that explains all observations.

Sub-(Super-)Region Explanation Problem-Minimum Cardinality (Sub/Sup-REP-MC)

INPUT: Given a space, $\mathcal{S}$, distance interval $[\alpha, \beta]$, set of observations $\mathcal{O}$, and set of regions R.

OUTPUT: Set $R' \subseteq R$ of minimum cardinality, where for each $o \in \mathcal{O}$, there is an $r \in R$ s.t. r sub-(super-) explains o.

Our second optimization problem fixes the number of regions returned in the solution, but maximizes the number of observations that are explained.

Sub-(Super-)Region Explanation Problem-Maximum Explaining (Sub/Sup-REP-ME)

INPUT: Given a space $\mathcal{S}$, distance interval $[\alpha, \beta]$, set $\mathcal{O}$ of observations, set R of regions, and natural number $k \in [1, |\mathcal{O}|]$.

OUTPUT: Set $R' \subseteq R$, where $|R'| \leq k$ s.t. the number of $o \in \mathcal{O}$ where there is an $r \in R$ s.t. r sub-(super-) explains o is maximized.

Consider the following Example.

Example 5.2.3. *Consider the scenario from Example 5.2.2. Consider an instance of Sup-REP with $k = 7$. The set $\{r_a, r_b, r_c, r_d, r_e, r_f, r_g\}$ is a solution to this problem. Now consider Sup-REP-MC with $k = 6$, the set $\{r_a, r_c, r_d, r_e, r_f, r_g\}$ is a solution to this problem. Finally, consider Sup-REP-ME with $k = 2$. The set $\{r_c, r_d\}$ is a solution to this problem.*

We now consider a special case of these problems that arises when the set R of regions is created by a partition of the space based on the set of observations ($\mathcal{O}$) and concentric circles of radii α and β drawn around each $o \in \mathcal{O}$. We can associate regions in such a case with subsets of $\mathcal{O}$. For a given subset $\mathcal{O}'$, we say that there is an associated set of *induced regions* (denoted $R_{\mathcal{O}'}$), defined as follows:

$$R_{\mathcal{O}'} = \{\{x| \quad \forall o \in \mathcal{O}', d(x,o) \in [\alpha,\beta] \wedge \\ \forall o' \notin \mathcal{O}', d(x,o') \notin [\alpha,\beta]\} \quad \}$$

We note that for a given subset of observations, it is possible to have a set of induce regions, $R_{\mathcal{O}'}$ that has more than one element. For example, consider set $R_\emptyset = \{r_1, r_{12}\}$ in Figure 5.2. For a given set of observations $\mathcal{O}$, we will use the notation $R_\mathcal{O}$ do denote the set of all induce regions. Formally:

$$R_\mathcal{O} = \bigcup_{\substack{\mathcal{O}' \in 2^\mathcal{O} \\ R_{\mathcal{O}'} \neq \emptyset}} R_{\mathcal{O}'}$$

We illustrate the idea of induce regions in the following example.

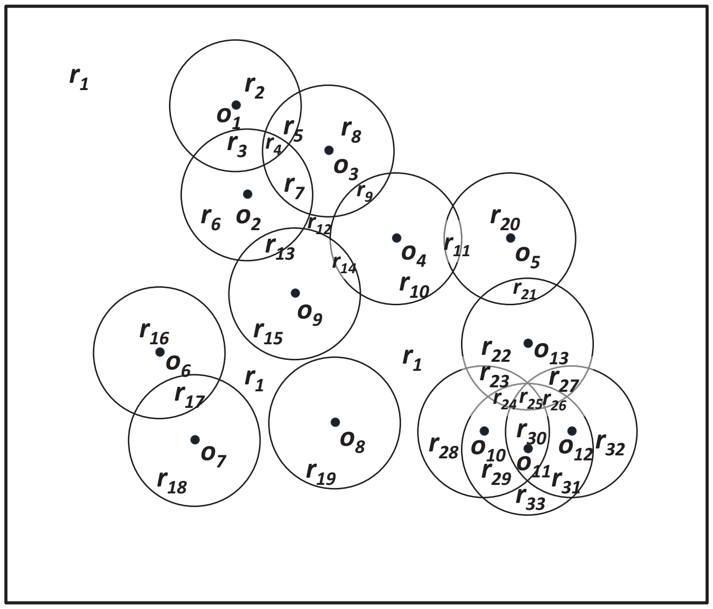


Figure 5.2: Space $\mathcal{S}$ and the regions in set $R_{\mathcal{O}}$.

Example 5.2.4. *In order to identify locations of drug safe-houses, police create* 33 ***induced*** *regions in* $\mathcal{S}$ *by drawing* 5*km radius circles around all observations (see Figure 5.2), the set of which is denoted* $R_{\mathcal{O}} = \{r_1, \ldots, r_{33}\}$.

For the special case where $R_{\mathcal{O}}$ is the set of regions, we have the following result.

Lemma 17. *Suppose* $\mathcal{O}$ *is a set of observation and* $R_{\mathcal{O}}$ *is the induced region. A region* $r \in R_{\mathcal{O}}$ *sub-explains an observation* $o \in \mathcal{O}$ *iff it super-explains* o.

By this result, for the special case of induced regions, we only need one decision problem.

Induced Region Explanation Problem (I-REP)

INPUT: Given a space, $\mathcal{S}$, distance interval $[\alpha, \beta]$, set $\mathcal{O}$ of observations, and natural number $k \in [1, |\mathcal{O}|]$.

OUTPUT: Set $R' \subseteq R_{\mathcal{O}}$, where $|R'| \leq k$ and for each $o \in \mathcal{O}$, there is an $r \in R$ s.t. r sub-explains o.

As mentioned earlier, the sizes of regions can be regulated by our choice of R. However, we may also explicitly require that all regions must be less than a certain area. Consider the following variant of Sup-REP.

Area-Constrained Super-Region Explanation Problem (AC-Sup-REP)

INPUT: Given a space, $\mathcal{S}$, distance interval $[\alpha, \beta]$, set $\mathcal{O}$ of observations, set R of regions, area A, and natural number $k \in [1, |\mathcal{O}|]$.

OUTPUT: Set $R' \subseteq R$, where $|R'| \leq k$ and each $r \in R'$ has an area $\leq A$ and for each $o \in \mathcal{O}$, there is an $r \in R$ s.t. r super-explains o.

The following proposition tells us that AC-Sup-REP is at least as hard as I-REP, yet no harder than Sup-REP (an analogous result can easily be shown for an area-constrained version of Sub-REP). We note that essentially, we eliminate the regions whose area is above area A, which gives us an instance of Sup-REP. To go the other direction, we directly encode I-REP into an instance of AC-Sup-REP and have A be larger than the area of any region.

Theorem 24. *I-REP is polynomially reducible to AC-Sup-REP.*

AC-Sup-REP is polynomially reducible to Sup-REP.

In our final observation of this section, we note that the set $R_{\mathcal{O}}$ can be used as a "starting point" in determining regions. For instance, supplemental information on area that may be restricted from being partnered with an observation may also be considered and reduce the area of (or eliminate altogether) some regions in the set. Consider the following example.

Example 5.2.5. *Consider the scenario from Example 5.2.4. The police may eliminate a river running through the area and certain other ares from their search. These "restricted areas" are depicted in Figure 5.3. Note that several regions from Figure 5.2 are either eliminated or have decreased in size. However, by eliminating these areas, the police have also pruned some possibilities from their search. For example, regions r_9, r_{13} were totally eliminated from consideration.*

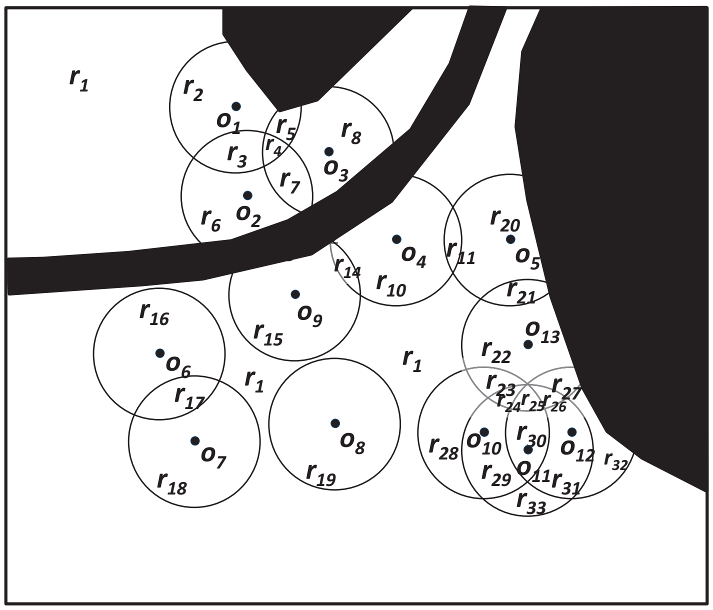


Figure 5.3: A set of regions in $\mathcal{S}$ created based on the distance $\beta = 5km$ as well as restricted areas (shown in black).

5.3 Complexity

In this section, we show that Sub-REP, Sup-REP, and I-REP are NP-Complete and that the associated optimization problems are NP-Hard. We also show that the optimization problems Sub-REP-MC, Sup-REP-MC, and I-REP-MC cannot be approximated by a fully polynomial-time approximation scheme (FPTAS) unless $P = NP$. We also note that the complexity of the area-constrained versions of these problems follows directly from the results of this section by the reduction of Theorem 24 (page 219).

We first prove that I-REP is NP-complete, which then allows us to correctly identify the complexity classes of the other problems by leveraging Lemma 17. First, we introduce the problem "circle covering" (CC) that was proven to be NP-complete in [125].

Circle Covering (CC)

INPUT: A space $\mathcal{S}'$, set P of points, real number β', natural number k'.

OUTPUT: "Yes" if there is a set of points, Q in $\mathcal{S}'$ such that all points in P are covered by discs centered on points in Q of radius β' where $|Q| \leq k'$ – "no" otherwise.

The theorem below establishes that I-REP is NP-complete.

Theorem 25. *I-REP is NP-Complete.*

Proof Sketch. *Clearly, a solution to I-REP can be verified in PTIME. To show NP-hardness, we show that CC $\leq_p$ I-REP by the following transformation: $\mathcal{S} = \mathcal{S}'$, $\mathcal{O} = P$, $\beta = \beta'$, $\alpha = 0$, and $k = k'$. ($\Leftarrow$) Given a solution to the instance of I-REP, we can simply pick a point in each returned region, and center a circle on it of radius β' - which will be a solution to CC. Likewise, ($\Rightarrow$) given a solution to CC, we can be assured that each point in the solution is enclosed by exactly one region from the set $R_{\mathcal{O}}$, which would ensure a solution to I-REP.* ■

Further, as the optimization version of circle covering is known to have no FPTAS unless $P = NP$ [70], by the nature of the construction in Theorem 25, we can be assured of the same result for I-REP-MC.

Corollary 8. *I-REP-MC cannot be approximated by a fully polynomial-time ap-*

proximation scheme (FPTAS) unless $P = NP$.

So, from the above Theorem and Corollary and Lemma 17, we get the following results:

Corollary 9. *1. Sub-REP and Sup-REP are NP-Complete.*

2. Sub-REP-MC, Sup-REP-MC, I-REP-MC, Sub-REP-ME, Sup-REP-ME, and I-REP-ME are NP-Hard.

3. Sub-REP-MC, Sup-REP-MC cannot be approximated by a FPTAS unless $P = NP$.

5.4 Algorithms

In this section we devise algorithms to address the optimization problems associated with Sup-REP, Sub-REP, and I-REP. First, we show that these optimization problems reduce to either instances of set-cover (for Sub/Sup-REP-MC) or max-k-cover (for Sub/Sup-REP-ME). These problems are well-studied and there are algorithms that provide exact and approximate solutions. We then provide a new greedy-algorithm for Sub/Sup-REP-MC that also provides an approximation guarantee. This is followed by a discussion of approximation for I-REP-ME for the case where $\alpha = 0$. Finally, we discuss some practical issues dealing with implementation.

5.4.1 Exact and Approximate Solutions by Reduction

In this section we show that the -MC problems can reduce to set-cover and that the -ME problem can reduce to max-k-cover. First, we introduce the two problems in question. First, we present set-cover [136].

Set-Cover

INPUT: Set of elements S, family of subsets of S, $\mathcal{H} = H_1, \ldots, H_m$.

OUTPUT: Subset $\mathcal{H}' \subseteq \mathcal{H}$ of minimum cardinality s.t. $\bigcup_{H_i \in \mathcal{H}'} H_i \supseteq S$.

Next, we present max-k-cover [46], which is often regarded as the dual of set-cover:

Max-k-Cover

INPUT: Set of elements S, family of subsets of S, $\mathcal{H} = H_1, \ldots, H_m$, natural number $k \leq |S|$.

OUTPUT: Subset $\mathcal{H}' \subseteq \mathcal{H}$ s.t. $|\mathcal{H}'| \leq k$ where $|\bigcup_{H_i \in \mathcal{H}'} H_i \cap S|$ is maximized.

The key to showing that Sub/Sup-REP optimization problems can reduce to one of these problems is to determine the family of subsets. We accomplish this as follows: for each region $r \in R$, we find the subset of $\mathcal{O}$ that can be partnered with r. We shall refer to this set as $\mathcal{O}_r$. This gives us the following algorithm for the optimization problems (we simply omit the k parameter for the -MC problems that reduce to Set-Cover):

```
REDUCE-TO-COVERING($\mathcal{O}$ set of observations, $R$ set of regions, $k$ natural number)
returns instance of covering problem $\langle S, \mathcal{H}, k \rangle$

    1. For each $r \in R$, find $\mathcal{O}_r$ (i.e. $o$ is in $\mathcal{O}_r$ iff $r$ sub/super-explains $o$)

    2. Return $\langle \mathcal{O}, \bigcup_{r \in R}\{\mathcal{O}_r\}, k \rangle$
```

Proposition 42. *REDUCE-TO-COVERING requires $O(|\mathcal{O}| \cdot |R|)$ time.*

The following theorem shows that REDUCE-TO-COVERING correctly reduces a Sub/Sup-REP optimization problem to set-cover or max-k-cover as appropriate.

Theorem 26. *Sub/Sup-REP-MC polynomially reduces to Set-Cover and Sub/Sup-REP-ME polynomially reduces to Max-k-Cover.*

This result allows us to leverage any exact approach to the above optimization problems to obtain a solution to an optimization problem associated with Sub/Sup-REP. A straightforward algorithm for any of the optimization problems would run in time exponential in $|\mathcal{O}|$ or k and consider every $|\mathcal{O}|$ or k sized subset of $\bigcup_{r \in R}\{\mathcal{O}_r\}$. Clearly this is not practical for real-world applications. Fortunately, there are several well-known approximation techniques for both these problems. First, we address the Sub/Sup-REP-ME problems, which reduce to Max-k-Cover. As the Max-k-Cover problem reduces to the maximization of a submodular function over a uniform matroid, we can leverage the greedy approximation algorithm of [127] to our problem. We do so below.

Suppose 'f' denotes the maximum number of observations that can be partnered with a given region.

```
GREEDY-REP-ME($\mathcal{O}$ set of observations, $R$ set of regions, $k$ natural number)
returns $R' \subseteq R$

1. Let $\mathbf{O} = \bigcup_{r \in R}\{\mathcal{O}_r\}$ (obtained by REDUCE-TO-COVERING)
2. Let $\mathcal{O}' = \mathcal{O}$, set $R' = \emptyset$
3. While $k \neq 0$ loop
    (a) Let the element $\mathcal{O}_r$ be the member of $\mathbf{O}$ s.t. $|\mathcal{O}_r \cap \mathcal{O}'|$ is maximized.
        $R' = R' \cup r$
        $\mathcal{O}' = \mathcal{O}' - (\mathcal{O}_r \cap \mathcal{O}')$
        $k - -$
4. Return $R'$
```

Proposition 43. *GREEDY-REP-ME runs in $O(k \cdot |R| \cdot f)$ time and returns a solution such that the number of observations in $\mathcal{O}$ that have a partner region in R' is within a factor $\left(\frac{e}{e-1}\right)$ of optimal.*

Example 5.4.1. *Consider Example 5.2.2 (page 213), where the set of regions is $R = \{r_a, r_b, r_c, r_d, r_e, r_f, r_g\}$. Suppose the police want to run GREEDY-REP-ME to solve an instance of Sup-REP-ME associated with this situation with $k = 3$. Initially set $\mathcal{O}' = \{o_1, \ldots, o_{13}\}$. On the first iteration of the outer loop, it identifies set $\mathcal{O}_{r_c} = \{o_2, o_3, o_4, o_9\}$ where the cardinality of $\mathcal{O}_{r_c} \cap \mathcal{O}'$ is maximum. Hence, it picks region r_c. The set $\mathcal{O}' = \{o_1, o_5, \ldots, o_8, o_{10}, \ldots o_{13}\}$. On the second iteration, it identifies $\mathcal{O}_{r_e} = \{o_5, o_{13}\}$, which intersected with $\mathcal{O}'$ provides a maximum cardinality, causing r_e to be picked. Set $\mathcal{O}'$ is now $\{o_1, o_6, \ldots, o_8, o_{10}, \ldots, o_{12}\}$. On the last iteration,* *it identifies $\mathcal{O}_{r_g} = \{o_{11}, o_{12}\}$, again the maximum cardinality when intersected with $\mathcal{O}'$. The element is picked and the solution is r_c, r_e, r_g, and the observations super-explained are $\{o_2, o_3, o_4, o_5, o_9, o_{11}, o_{12}, o_{13}\}$.*

Likewise, we can leverage the greedy algorithm for set-cover [136] applied to Sub/Sup-REP-MC.

```
GREEDY-REP-MC($\mathcal{O}$ set of observations, $R$ set of regions, ) returns $R' \subseteq R$

  1. Let $\mathbf{O} = \bigcup_{r \in R}\{\mathcal{O}_r\}$ (obtained by REDUCE-TO-COVERING)
  2. Let $\mathcal{O}' = \mathcal{O}$, set $R' = \emptyset$
  3. While not $\mathcal{O}' \equiv \emptyset$ loop
       (a) Let the element $\mathcal{O}_r$ be the member of $\mathbf{O}$ s.t. $|\mathcal{O}_r \cap \mathcal{O}'|$ is maximized.
           $R' = R' \cup r$
           $\mathcal{O}' = \mathcal{O}' - (\mathcal{O}_r \cap \mathcal{O}')$
  4. Return $R'$
```

Proposition 44. *GREEDY-REP-MC runs in $O(|\mathcal{O}| \cdot |R| \cdot f)$ time and returns a solution whose cardinality is within a factor of $1 + \ln(f)$ of optimal.*

Example 5.4.2. *Consider the scenario from Example 5.4.1. To explain all points, the police can create an instance of Sup-REP-MC and use GREEDY-REP-MC. The algorithm proceeds just as GREEDY-REP-ME in the first three steps (as in Example 5.4.1, but will continue on until all observations are super-explained. So, GREEDY-REP-MC proceeds for three more iterations, selecting r_f ($\mathcal{O}_{r_f} = \{o_8, o_{10}\}$),*

r_d ($\mathcal{O}_{r_d} = \{o_6, o_7\}$), and finally r_a ($\mathcal{O}_{r_a} = \{o_1\}$). The solution returned is:

$$\{r_c, r_e, r_g, r_f, r_d, r_a\}$$

We now focus on speeding up the set-cover reduction via the GREEDY-REP-MC2 algorithm below.

In the rest of this section, we use 'Δ' to denote the maximum number of different regions that can be partnered with a given observation.

Proposition 45. *GREEDY-REP-MC2 runs in $O(\Delta \cdot f^2 \cdot |\mathcal{O}| + |\mathcal{O}| \cdot \ln(|\mathcal{O}|)$ time and returns a solution whose cardinality is within a factor of $1 + \ln(f)$ of optimal.*

Although GREEDY-REP-MC2 considers regions in a different order than GREEDY-REP-MC, it maintains the same approximation ratio. This is because the region (in set GRP_o) that is partnered with the greatest number of uncovered observations is selected at each iteration, allowing us to maintain the approximation guarantee. There are two selections at each step: the selection of the observation (in which we use a minimal key value based on related observations) and a greedy selection in the inner loop. Any selection of observations can be used at each step and the approximation guarantee is still maintained. This allows for a variety of different heuristics. Further, the use of a data structure such as a Fibonacci Heap allows us to actually obtain a better time complexity than GREEDY-REP-MC.

Example 5.4.3. *Consider the situation in Example 5.2.4 where the police are considering regions $R_{\mathcal{O}} = \{r_1, \ldots, r_{33}\}$ that are induced by the set of observations and wish to solve I-REP-MC using GREEDY-REP-MC. On the first iteration of the loop*

```
GREEDY-REP-MC2($\mathcal{O}$ set of observations, $R$ set of regions, ) returns $R' \subseteq R$

1. Let $\mathbf{O} = \bigcup_{r \in R}\{\mathcal{O}_r\}$ (obtained by REDUCE-TO-COVERING)
2. For each observation $o \in \mathcal{O}$, let $GRP_o = \{\mathcal{O}_r \in \mathbf{O} | o \in \mathcal{O}_r\}$
3. For each observation $o \in \mathcal{O}$, let $REL_o = \{o' \in \mathcal{O} | o' \in \bigcup_{\mathcal{O}_r \in GRP_o} \mathcal{O}_r\}$ and let
   $key_o = |REL_o|$
4. Let $\mathcal{O}' = \mathcal{O}$, set $R' = \emptyset$
5. While not $\mathcal{O}' \equiv \emptyset$ loop
   (a) Let $o$ be the element of $\mathcal{O}$ where $key_o$ is minimal.
   (b) Let the element $\mathcal{O}_r$ be the member of $GRP_o$ s.t. $|\mathcal{O}_r \cap \mathcal{O}'|$ is maximized.
   (c) If there are more than one set $\mathcal{O}_r$ that meet the criteria of line 5b, pick the set
       w. the greatest cardinality.
   (d) $R' = R' \cup r$
   (e) For each $o' \in \mathcal{O}_r \cap \mathcal{O}'$, do the following:
       i. $\mathcal{O}' = \mathcal{O}' - o'$
       ii. For each $o'' \in \mathcal{O}' \cap REL_{o'}$, $key_{o''} - -$
6. Return $R'$
```

at line 5, the algorithm picks o_8*, as* $key_{o_8} = 1$*. The only possible region to pick is* r_{19}*, which can only be partnered with* o_8*. There are no observations related to* o_8 *other than itself, so it proceeds to the next iteration. It then selects* o_6 *as* $key_{o_6} = 2$ *because* $REL_{o_6} = \{o_6, o_7\}$*. It then greedily picks* r_{17} *which sub-explains both* o_6, o_7*. As all observations related to* o_6 *are now sub-explained, the algorithm proceeds with the next iteration. The observation with the lowest key value is* o_5 *as* $key_{o_5} = 3$ *and* $REL_{o_5} = \{o_4, o_5, o_{13}\}$*. It then greedily picks region* r_{21} *which sub-explains* o_5, o_{13}*. The algorithm then reduces the key value associated with* o_4 *from* 4 *to* 3 *and decrements the keys associated with* o_{10}, o_{11}, o_{12} *(the un-explained observations related to* o_{13}*) also from* 4 *to* 3*. In the next iteration, the algorithm picks* o_9 *as* $key_{o_9} = 3$*. It greedily picks* r_{12} *which sub-explains* o_9, o_2*. It then decreases* key_{o_4} *to* 2 *and also decreases the keys associated with* o_1 *and* o_3*. At the next iteration, it picks* o_1 *as* $key_{o_1} = 2$*. It greedily selects* r_4*, which sub-explains* o_1, o_3 *and decreases the* key_{o_4} *to* 1 *which causes* o_4 *to be selected next, followed by a greedy selection of* r_{11} *– no keys are updated at this iteration. In the final iteration, it selects* o_{10} *as* $key_{o_{10}} = 3$*. It greedily selects* r_{25}*, which sub-explains all un-explained observations. The algorithm terminates and returns* $\{r_{11}, r_{12}, r_{17}, r_{19}, r_{21}, r_{25}\}$*.*

5.4.2 Approximation for a Special Case

In Section 5.3, we showed that circle covering is polynomially reducible to I-REP-MC. Let us consider a special (but natural) case of I-REP-MC where $\alpha = 0$, i.e. there is no minimum distance between an observation and a parter point that caused it. We shall call this special case I-REP-MCZ. There is a great similarity between this problem and circle-covering. It is trivial to modify our earlier complexity proof to obtain the following result.

Corollary 10. *I-REP-MCZ is polynomially reducible to CC.*

Further, we can adopt any algorithm that provides a constructive result for CC to provide a result for I-REP-MCZ in polynomial time with the following algorithm. Given some point p, it identifies the set $\mathcal{O}_r$ associated with the region that encloses that point.

```
FIND-REGION($\mathcal{S}$ space, $\mathcal{O}$ observation set, $\beta$ real, $p$ point) returns set $\mathcal{O}_r$

  1. Set $\mathcal{O}_r = \emptyset$

  2. For each $o \in \mathcal{O}$, if $d(p, o) \leq \beta$ then $\mathcal{O}_r = \mathcal{O}_r \cup \{o\}$

  3. Return $\mathcal{O}_r$.
```

Proposition 46. *The algorithm, FIND-REGION runs $O(|\mathcal{O}|)$ time, and region r (associated with the returned set $\mathcal{O}_r$) contains p.*

By pre-processing the regions, we can compute $\mathcal{O}_r$ a-priori and simply pick a region r associated with the output for FIND-REGION. While there may be more than one such region, any one can be selected as, by definition, they would support the same observations.

Example 5.4.4. *Paleontologists working in a $30 \times 26km$ area represented by space $\mathcal{S}$ have located scattered fossils of prehistoric vegetation at $\mathcal{O} = \{o_1, o_2, o_3, o_4\}$. Pre-*

vious experience has led the paleontologists to believe that a fossil site will be within 3km of the scattered fossils. In Figure 5.4, the observations are labeled and circles with radius of 3 *km are drawn (shown with solid lines). Induced regions* $r_1, \ldots, r_6$ *are also labeled. As the paleontologists have no additional information, and* $\alpha = 0$*, they can model their problem as an instance of I-REP-MCZ with* $k = 3$*. They can solve this problem by reducing it to an instance of circle-covering. The circle-covering algorithm returns three points -* p_1, p_2, p_3 *(marked with an 'x' in Figure 5.4). Note that each point in the solution to circle-covering falls in exactly one region (when using induced regions). The algorithm* ***FIND-REGION*** *returns the set* $\{o_1, o_2\}$ *for point* p_1*, which corresponds with region* r_2*. It returns set* $\{o_3\}$ *for* p_2*, corresponding with* r_6 *and returns set* $\{o_4\}$ *for* p_3*, corresponding with* r_5*. Hence, the algorithm returns regions* r_2, r_6, r_5*, which explains all observations.*

Any algorithm that provides a constructive result for CC can provide a constructive result for I-REP-MCZ. Because of this one-to-one mapping between the problems, we can also be assured that we maintain an approximation ratio of any approximation technique.

Corollary 11. *An* $a-$*approximation algorithm for CC is an* a*-approximation for I-REP-MCZ.*

This is useful as we can now use approximation algorithms for CC on I-REP-MCZ. Perhaps the most popular approximation algorithms for CC are based on the "shifting strategy" [70]. To leverage this strategy, we would divide the space, $\mathcal{S}$, into strips of width $2 \cdot \beta$. The algorithm considers groups of ℓ consecutive strips – ℓ

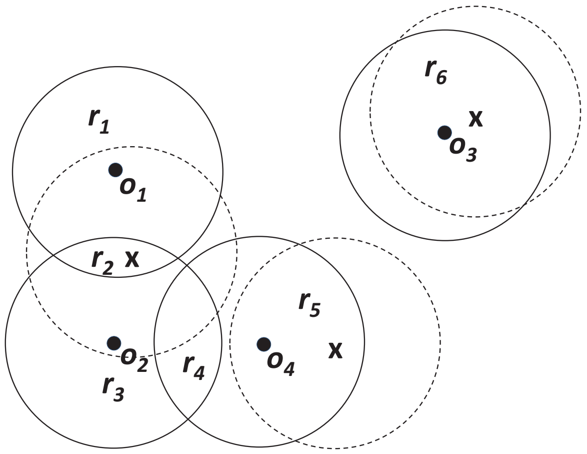


Figure 5.4: Given the instance of I-REP-MCZ for Example 5.4.4 as input for circle-covering, a circle-covering algorithm returns points p_1, p_2, p_3 (points are denoted with an "x", dashed circles are the area of $3km$ from the point).

is called the "shifting parameter." A local algorithm **A** is applied to each group of strips. The union of all solutions is a feasible solution to the problem. The algorithm then shifts all strips by $2 \cdot \beta$ and repeats the process, saving the feasible solution. This can be done a total of $\ell - 1$ times, and the algorithm simply picks the feasible solution with minimal cardinality. In [70], the following lemma is proved (we state it in terms of I-REP-MCZ – which is done by an application of Corollary 11):

Lemma 18 (Shifting Lemma [70])**.** *Let $\boldsymbol{a}_{S(\boldsymbol{A})}$ be the approximation factor of the shifting strategy applied with local algorithm $\boldsymbol{A}$ and $\boldsymbol{a}_{\boldsymbol{A}}$ be the approximation factor for the local algorithm. Then:*

$$\boldsymbol{a}_{S(\boldsymbol{A})} = \boldsymbol{a}_{\boldsymbol{A}} \cdot \left(1 + \frac{1}{\ell}\right).$$

Further, the shifting strategy can actually be applied twice, solving the local algorithm in squares of size $2 \cdot \beta \cdot \ell \times 2 \cdot \beta \cdot \ell$. This gives the following result:

$$\boldsymbol{a}_{S(S(\mathbf{A}))} = \boldsymbol{a}_{\mathbf{A}} \cdot \left(1 + \frac{1}{\ell}\right)^2.$$

A good survey of results based on the shifting strategy can be found in [48], which also provides a linear-time algorithm (this result is later generalized by [52] for multiple dimensions). The following result leverages this for I-REP-MCZ by Corollary 11 (and is proved in [52]).

Proposition 47. *I-REP-MCZ can be solved with an approximation ratio of* $\boldsymbol{x} \cdot \left(1 + \frac{1}{\ell}\right)^2$ *in* $O(K_{\ell,\rho} \cdot |\mathcal{O}|)$ *time. Where* p *is the maximum number of points in a finite lattice over a square of side length* $2 \cdot \beta \cdot \ell$ *s.t. each observation in such a square lies directly on a point in the lattice and* $\boldsymbol{x} \in \{3, 4, 5, 6\}$ *(and is determined by* β*, see [48] for details) and* $K_{\ell,\rho}$ *is defined as follows.*

$$K_{\ell,\rho} = \ell^2 \cdot \sum_{i=1}^{\lceil \ell \cdot \sqrt{2} \rceil^2 - 1} \binom{p}{i} \cdot i$$

An alternative to the shifting strategy leverages techniques used for the related problem of geometric dominating set. In [104], the authors present a $1 + \epsilon$ approximation that runs in $O(|\mathcal{O}|^{O(\frac{1}{\epsilon^2} \cdot \lg^2(\frac{1}{\epsilon}))})$ time.

5.4.3 Practical Considerations for Implementation

We now describe some practical implementation issues. Our primary aim is to find a set of regions that resembles the set of induced regions, $R_{\mathcal{O}}$. There are several reasons for doing this. One reason is to implement a fast heuristic to deal with I-REP optimization problems, specifically when $\alpha \neq 0$. Another, is that such a set of induced regions in the space may be a starting point for creating a set of regions that may include other data, such as that shown in Example 5.2.5.

As most GIS systems view space as a set of discrete points, we discretized the space using the REGION-GEN algorithm below. The parameter g is the spacing of a square grid that overlays the space.

Proposition 48. *REGION-GEN has a time complexity* $\Theta(|\mathcal{O}| \cdot \frac{\pi \cdot \beta^2}{g^2})$.

Example 5.4.5. *Consider the scenario from Example 5.4.4. Suppose the paleontologists now want to generate regions using REGION-GEN instead of using induced regions. The algorithm REGION-GEN overlays a grid on the space in consideration. Using an array representing the space, it records the observations that can be explained by each grid point (Figure 5.5, top). As it does this, any grid point that can explain an observation is stored in list L. The algorithm then iterates through list L, creating entries in a hash table for each subset of observations, enclosing all points that explain the same observation with a minimally-enclosing rectangle. Figure 5.5 (bottom) shows the resulting regions $r_1, \ldots, r_6$.*

One advantage to using REGION-GEN is that we already have the observations that a region super-explains stored – simply consider the bit-string used to index the region in the hash table. Another thing that can be done, for use in an algorithm such as GREEDY-MC2, where the regions are organized by what observation they support, can also be easily done during the running of this algorithm at an additional cost of f (the number of observations that can be partnered with a given region) - by updating an auxiliary data structure at line 6a.

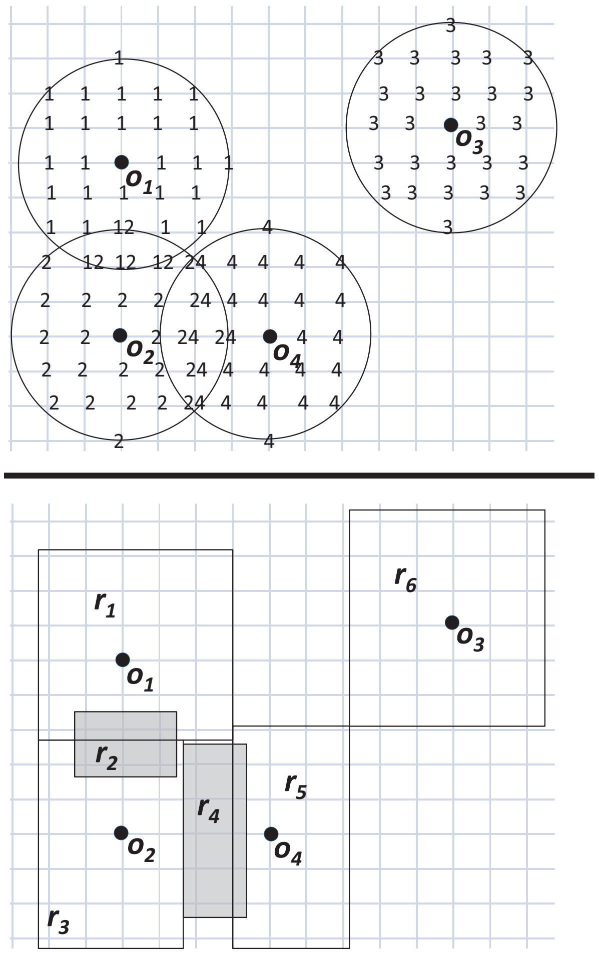


Figure 5.5: REGION-GEN applied to the paleontology example (Example 5.4.4). First, it identifies observations associated with grid points (top). It then creates minimally-enclosing rectangles around points that support the same observations (bottom).

```
REGION-GEN($\mathcal{S}$ space, $\mathcal{O}$ observation set, $\alpha, \beta, g$ reals returns set $R$

1. Overlay a grid of spacing $g$ on space $\mathcal{S}$. With each grid point, $p$, associate set
   $\mathcal{O}_p = \emptyset$. This can easily be represented with an array.
2. Initialize list $L$ of pointers to grid-points.
3. For each $o \in \mathcal{O}$, identity all grid points within distance $[\alpha, \beta]$. For each point
   $p$ meeting this criteria, if $\mathcal{O}_p = \emptyset$, add $p$ to $L$. Also, set $\mathcal{O}_p = \mathcal{O}_p \cup \{o\}$
4. For some subset $\mathcal{O}' \subset \mathcal{O}$, let $str(\mathcal{O}')$ be a bit string of length $|\mathcal{O}|$ where every
   position corresponding to an element of $\mathcal{O}'$ is 1 and all other positions are 0.
5. Let $T$ be a hash table of size $\lceil |\mathcal{O}| \cdot \frac{\pi \cdot \beta^2}{g^2} \rceil$ regions indexed by bit-strings of length
   $|\mathcal{O}|$
6. For each $p \in L$, do the following:
   (a) If $T[str(\mathcal{O}_p)] =$ null then initialize this entry to be a rectangle that en-
       closes point $p$.
   (b) Else, expand the region at location $T[str(\mathcal{O}_p)]$ to be the minimum-
       enclosing rectangle that encloses $p$ and region $T[str(\mathcal{O}_p)]$.
7. Return all entries in $T$ that are not null.
```

5.5 Experimental Results

We implemented REGION-GEN and GREEDY-MC2 in approximately 3000 lines of Java code and conducted several experiments on a Windows-based computer with an Intel $x86$ processor. Our goal was to show that solving the optimization problem Sup-REP-MC would provide useful results in a real-world scenario. We looked at counter-insurgency data from [72] that included data on improvised-explosive device attacks in Baghdad and cache sites where insurgents stored weapons. Under the assumption that the attacks required support of a cache site a certain distance away, could we use attack data to locate cache sites using an instance of Sup-REP-MC solved with GREEDY-MC2 using regions created with REGION-GEN? In our framework, the observations were attacks associated with a cache (which was a partner). The goal was to find relatively small regions that contained partners (caches). We evaluated our approach based on the following criteria:

1. Do the algorithms run in a reasonable amount of time?
2. Does GREEDY-MC2 return regions of a relatively small size?
3. Do the regions returned by GREEDY-MC2 usually contain a partner (cache)?
4. Is the partner (cache) density within regions returned by GREEDY-MC2 significantly greater than the partner density of the space?
5. How does the spacing between grid points affect the runtime and accuracy of the algorithms?

Overall, the experiments indicate that REGION-GEN and GREEDY-MC2 satisfactorily meet the requirements above. For example, for our trials considering locating regions with weapons cache sites (partners) in Baghdad given recent IED attacks (observations), with a grid spacing $g = 100m$, the combined (mean) run-time on a Windows-based laptop was just over 2 seconds. The algorithm produced (mean) 15.54 regions with an average area of $1.838km^2$. Each region, on average, enclosed 1.739 cache sites. If it did not contain a cache site, it was (on average) 275m away from one. The density of caches within returned regions was $8.09caches/km^2$ - significantly higher than the overall density for Baghdad of $0.488caches/km^2$.

The rest of this section is organized as follows. Section 5.5.1 describes the data set we used for our tests and experimental set-up. Issue 1 is addressed in Section 5.5.2. We shall discuss the area (issue 2) of the regions returned in Section 5.5.3 and follow this with a discussion of issue 3 in Section 5.5.4. We shall discuss issue 4 in Section 5.5.5. Throughout all the sections, we shall describe results for a variety of different grid-spacings, hence addressing issue 5.

5.5.1 Experimental Set-Up

We used the *Map of Special Groups Activity in Iraq* available from the Institute for the Study of War [72]. The map plots over 1000 insurgent activities attributed to what are termed as "Special Groups" - groups with access to certain advanced weaponry. This data set contains events for 21 months between February 2007 and November 2008. The activity types include the following categories.

1. Attacks with probable links to Special Groups
2. Discoveries of caches containing weapons associated with Special Groups
3. Detainments of suspected Special Groups criminals

4. Precision strikes against Special Groups personnel

We use this data for two geographic areas: the Baghdad urban area and the Sadr City district. In our experiment, we will view the attacks by the special groups (item 1) as observations and attempt to determine the minimum set of cache sites (item 2), which we shall view as partners. Hence, a region returned by GREEDY-MC2 encloses a partner iff a cache falls within the region.

For distance constraints, we used a simple algorithm to learn the parameter β (α was set to zero). This was done using the first 7 months of attack data ($\frac{1}{3}$ of the available months) and 14 months of cache data. We used the following simple algorithm, FIND-BETA, to determine these values. Note we set β_{max} to 2.5 km.

We ran the experiments on a Lenovo T400 ThinkPad laptop with a 2.53 GHz Intel Core 2 Duo T9400 processor and 4GB of RAM. The computer was running Windows Vista 64-bit Business edition with Service Pack 1 installed.

As the relationship between attacks and cache sites may differ varied on terrain, we ran tests with two different geographic areas. First, we considered the entire Baghdad urban area. Then, we considered just the Sadr City district. We ran FIND-BETA with a β_{max} of 2.5 km on both areas prior to testing the algorithms. There were 73 observations (attacks) for Baghdad and 40 for Sadr City. Table 5.1 shows the exact locations and dimensions of the areas considered.

We conducted two types of tests: tests focusing on GREEDY-MC2 and tests focusing on REGION-GEN.

Algorithm 20 Determines β value from historical data

```
FIND-BETA($\mathcal{O}_h$ historical, time-stamped observations,

$\mathcal{E}_h$ historical, time-stamped partners, $\beta_{max}$ real)

  1. Set $\beta = \beta_{max}$

  2. Set Boolean variable $flag$ to TRUE

  3. For each $o \in \mathcal{O}_h$, do the following:

     (a) For each $p \in \mathcal{E}_h$ that occurs after $o$, do the following.

         i. Let $d$ be the Euclidean distance function.

         ii. If $flag$, and $d(o,p) \leq \beta_{max}$ then set $\beta = d(o,p)$

         iii. If not $flag$, then do the following:

              A. If $d(o,p) > \beta$ and $d(o,p) \leq \beta_{max}$ then set $\beta = d(o,p)$

  4. Return real $\beta$
```

Area	Lower-Left Latitude	Lower-Left Longitude	E-W Distance	N-S Distance
Baghdad	33.200° N	44.250° E	27 km	25 km
Sadr City	33.345° N	44.423° E	7 km	7 km

Table 5.1: Locations and dimensions of areas considered

For the tests of GREEDY-MC2, we used multiple setting for the grid spacing. We tested grid grid spacings at every 10 meter interval in the range of $[70, 1000]$ meters - giving a total of 93 different values for g. Due to the fact that REGION-GEN produces a deterministic result, we ran that algorithm only once per grid setting. However, we ran 100 trials of GREEDY-MC2 per each parameter g. This was done for both Baghdad and Sadr City - giving a total of $18,600$ experiments.

To study the effects of grid-spacing on the run-time of REGION-GEN, we also ran 25 trials for each grid spacing setting for both geographic areas - giving a total of $4,650$ experiments. To compare the algorithms running with different settings for g in a statistically valid manner, we used ANOVA [50] to determine if the differences among grid spacings are statistically significant. For some test results, we conducted linear regression analysis.

5.5.2 Running Time

Overall, the run-times provided by the algorithms were quite reasonable. For example, for the Baghdad trials, 73 attacks were considered for an area of $675m^2$. With a grid spacing $g = 100m$, REGION-GEN ran in $2340ms$ and GREEDY-MC2 took less than $30ms$.

For GREEDY-MC2, we found that run-time generally decreased as g increased. For Baghdad, the average run times ranged over $[1.39, 34.47]ms$. For Sadr City, these times ranged over $[0.15, 4.97]ms$. ANOVAs for both Baghdad and Sadr City run-times gave p-values of $2.2 \cdot 10^{-16}$, which suggests with well over 99% probability that the algorithm run with different grid settings will result in different run-times. We also recorded the number of regions considered in each experiment (resulting from the output of REGION-GEN). Like run-times, we found that the number of regions considered also decreased as the grid spacing increased. For Baghdad, the number of considered regions ranged over $[88, 1011]$. For Sadr City, this number ranged over $[25, 356]$. ANOVAs for both Baghdad and Sadr City number of considered regions gave p-values of $2.2 \cdot 10^{-16}$, which suggests with well over 99% probability that the algorithm run with different grid settings will result in different numbers of considered regions. Note that this is unsurprising as REGION-GEN run deterministically. We noticed that, generally, only grid spacings that were near the same value would lead to the same number of considered regions.

The most striking aspect of the run-time/number of regions considered results for GREEDY-MC2 is that these two quantities seem closely related (see Figure 5.6). This most likely results from the fact that the number of regions that can be associated with a given observation (Δ) increases as the number of regions increases. This coincides with our analysis of GREEDY-MC2 (see Proposition 45).

We also studied the average run-times for REGION-GEN for the various different settings for g. For Baghdad, the average run times ranged over $[16.84, 9184.72]ms$. For Sadr City, these times ranged over $[0.64, 308.92]ms$. ANOVAs for both Baghdad and Sadr City run-times gave p-values of $2.2 \cdot 10^{-16}$, which suggests with well over 99% probability that the algorithm run with different grid settings will result in

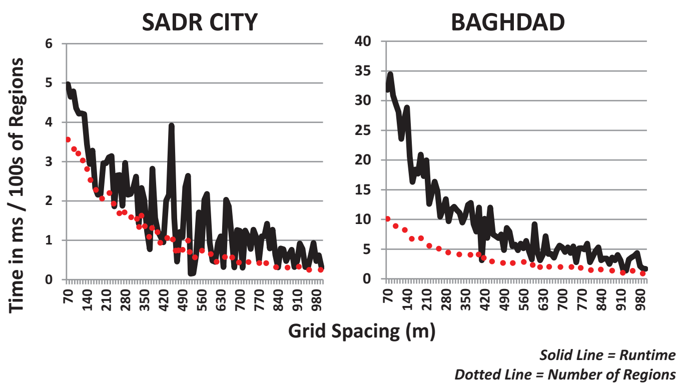


Figure 5.6: The run-time of GREEDY-MC2 in ms compared with the number of regions considered.

different run-times. Our analysis of REGION-GEN (See Proposition 48) states that the algorithm runs in time $O(\frac{1}{g^2})$. We found striking similarities with this analysis and the experimental results (see Figure 5.7).

5.5.3 Area of Returned Regions

In this section, we examine how well the REGION-GEN/GREEDY-MC2 suite of algorithms address the issue of returning regions that are generally small. Although not inherently part of the algorithm, our intuition is that the Sup-REP-MC optimization problem will generally return small regions based on the set R produced by REGION-GEN. The reason for this is that we would expect that smaller regions generally support more observations (note that this is not always true, even for induced regions, but our conjecture is that it is often the case for induced regions or the output of REGION-GEN).

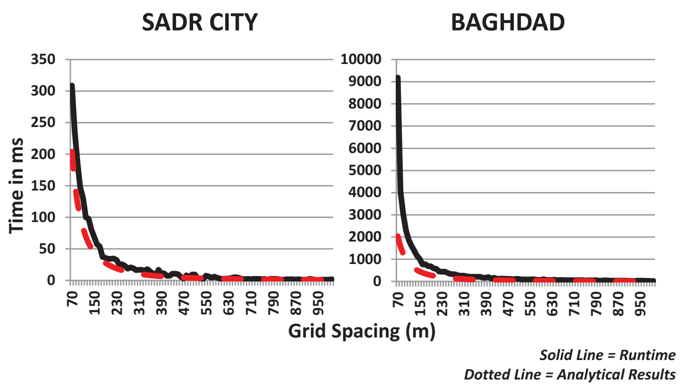


Figure 5.7: A comparison between analytical ($O(\frac{1}{g^2})$) and experimental results for the run-time of REGION-GEN compared with grid spacing (g).

To define "small" we look at the area of a circle of radius β as a basis for comparison. As different grid settings led to different values for β, we looked at the smallest areas. For a given trial, we looked at the average area of the returned regions.

For Baghdad, the average areas ranged over $[0.611, 2.985]km^2$. For Sadr City, these times ranged over $(0.01, 0.576]km^2$. ANOVAs for both Baghdad and Sadr City run-times gave p-values of $2.2 \cdot 10^{-16}$, which suggests with a 99% probability that the algorithm run with different grid settings will result in different average areas. Plotting the areas compared with the established "minimum area" described earlier in this section clearly shows that REGION-GEN/GREEDY-MC2 produce solutions with an average area that is about half of this value - refer to Figure 5.8.

Overall, there seemed to be little relation between grid spacing and average area of the returned set of regions - based on grid spacings in $[70, 1000]m$. As

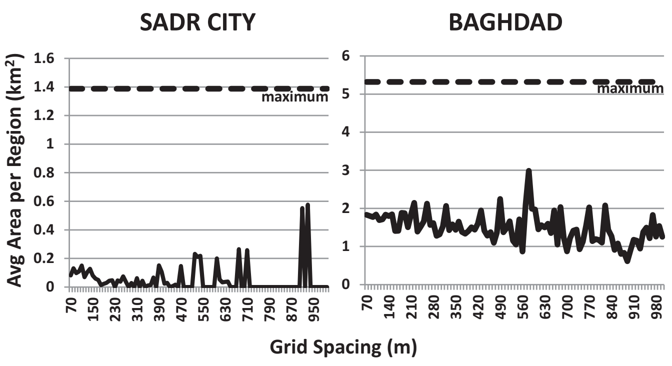


Figure 5.8: Average areas for solutions provided by REGION-GEN/GREEDY-MC2 for Baghdad and Sadr City.

an example, we provided screen-shots of GREEDY-MC2 for $g = 100$ and $g = 1000$ (Figure 5.9). Anecdotally, we noticed that larger grid spacing led to more "pinpoint" regions - regions encompassing only one point in the grid (and viewed as having an area of 0). This is most likely due to the fact that overlaps in the circles around observations points would overlap on fewer grid points for larger values of g. Another factor is that different settings for g led to some variation of the value β - which also affects accuracy (note for our analysis we considered only the smallest values of β as an upper bound for the area - see Figure 5.8.

5.5.4 Regions that Contain Caches

In this section we discuss the issue of ensuring that most of the returned regions enclose at least one partner (cache in the case of our experiments). One measure of this aspect is to look at the average number of caches enclosed per region in a given result. We found, that for Baghdad, we generally enclosed more than 1 cache

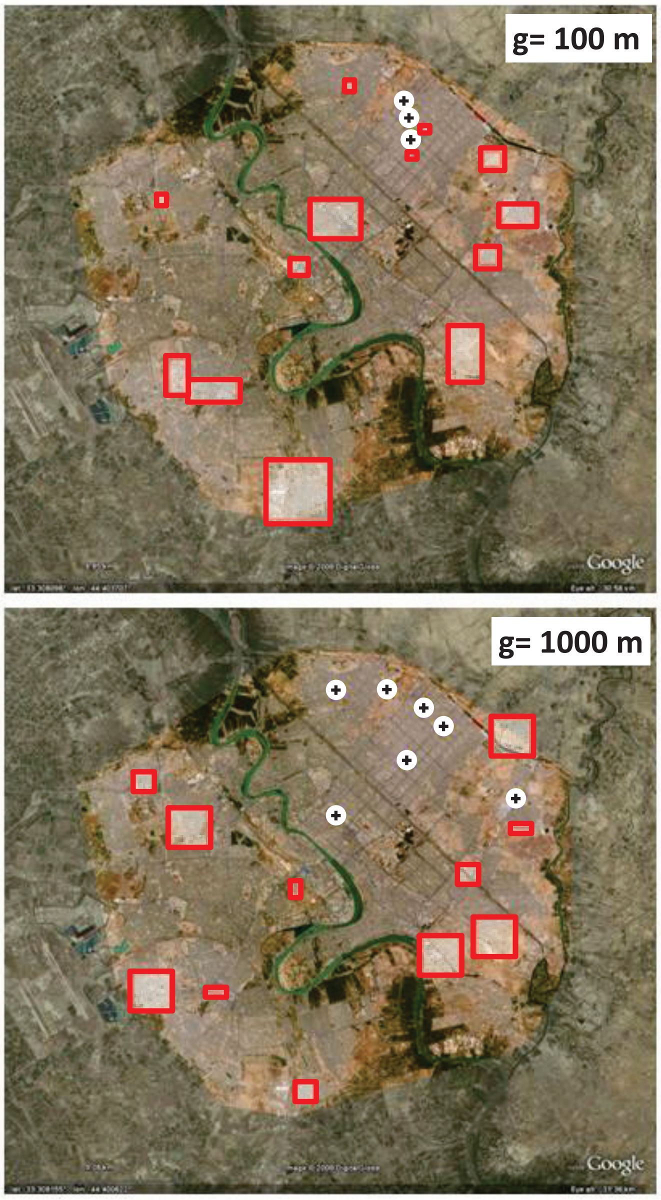


Figure 5.9: Results from two runs of GREEDY-MC2 - $g = 100m$ (top), $g = 1000m$ (bottom). Pinpoint-regions are denoted with plus-signs. Notice that the average areas of the results are comparable.

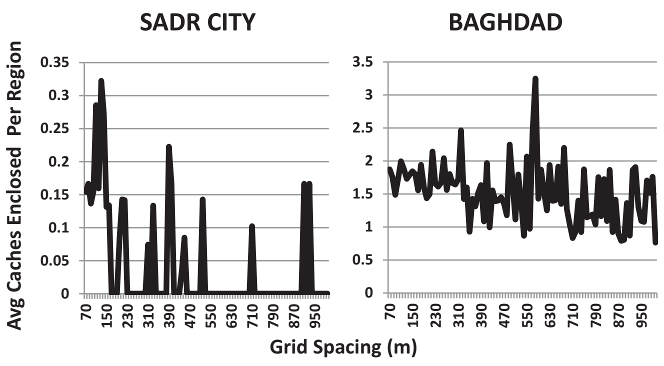


Figure 5.10: Average caches enclosed per region for Baghdad and Sadr City for various grid-spacing settings.

per region in a given result - this number was in the range $[0.764, 3.25]$. The results for Sadr City were considerably lower - in the range $[0, 0.322]$. ANOVAs for both Baghdad and Sadr City gave p-values of $2.2 \cdot 10^{-16}$, which suggests with a 99% probability that the algorithm run with different grid settings will result in different average number of enclosed caches. However, we did not observe an obvious trend in the data (see Figure 5.10).

As an alternative metric - we look at the number of regions in provided by GREEDY-MC2 that contain at least one region. Figure 5.12 shows the number of regions returned in the output. For Baghdad, generally less than half the regions in the output will enclose a cache - the number of enclosing regions was in $[1, 8]$, while the total number of regions was in $[10.49, 22]$. This result, along with the average number of caches enclosed by a region - may indicate that while sometimes GREEDY-MC2 may find regions that enclose many caches, there are often regions

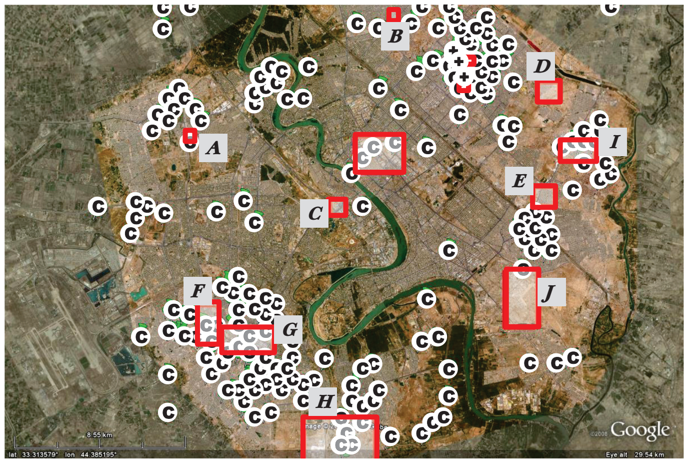


Figure 5.11: The output of GREEDY-MC2 for Baghdad with $g = 100m$ compared with the locations of actual cache sites (denoted with a "C"). Notice that regions A-E do not contain any cache sites while regions G-I all contain numerous cache sites.

that enclose no caches as well. This may indicate that for Baghdad, some attacks-cache relationships conform to our model and others do not - perhaps there is another discriminating attribute about the attacks not present in the data that may account for this phenomenon. For example, perhaps some attacks were preformed by some group that had a capability to store weapons in a cache located further outside the city, or perhaps some groups had the capability to conduct attacks using cache sites that were never found. We illustrate this phenomenon with an example output in Figure 5.11. Note that in the figure, regions A-E do not contain any cache sites while regions G-I all contain numerous cache sites.

For Sadr City, the number of caches that contain one region was significantly

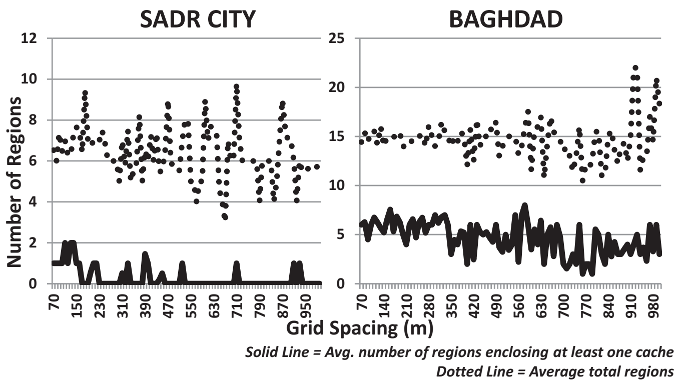


Figure 5.12: Regions in the output that enclose at least one partner (cache) and total number of regions returned for Baghdad and Sadr City.

lower - in the range $[0, 2]$, while the total number of returned regions was in $[3, 9.8]$. ANOVAs for both Baghdad and Sadr City gave p-values of $2.2 \cdot 10^{-16}$, which suggests with well over 99% probability that the algorithm run with different grid settings will result in different number of caches that enclose a region.

We believe that the low numbers for caches enclosed by regions for Sadr City were directly related to the smaller areas of regions. However, the mean of the average area of a returned set of regions was 0 for 49 of the 94 different grid settings (for Sadr City). This means that for the majority of grid settings, the solution consisted only of "pinpoint" regions (see Section 5.5.3 for a description of "pinpoint" regions).

Obviously, it is unlikely for a pinpoint region to contain a cache site merely due to its infinitesimally small area. To better account for this issue - we develop another metric - distance to nearest cache. If a region contains a cache, the value for this metric is 0. Otherwise, it is the distance to the closest cache outside of the region. For

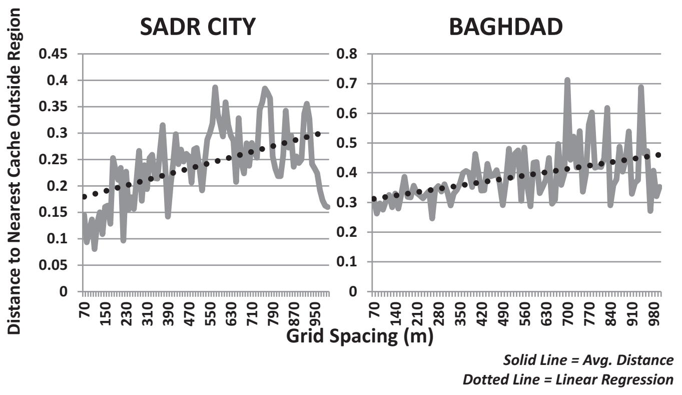


Figure 5.13: Distance to nearest cache vs. grid spacing.

Baghdad, we obtained distances in $[0.246, 0.712]km$, for Sadr City, $[0.080, 0.712]km$. ANOVAs for both Baghdad and Sadr City gave p-values of $2.2 \cdot 10^{-16}$, which suggests with well over 99% probability that the algorithm run with different grid settings will result in different distances to the nearest cache. Using linear regression, we observed that this distance increases as grid spacing increases. For Baghdad, we obtained $R^2 = 0.2396$ and $R^2 = 0.2688$ for Sadr City. See Figure 5.13 for experimental results and the results of the liner regression analysis.

5.5.5 Partner Density

T consider the density of partners in the regions, we compare the number of enclosed partners to the overall partner density of the area in question. For Baghdad, there were 303 caches in an area $27 \times 24km$ - giving a density of $0.488 caches/km^2$. For Sadr City, there were 64 caches in an area $7 \times 7km$ - giving a density of $1.306 caches/km^2$. In our experiments, we looked at the cache density for each output. For Baghdad, the density was significantly higher - in $[0.831, 34.9] cache/km^2$.

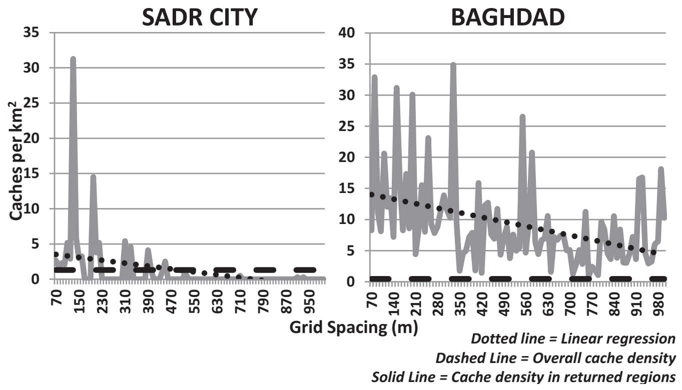


Figure 5.14: Cache density of outputs produced by GREEDY-MC2 for Baghdad and Sadr City compared with overall cache density and linear-regression analysis.

If we consider $g \in [70, 200]$, the density is in $[7.19, 32.9] cache/km^2$. For $g = 100$, the density was $8.09 caches/km^2$. Most likely due to the issue of "pinpoint" regions described in Section 5.5.3, the results for Sadr City, were often lower than the overall density (in $[0, 31.3] cache/km^2$). For $g = 100$, the density was $2.08 caches/km^2$. We illustrate these results compared with overall cache density in Figure 5.14.

ANOVAs for both Baghdad and Sadr City gave p-values of $2.2 \cdot 10^{-16}$, which suggests with well over 99% probability that the algorithm run with different grid settings will result in different cache densities. Using linear regression, we observed that this cache density decreases as grid spacing increases. For Baghdad, we obtained $R^2 = 0.1614$ and $R^2 = 0.1395$ for Sadr City. See Figure 5.14 for experimental results and the results of the liner regression analysis.

Although partner density is a useful metric, it does not tell us anything about partners that lie close to a region - although still outside. For example, consider Figure 5.11. Although region A does not enclose any caches, there is a cache just

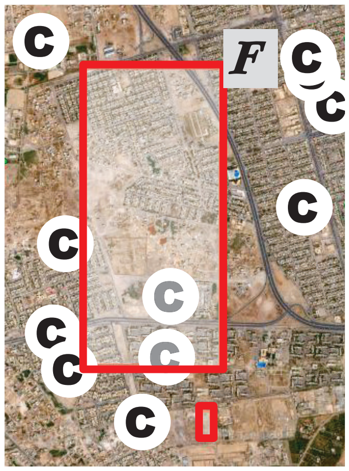


Figure 5.15: Close-up of region F from Figure 5.11. While region F contains 1 cache, there are 4 other caches $< 250m$ from the boundary of that region. The area-quadrupling metric helps us account for such scenarios.

outside - region B is similar. Also consider the cluster of caches south of region E and north of region J - in this situation it appears as though GREEDY-MC2 mis-positioned a region. We include a close-up of region F in Figure 5.15, which encloses a cache, but there are also 4 other caches at a distance of $250m$ or less.

In order to account for such phenomena, we created an area-quadrupling metric - that is we uniformly double the sides of each region in the output. Then, we calculated the density of the output with area-quadrupled regions. For Baghdad, this density was in $[0.842, 30.3] caches/km^2$. For Sadr City, this density was in $[0, 12.3] caches/km^2$. These results are depicted in Figure 5.16.

As the regions for Sadr City were often smaller than those in Baghdad, we found that the cache density for area-quadrupled regions was often higher for Sadr City (i.e. a region in Sadr City would have nearby cache sites). An example is

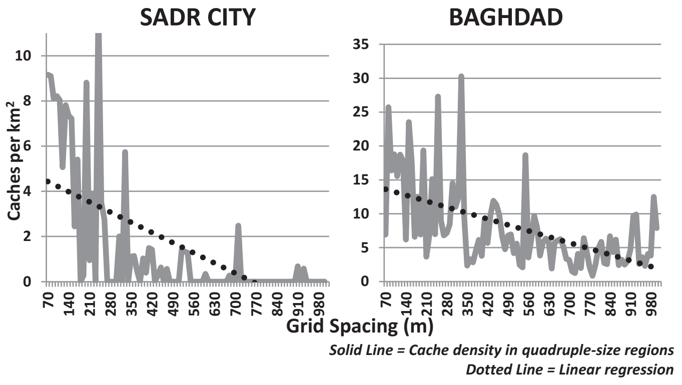


Figure 5.16: Area quadrupled cache density of output produced by GREEDY-MC2 with linear-regression analysis.

shown in Figure 5.15.

ANOVAs for both Baghdad and Sadr City gave p-values of $2.2 \cdot 10^{-16}$, which suggests with well over 99% probability that the algorithm run with different grid settings will result in different cache densities for area-quadrupled regions. We also conducted linear regression analysis, and like the normal partner density, we found that cache density decreases as grid spacing increases. However, this liner analysis was more closely correlated with the data than the analysis for non-area quadrupled density. For Baghdad, we obtained $R^2 = 0.3171$ (for non-area quadrupled, we obtained $R^2 = 0.1614$) and $R^2 = 0.3983$ (for non-area quadrupled, we obtained $R^2 = 0.1395$) for Sadr City. See Figure 5.16 for experimental results and the results of the liner regression analysis.

5.6 Chapter 5 Related Work

Facility location [164] may also appear similar to this work. However, facility location problems normally seek to locate a facility at an infinitesimal point with respect to some minimality criteria - not identify a region. Further, in a facility location problem, distance is often sought to be minimized - so a "closer" facility may be more optimal. In our formulation, we restrict distance with α, β, but a more optimal region is not necessarily closer to its associated observation. Rather, a region is often more optimal provided if it supports multiple partners. This may, in fact, make regions further from their observations. Another problem, which influences some facility location work, is the *k-means* problem [116]. This type of "clustering" technique looks to group points together and possibly locate a "center." While there is an implicit grouping of observations by the algorithms of this paper, we are attempting to find regions that explain them rather than simply group them. Moreover, Chapter 4 shows experimentally that the methods for solving GAPs significantly outperform simply applying k-means algorithms. This fact illustrates that the problem of this paper (and other work in geospatial abduction) is fundamentally different from work in clustering. Perhaps some of the closest work to our problem is in the study of the circle-covering problem [125, 70, 58, 16]. The problem of this paper is more general than circle-covering although special case of the region-explanation problem does reduce to circle-covering, as described in Section 5.4.2 (page 230).

5.7 Chapter Summary

In this chapter we explored a variant of "geospatial abduction" (which was introduced in chapter 4) called *region-based geospatial abduction problems* where the user wishes to identify a set of regions that best explain a given set of observations. This has several important applications including criminology [144], marketing [55], natural science [143], and the military [170]. We explored properties and the complexity of several variants of this problem, including variants where the space is induced by a distance from the observations, as well as when the regions are irregular shapes (including non-convex). As most of the problems were NP-hard, we illustrated a variety of approximation techniques, often with guarantees, to address these problems. We also implemented some of our algorithms and evaluated with a real-world counterinsurgency [72] data-set to find weapons cache sites based on attack data in Baghdad, Iraq and produced regions that had an average density of over 8 caches per square kilometer, significantly higher than the city wide density of 0.4.

There are many interesting open questions relating to this type of abduction problem. Future work may include studies of the counting version of the problem, where we may consider all possible solutions to a given region explanation problem according to a probability distribution and determine the "most probable" regions. Another aspect to consider would be time – perhaps in some applications the locations of the partners are in a certain region only at a certain time.

Chapter 6

Adversarial Geospatial Abduction

Given an instance of a geospatial abduction problem from Chapter 4, where do we look for partners if an adversary is aware of the algorithms we are using? We study this situation, along with a complementary problem in this chapter.[1]

6.1 Chapter Introduction

Geospatial abduction problems (GAPs) were introduced in Chapter 4 to find a set of locations that "best explain" a given set of locations of observations. We call these inferred sets of locations "explanations". There we described many such applications of GAPs.

Chapter 4 defined *geospatial abduction problems* (GAPs) and studied a version of the problem where the adversary (the "bad guy" or the entity that wishes to evade detection) does not reason about the agent (the "good guy" or the entity

[1]This chapter is based on [154] completed in cooperation with John Dickerson and V.S. Subrahmanian.

that wants to detect the adversary). Despite this significant omission, they were able to accurately predict the locations of weapons caches in real-world data about IED attacks in Baghdad. In this chapter, we introduce *adversarial* geospatial abduction problems where both the agent and the adversary reason about each other. Specifically, we:

1. Axiomatically define *reward functions* to be any functions that satisfy certain basic axioms about the similarity between an explanation chosen by the adversary (e.g. where the serial killer lives and works or where the insurgents put their IED caches) and define notions of expected detriment (to the adversary) and expected benefit (to the agent).

2. Formally define the *optimal adversary strategy* (OAS) that minimizes chances of detection of the adversary's chosen explanation and the *maximal counter-adversary strategy* (MCA) that maximizes the probability that the agent will detect the adversary's chosen explanation.

3. Provide a detailed set of results on the computational complexity of these problems, the counting complexity of these problems, and the possibility of approximation algorithms with approximation guarantees for both OAS and MCA.

4. Develop mixed integer linear programming algorithms (MILPs) for OAS and two algorithms, MCA-LS and MCA-GREEDY-MONO, to solve MCA with certain approximation guarantees. MCA-LS has no assumptions, while MCA-GREEDY-MONO assumes monotonicity.

5. Develop a prototype of our MILP algorithms to solve the OAS problem, using our techniques for variable reduction on top of a integer linear program solver. We demonstrate the ability to achieve near-optimal solutions as well as a correct reduction of variables by 99.6% using a real-world data set.

6. Develop a prototype implementation that shows that both MCA-LS and MCA-GREEDY-MONO are highly accurate and have very reasonable time frames. Though MCA-GREEDY-MONO is slightly faster than MCA-LS, we found that on every single run, MCA-LS found the exact optimal benefit even though its theoretical lower bound approximation ratio is only 1/3. As MCA-LS does not require any additional assumptions and as its running time is only slightly slower than that of MCA-GREEDY-MONO, we believe this algorithms has a slight advantage.

The main contributions of the chapter are as follows. Section 6.2 first reviews the GAP framework of Chapter 4. Section 6.3 extends GAPs to the adversarial case using axiomatically defined reward function (Section 6.2). Section 6.4 complexity results and several exact algorithms using MILPs for the OAS problem. Section 6.5 provides complexity results and develops exact and approximate methods MCA —including an approximation technique that provides the best possible guarantee unless P=NP. We then briefly describe our prototype implementation and describe a detailed experimental analysis of our algorithms. Finally, related work is then described in Section 6.7.

6.2 Overview of GAPs

We utilize the same definitions of a space, observations, feasibility, partners, and explanations as we did in Chapter 4. We note in that chapter we often sought to find an explanation of minimal cardinality, a common parsimony requirement. Alternatively, another requirement that can be imposed on an explanation is *irredundancy.*

Definition 58. *An explanation $\mathcal{E}$ is* ***irredundant*** *iff no strict subset of $\mathcal{E}$ is an explanation.*

Intuitively, if we can remove any element from an explanation – and this action causes it to cease to be a valid explanation – we say the explanation is irredundant.

Example 6.2.1. *Figure 6.1 shows a map of a drug plantation depicted in a 18×14 grid. The distance between grid squares is 100 meters. Observation set $\mathcal{O} = \{o_1, o_2, o_3, o_4, o_5\}$ represents the center of mass of the poppy fields. Based on an informant or from historical data, drug enforcement officials know that there is a drug laboratory located $150 - 320$ meters from the center mass of each field (i.e. in a geospatial abduction problem, we can set $[\alpha, \beta] = [150, 320]$). Further, based on the terrain, the drug enforcement officials are able to discount certain areas (shown in black on Figure 6.1, a feasibility predicate can easily be set up accordingly). Based on Figure 6.1, the set $\{p_{40}, p_{46}\}$ is an irredundant explanation. The sets $\{p_{42}, p_{45}, p_{48}\}$ and $\{p_{40}, p_{45}\}$ are also irredundant explanations.*

In Chapter 4, we showed the problem of finding an explanation of size k to be NP-Complete based on a reduction from the known NP-Complete problem

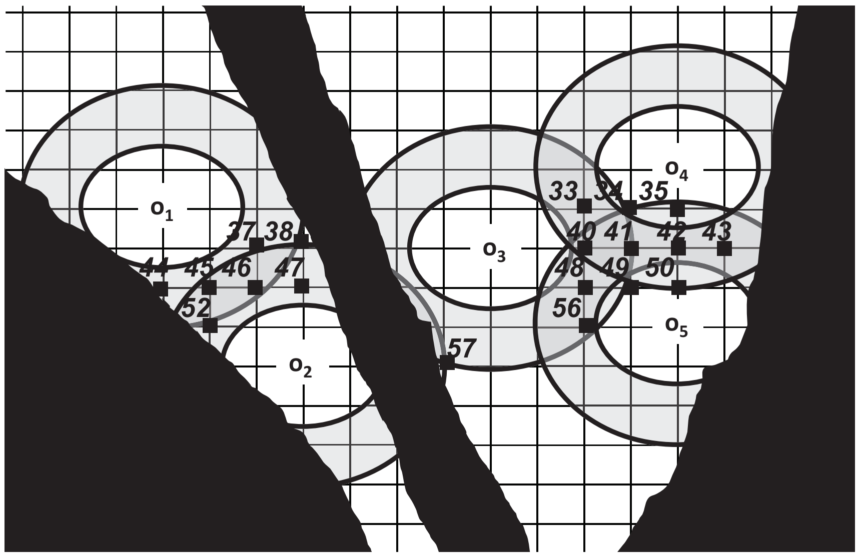


Figure 6.1: Map of poppy fields for Example 6.2.1. For each labeled point p_i, the "p" is omitted for readibility.

Geometric Covering by Discs (GCD) seen in [76]. As with most decision problems, we define the associated counting problem, #GCD, as the number of "yes" answers to the GCD decision problem. The result below, which is new, shows that #GCD is #P-complete and, moreover, that there is no fully-polynomial random approximation scheme for #GCD unless NP equals the complexity class RP.[2]

Lemma 19. *#GCD is #P-complete and has no FPRAS unless NP=RP.*

We can leverage the above result to derive a complexity result for the counting version of k-**SEP**.

[2]RP is the class of decision problems for which there is a randomized polynomial algorithm that, for any instance of the problem, returns "false" with probability 1 when the correct answer to the problem instance is false, and returns "true" with probability $(1-\epsilon)$ for a small $\epsilon > 0$ when the correct answer to the problem instance is "true."

Theorem 27. *The counting version of k-**SEP** is #P-Complete and has no FPRAS unless NP=RP.*

6.3 Geospatial Abduction as a Two-Player Game

Throughout this chapter, we view geospatial abduction as a two-player game where an ***agent*** attempts to find an "explanation" for a set of observations caused by the ***adversary*** who wants to hide the explanation from the agent.

Each agent chooses a *strategy* which is merely a subset of $\mathcal{S}$. Though "strategy" and "observation" are defined identically, we use separate terms to indicate our intended use. In the IED example, the adversary's strategy is a set of points where to place his cache, while the agent's strategy is a set of points that he thinks hold the weapons caches. Throughout this chapter, we use $\mathcal{E}_{gt}$ (resp. $\mathcal{C}$) to denote the strategy of the adversary (resp. agent).

Given a pair $(\mathcal{E}_{gt}, \mathcal{C})$ of adversary-agent strategies, a reward function measures how similar the two sets are. The more similar, the better it is for the agent. As reward functions can be defined in many ways, we choose an axiomatic approach so that our framework applies to many different reward functions including ones that people may invent in the future.

Definition 59 (Reward Function)**.** *A* reward function *is any function* $\boldsymbol{rf} : 2^{\mathcal{S}} \times 2^{\mathcal{S}} \rightarrow [0, 1]$ *that for any k-explanation $\mathcal{E}_{gt} \not\equiv \emptyset$ and set $\mathcal{C} \subseteq \mathcal{S}$, the function satisfies:*

1. *If $\mathcal{C} = \mathcal{E}_{gt}$, then $\boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C}) = 1$*

2. For $\mathcal{C}, \mathcal{C}'$ then

$$\boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C} \cup \mathcal{C}') \leq \boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C}) + \boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C}') - \boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C} \cap \mathcal{C}').$$

We now define the payoffs for the agent and adversary.

Observation 6.3.1. *Given adversary strategy $\mathcal{E}_{gt}$, agent strategy $\mathcal{C}$, and reward function $\boldsymbol{rf}$, the payoff for the agent is $\boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C})$ and the payoff for the adversary is $-\boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C})$.*

It is easy to see that for any reward function and pair $(\mathcal{E}_{gt}, \mathcal{C})$, the corresponding game is a *zero-sum game* [102]. Our complexity analysis assumes all reward functions are polynomially computable. All the specific reward functions we propose in this chapter satisfy this condition.

The basic intuition behind the reward function is that the more the strategy of the agent resembles that of the adversary, the closer the reward is to 1. Axiom 1 says that if the agent's strategy is the same set as adversary's, then the reward is 1. Axiom 2 says that adding a point to $\mathcal{C}$ cannot increase the reward to the agent if that point is already in $\mathcal{C}$, i.e. double-counting of rewards is forbidden.

The following theorem tells us that every reward function is *submodular*, i.e. the marginal benefit of adding additional points to the agent's strategy decreases as the cardinality of the strategy increases.

Proposition 49 (Submodularity of Reward Functions)**.** *Every reward function is submodular, i.e. If $\mathcal{C} \subseteq \mathcal{C}'$, and point $p \in \mathcal{S}$ s.t. $p \notin \mathcal{C}$ and $p \notin \mathcal{C}'$, then $\boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C} \cup \{p\}) - \boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C}) \geq \boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C}' \cup \{p\}) - \boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C}')$.*

Some readers may wonder why $\boldsymbol{rf}(\mathcal{E}_{gt}, \emptyset) = 0$ is not an axiom. While this is true of many reward functions, there are reward functions where we may wish to penalize the agent for "bad" predictions. Consider the following reward function.

Definition 60 (Penalizing Reward Function)**.** *Given a distance dist, we define the* ***penalizing reward function***, $\boldsymbol{prf}^{(dist)}(\mathcal{E}_{gt}, \mathcal{C})$, *as follows:*

$$\frac{1}{2} + \frac{|\{p \in \mathcal{E}_{gt} | \exists p' \in \mathcal{C} \text{ s.t. } d(p,p') \leq dist\}|}{2 \cdot |\mathcal{E}_{gt}|} - \frac{|\{p \in \mathcal{C} | \nexists p' \in \mathcal{E}_{gt} \text{ s.t. } d(p,p') \leq dist\}|}{2 \cdot |\mathcal{S}|}$$

Proposition 50. $\boldsymbol{prf}$ *is a valid reward function.*

Example 6.3.1. *Consider Example 6.2.1 and the explanation* $\mathcal{E}_{gt} \equiv \{p_{40}, p_{46}\}$ *(resembling actual locations of the drug labs), the set* $\mathcal{C} \equiv \{p_{38}, p_{41}, p_{44}, p_{56}\}$ *(representing areas that the drug enforcement officials wish to search), distance dist* $= 100$ *meters.There is only one point in* $\mathcal{E}_{gt}$ *that is within* 100 *meters of a point in* $\mathcal{C}$ *(point* p_{40}*) and* 3 *points in* $\mathcal{C}$ *more than* 100 *meters from any point in* $\mathcal{E}_{gt}$ *(points* p_{38}, p_{44}, p_{56}*). These relationships are shown visually in Figure 6.2. Hence,* $\boldsymbol{prf}^{(dist)}(\mathcal{E}_{gt}, \mathcal{C}) = 0.5 + 0.25 - 0.011 = 0.739$.

$\boldsymbol{prf}$ penalizes the agent if he poorly selects points in $\mathcal{S}$. The agent starts with a reward of 0.5. The reward increases if he finds points close to elements of $\mathcal{E}_{gt}$ — otherwise it decreases.

A reward function is *zero-starting* if $\boldsymbol{rf}(\mathcal{E}_{gt}, \emptyset) = 0$, i.e. the agent gets no reward if he infers nothing.

Definition 61. *A reward function,* $\boldsymbol{rf}$, *is* ***monotonic*** *if (i) it is zero-starting and (ii) if* $\mathcal{C} \subseteq \mathcal{C}'$ *then* $\boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C}) \leq \boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C}')$.

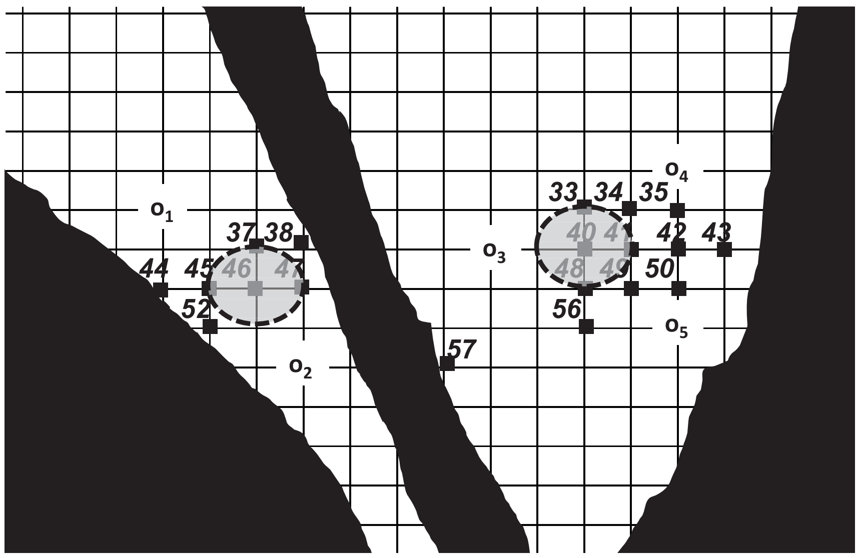


Figure 6.2: Dashed circles encompass all feasible points within 100 meters from explanation $\{p_{40}, p_{45}\}$.

We now define several example monotonic reward functions.

The intuition behind the *cutoff reward function* $\boldsymbol{crf}$ is simple: for a given distance $dist$ (the "cut-off" distance), if for every $p \in \mathcal{E}_{gt}$, there exists $p' \in \mathcal{C}$ such that $d(p, p') \leq dist$, then p' is considered "close to" p.

Definition 62 (Cutoff Reward Function)**.** *Reward function based on a cut-off distance, dist.*

$$\boldsymbol{crf}^{(dist)}(\mathcal{E}_{gt}, \mathcal{C}) := \frac{card(\{p \in \mathcal{E}_{gt} | \exists p' \in \mathcal{C} \ s.t. \ d(p,p') \leq dist\})}{card(\mathcal{E}_{gt})}$$

The following proposition shows that the cutoff reward function is a valid, monotonic reward function.

Proposition 51. *$\boldsymbol{crf}$ is a valid, monotonic reward function.*

Example 6.3.2. *Consider Example 6.3.1. Here,* $\boldsymbol{crf}^{(dist)}(\mathcal{E}_{gt}, \mathcal{C})$ *returns* 0.5 *as one element of* $\mathcal{E}_{gt}$ *is within* 100 *meters of an element in* $\mathcal{C}$*.*

By allowing a more general notion of "closeness" between points $p \in \mathcal{E}_{gt}$ and $p' \in \mathcal{E}$, we are able to define another reward function, the *falloff reward function*, $\boldsymbol{frf}$. This function provides the most reward if $p = p'$ but, unlike the somewhat binary $\boldsymbol{crf}$, gently lowers this reward to a minimal zero as distances $d(p, p')$ grow.

Definition 63 (Falloff Reward Function)**.** *Reward function with value based on minimal distances between points.*

$$\boldsymbol{frf}(\mathcal{E}_{gt}, \mathcal{C}) := \begin{cases} 0 & \text{if } \mathcal{C} = \emptyset \\ \sum_{p \in \mathcal{E}_{gt}} \frac{1}{|\mathcal{E}_{gt}| + \min_{p' \in \mathcal{C}}(d(p,p')^2)} & \text{otherwise} \end{cases}$$

with $d(p, p') := \sqrt{(p_x - p'_x)^2 + (p_y - p'_y)^2}$*. In this case, the agent's reward is inversely proportional to the square of the distance between points, as the search area required grows proportionally to the square of this distance.*

Proposition 52. $\boldsymbol{frf}$ *is a valid, monotonic reward function.*

In practice, an agent may assign different weights to points in $\mathcal{S}$ based on the perceived importance of their partner observations in $\mathcal{O}$. The "weighted reward function" $\boldsymbol{wrf}$ gives greater reward for being "closer" to points in $\mathcal{E}_{gt}$ that have high weight than those with lower weights.

Definition 64 (Weighted Reward Function)**.** *Given weight function* $W : \mathcal{S} \to \mathbb{R}^+$*, and a cut-off distance dist we define the weighted reward function to be:*

$$\boldsymbol{wrf}^{(W,dist)}(\mathcal{E}_{gt}, \mathcal{C}) := \frac{\sum_{\{p \in \mathcal{E}_{gt} | \exists p' \in \mathcal{C} \text{ s.t. } d(p,p') \leq dist\}} W(p)}{\sum_{p' \in \mathcal{E}_{gt}} W(p')}$$

Proposition 53. ***wrf*** *is a valid, monotonic reward function.*

It is easy to see that the weighted reward function is a generalization of the cutoff reward function where all weights are 1.

6.3.1 Incorporating Mixed Strategies

In this section, we introduce pdfs over strategies (or "mixed strategies" [102]) and introduce the notion of "expected reward." We first present *explanation/strategy functions* which return an explanation (resp. strategy) of a certain size for a given set of observations.

Definition 65 (Explanation/Strategy Function)**.** *An* explanation (resp. strategy) function *is any function* $\mathsf{ex\_fcn} : 2^{\mathcal{S}} \times \mathbb{N} \to 2^{\mathcal{S}}$ *(resp.* $\mathsf{sf} : 2^{\mathcal{S}} \times \mathbf{N} \to 2^{\mathcal{S}}$*) that, given a set* $\mathcal{O} \subseteq \mathcal{S}$ *and* $k \in \mathbb{N}$*, returns a set* $\mathcal{E} \subseteq \mathcal{S}$ *such that* $\mathcal{E}$ *is a* k*-sized explanation of* $\mathcal{O}$ *(resp.* $\mathcal{E}$ *is a* k*-sized subset of* $\mathcal{S}$*). Let* $\mathbf{EF}$ *be the set of all explanation functions.*

Example 6.3.3. *Following from Example 6.2.1, we shall define two functions* $\mathsf{ex\_fcn}_1, \mathsf{ex\_fcn}_2$*, which for set* $\mathcal{O}$ *(defined in Example 6.2.1 and* $k \leq 3$*, give the following sets:*

$$\begin{aligned} \mathsf{ex\_fcn}_1(\mathcal{O}, 3) &= \{p_{42}, p_{45}, p_{48}\} \\ \mathsf{ex\_fcn}_2(\mathcal{O}, 3) &= \{p_{40}, p_{46}\} \end{aligned}$$

These sets may correspond to explanations from various sources. Perhaps they correspond to the answer of an algorithm that drug-enforcement officials use to solve GAPs. Conversely, they could also be the result of a planning session by the drug cartel to determine optimal locations for the drug labs.

In theory, the set of all explanation functions can be infinitely large; however, it makes no sense to look for explanations containing more points than $\mathcal{S}$ — so we assume explanation functions are only invoked with $k \leq (M+1) \times (N+1)$.

A strategy function is appropriate for an agent who wants to select points resembling what the adversary selected, but is not required to produce an explanation. Our results typically do not depend on whether an explanation or strategy function is used (when they do, we point it out). Therefore, for simplicity, we use "explanation function" throughout the chapter. In our complexity results, we assume that explanation/strategy functions are computable in constant time.

Both the agent and the adversary do not know the explanation function (where is the adversary going to put his weapons caches? where will US forces search for them?) in advance. Thus, they use a pdf over explanation functions to estimate their opponent's behavior, yielding a "mixed" strategy.

Definition 66 (Explanation Function Distribution)**.** *Given a space $\mathcal{S}$, real numbers α, β, feasibility predicate* feas*, and an associated set of explanation functions* $\mathbf{EF}$*, an* explanation function distribution *is a finitary*[3] *probability distribution* $\mathit{exfd} : \mathbf{EF} \to [0,1]$ *with* $\sum_{\mathit{ex\_fcn} \in \mathbf{EF}} \mathit{exfd}(\mathit{ex\_fcn}) = 1$*. Let* $\mathbf{EFD}$ *be a set of explanation function distributions.*

We use $|\mathsf{exfd}|$ to denote the cardinality of the set $\mathbf{EF}$ associated with exfd.

Example 6.3.4. *Following from Example 6.3.3, we shall define the explanation function distribution* exfd_{drug} *that assigns a uniform probability to explanation func-*

[3] That is, exfd assigns non-zero probabilities to only finitely many explanation functions.

tions in the set $\mathsf{ex\_fcn}_1, \mathsf{ex\_fcn}_2$ *(i.e.* $\mathsf{exfd}_{drug}(\mathsf{ex\_fcn}_1) = 0.5$*).*

We now define an "expected reward" that takes into account these mixed strategies specified by explanation function distributions.

Definition 67 (Expected Reward)**.** *Given a reward function* $\boldsymbol{rf}$*, and explanation function distributions* $\mathsf{exfd}_{adv}, \mathsf{exfd}_{ag}$*, the* expected reward *is the function* $\mathsf{EXR}^{(\boldsymbol{rf})}$: $\mathbf{EFD} \times \mathbf{EFD} \to [0,1]$*. For some explanations function distributions* $\mathsf{exfd}_{adv}, \mathsf{exfd}_{ag}$*, we define* $\mathsf{EXR}^{(\boldsymbol{rf})}(\mathsf{exfd}_{adv}, \mathsf{exfd}_{ag})$ *as follows:*

$$\sum_{\mathsf{ex\_fcn}_{adv} \in \mathbf{EF}_{adv}} \left(\mathsf{exfd}_{adv}(\mathsf{ex\_fcn}_{adv}) \cdot \sum_{\mathsf{ex\_fcn}_{ag} \in \mathbf{EF}_{ag}} \mathsf{exfd}_{ag}(\mathsf{ex\_fcn}_{ag}) \cdot \boldsymbol{rf}(\mathsf{ex\_fcn}_{adv}, \mathsf{ex\_fcn}_{ag}) \right)$$

However, in this chapter, we will generally not deal with expected reward directly, but two special cases - expected adversarial detriment and expected agent benefit - in which the adversary's and agent's strategies are *not* mixed respectively. We explore these two special cases in the next two sections.

6.4 Selecting a Strategy for the Adversary

In this section, we look at how an adversary would select points (set $\mathcal{E}_{gt}$) in the space he would use to cause observations $\mathcal{O}$. For instance, in the IED example, the adversary needs to select $\mathcal{E}_{gt}$ and O so that $\mathcal{E}_{gt}$ is an explanation for $\mathcal{O}$. We assume the adversary has a probabilistic model of the agent's behavior (an explanation function distribution) and that he wants to eventually find an explanation (e.g. to put his weapons caches at). Hence, though he can use expected reward to measure how close the agent will be to his explanation, only the agent's strategy is mixed.

His actions are concrete. Hence, we introduce a special case of expected reward – expected adversarial detriment.

Definition 68 (Expected Adversarial Detriment)**.** *Given any reward function* $\boldsymbol{rf}$*, and explanation function distribution* exfd*, the* expected adversarial detriment *is the function* $\mathsf{EXR}^{(\boldsymbol{rf})} : \mathbf{EFD} \times 2^{\mathcal{S}} \to [0,1]$ *defined as follows:*

$$\mathsf{EXR}^{(\boldsymbol{rf})}(\mathsf{exfd}, \mathcal{E}_{gt}) = \sum_{\mathsf{ex\_fcn} \in \mathbf{EF}} \boldsymbol{rf}(\mathcal{E}_{gt}, \mathsf{ex\_fcn}(\mathcal{O}, k)) \cdot \mathsf{exfd}(\mathsf{ex\_fcn})$$

Intuitively, the expected adversarial detriment is the fraction of partner locations the agent may uncover. Consider the following example.

Example 6.4.1. *Following from the previous examples, suppose the drug cartel is planning three drug labs. Suppose they have information that drug-enforcement agents will look for drug labs using* exfd_{drug} *(Example 6.3.4). One suggestion the adversary may consider is to put the labs at locations* p_{41}, p_{52} *(see Figure 6.1). Note that this explanation is optimal wrt cardinality. With dist* $= 100$ *meters, they wish to compute* $\mathsf{EXR}^{(\boldsymbol{crf})}(\mathsf{exfd}_{drug}, \{p_{41}, p_{52}\})$*. We first need to find the reward associated with each explanation function (see Example 6.3.3):*

$$\begin{aligned} \boldsymbol{crf}^{(dist)}(\{p_{41}, p_{52}\}, \mathsf{ex\_fcn}_1(\mathcal{O}, 3)) &= 1 \\ \boldsymbol{crf}^{(dist)}(\{p_{41}, p_{52}\}, \mathsf{ex\_fcn}_2(\mathcal{O}, 3)) &= 0.5 \end{aligned}$$

Thus, $\mathsf{EXR}^{(\boldsymbol{crf})}(\mathsf{exfd}_{drug}, \{p_{41}, p_{52}\}) = 0.5 \cdot 1 + 0.5 \cdot 0.5 = 0.75$*. Hence, this is probably not the best location for the cartel to position the labs wrt* $\boldsymbol{crf}$ *and* exfd *– the expected adversarial detriment of the drug-enforcement agents is large.*

The expected adversarial detriment is a quantity that the adversary would seek to minimize — this now defined as an *optimal adversarial strategy* below.

Definition 69 (Optimal Adversarial Strategy)**.** *Given a set of observations $\mathcal{O}$, natural number k, reward function **rf**, and explanation function distribution exfd, an **optimal adversarial strategy** is a k-sized explanation $\mathcal{E}_{gt}$ for $\mathcal{O}$ such that $\mathsf{EXR}^{(\boldsymbol{rf})}(\mathsf{exfd}, \mathcal{E}_{gt})$ is minimized.*

6.4.1 The Complexity of Finding an Optimal Adversarial Strategy

In this section, we formally define the optimal adversary strategy (OAS) problem and study its complexity.

OAS Problem

INPUT: Space $\mathcal{S}$, feasibility predicate, feas, real numbers α, β, set of observations, $\mathcal{O}$, natural number k, reward function $\boldsymbol{rf}$, and explanation function distribution exfd.

OUTPUT: The optimal adversarial strategy, $\mathcal{E}_{gt}$.

We show that the known NP-hard problem *Geometric Covering by Discs* (see Section 6.2) is polynomially reducible to OAS - this establishes NP-hardness.

Theorem 28. *OAS is NP-hard.*

The proof of the above theorem yields two insights. First, OAS is NP-hard even if the reward function is monotonic (or anti-monotonic). Second, OAS remains NP-hard even if the cardinality of $\mathsf{\mathbf{EF}}$ is small - in the construction we only have one explanation function. Thus, we cannot simply pick an "optimal" function from $\mathsf{\mathbf{EF}}$. To show an upper bound, we define OAS-DEC to be the decision problem associated with OAS. If the reward function is computable in polynomial time, OAS-DEC is in NP.

<u>OAS-DEC</u>

INPUT: Space $\mathcal{S}$, feasibility predicate, feas, real numbers α, β, set of observations, $\mathcal{O}$, natural number k, reward function $\boldsymbol{rf}$, explanation function distribution exfd, and number $R \in [0, 1]$.

OUTPUT: "Yes" if there exists an adversarial strategy, $\mathcal{E}_{gt}$ such that $\mathsf{EXR}^{(\boldsymbol{rf})}(\mathsf{exfd}, \mathcal{E}_{gt}) \leq R$ – "no" otherwise.

Theorem 29. *If the reward function is computable in PTIME, then OAS-DEC is NP-complete.*

Suppose we have an NP oracle that can return an optimal adversarial strategy - lets call it $\mathcal{E}_{gt}$. Quite obviously, this is the **best response** of the adversary to the mixed strategy of the agent. Now, how does the agent respond to such a strategy? If we were to assume that such a solution were unique, then the agent would simply have to find an strategy $\mathcal{C}$ such that $\boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C})$ is maximized. This would be a special case of the problem we discuss in Section 6.5. However, this is not necessarily the case. A natural way to address this problem is to create a uniform probability distribution over all optimal adversarial strategies and optimize the expected reward – again a special case of what is to be discussed in Section 6.5. However, obtaining the set of explanations is not an easy task. Even if we had an easy way to exactly compute an optimal adversarial strategy, finding *all* such strategies is an even more challenging problem. In fact, it is at least as hard as the counting version of GCD – which we already have shown to be #P-hard and difficult to approximate. Let us consider the following theorem.

Theorem 30. *Finding the set of all adversarial optimal strategies that provide a "yes" answer to OAS-DEC is #P-hard.*

6.4.2 Pre-Processing and Naive Approach

In this section, we present several algorithms to solve OAS. We first present a simple routine for pre-processing followed by a naive enumeration-based algorithm.

We use Δ to denote the maximum number of partners per observation and f to denote the maximum number of observations supported by a single partner. In general, Δ is bounded by $\pi(\beta^2 - \alpha^2)$, but may be lower depending on the feasible points in $\mathcal{S}$. Likewise, f is bounded by $\min(|\mathcal{O}|, \Delta)$ but may be much smaller depending on the sparseness of the observations.

Pre-Processing Procedure. Given a space $\mathcal{S}$, a feasibility predicate feas, real numbers $\alpha, \beta \in [0, 1]$, and a set $\mathcal{O}$ of observations, we create two lists (similar to a standard invertex index) as follows.

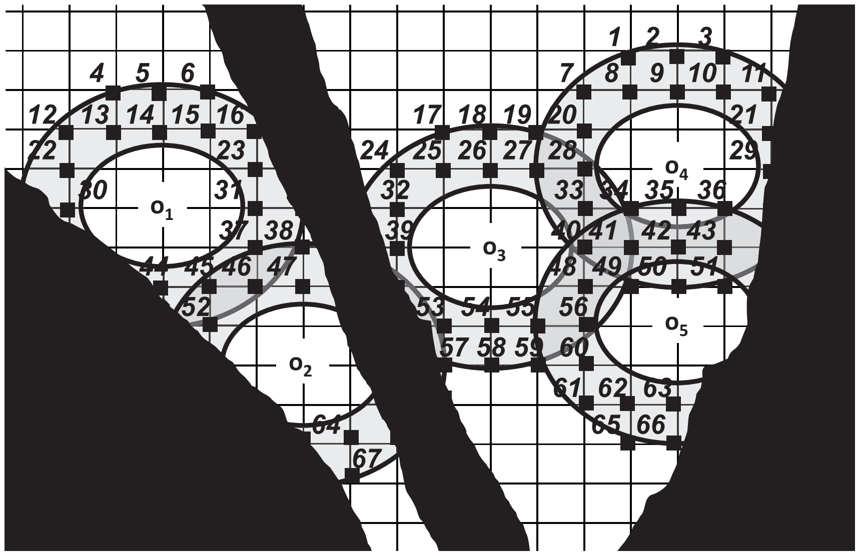


Figure 6.3: Set L of all possible partners for our drug laboratory location example.

- **Matrix** M**.** M is an array of size $\mathcal{S}$. For each point $p \in \mathcal{S}$, $M[p]$ is a *list of pointers* to observations. $M[p]$ contains pointers to each observation o such that $\mathsf{feas}(p)$ is true and such that $d(o,p) \in [\alpha, \beta]$.

- **List** L**.** List L contains a pointer to position $M[p]$ in the array M iff there exists an observation $o \in \mathcal{O}$ such that $\mathsf{feas}(p)$ is true and such that $d(o,p) \in [\alpha, \beta]$..

It is easy to see that we can compute M and L in $O(|\mathcal{O}| \cdot \Delta)$ time. The example below shows how M, L apply to our running drug example.

Example 6.4.2. *Consider our running example concerning the location of drug laboratories that started with Example 6.2.1. The set L consists of* $\{p_1, \ldots, p_{67}\}$. *The matrix M returns lists of observations that can be associated with each point. For example,* $M(p_{40}) = \{o_3, o_4, o_5\}$ *and* $M(p_{46}) = \{o_1, o_2\}$.

Naive Approach. After pre-processing, a straight-forward exact solution to OAS would be to enumerate all subsets of L that have a cardinality less than or equal to k. Let us call this set L^*. Next, we eliminate all elements of L^*. that are not valid explanations. Finally, for each element of L^*, we compute the expected adversarial detriment - and return the element of L^* for which this value is the least. Clearly, this approach is impractical as the cardinality of L^* can be very large. Further, this approach does not take advantage of the specific reward functions. We now present mixed integer linear programs (MILPs) for ***wrf*** and ***frf*** and later look at ways to reduce the complexity of solving these MILPs.

6.4.3 Mixed Integer Linear Programs for OAS under *wrf*, *crf*, *frf*

We present mixed integer linear programs (MILPs) to solve OAS exactly for some specific reward functions. First, we consider the reward function ***wrf***. Later, in Section 6.4.4, we show how to improve efficiency by correctly reducing the number of variables in such MILPs. Note that these constraints can also be used for ***crf*** as ***wrf*** generalizes ***crf***.

Definition 70 (***wrf*** MILP)**.** *We associate an integer-valued variable X_i with each $p_i \in L$.*

Minimize:

$$\sum_{\mathsf{ex\_fcn}_j \in \mathbf{EF}} \left(\mathsf{exfd}(\mathsf{ex\_fcn}_j) \cdot \sum_{p_i \in L} \left(X_i \cdot \left(\frac{w_i \cdot c_{i,j}}{\sum_{p_i \in L} w_i \cdot X_i} \right) \right) \right)$$

subject to:

1. $X_i \in \{0, 1\}$

2. *Constraint* $\sum_{p_i \in L} X_i \leq k$

3. *For each* $o_j \in \mathcal{O}$*, add constraint*

   $\sum_{p_i \in L_{d(o_j,p_i)\in[\alpha,\beta]}} X_i \geq 1$

4. *For each* $p_i \in L$ *and* $\textsf{ex\_fcn}_j \in \textsf{\textbf{EF}}$*, let constant* $c_{i,j} = 1$ *iff* $\exists p' \in \textsf{ex\_fcn}(\mathcal{O}, k)$ *s.t.* $d(p', p_i) \leq dist$ *and* 0 *otherwise.*

Example 6.4.3. *Continuing from Examples 6.4.1 (page 270) and 6.4.2, suppose the drug cartel wishes to produce an adversarial strategy* $\mathcal{E}_{gt}$ *using* ***wrf****. Consider the case where we use* ***crf****,* $k \leq 3$*, and* $dist = 100$ *meters as before (see Example 6.4.1). Clearly, there are* 67 *variables in these constraints, as this is the cardinality of set L (as per Example 6.4.2). The constants* $c_{i,1}$ *are* 1 *for elements in the set* $\{p_{35}, p_{40}, p_{41}, p_{42}, p_{43}, p_{44}, p_{45}, p_{46}, p_{49}, p_{49}, p_{50}, p_{52}, p_{56}\}$ *(and* 0 *for all others). The constants* $c_{i,2}$ *are* 1 *for elements in the set* $\{p_{33}, p_{37}, p_{40}, p_{41}, p_{45}, p_{46}, p_{47}, p_{48}\}$ *(and* 0 *for all others).*

We can create a MILP for ***frf*** as follows.

Definition 71 (***frf*** MILP)**.** *Minimize:*

$$\sum_{\textsf{ex\_fcn}_j \in \textsf{\textbf{EF}}} \left(\textsf{exfd}(\textsf{ex\_fcn}_j) \cdot \sum_{p_i \in L} \left(X_i \cdot \left(\frac{1}{c_{i,j} + \sum_{p_i \in L} X_i}\right) \right) \right)$$

subject to:

1. $X_i \in \{0, 1\}$

2. *Constraint* $\sum_{p_i \in L} X_i \leq k$

3. *For each* $o_j \in \mathcal{O}$*, add constraint*

   $\sum_{p_i \in L_{d(o_j,p_i)\in[\alpha,\beta]}} X_i \geq 1$

4. *For each* $p_i \in L$ *and* $\textsf{ex\_fcn}_j \in \textbf{EF}$*, let constant* $c_{i,j} = \min_{p' \in \textsf{ex\_fcn}(\mathcal{O},k)}(d(p_i, p')^2)$.

The following theorem tells us that solving the above MILPs correctly yields a solution for the OAS problem under both ***wrf*** or ***frf***.

Proposition 54. *Suppose* $\mathcal{S}$ *is a space,* $\mathcal{O}$ *is an observation set,* $[\alpha, \beta] \subseteq [0,1]$ *and suppose the* ***wrf*** *and* ***frf*** *MILPs are defined as above.*

1. *Suppose* $\mathcal{E}_{gt} \equiv \{p_1, \ldots, p_n\}$ *is a solution to OAS with* ***wrf****(resp.* ***frf****). Consider the assignment that assigns* 1 *to each* $X_1, \ldots, X_n$ *corresponding to the* p_i*'s and* 0 *otherwise. This assignment is an optimal solution to the MILP.*

2. *Given the solution to the constraints, if for every* $X_i = 1$*, we add point* p_i *to set* $\mathcal{E}_{gt}$*, then* $\mathcal{E}_{gt}$ *is a solution to OAS with* ***wrf****(resp.* ***frf****).*

Setting up either set of constraints can be performed in polynomial time – where computing the $c_{i,j}$ constants is the dominant operation.

Proposition 55. *Setting up the* ***wrf/frf*** *Constraints can be accomplished in* $O(|\textbf{EF}| \cdot k \cdot |\mathcal{O}| \cdot \Delta)$ *time (provided the weight function* W *can be computed in constant time).*

The number of variables for either set of constraints is related to the size of L - which depends on the number of observations, spacing of $\mathcal{S}$, and α, β.

Proposition 56. *The* ***wrf/frf*** *Constraints have* $O(|\mathcal{O}| \cdot \Delta)$ *variables and* $1 + |\mathcal{O}|$ *constraints.*

The MILPs for ***wrf*** and ***frf*** appear non-linear as the objective function is fractional. However, as the denominator is non-zero and strictly positive, the Charnes-

Cooper transformation [22] allows us to quickly (in the order of number of constraints multiplied by the number of variables) transform the constraints into a purely integer-linear form. Many linear and integer-linear program solvers include this transformation in their implementation.

Proposition 57. *The **wrf/frf** constraints can be transformed into a purely linear-integer form in $O(|\mathcal{O}|^2 \cdot \Delta)$ time.*

We note that a linear relaxation of any of the above three constraints can yield a lower bound on the objective function in $O(|L|^{3.5})$ time.

Proposition 58. *Given the constraints of Definition 70 or Definition 71, if we consider the linear program formed by setting all X_i variables to be in $[0, 1]$, then the value returned by the objective function will be a lower bound on the value returned by the objective function for the mixed integer-linear constraints, and this value can be obtained in $O(|\mathcal{O}|^{3.5} \cdot \Delta^{3.5})$ time.*

Likewise, if we solve the mixed integer linear program with a reduced number of variables, we are guaranteed that the solution will cause the objective function to be an upper bound for the original set of constraints.

Proposition 59. *Consider the MILPs in Definition 70 and Definition 71. Suppose $L' \subset L$ and every variable X_i associated with some $p_i \in L'$ is set to 0. The resulting solution is an upper bound on the objective function for the constraints solved on the full set of variables.*

6.4.4 Correctly Reducing the Number of Variables for ***crf***

As the complexity of solving MILPs is closely related to the number of variables in the MILP, the goal of this section is to reduce the number of variables in the MILP associated above with the ***crf*** reward function. In this section, *we show that if we can find a certain type of explanation called a δ-core optimal explanation, then we can "build-up" an optimal adversarial strategy in polynomial time.* It also turns out that finding these special explanations can be accomplished using a MILP which will often have significantly less variables than the MILP's of the last section. First, we consider the ***wrf*** constraints applied to ***crf*** which is a special case of ***wrf***. The objective function for this caseis:

$$\sum_{\mathsf{ex\_fcn}_j \in \mathbf{EF}} \left(\mathsf{exfd}(\mathsf{ex\_fcn}_j) \cdot \sum_{p_i \in L} \left(X_i \cdot \left(\frac{c_{i,j}}{\sum_{p_i \in L} X_i} \right) \right) \right)$$

where for each $p_i \in L$ and $\mathsf{ex\_fcn}_j \in \mathbf{EF}$, $c_{i,j} = 1$ iff $\exists p' \in \mathsf{ex\_fcn}(\mathcal{O}, k)$ s.t. $d(p', p_i) \leq$ *dist* and 0 otherwise. If we re-arrange the objective function, we see that with each X_i - variable associated with point $p_i \in L$, there is an associated constant - $const_i$:

$$const_i = \sum_{\mathsf{ex\_fcn}_j \in \mathbf{EF}} \mathsf{exfd}(\mathsf{ex\_fcn}_j) \cdot c_{i,j}.$$

This lets us re-write the objective function as:

$$\frac{\sum_{p_i \in L} X_i \cdot const_i}{\sum_{p_i \in L} X_i}.$$

Example 6.4.4. *Continuing from Example 6.4.3, $const_i = 0.5$ for the following elements: $\{p_{33}, p_{35}, p_{37}, p_{42}, p_{43}, p_{44}, p_{47}, p_{49}, p_{50}, p_{52}, p_{56}\}$; $const_i = 1$ for these elements: $\{p_{40}, p_{41}, p_{45}, p_{46}, p_{48}\}$, and 0 for all others.*

In many covering problem where we wish to find a cover of minimal cardinality, we could reduce the number of variables in the integer program by considering equivalent covers as duplicate variables. However, for OAS, this technique can not be easily applied. The reason for this is because an optimal adversarial explanation is not necessarily irredundant (see Definition 58, page 260). Consider the following. Suppose, we wish to find an optimal adversarial strategy of size k. Let P be a irredundant cover of size $k-1$. Suppose there is some element $p' \in P$ that covers only one observation - o'. Hence, there is no $p \in P - \{p'\}$ that covers o' by the definition of an irredundant cover. Suppose there is also some $p'' \notin P$ that also covers o'. Now, let $m = \sum_{p_i \in P - p'} const_i$. Let the $const'$ be the value associated with both p' and p''. Consider the scenario where $const' < \frac{m}{k-2}$. Suppose by way of contradiction, that the optimal irredundant cover is also the optimal adversarial strategy. Then, by the definition of an optimal adversarial strategy we know that the set P is more optimal than $P \cup \{p''\}$. This would mean that $\frac{m+const'}{k-1} < \frac{m+2 \cdot const'}{k}$. This leads us to infer that $m < const' \cdot (k-2)$, which clearly contradicts $const' < \frac{m}{k-2}$. It is clear that a solution to OAS need not be irredundant.

However, we do leverage the idea of an irredundant cover in a different exact approach in this section which may provide a speedup over the exact algorithms of the previous section. The main intuition is that each OAS solution contains an irredundant cover, and if we find such a cover, we can build an optimal adversarial strategy in polynomial time. First, we define a *core* explanation.

Definition 72 (Core Explanation)**.** *Given an observation set $\mathcal{O}$ and set L of possible* *partners, an explanation $\mathcal{E}_{core}$ is a* ***core explanation*** *iff:*

1. *There are no two elements $p, p' \in \mathcal{E}_{core}$ such that $\forall o \in \mathcal{O}$ s.t. o, p are partners, then o, p' are also partners.*
2. *For any $p_i \in \mathcal{E}_{core}$, there does not exist $p_j \in L$ such that:*
   - *$\forall o \in \mathcal{O}$ s.t. o, p_i are partners, then o, p_j are also partners.*
   - *$const_j < const_i$*

We now show that any optimal adversarial strategy contains a subset that is a core explanation.

Theorem 31. *If $\mathcal{E}_{gt}$ is an optimal adversarial strategy, there exists a core explanation $\mathcal{E}_{core} \subseteq \mathcal{E}_{gt}$.*

Example 6.4.5. *Continuing from Example 6.4.4, consider the set $\mathcal{E}_{gt} \equiv \{p_{34}, p_{38}, p_{57}\}$ (which would correspond to drug lab locations as planned by the cartel). Later, we show that this is an optimal adversarial strategy (the expected adversarial detriment associated with $\mathcal{E}_{gt}$ is 0). Consider the subset p_{34}, p_{38}. As p_{34} explains observations o_3, o_4, o_5 and p_{38} explains observations o_1, o_2, this set is also an explanation. Obviously, it is of minimal cardinality. Hence, the set $\{p_{34}, p_{38}\}$ is a* ***core explanation*** *of $\mathcal{E}_{gt}$.*

Suppose we have an oracle that, for a given k, $\mathcal{O}$, and exfd returns a core explanation $\mathcal{E}_{core}$ that is guaranteed to be a subset of the optimal adversarial strategy associated with k, $\mathcal{O}$, and exfd. The following theorem says we can find the optimal

Algorithm 21 BUILD-STRAT

```
INPUT: Partner list $L$, core explanation $\mathcal{E}_{core}$, natural number $k$

OUTPUT: Optimal adversarial strategy $\mathcal{E}_{gt}$

  1. If $|\mathcal{E}_{core}| = k$, return $\mathcal{E}_{core}$
  2. Set $\mathcal{E}_{gt} = \mathcal{E}_{core}$. Let $k' = |\mathcal{E}_{core}|$
  3. Sort the set $L - \mathcal{E}_{core}$ by $const_i$. Let $L' = \{p_1, \ldots, p_{k-k'}\}$ be the $k - k'$ elements
     of this set with the lowest values for $cosnt_i$
  4. For each $p_i \in L'$ let $P_i$ be the set $\{p_1, \ldots, p_i\}$
  5. For each $P_i$ let $S_i = \sum_{j \leq i} const_j$
  6. Let $ans = \min_{p_i \in L'}(\{\frac{k' \cdot \mathsf{EXR}^{(\mathit{rf})}(\mathsf{exfd}, \mathcal{E}_{core}) + S_i}{k'+i}\})$
  7. Let $P_{ans}$ be the $P_i$ associated with $ans$
  8. If $const_1 \geq \mathsf{EXR}^{(\mathit{rf})}(\mathsf{exfd}, \mathcal{E}_{core})$, return $\mathcal{E}_{core}$, else return $\mathcal{E}_{core} \cup P_{ans}$
```

adversarial strategy in polynomial time. The key intuition is that we need not concern ourselves with covering the observations as $\mathcal{E}_{core}$ is an explanation. The algorithm BUILD-STRAT follows from this theorem.

Theorem 32. *If there is an oracle that for any given k, $\mathcal{O}$, and* exfd *returns a core explanation $\mathcal{E}_{core}$ that is guaranteed to be a subset of the optimal adversarial strategy associated with k, $\mathcal{O}$, and* exfd*, then we can find an optimal adversarial strategy in $O(\Delta \cdot |\mathcal{O}| \cdot \log(\Delta \cdot |\mathcal{O}|) + (k - |\mathcal{E}_{core}|)^2)$ time.*

We now introduce the notion of δ-core optimal. Intuitively, this is a core explanation of cardinality exactly δ that is optimal wrt expected adversarial detriment compared to all other core explanations of that cardinality.

Definition 73. *Given exfd, a core explanation, $\mathcal{E}_{core}$, is δ-**core optimal** iff:*

- $|\mathcal{E}_{core}| = \delta$
- *There does not exist another core explanation, $\mathcal{E}'_{core}$ of cardinality exactly δ, such that $\mathsf{EXR}^{(\mathsf{rew}(\mathcal{O},\delta))}(\mathsf{exfd}, \mathcal{E}'_{core}) < \mathsf{EXR}^{(\mathsf{rew}(\mathcal{O},\delta))}(\mathsf{exfd}, \mathcal{E}_{core})$*

From this, we obtain the following lemma that tells us that an OAS must contain a core explanation that is δ-core optimal.

Lemma 20. *Given an optimal adversarial strategy, $\mathcal{E}_{gt}$, if core explanation $\mathcal{E}_{core}$, of size δ, is a subset of $\mathcal{E}_{gt}$, then $\mathcal{E}_{core}$ is δ-core optimal.*

We now present a set of linear constraints to find a δ-**core optimal** explanation. Of course we can easily adopt the constraints of the previous section, but this would offer us no improvement in performance. We therefore create an MILP that should have a significantly smaller number of variables in most cases. First, given a set of possible partners L, we define set L^* - the reduced partner set - which often will have a cardinality much smaller than L. Later, we use this set in a new set of constraints to find a δ-core optimal explanation. We define L^* below.

Definition 74 (Reduced Partner Set). *Given observations $\mathcal{O}$, and set of possible* *partners L, we define reduced partner set L^{**} as follows:*

$$\begin{aligned} L^{**} \;\equiv\; & \{p_i \in L \mid \nexists p_j \in L \text{ s.t. } (const_j < const_i) \wedge (\forall o \in \mathcal{O} \text{ s.t. } o, p_i \text{ are partners}, \\ & o, p_j \text{ are also partners})\} \end{aligned}$$

We define L^* as follows:

$$\begin{aligned} L^{*} \;\equiv\; & \{p_i \in L^{**} \mid \nexists p_j \in L^{**} \text{ s.t. } (const_j = const_i) \wedge (\forall o \in \mathcal{O} \text{ s.t. } o, p_i \text{ are partners}, \\ & o, p_j \text{ are also partners})\} \end{aligned}$$

Lemma 21. *1. If explanation $\mathcal{E}$ is δ-core optimal, then $\mathcal{E} \subseteq L^{**}$.*

2. If for some natural number δ, there exists an explation of size δ, then there exists a δ-core optimal explanation $\mathcal{E}$ s.t. $\mathcal{E} \subseteq L^*$.

Example 6.4.6. *Let us continue from Example 6.4.5. Based on pre-processing and the computation of $const_i$, we can easily produce the data of Table 6.1 in polynomial time. Based on this, we obtain a **reduced partner set** $L^* \equiv \{p_{34}, p_{38}, p_{57}\}$.*

We now present the δ-core constraints. Notice that the cardinality requirement in these constraints is "=" and not "$\leq$". This is because Lemma 20 ensures us of a core-explanation that is δ-core optimal, meaning that the core explanation must have cardinality exactly δ. This also allows us to eliminate variables from the denominator of the objective function, as the denominator must equal δ as well.

Definition 75 (δ-core MILP)**.** *Given parameter δ, and reduced partner set L^*, we define the δ-core constraints by first associating a variable X_i with each $p_i \in L^*$. Then: Minimize:*

$$\frac{1}{\delta} \sum_{p_i \in L^*} X_i \cdot const_i$$

Supported Observations	$const_i = 0$	$const_i = 0.5$	$const_i = 1$
o_1	$p_4 - p_6, p_{12} - p_{16}, p_{22} - p_{23}, p_{30} - p_{31}$	p_{44}	
o_1, o_2	p_{38}	p_{37}, p_{52}	p_{45}, p_{46}
o_2	p_{64}, p_{67}	p_{47}	
o_2, o_3	p_{57}		
o_3	$p_{17} - p_{19}, p_{24} - p_{26}, p_{32}, p_{39}, p_{58} - p_{59}$		
o_3, o_4	$p_{27} - p_{28}$	p_{33}	
o_4	$p_1 - p_3, p_7 - p_{11}, p_{20} - p_{21}, p_{29}, p_{51}$	p_{50}	
o_3, o_4, o_5	$p_{34}, p_{53} - p_{54}$	p_{49}	$p_{40} - p_{41}$
o_5	$p_{36}, p_{60} - p_{66}$	p_{35}	
o_4, o_5		$p_{42} - p_{43}$	
o_3, o_5	p_{55}	p_{56}	p_{48}

Table 6.1: The set L partitioned by $const_i$ and supported observations.

subject to:

1. $X_i \in \{0,1\}$
2. *Constraint* $\sum_{p_i \in L} X_i = \delta$
3. *For each* $o_j \in \mathcal{O}$, *add constraint*

   $\sum_{p_i \in L^* d(o_j,p_i) \in [\alpha,\beta]} X_i \geq 1$

Example 6.4.7. *Using set L^* from Example 6.4.6, we can create δ-core constraints as follows:*

Minimize

$$\frac{1}{\delta}\left(X_{34} \cdot const_{34} + X_{38} \cdot const_{38} + X_{57} \cdot const_{57}\right)$$

subject to:

1. $X_{34}, X_{38}, X_{57} \in \{0,1\}$
2. $X_{34} + X_{38} + X_{57} = \delta$
3. $X_{38} \geq 1$ *(for observation o_1)*
4. $X_{38} + X_{57} \geq 1$ *(for observation o_2)*
5. $X_{34} + X_{57} \geq 1$ *(for observation o_3)*
6. $X_{34} \geq 1$ *(for observations o_4, o_5)*

In the worst case, the set $L^* \equiv L$. Hence, we can assert that:

Proposition 60. *The δ-core constraints require $O(\Delta \cdot |\mathcal{O}|)$ variables and $1 + |\mathcal{O}|$ constraints.*

Proposition 61. *Given δ-core constraints:*

1. *Given set δ-core optimal explanation $\mathcal{E}_{core} \equiv \{p_1, \ldots, p_n\}$, if variables $X_1, \ldots, X_n$ - corresponding with elements in $\mathcal{E}_{gt}$ are set to 1 - and the rest of the variables are set to 0, the objective function of the constraints will be minimized.*

2. *Given the solution to the constraints, if for every $X_i = 1$, we add point p_i to set $\mathcal{E}_{core}$, then $\mathcal{E}_{core}$ is a δ-core optimal solution.*

We now have all the pieces required to leverage core-explanation and reduced partner sets to find an optimal adversarial strategy. By Theorem 6.4.5, we know that any optimal adversarial strategy must have a core explanation. Further, by Lemma 20, such a core explanation is δ-core optimal. Using a (usually) much smaller mixed-integer-linear program, we can find such an explanation. We can then find the optimal adversarial strategy in polynomial time using BUILD STRAT. Though we do not know what δ is, we know it must be in the range $[1, k]$. Further, using a relaxation of the OPT-KSEP-IPC constraints for solving geospatial abduction problems (as presented in [158], we can easily obtain a lower bound tighter than 1 on δ. Hence, if we solve k such (most likely, small) mixed-integer-linear programs, we are guaranteed that at least one of them must be a core explanation for an optimal adversarial strategy. We note that these k MILP's can be solved in parallel (and the following k instances of BUILD-STRAT can also be run in parallel as well). An easy comparison of the results of the parallel processes would be accomplished at the end. As L^* is likely to be significantly smaller than L, this could yield a significant reduction in complexity. Further, various relaxations of this technique can be used - i.e. only using one value of δ.

Example 6.4.8. *Continuing from Example 6.4.7 where the cartel members are attempting to find an OAS to best position drug laboratories, suppose they used the relaxation of* OPT-KSEP-IPC *(from [158]) to obtain a lower bound on the cardinality of an explanation and found it to be* 2*. With* $k = 3$*, they would solve two MILP's of the form of Example 6.4.7 - one with* $\delta = 2$ *and one with* $\delta = 3$*. The solution to the first MILP would set* X_{34} *and* X_{38} *both to* 1 *while the second MILP would set* X_{34}, X_{38}*, and* X_{57} *all to* 1*. As the expected adversarial detriment for both solutions is* 0*, they are both optimal and running* BUILD-STRAT *is not necessary. Either* $\{p_{34}, p_{38}\}$ *or* $\{p_{34}, p_{38}, p_{57}\}$ *can be returned as an OAS.*

6.5 Finding a Counter-Adversary Strategy

Now that we have examined ways in which the adversary can create a strategy based on probabilistic knowledge of the agent, we consider how the agent can devise an "optimal" strategy to counter the adversary. As before, we use a special case of expected reward (Definition 6.3.1 from Section 67.

Definition 76 (Expected Agent Benefit)**.** *Given a reward function* ***rf****, and explanation function distribution* **exfd***, the* expected agent benefit *is the function* $\mathit{EXB}^{(\boldsymbol{rf})} : 2^{\mathcal{S}} \times \textbf{EFD} \rightarrow [0, 1]$ *defined as follows:*

$$\mathit{EXB}^{(\boldsymbol{rf})}(\mathcal{C}, \mathit{exfd}) = \sum_{\mathit{ex\_fcn} \in \textbf{EF}} \boldsymbol{rf}(\mathit{ex\_fcn}(\mathcal{O}, k), \mathcal{C}) \cdot \mathit{exfd}(\mathit{ex\_fcn})$$

Example 6.5.1. *Following from Examples 6.2.1 and 6.3.4, suppose drug-enforcement agents have information that the cartel is placing drug labs according to exfd_{drug}. (such information could come from multiple runs of the* GREEDY-KSEP-OPT2 *algorithm of [158]). The drug-enforcement agents wish to consider the set $\mathcal{C} \equiv \{p_{41}, p_{52}\}$. First, they must calculate the reward associated with each explanation function (note that $k = 3, dist = 100$ and* ***rf*** *=* ***crf****).*

$$\begin{aligned}\boldsymbol{crf}^{(dist)}(\mathsf{ex\_fcn}_1(\mathcal{O}, 3), \{p_{41}, p_{52}\}) &= 0.67\\ \boldsymbol{crf}^{(dist)}(\mathsf{ex\_fcn}_2(\mathcal{O}, 3), \{p_{41}, p_{52}\}) &= 0.5\end{aligned}$$

(as an aside, we would like to point out the asymmetry in ***crf*** *- compare these computations with the results of Example 6.4.1). Hence, $\mathsf{EXB}^{(\boldsymbol{crf})}(\{p_{41}, p_{52}\}, \mathsf{exfd}_{drug}) =$ 0.634.*

We now define a maximal counter-adversary strategy. This is the agent's **best response** to the mixed strategy of an adversary.

Definition 77 (Maximal Counter-Adversary Strategy (MCA))**.** *Given a reward function* ***rf*** *and explanation function distribution* exfd*, a* ***maximal counter-adversary strategy****, $\mathcal{C}$, is a subset of $\mathcal{S}$ such that $\mathsf{EXB}^{(\boldsymbol{rf})}(\mathcal{C}, \mathsf{exfd})$ is maximized.*

Note that MCA does not include a cardinality constraint. This is because we do not require reward functions to be monotonic. In the monotonic case, we can trivially return all feasible points in $\mathcal{S}$ and be assured of a solution that maximizes the expected agent benefit. Therefore, for the monotonic case, we include an extra parameter $B \in \{1, \ldots, |\mathcal{S}|\}$ (for "budget") which will serve as a cardinality requirement for $\mathcal{C}$. This cardinality requirement for $\mathcal{C}$ is necessarily the same as for $\mathcal{E}_{gt}$

as the agent and adversary may have different sets of resources. Also, we do not require that $\mathcal{C}$ be an explanation. We discuss the special case where the solution to the MCA problem is required to be an explanation in the appendix.

6.5.1 The Complexity of Finding a Maximal Counter-Adversary Strategy

We now formally define the problem of finding a maximal counter-adversary strategy.

<u>**MCA Problem**</u>

INPUT: Space $\mathcal{S}$, feasibility predicate, feas, real numbers α, β, set of observations, $\mathcal{O}$, natural numbers k, B, reward function $\boldsymbol{rf}$, and explanation function distribution exfd.

OUTPUT: The maximal counter-adversary strategy, $\mathcal{C}$.

MCA is NP-hard via a reduction of the GCD problem.

Theorem 33. *MCA is NP-hard.*

The proof of the above result shows that MCA is NP-hard even if the reward function is monotonic. Later, in Section 6.5.3, we also show that MCA can encode the NP-hard MAX-K-COVER problem [46] as well (which provides an alternate proof for NP-hardness of MCA). We now present the decision problem associated with MCA and show that it is NP-complete under reasonable conditions.

<u>MCA-DEC</u>

INPUT: Space $\mathcal{S}$, feasibility predicate, feas, real numbers α, β, set of observations, $\mathcal{O}$, natural numbers k, B, reward function $\boldsymbol{rf}$, explanation function distribution exfd, and number $R \in [0, 1]$.

OUTPUT: The counter-adversary strategy, $\mathcal{C}$ such that $\mathsf{EXB}^{(\boldsymbol{rf})}(\mathcal{C}, \mathsf{exfd}) \geq R$.

Theorem 34. ***MCA-DEC*** *is NP-complete, provided the reward function can be evaluated in PTIME.*

Not only is **MCA-DEC** NP-hard, under the same assumptions as above, the counting version of the problem is #P-complete and moreover, it has no fully polynomial random approximation scheme.

Theorem 35. *Counting the number of strategies that provide a "yes" answer to* ***MCA-DEC*** *is #P-complete and has no FPRAS unless NP==RP.*

Theorem 35 tells us that MCA may not have a unique solution. Therefore, setting up a mixed-strategy of all MCA's to determine the "best response" to the MCA of an agent by an adversary would be an intractable problem. This mirrors our result of the previous section (Theorem 30, page 273).

6.5.2 MCA in the General Case: Exact and Approximate Algorithms

We now describe exact and approximate algorithms for finding a maximal counter-adversary strategy in the general case. Note that throughout this section (as well as in Section 6.5.3), we assume that the same pre-processing for **OAS** is used (cf. Section 6.4.2). We will use the symbol L to refer to the set of all possible partners.

An Exact Algorithm For MCA. A naive, exact, and straightforward approach to the MCA problem would simply consider all subsets of L and pick the one which maximizes the expected agent benefit. Obviously, this approach has a complexity $O(\sum_{i=0}^{|\mathcal{S}|} \binom{|L|}{i})$ and is not practical. This is unsurprising as we showed this to be an NP-complete problem.

Approximation in the General Case. Despite the impractical time complexity associated with an exact approach, it is possible to approximate MCA with guarantees – even in the general case. This is due to the fact that when exfd is fixed, the expected agent benefit is submodular.

Theorem 36. *For a fixed $\mathcal{O}, k\mathsf{exfd}$, the expected agent benefit, $\mathsf{EXB}^{(rf)}(\mathcal{C}, \mathsf{exfd})$ has the following properties:*

1. $\mathsf{EXB}^{(rf)}(\mathcal{C}, \mathsf{exfd}) \in [0, 1]$

2. For $\mathcal{C} \subseteq \mathcal{C}'$ and some point $p \in \mathcal{S}$ where $p \notin \mathcal{C}'$, the following is true:

$$\mathsf{EXB}^{(\boldsymbol{rf})}(\mathcal{C} \cup \{p\}, \mathsf{exfd}) - \mathsf{EXB}^{(\boldsymbol{rf})}(\mathcal{C}, \mathsf{exfd}) \geq \mathsf{EXB}^{(\boldsymbol{rf})}(\mathcal{C}' \cup \{p\}, \mathsf{exfd}) - \mathsf{EXB}^{(\boldsymbol{rf})}(\mathcal{C}', \mathsf{exfd})$$

(i.e. expected agent benefit is sub-modular for MCA)

It follows immediately that MCA reduces to the maximization of a submodular function. We now present the MCA-LS algorithm that leverages this submodularity.

The following two propositions leverage Theorem 36 and Theorem 3.4 of [47].

Proposition 62. *MCA-LS has time complexity of $O(\frac{1}{\epsilon} \cdot |L|^3 \cdot F(\mathsf{exfd}) \cdot \lg(|L|)$ where $F(\mathsf{exfd})$ is the time complexity to compute $\mathsf{EXB}^{(\boldsymbol{rf})}(\mathcal{C}, \mathsf{exfd})$ for some set $\mathcal{C} \subseteq L$.*

Proposition 63. *MCA-LS is an $(\frac{1}{3} - \frac{\epsilon}{|L|})$-approximation algorithm for MCA.*

Example 6.5.2. *Let us consider our running example where drug-enforcement agents are attempting to locate illegal drug laboratories in the area depicted in Figure 6.1. The agents have information that there are k or less drug laboratories that support the poppy fields (set of observations $\mathcal{O}$) and that they are positioned according to exfd_{drug} (see Example 6.3.4, page 268). The agents wish to find a maximal counter-adversarial strategy using the **prf** reward function (see page 60). They decide to use MCA-LS to find such a strategy with $\epsilon = 0.1$. Initially (at line 3), the algorithm selects point p_{48} (renumbering as p_1, note that in this example we shall use p_i and inc_i numbering based on Example 6.2.1 rather than what the algorithm uses). Hence, $inc_{40} = 0.208$ and $cur\_val = 0.708$. As the elements are sorted, the next point to be considered in the loop at line 4 is p_{40} which has an incremental increase of 0, so it is not picked. It then proceeds to point p_{41} - which gives an incremental increase*

Algorithm 22 (MCA-LS)

INPUT: Reward function $\boldsymbol{rf}$, set $\mathcal{O}$ of observations, explanation function distribution exfd, possible partner set L, real number $\epsilon > 0$

OUTPUT: Set $\mathcal{C} \subset \mathcal{S}$

```
1. Set $\mathcal{C}^* = L$, for each $p_i \in \mathcal{C}^*$ let $inc_i = \mathsf{EXB}^{(\boldsymbol{rf})}(\{p\}, \mathsf{exfd}) - \mathsf{EXB}^{(\boldsymbol{rf})}(\emptyset, \mathsf{exfd})$.

2. Sort the $p_i$'s in $\mathcal{C}^*$ from greatest to least by $inc_i$ (i.e. $p_1$ is the element with the greatest $inc_i$).

3. $\mathcal{C} = \{p_1\}$, $\mathcal{C}^* = \mathcal{C}^* - \{p_1\}$, $cur\_val = inc_1 + \mathsf{EXB}^{(\boldsymbol{rf})}(\emptyset, \mathsf{exfd})$, $flag1 = \mathsf{true}$, $i = 2$

4. While $flag1$

   (a) $new\_val = cur\_val + inc_i$

   (b) If $new\_val > (1 + \frac{\epsilon}{|L|^2}) \cdot cur\_val$ then

       i. If $\mathsf{EXB}^{(\boldsymbol{rf})}(\mathcal{C} \cup \{p_i\}, \mathsf{exfd}) > (1 + \frac{\epsilon}{|L|^2}) \cdot \mathsf{EXB}^{(\boldsymbol{rf})}(\mathcal{C}, \mathsf{exfd})$ then:
          $\mathcal{C} = \mathcal{C} \cup \{p_i\}$, $\mathcal{C}^* = \mathcal{C}^* - \{p_i\}$, $cur\_val = \mathsf{EXB}^{(\boldsymbol{rf})}(\mathcal{C} \cup \{p_i\}, \mathsf{exfd})$

   (c) If $new\_val \leq (1 + \frac{\epsilon}{|L|^2}) \cdot cur\_val$ **or** if $p_i$ is the last element then

       i. $j = 1$, $flag2 = \mathsf{true}$, number each $p_j \in \mathcal{C}$

       ii. While $flag2$

           A. If $\mathsf{EXB}^{(\boldsymbol{rf})}(\mathcal{C} - \{p_j\}, \mathsf{exfd}) > (1 + \frac{\epsilon}{|L|^2}) \cdot \mathsf{EXB}^{(\boldsymbol{rf})}(\mathcal{C}, \mathsf{exfd})$ then:
              $\mathcal{C} = \mathcal{C} - \{p_j\}$, $cur\_val = \mathsf{EXB}^{(\boldsymbol{rf})}(\mathcal{C} - \{p_j\}, \mathsf{exfd})$
              For each $p_i \in \mathcal{C}^*$ let $inc_i = \mathsf{EXB}^{(\boldsymbol{rf})}(\mathcal{C} \cup \{p_i\}, \mathsf{exfd}) - \mathsf{EXB}^{(\boldsymbol{rf})}(\mathcal{C}, \mathsf{exfd})$.
              Sort the $p_i$'s in $\mathcal{C}^*$ from greatest to least by $inc_i$
              $i = 0$, $flag2 = \mathsf{false}$

           B. Else,
              If $p_j$ was the last element of $\mathcal{C}$ then set $flag1, flag2 = \mathsf{false}$
              Otherwise, $j++$

   (d) $i++$

5. If $\mathsf{EXB}^{(\boldsymbol{rf})}(L - \mathcal{C}, \mathsf{exfd}) > \mathsf{EXB}^{(\boldsymbol{rf})}(\mathcal{C}, \mathsf{exfd})$ then set $\mathcal{C} = L - \mathcal{C}$

6. Return $\mathcal{C}$
```

of 0.084 *and is added to* $\mathcal{C}$ *so cur_val* $= 0.792$. *Point* p_{45} *is considered next, which gives an incremental increase of* 0.208 *and is picked, so now cur_val* $= 1.0$. *The algorithm then considers point* p_{46}*, which does not afford any incremental increase. After considering points* $p_{33}, p_{35}, p_{37}, p_{42}, p_{43}, p_{44}, p_{47}, p_{49}, p_{50}, p_{52}, p_{56}$ - *and finds the all give a negative incremental increase (and thus, are not picked), the algorithm finds that the old incremental increase of the next element,* p_1*, would cause the "if" statement at line 4c to be true, thus proceeding to the inner loop inside that "if" statement (line 4(c)iiA). This loop considers if the removal of any picked elements* - p_{48}, p_{41}, p_{45} *causes the expected agent benefit to increase. However, in this example, if any of the elements are removed, the expected agent benefit decreases. Hence, the boolean* $flag1$ *is set to false and the algorithm exits the outer loop. The algorithm then returns the set* $\mathcal{C} \equiv \{p_{48}, p_{41}, p_{45}\}$ *which is optimal.*

6.5.3 Finding a Maximal Counter-Adversary Strategy, the Monotonic Case

In the previous section we showed that a $\frac{1}{3}$ approximate solution to MCA can be found in polynomial time *even without any monotonicity restriction.* In this section, we show that under the additional assumptions of monotonicity of reward functions, we can obtain a better 63% approximation ratio with a faster algorithm. Here, we also have the additional cardinality requirement of B for the set $\mathcal{C}$ (as described in Section 6.5). We first show that expected agent benefit is monotonic when the reward function is.

Corollary 12. *For a fixed $\mathcal{O}, k$exfd, if the reward function is monotonic, then the expected agent benefit, $\mathsf{EXB}^{(rf)}(\mathcal{C}, \mathsf{exfd})$ is also monotonic.*

Thus, when we have a monotonic reward function, the MCA problem reduces to the maximization of a monotonic, normalized[4] submodular function w.r.t. a uniform matroid[5] – this is a direct consequence of Theorem 36 and Corollary 12. Therefore, we can leverage the result of [127], to develop the MCA-GREEDY-MONO algorithm below. We improve performance by including "lazy evaluation" using the intuition is that the incremental increase caused by some point p at iteration i of the algorithm is greater than or equal to the increase caused by that point at a later iteration. As with MCA-LS, we also sort elements by the incremental increase, which may allow the algorithm to exit the inner-loop earlier. In most non-trivial instances of MCA, this additional sorting operation will not affect the complexity of the algorithm (i.e. under the assumption that the time to compute $\mathsf{EXB}^{(rf)}$ is greater than $\lg(|L|)$, we make this same assumption in MCA-LS as well).

Proposition 64. *The complexity of MCA-GREEDY-MONO is $O(B \cdot |L| \cdot F(\mathsf{exfd}))$ where $F(\mathsf{exfd})$ is the time complexity to compute $\mathsf{EXB}^{(rf)}(\mathcal{C}, \mathsf{exfd})$ for some set $\mathcal{C} \subseteq L$ of size B. In the first iteration of the algorithm,*

Corollary 13. *MCA-GREEDY-MONO is an $(\frac{e}{e-1})$-approximation algorithm for MCA (when the reward function is monotonic).*

In addition to the fact that MCA-GREEDY-MONO is an $(\frac{e}{e-1})$-approximation

[4]As we include zero-starting in our definition of monotonic.

[5]In our case, the uniform matroid consists of all subsets of L of size B or less.

Algorithm 23 (MCA-GREEDY-MONO)

INPUT: Monotonic reward function $\boldsymbol{rf}$, set $\mathcal{O}$ of observations, real number $B > 0$, explanation function distribution exfd, possible partner set L, real number $\epsilon > 0$

OUTPUT: Set $\mathcal{C} \subset \mathcal{S}$

```
1. Initialize $\mathcal{C} = \emptyset$ and $\mathcal{C}^* = L$
2. For each $p_i \in \mathcal{C}^*$, set $inc_i = 0$
3. Set $last\_val = \mathsf{EXB}^{(\boldsymbol{rf})}(\mathcal{C}, \mathsf{exfd})$
4. While $|\mathcal{C}| \leq B$
    (a) $p_{best} = \mathsf{null}$, $cur\_inc = 0$
    (b) For each $p_i \in \mathcal{C}^*$, do the following
        i. If $inc_i < cur\_inc$, break loop and goto line 4c.
        ii. Let $inc_i = \mathsf{EXB}^{(\boldsymbol{rf})}(\mathcal{C} \cup \{p\}, \mathsf{exfd}) - last\_val$
        iii. If $inc_i \geq cur\_inc$ then $cur\_inc = inc_i$ and $p_{best} = p$
    (c) $\mathcal{C} = \mathcal{C} \cup \{p_{best}\}$, $\mathcal{C}^* = \mathcal{C}^* - \{p_{best}\}$
    (d) Sort $\mathcal{C}^*$ in descending order by $inc_i$.
    (e) Set $last\_val = \mathsf{EXB}^{(\boldsymbol{rf})}(\mathcal{C}, \mathsf{exfd})$
5. Return $\mathcal{C}$
```

algorithm for MCA, it also provides the best possible approximation ratio unless $P = NP$. This is done by a reduction of MAX-K-COVER [46].

Theorem 37. *MCA-GREEDY-MONO provides the best approximation ratio for MCA (when the reward function is monotonic) unless $P = NP$.*

The following example illustrates how MCA-GREEDY-MONO works.

Example 6.5.3. *Consider the situation from Example 6.5.2, where the drug-enforcement agents are attempting to locate illegal drug labs. Suppose they want to locate the labs, but use the **crf** reward function, which is monotonic and zero-starting. They use the cardinality requirement $B = 3$ in MCA-GREEDY-MONO. After the first iteration of the loop at line 4, the algorithm selects point p_{48} as it affords an incremental increase of 0.417. On the second iteration, it selects point p_{46}, as it also affords an incremental increase of 0.417, so $last\_val = 0.834$. Once p_{46} is considered, the next point considered is p_{33}, which had a previous incremental increase (calculated in the first iteration) of 0.25, so the algorithm can correctly exit the loop to select the final element. On the last iteration of the outer loop, the algorithm selects point p_{35}, which gives an incremental increase of 0.166. Now the algorithm has a set of cardinality 3, so it exits the outer loop and returns the set $\mathcal{C} = \{p_{48}, p_{46}, p_{35}\}$, which provides an expected agent benefit of 1, which is optimal. Note that this would not be an optimal solution for the scenario in Example 6.5.2 which uses **prf** as p_{35} would incur a penalty (which it does not when using **crf** as in this example).*

6.6 Implementation and Experiments

In this section, we describe prototype implementations and experiments for solving the OAS and MCA problems. For OAS, we create a MILP for the ***crf*** case and reduce the number of variables with the techniques we presented in Section 6.4. For MCA, we implement both the MCA-LS and MCA-GREEDY-MONO.

We carried out all experiments for MCA on an Intel Core2 Q6600 processor running at 2.4GHz with 8GB of memory available, using code written in Java 1.6; all runs were performed in Windows 7 Ultimate 64-bit using a 64-bit JVM, and made use of a single core. We also used functionality from the previously-implemented SCARE software from Chapter 4 to calculate, for example, the set of all possible partners L.

Our experiments are based on 21 months of real-world Improvised Explosive Device (IED) attacks in Baghdad[6], see Chapter 4. The IED attacks in this 25×27 km region constitute our observations. The data also includes locations of caches associated with those attacks discovered by US forces. These constitute partner locations. We used data from the International Medical Corps to define feasibility predicates based on ethnic makeup, location of US bases, and geographic features. We overlaid a grid of 100m $\times$ 100m cells—about the size of a standard US city block. We split the data into two parts; the first 7 months of data were used as a "training" set to learn the $[\alpha, \beta]$ parameters and the next 14 months of data were used for the observations. We created an explanation function distribution based

[6]Attack and cache location data provided by the Institute for the Study of War.

on multiple runs of GREEDY-KSEP-OPT2 algorithm described in Chapter 4.

We also made use of classes and methods from our previously-implemented SCARE software from Chapter 4 to provide features such as pre-processing (see the discussion in Section 6.4.2, page 273). We carried out all experiments for OAS on an Intel Core2 Q6600 processor running at 2.4GHz with 8GB of memory available, using Java 1.6; all runs were performed in Windows 7 Ultimate 64-bit using a 64-bit JVM, and made use of a single core.

6.6.1 OAS Implementation

We now present experimental results for the version of OAS, with the ***crf*** reward function, based on the constraints in Definition 70 and variable-reduction techniques of Section 6.4.4. First, we discuss promising real-world results for the calculation of the reduced partner set L^*, described in Definition 72. Then, we show that an optimal adversarial strategy can be computed quite tractably using the methods discussed in Section 6.4.4. Our implementation was written on top of the QSopt[7] MILP solver and used 900 lines of Java code.

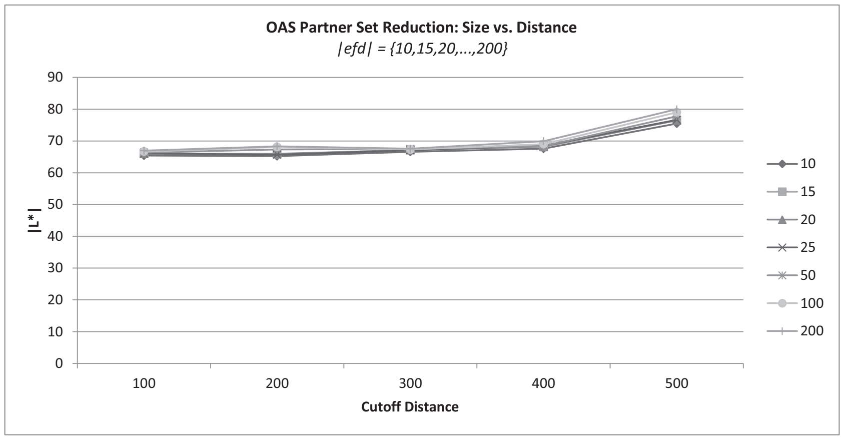
Reduced Partner Set. As discussed in Section 6.4.2, producing an optimal adversarial strategy for any reward function relies heavily on efficiently solving a (provably worst-case intractable) integer linear program. The number of integer variables in these programs is based solely on the size of the partner set L; as such, the *ability to experimentally solve OAS* relies heavily on the size of this set.

[7]http://www2.isye.gatech.edu/ wcook/qsopt/index.html

Our real-world data created a partner set L with cardinality 22,692. We then applied the method from Definition 72 to reduce this original set L to a smaller subset of possible partners L^*, while retaining the optimality of the final solution. This simple procedure, while dependent on the explanation function distribution exfd as well as the cutoff distance for ***crf***, always returned a reduced partner set L^* with cardinality between 64 and 81. This represents around a 99.6% decrease in the number of variables required in the subsequent integer linear programs!

Figure 6.4 provides more detailed accuracy and timing results for this reduction. Most importantly, regardless of parameters chosen, our real-world data is reduced by orders of magnitude across the board. We see a slight increase in the size of the reduced set L^* as the size of the explanation function distribution exfd increases. This can be traced back to the strict inequality in Definition 74. As we increase the number of nontrivial explanation functions in exfd, the number of nonzero constants $const_i$ increases. This results in a higher number of candidates for the intermediary set L^{**}. We see a similar result as we increase the penalizing cutoff distance. Again, this is a factor of the strict inequality in Definition 74 in conjunction with a higher fraction of nonzero $const_i$ constants.

Interestingly, Figure 6.4 shows a slight *decrease* in the runtime of the reduction as we increase the penalizing cutoff distance. Initially, this seems counterintuitive; with more nontrivial constants $const_i$, the construction of the intermediary set L^{**} requires more work. However, this extra work pays off during the computation of the final reduced set L^*. In our experiments, the reduction from L to L^{**} took less time than the final reduction from L^{**} to L^*. This is due to frequent short circuiting



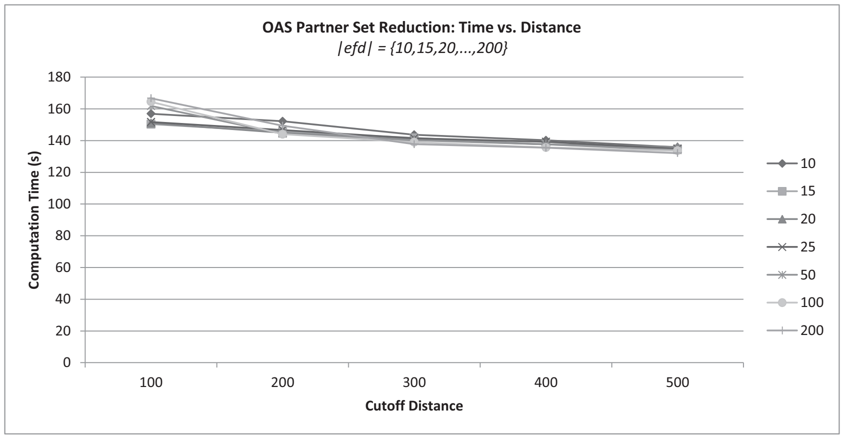


Figure 6.4: The size of the reduced partner set L^* (left) and the time required to compute this reduction (right). Regardless of parameters chosen, we see a 99.6% decrease in possible partners—as well as integer variables in our linear program—in under 3 minutes.

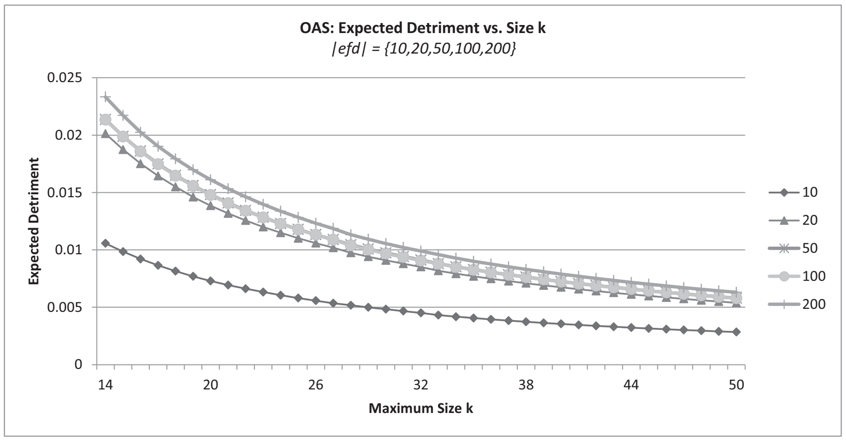
in the computation of the right-hand side of the conjunction during L^{**} creation. As we increase the penalizing cutoff distance, the size of L^{**} actually *decreases*, resulted in a decrease in the longer computation of L^*. As seen above, this decrease in L^{**} did not correspond to a decrease in the size of L^*.

Optimal Adversarial Strategy. Using the set L^*, we now present results to find an optimal adversarial strategy using δ-core optimal explanations. This is done by minimizing the MILP of Section 6.4.4, then feeding this solution into BUILD-STRAT. Since we do not know the value of δ in advance, we must perform this combined operation multiple times, choosing the best—lowest expected detriment—adversarial strategy as optimal.

A note on the lower bound for δ: as shown by Chapter 4, finding a *minimum-cardinality* explanation is NP-hard. Because of this, it is computationally difficult to find a tight lower bound for δ. However, this lower bound can be estimated empirically. For instance, for our set of real-world data from Baghdad, an explanation of cardinality below 14 has never been returned—even across tens of thousands of runs of GREEDY-KSEP-OPT2. Building on this strong empirical evidence, the minimum δ used in our experiments is 14.

Figure 6.5 shows both timing and expected detriment results as the size of the explanation function |exfd| and maximum strategy cardinality k are varied. Note that a lower expected detriment is better for the adversary, with zero representing no probability of partner discovery by the reasoning agent. As the adversary is allowed larger and larger strategies, its expected detriment smoothly decreases toward zero. Intuitively, as the number of nontrivially-weighted explanation functions in exfd increases, the expected detriment increases as well. This is a side effect of a larger |exfd| allowing the reasoning agent to cover a larger swath of partner locations.

Recall that, as the maximum k increases, we must solve linear programs for



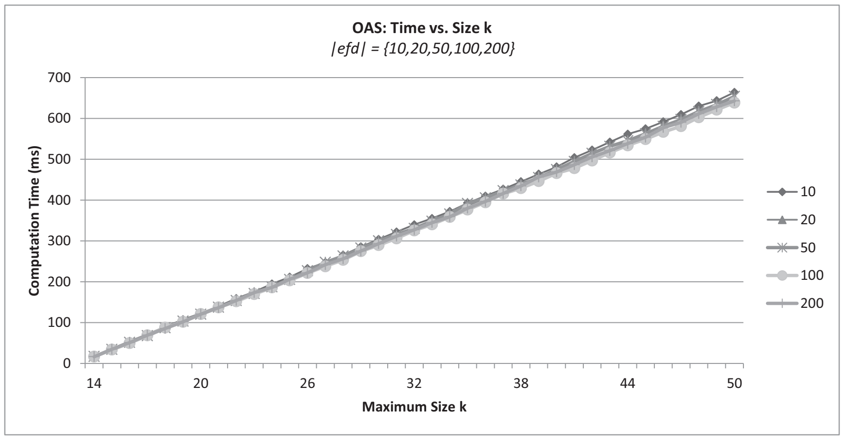


Figure 6.5: Expected detriment of the optimal adversarial strategy (left, lower is better) and the runtime of the integer linear program required to produce this strategy in milliseconds (right). Note the smooth decrease toward zero detriment as k increases, corresponding with a near-linear increase in total runtime.

each $\delta \in \{k_{low}, k\}$. This is mirrored in the timing results in Figure 6.5, which assumes $k_{low} = 14$. As k increases, we see a near linear increase in the total runtime of the set of integer programs. Due to the reduced set L^*, we are able to solve dozens of integer programs in less than 800ms; were we to use the unreduced partner set L, this would be intractable. Note that the runtime graph includes that of BUILD-STRAT which always ran in under sixteen milliseconds.

6.6.2 MCA Implementation

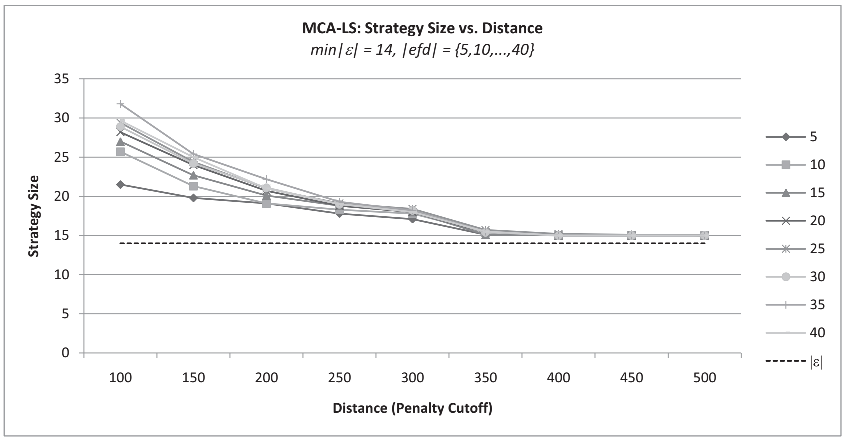
First, we briefly discuss an implementation of the naive MCA algorithm discussed in section 6.5.2. Next, we provide promising results for the MCA-LS algorithm using the ***prf*** reward function. Finally, we give results for the MCA-GREEDY-MONO using the monotonic ***crf*** reward function, and qualitatively compare and constrast the results from both algorithms.

MCA-Naive. The naive, exact solution to MCA—considering all subsets of L with cardinality $k_{\mathcal{C}}$ or more and picking the one which maximizes the expected agent benefit—is inherently intractable. This approach has a complexity $O(\binom{|L|}{k_{\mathcal{C}}})$, and is made worse by the large magnitude of the set L. In our experimental setup, we typically saw $|L| > 20,000$; as such, for even the trivially small $k_{\mathcal{C}} = 3$, we must enumerate and rank over a trillion subsets. For any realistic value of $k_{\mathcal{C}}$, this approach is simply unusable. Luckily, we will see that both MCA-LS and MCA-GREEDY-MONO provide highly tractable and accurate alternatives.

MCA-LS. In sharp contrast to the naive algorithm described above, the MCA-LS algorithm provides (lower-)bounded approximate results in a tractable manner. Interestingly, even though MCA-LS is an approximation algorithm, in our experiments on real-world data from Baghdad using the ***prf*** reward function, the algorithm returned strategies with an expected benefit of 1.0 on every run. Put simply, on our practical test data, MCA-LS always *completely maximized* the expected benefit. This significantly outperforms the lower-bound approximation ratio of 1/3. We would also like to point out that this is the first implementation (to the best of our knowledge) of the non-monotonic submodular maximization approximation algorithm of [47].

Since the expected benefit was maximal for every strategy $\mathcal{C}$ returned, we move to analyzing the particular structure of these strategies. Figure 6.6 shows a relationship between the size $|\mathcal{C}|$, the cutoff distance *dist*, and the cardinality of the expectation function distribution $|\mathsf{exfd}|$. Recall that ***prf*** penalizes any strategy that does not completely cover its input set of observations; as such, intuitively, we see that MCA-LS returns larger strategies as the penalizing cutoff distance decreases. If the algorithm can cover all possible partners across all expectation functions, it will not receive any penalty. Still, even when *dist* is 100m, the algorithm returns $\mathcal{C}$ only roughly twice the size as minimum-sized explanation found by GREEDY-KSEP-OPT2 (which, based on the analysis of Chapter 4, is very close to the minimum possible explanation). As the cutoff *dist* increases, the algorithm returns strategies with sizes converging, generally, to a baseline—the smallest-sized explanation found by the algorithm of Chapter 4, $|\mathcal{E}|$. This is an intuitive soft lower bound; given enough leeway from a large distance *dist*, a single point will cover all expected partners. This is not a strict lower bound in that, given two extremely close observations with similar expected partners, a single point may sufficiently cover both.

In Figure 6.7, we see results comparing overall computation time to both the



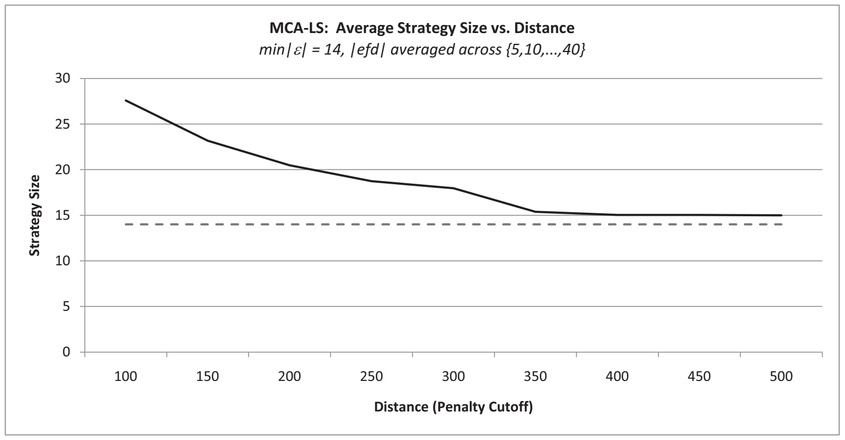
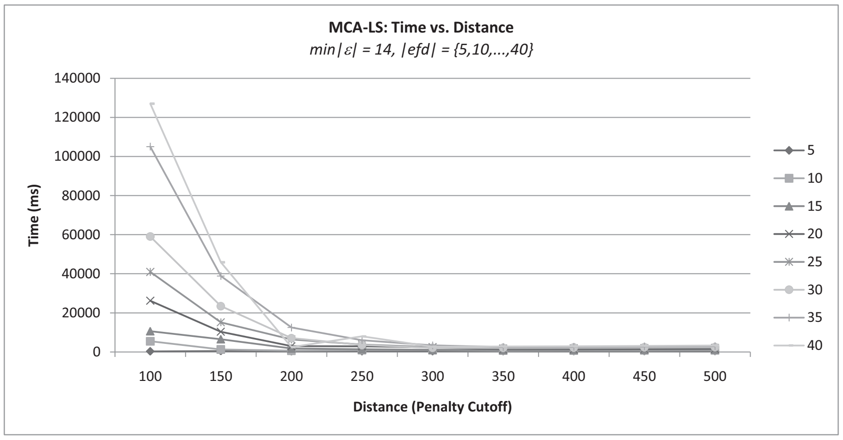


Figure 6.6: The average size of the strategy recommended by MCA-LS decreases as the distance cutoff increases. For these experiments, the minimum cardinality for a given explanation $\mathcal{E}$ considered is exfd was 14, which gives us a natural lower bound on the expected size of a strategy. Note the convergence to this bound at cutoff distances at and above 300 meters.

distance *dist* and the cardinality of exfd. For more strict (*i.e.*, smaller) values of *dist*, the algorithm—which, under ***prf***, is penalized for all uncovered observations across exfd—must spend more time forming a strategy $\mathcal{C}$ that minimizes penaliza-



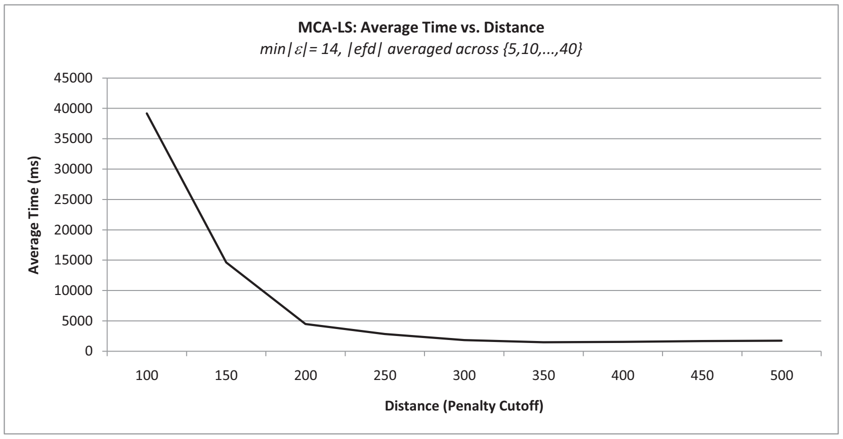
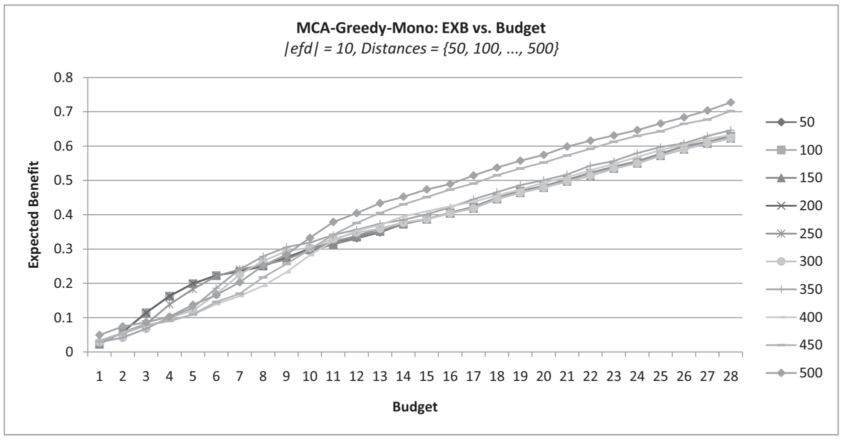


Figure 6.7: The runtime of MCA-LS decreases as the penalizing cutoff distance is relaxed. Note the relation to Figure 6.6; intuitively, larger recommended strategies tend to take longer to compute.

tion. Similarly, as the distance constraint is loosened, the algorithm completes more quickly. Finally, an increase in |exfd| results in higher computational cost; as explained in Proposition 62, this is due to an increase in $F(\mathsf{exfd})$, the time complexity of computing $\mathsf{EXB}^{(rf)}(\mathcal{C}, \mathsf{exfd})$. Comparing these results to Figure 6.6, we see that the runtime of MCA-LS is correlated to the size of the returned strategy $\mathcal{C}$.



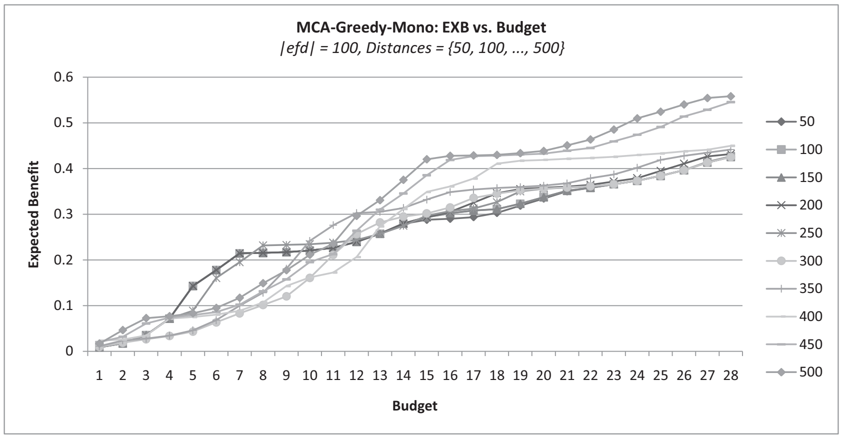


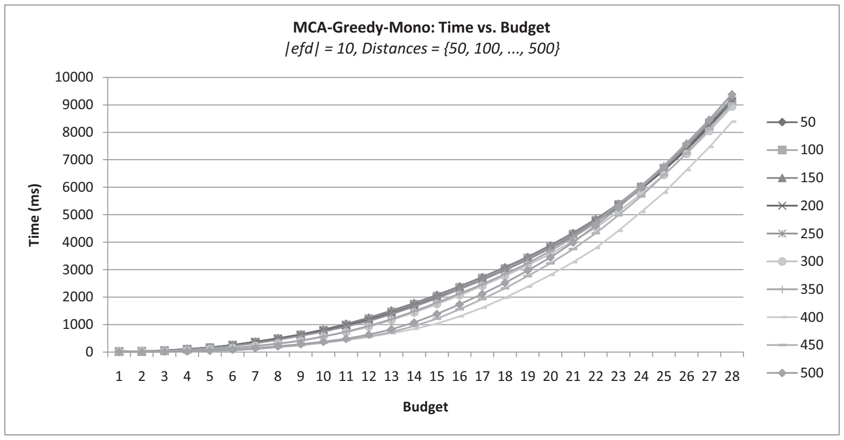
Figure 6.8: Expected benefit of the strategy returned by MCA-GREEDY-MONO as the budget increases, with |exfd| = 10 (left) and |exfd| = 100 (right). Note the decrease in expected benefit due to the increase in |exfd|. Similarly, note the increase in expected benefit given a larger cutoff distance.

MCA-GREEDY-MONO. As discussed in Section 6.5.3, MCA-GREEDY-MONO provides tighter approximation bounds than MCA-LS at the cost of a more restrictive (monotonic) reward function. For these experiments, we used the monotonic $\boldsymbol{rf} =$

crf. Recall that a trivial solution to MCA given a monotonic reward function is $\mathcal{C} = L$; as such, MCA-GREEDY-MONO uses a budget B to limit the maximum size $|\mathcal{C}| \ll |L|$. We varied this parameter $B \in \{1, \ldots, 28\}$.

Figure 6.8 shows the expected benefit $\mathsf{EXB}^{(\mathit{rf})}(\mathcal{C}, \mathsf{exfd})$ increase as the maximum allowed $|\mathcal{C}|$ increases. In general, the expected benefit of $\mathcal{C}$ increases as the distance constraint *dist* is relaxed. However, note the points with $B \in \{3, \ldots, 9\}$; we see that $\mathit{dist} \leq 100$ performs better than $\mathit{dist} > 100$. We believe this is an artifact of our real-world data. Finally, as $|\mathsf{exfd}|$ increases, the expected benefit of $\mathcal{C}$ converges more slowly to 1.0. This is intuitive, as a wider spread of possible partner positions will, in general, require a larger $|\mathcal{C}|$ to provide coverage.

Figure 6.9 shows that the runtime of MCA-GREEDY-MONO increases as predicted by Proposition 62. In detail, as we linearly increase budget B, we also linearly increase the runtime of our $F(\mathsf{exfd}) = \mathsf{EXB}^{(\mathit{rf})}(\mathcal{C}, \mathsf{exfd})$. In turn, the overall runtime $O(B \cdot |L| \cdot F(\mathsf{exfd}))$ increases quadratically in B, for our specific reward function. Finally, note the increase in runtime as we increase $|\mathsf{exfd}| = 10$ to $|\mathsf{exfd}| = 100$. Theoretically, this increases $F(\mathsf{exfd})$ linearly; in fact, we see almost exactly a ten-fold increase in runtime given a ten-fold increase in $|\mathsf{exfd}|$.



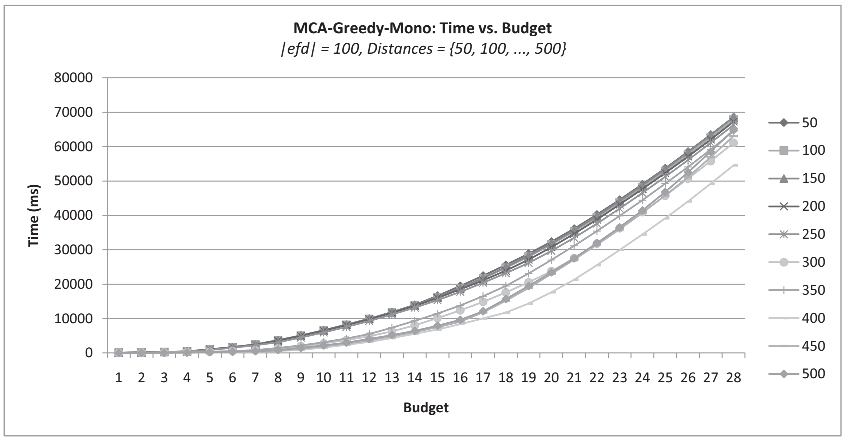


Figure 6.9: Runtime of MCA-GREEDY-MONO as the budget increases, with |exfd| = 10 (left) and |exfd| = 100 (right). Note the increase in runtime due to the extra determinism of a larger exfd.

6.7 Chapter 6 Related Work

A similar motivation to this chapter exists in the field of (multi-)agent security, where the central idea is to protect a set of targets from adversaries. These games are typically modeled on top of graphs, with agents and adversaries competing to protect or penetrate a set of targets. [135] represents the adversary's behavior through a probability distribution over states, indicating the probability of that state being targeted; no real graph structure is considered, much less a geospatial model. [1] and [2] consider an environment with more hidden information, and attempt to detect adversarial penetrations across the routes (represented as paths on a graph) of patrolling agents. [139] solves Stackelberg (leader-follower) games under the assumption of bounded reasoning rationality, again on a graph network. [35] explores protecting dynamic targets from rational adversaries on real-world road networks.

6.8 Chapter Summary

Geospatial abduction was introduced in Chapter 4 and used to infer a set of partner locations from a set of observations, given a feasibility predicate and an interval $[\alpha, \beta] \subseteq [0, 1]$. Chapter 4 developed exact and approximate algorithms for GAPs. In particular, no adversary was assumed to exist there. In this chapter, we study the case of geospatial abduction where there is an explicit adversary who is interested in ensuring that the agent does not detect the partner locations. This is the case with real world serial killers and insurgents who launch IED attacks.We develop a game-theoretic framework for reasoning about the best strategy that an adversary might adopt (based on the minimizing the adversary's detriment) and the best strategy that the agent could adopt to counter the adversary's strategy.

We consider the adversarial geospatial abduction problem to be a two player game—an agent ("good" guy) and an adversary ("bad" guy). The adversary is attempting to cause certain observable events to occur (e.g. murders or IED attacks) but make it hard to detect the associated set of partner locations (e.g. location of the serial killers home/office, or the locations of weapons caches supporting the IED attacks). We use an axiomatically-defined "reward function" to determine how similar two explanations are to each other. We study the problems of finding the best response for an agent and adversary to a mixed strategy (based on a probability distribution over explanations) of the opponent. We formalize these problems as the "optimal adversarial strategy" (OAS) and maximal counter-adversary strategy (MCA) problem. We show both OAS and MCA to be NP-hard and provide exact and approximate methods for solving them. When reasoning about the best possible strategy for the adversary, we present a mixed integer programming based algorithm and show that the MILP in question can be greatly reduced through the elimination of many variables using the concept of a δ-core explanation. Our experiments are carried out on real world data about IED attacks over a period of 21 months in Baghdad.

When reasoning about the best possible strategy for the adversary, we present two algorithms. The MCA-LS algorithm is very general and leverages submodularity of reward functions. The MCA-GREEDY-MONO algorithm assumes the reward function is monotonic. Both MCA-LS and MCA-GREEDY-MONO are highly accurate and have very reasonable time frames. Though MCA-GREEDY-MONO is slightly faster than MCA-LS, we found that on every single run, MCA-LS found the exact optimal benefit even though its theoretical lower bound approximation ratio is only 1/3—a truly remarkable performance. As MCA-LS does not require any additional assumptions and as its running time is only slightly slower than that of MCA-GREEDY-MONO, we believe this algorithms has a slight advantage.

Chapter 7

Geospatial Optimization

The next two chapters deal with optimal selection of agent actions. Here the agent has the as set of actions that modify attributes of a geospatial region and he wishes to select a limited number of such actions (with respect to some budget) in a manner that either achieves some goal (goal-based geospatial optimization) and/or maximizes a benefit function (benefit-maximizing geospatial optimization). Additionally, there are certain combinations of actions that cannot be performed together. In this chapter, we study the complexity of geospatial optimization problems and present algorithm for solving such problems - either exactly or within a certain factor of optimal.[1]

7.1 Chapter Introduction

There are numerous applications which require the ability to take certain actions (e.g. distribute money, medicines, people etc.) over a geographic region. For

[1]This chapter is based research that was completed in cooperation with V.S. Subrahmanian.

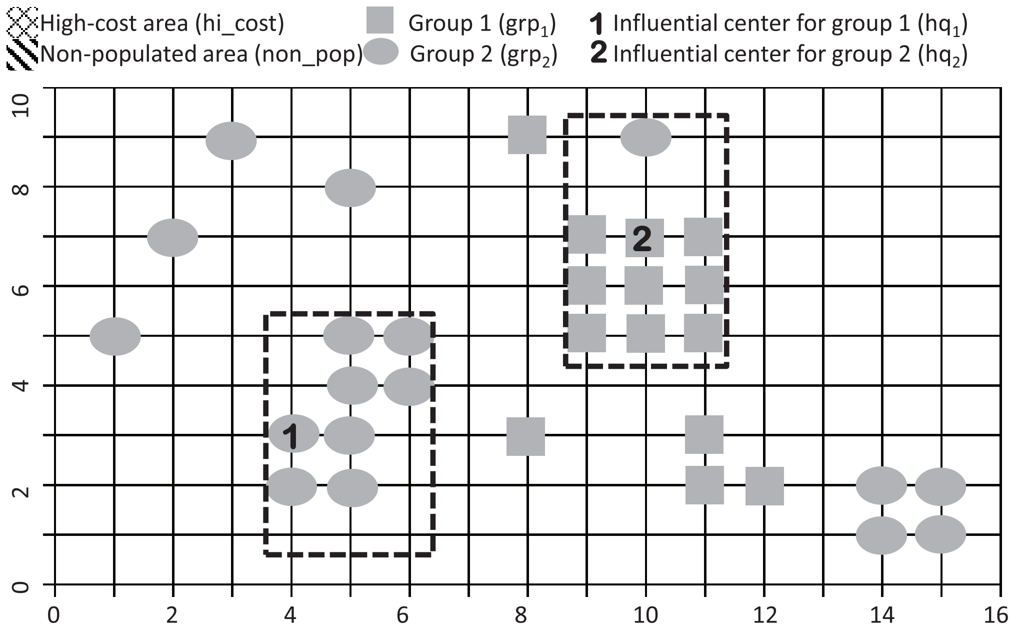

High-cost area (hi_cost)
Non-populated area (non_pop)
Group 1 (grp1)
Group 2 (grp2)
1 Influential center for group 1 (hq1)
2 Influential center for group 2 (hq2)
1
2
0
2
4
6
8
10
12
14
16

Figure 7.1: Locations in a district - contingency groups and unpopulated areas.

instance, a disaster relief organization must allocate people and supplies in a region after a disaster. A public health organization needs to allocate limited vaccine stocks to people across the region. A government needs to allocate funds for education or unemployment training across a region.

Figure 7.1 shows a 2-dimensional map of a region. A political candidate can only make so many campaign stops and public appeals. We assume that a map $\mathcal{M}$ is discrete (this is a common assumption in most GIS systems) and has coordinates drawn from $[0, \dots, M] \times [0, \dots N]$ where the bottom left corner of the map is the point $(0, 0)$. The candidate wants to identify the best places to campaign or make public appeals to maximize his exposure. Additionally, the map shows un-populated areas, areas where campaigning costs are high, and areas dominated by one of two constituent groups. All of these factors may affect the set of locations the candidate selects to optimize his exposure.

In this chapter, we introduce *geographic optimization problems* or GOPs that capture and solve problems such as those mentioned above. The organization and contribution of the chapter is as follows. Section 7.2 formally defines GOPs - specifically we introduce goal-based and benefit-maximizing GOPs (GBGOP and BMGOP respectively). Section 7.3 shows that both GBGOP and BMGOP are NP-hard (with the associated decision problems in the complexity class NP). Additionally, we prove non-trivial theoretical limits on approximation: if GBGOP were to be approximated within the logarithm of the input then NP would have a slightly super-polynomial oracle. BMGOP cannot be approximated within a guaranteed factor greater than 0.63 unless P=NP. Section 7.4 presents integer programs to solve both GBGOP and BMGOP using an IP solver like CPLEX. In Section 7.5, we show how to correctly reduce the number of variables in the integer constraints for GBGOP. We then develop the BMGOP-Compute algorithm in Section 7.6 that can quickly approximate a BMGOP in polynomial time and provides an approximation guarantee.

7.2 GOPs Formalized

Throughout this chapter, we assume that $\mathcal{M} = [0, \dots, M] \times [0, \dots, N]$ is an arbitrary, but fixed "map". We define a logical language $\mathcal{L}$ whose constant symbols are members of $\mathcal{M}$ and that has an infinite set $\mathcal{L}_{var}$ of variable symbols disjoint from $\mathcal{M}$. $\mathcal{L}$ has a set $\mathcal{G} = \{g_1, \dots, g_n\}$ of unary predicate symbols. As usual, a term is either a constant symbol or variable symbol. If t is a term, then $g_i(t)$ is an *atom*. If t is a constant, then $g_i(t)$ is *ground*. Intuitively, if $p \in \mathcal{M}$, then $g_i(p)$ says that point p has property g_i. We use $B_{\mathcal{L}}$ to denote the set of all ground atoms. Well-formed formulas (wffs) are defined in the usual way. (i) Every atom is a wff. (ii) If F, G are wffs, then so are $F \wedge G, F \vee G, \neg F$ are all wffs.

Example 7.2.1. *Consider the map $\mathcal{M}_{cpgn}$ in Figure 7.1 with the predicates of $\mathcal{G}$ including the following:*

$$\{hi\_cost, non\_pop, grp_1, grp_2, hq_1, hq_2\}$$

The predicate exposure not depicted in the figure corresponds to a candidate receiving exposure in a certain area. $hi\_cost((1,9)), hq_1((4,3)), non\_pop((8,1)),$ and $grp_2((5,8))$ are all examples of ground atoms.

A *state* is any subset of $B_{\mathcal{L}}$. We use $\mathbf{S}$ to denote the set of all states. Satisfaction of formulas is defined in the obvious way. State s satisfies a ground atom A, denoted $s \models A$, iff $A \in s$. $s \models F \vee G$ iff $s \models F$ or $s \models G$. $s \models F \wedge G$ iff $s \models F$ and $s \models G$. $s \models \neg F$ iff s does not satisfy F.

Example 7.2.2. *The shading shown in Figure 7.1 defines a state. For example, $hi\_cost((1,9)) \in s_{cpgn}$ while $exposure((1,9)) \notin s_{cpgn}$.*

An action maps points to sets of ground atoms.

Definition 78 (Action)**.** *An action is a mapping $a : \mathcal{M} \to 2^{B_{\mathcal{L}}}$. We use $\mathcal{A}$ to denote the set of actions. An action-point pair is any member of $\mathcal{A} \times \mathcal{M}$.*

An action-point pair (a, p) is executed if action a takes place at point p. Thus, one can think of (a, p) as saying that action a occurs at point p. The *result of executing a set SOL of action-point pairs in state s_0 is denoted $appl(SOL, s_0)$ and is the set $(s_0 \cup \{a(p) \mid (a, p) \in SOL\})$.*

Example 7.2.3. *Continuing with example 7.2.2, our candidate has actions $\mathcal{A}_{cpgn} = \{nor, appeal_1, appeal_2\}$ where nor refers to a normal campaign stop and $appeal_1, appeal_2$ refer to public appeals to constituent groups 1 and 2 respectively. The actions map to ground atoms as follows.*

$$\begin{aligned} nor(p) = \quad & \{exposure(p')| \quad \neg non\_pop(p') \wedge d(p, p') \leq 1\} \\ appeal_i(p) = \quad & \{exposure(p')| \quad hq_i(p) \wedge grp_i(p')\} \end{aligned}$$

The first action says that when a normal campign stop is made at point p and p′ is a populated place one distance unit or less from p, then the candidate has exposure at place p′ as well. The second action says that if the candidate makes an appeal (action) at point p and p is the headquarters of interest group grp_i, then the candidate has obtained exposure in all places associated with interest group grp_i.

Definition 79 (Cost Function)**.** *A* ***cost function****, $\mathsf{C} : \mathcal{A} \times \mathcal{M} \to [0, 1]$.*

Throughout this chapter, we assume the cost function is arbitrary but fixed and can be computed in constant time. We also assume that if $\mathcal{A} \times \mathcal{M} = \{(a_1, p_1), \ldots, (a_m, p_m)\}$, then c_i is used to denote $\mathsf{C}(a_i, p_i)$.

Example 7.2.4. *The cost function for our example is $\mathsf{C}^{(s)}_{cpgn}$ and is defined (based on some state s) as follows: $\mathsf{C}^{(s)}_{cpgn}(a, p) = 1$ if $hi\_cost(p) \in s$ and 0.5 otherwise.*

We also assume the existence of a set of integrity constraints IC that specify that certain actions cannot be jointly taken if some conditions hold w.r.t. the state — such constraints were defined before by [40].

Definition 80 (Integrity Constraint)**.** *If Φ is a set of action-point pairs and χ is a wff, then $\Phi \hookleftarrow \chi$ is an **integrity constraint**.*

When $\Phi \hookleftarrow \chi$ is ground (this is where χ is ground), this says that if χ is true, then only one action-point pair in Φ may be executed. Formally, suppose s is a state and Φ' is a set of action-point pairs and $\Phi \hookleftarrow \chi$ is ground. $(s, \Phi') \models \Phi \hookleftarrow \chi$ iff either $s \not\models \chi$ or $s \models \chi$ **and** $|\Phi \cap \Phi'| \leq 1$. (s, Φ') satisfies an integrity constraint iff it satisfies all ground instances of it. $(s, \Phi') \models IC$ where IC is a set of integrity constraints iff (s, Φ') satisfies every constraint in that set. Given a state s and set IC of integrity constraints, we use IC_s to denote the set of all ground instances of integrity constraints in IC where the associated wff χ is satisfied by s[2].

Example 7.2.5. *Continuing Example 7.2.4, let IC_{cpgn} be:*

$$\{\{appeal_1((4,3)), appeal_2((10,7))\} \hookleftarrow \mathsf{TRUE}\}$$

This constraint says that an appeal can be made to either group 1 or group 2 at their center of influence, but not both — for instance, these two groups may have opposing views.

We now introduce the *goal-based geospatial optimization problem* (GBGOP). This problem takes as input a map $\mathcal{M}$, initial state s_0, set of actions $\mathcal{A}$, cost function

[2]Formally, $IC_s = \{(\Phi \hookleftarrow \chi) \in IC | s \models \chi\}$

C, integrity constraints IC, positive real number $\mathbf{c}$, and disjoint sets $\Theta_{in}, \Theta_{out} \subseteq B_{\mathcal{L}}$. Intuitively, $\mathbf{c}$ restricts the total cost and Θ_{in} (resp. Θ_{out}) is a set of atoms that must be true (resp. false) after the actions are applied. Our optimality criteria for a GBGOP is to minimize the cardinality of the action-point pairs. A GBGOP can be viewed as an abductive inference problem (i.e. find a set of actions that lead to the current state) - where minimal cardinality is a common parsimony requirement.

Definition 81 (GBGOP Solution, Optimal Solution)**.** *A* solution *to a GBGOP* $(\mathcal{M}, s_0, \mathcal{A}, \mathsf{C}, IC, \mathbf{c}, \Theta_{in}, \Theta_{out})$ *is a set* $SOL \subseteq \mathcal{A} \times \mathcal{M}$ *such that: (i)* $\Sigma_{(a_i,p_i)\in SOL} c_i \leq \mathbf{c}$*, (ii)* $(s_0, SOL) \models IC$*, and (iii)* $appl(s_0, SOL) \models \bigwedge_{A_i \in \Theta_{in}} A_i \wedge \bigwedge_{A_j \in \Theta_{out}} \neg A_j$*.*

A solution SOL *is* optimal *iff there is no other solution* SOL' *such that* $|SOL'| \leq |SOL|$*.*

Our next type of problem is a *benefit-maximizing geospatial optimization problem* (BMGOP) that also considers a benefit function, defined as follows.

Definition 82 (Benefit Function)**.** *The* ***benefit function****,* $\mathsf{B} : B_{\mathcal{L}} \rightarrow \Re^{+}$ *maps atoms to positive real numbers.*

Example 7.2.6. *In our running example, we use the benefit function* B_{cpgn} *where* $\mathsf{B}_{cpgn}(A) = 1$ *if* A *has the form* $exposure()$ *and* 0 *otherwise.*

As with cost, we assume the benefit function to be arbitrary but fixed and computable in constant time. We also assume that if $B_{\mathcal{L}} = \{A_1, \ldots, A_n\}$, then $\mathsf{B}(A_i)$ is denoted b_i. A BMGOP takes as input, $\mathcal{M}$, s_0, $\mathcal{A}$, C, IC, and $\mathbf{c}$ - all defined the same as for a GBGOP. Additionally it takes benefit function B and natural number k. Here k is a bound on the number of actions the agent can take as we attempt to maximize benefit as an optimality criteria.

Definition 83 (BMGOP Solution, Optimal Solution)**.** *A* solution *to a BMGOP* $(\mathcal{M}, s_0, \mathsf{B}, \mathcal{A}, \mathsf{C}, IC, k, \mathbf{c})$ *is a set* $SOL \subseteq \mathcal{A} \times \mathcal{M}$ *such that: (i)* $|SOL| \leq k$ *and (ii)* $\Sigma_{(a_i,p_i) \in SOL} c_i \leq \mathbf{c}$*, and (iii)* $(s_0, SOL) \models IC$*.*

A solution SOL is optimal *iff there is no other solution SOL′ such that:*

$$\sum_{A_i \in appl(SOL, s_0)} b_i < \sum_{A_i \in appl(SOL', s_0)} b_i$$

7.3 Complexity Results

Here, we provide complexity results for GBGOPs and BMGOPs. First, we establish both as being at least NP-hard.

Theorem 38. *Given GBGOP* $(\mathcal{M}, s_0, \mathcal{A}, \mathsf{C}, IC, \mathbf{c}, \Theta_{in},$
$\Theta_{out})$*, finding an optimal solution* $SOL \subseteq \mathcal{A} \times \mathcal{M}$ *is NP-hard. This result holds even if for each* $a \in \mathcal{A}, p \in \mathcal{M}$*, it is the case that* $\forall g'(p') \in a(p)$*,* $p' = p$ *- i.e. each action only affects the point is is applied to.*

Proof Sketch. We embed the known NP-hard problem of SET-COVER [46] which takes as input a set of n elements, S and a family of m subsets of S, $\mathcal{H} \equiv \{H_1, \ldots, H_m\}$, and outputs $\mathcal{H}' \subseteq \mathcal{H}$ s.t. the union of the subsets covers all elements in S and $\mathcal{H}'$ is of minimal cardinality. We encode this problem into a GBGOP as follows: we set $\mathcal{G} = \{g_1, \ldots, g_n\}$ - each predicate in $\mathcal{G}$ corresponds to an element in S, the map, $\mathcal{M}$ consists of a single point, p, the actions $\mathcal{A} = \{a_1, \ldots, a_m\}$ s..t each action $a_i \mathcal{A}$ corresponds to an element in $\mathcal{H}$ and each is defined as follows: $a_i(p) = \bigcup_{x_j \in H_i} \{g_j(p)\}$. The cost function C returns 1 for each action-point pair, $\Theta_{in} = \bigcup_{g_i \in \mathcal{G}} \{g_i(p)\}$, $\Theta_{out} = \emptyset$, and finally, we set $s_0 = \emptyset$, $IC = \emptyset$, $\mathbf{c} = n$. □

Theorem 39. *Given BMGOP $(\mathcal{M}, s_0, \mathsf{B}, \mathcal{A}, \mathsf{C}, IC, k, \mathbf{c})$, finding an optimal solution $SOL \subseteq \mathcal{A}$ is NP-hard. This result holds even if for each $a \in \mathcal{A}, p \in \mathcal{M}$, it is the case that $\forall g'(p') \in a(p)$, $p' = p$ - i.e. each action only affects the point is is applied to).*

Proof Sketch. The problem MAX-K-COVER [46] is considered the dual of SET-COVER and accepts the same input as that problem, with an additional natural K. It outputs K subsets that covers a maximal amount of elements in S. The encoding reflects that of Theorem 38 except now we assign a benefit of 1 for each ground atom and set $k = K$. □

One may think that one can solve GOPs efficiently in practice by using fully polynomial time approximation schemes (FPTAS). However, by the nature of our constructions used in the NP-hardness results, this is not possible for either type of GOP under accepted theoretical assumptions.

Theorem 40. *If for some $\epsilon > 0$, there is a PTIME algorithm to approximate GBGOP within $(1-\epsilon) \cdot \ln(|\mathcal{A} \times \mathcal{M}|)$, then $NP \subset TIME(|\mathcal{A} \times \mathcal{M}|^{O(\lg \lg |\mathcal{A} \times \mathcal{M}|)})$ (NP has a slightly super-polynomial algorithm).*

Proof Sketch. Follows from Theorem 38 and [46, Theorem 4.4]. □

Theorem 41. *Finding an optimal solution to BMGOP cannot be approximated in*

PTIME within a ratio of $\frac{e-1}{e} + \epsilon$ *(approx.* 0.63*) for some* $\epsilon > 0$ *(where* e *is the inverse of the natural log) unless* ***P=NP****, even when* $IC = \emptyset$*.*

Proof Sketch. Follows from Theorem 39 and [46, Theorem 5.3]. □

Next, under some reasonable assumptions, the decision problems for GBGOP/BMGOP are in-NP.

Theorem 42. *Given GBGOP* $(\mathcal{M}, s_0, \mathcal{A}, \mathsf{C}, IC, \mathbf{c}, \Theta_{in},$

$\Theta_{out})$*, if the cost function and all actions* $a \in \mathcal{A}$ *can be polynomially computed, then determining if there is a solution SOL for the instance of the GBGOP s.t. for some real number* k*,* $|SOL| \leq k$ *is in-NP.*

Theorem 43. *Given BMGOP* $(\mathcal{M}, s_0, \mathsf{B}, \mathcal{A}, \mathsf{C}, IC, k, \mathbf{c})$*, if the cost function, benefit function, and all actions* $a \in \mathcal{A}$ *can be polynomially computed, then determining if there is a solution SOL for the instance of the BMGOP s.t. for some real number* val*,* $\sum_{A_i \in appl(SOL, s_0)} b_i \geq val$ *is in-NP.*

As stated earlier, a GBGOP may also be viewed as an abductive inference problem. Even though finding a solution (not necessarily optimal) to a GBGOP can trivially be conducted in PTIME[3], counting the number of solutions is #P-complete. This counting problem is difficult to approximate.

Theorem 44. *Counting the number of solutions to a GBGOP (under the assumptions of Theorem 42) is #P-complete.*

Proof Sketch. The MONSAT problem takes a set C of n clauses of K disjuncted literals (no negation) over set L of atoms (size m) and outputs "yes" iff there is a

[3]Return the set $\{(a_i, p_i) \in \mathcal{A} \times \mathcal{M} | a_i(p_i) \cap \Theta_{out} = \emptyset\}$

subset of L that satisfies all clauses in C. This problem has an obvious resemblance to SET-COVER (with no cardinality criteria) and we embed it into a GBGOP in a way similar to the construction of Theorem 38. The key here is to have the predicates correspond to clauses and actions correspond to lierals - each a_i is defined as follows: $a_i(p) = \{g_j(p) | \{\ell_i\} \models \phi_j\}$, where ℓ_i is the corresponding literal and g_j is the predicate that corresponds to clause ϕ_j. The reduction is parsimonious, and as #MONSAT is #P-hard, the hardness result follows. The membership in #P follows from Theorem 42. $\square$

Theorem 45. *For $\epsilon > 0$, approximating the number of solutions to a GBGOP within a factor of $2^{|\mathcal{A}\times\mathcal{M}|^{1-e}}$ is NP-hard.*

Proof Sketch. Follows from Theorem 44 and Theorem 3.2 of [145]. $\square$

7.4 Integer Programs for Solving GOPs

In this section, we present an integer programming (IP) algorithms for both GBGOP and BMGOP which provide exact solutions. Given a GBGOP, the IP associates an integer-valued variable X_i with each action-point pair $(a_i, p_i) \in \mathcal{A}\times\mathcal{M}$ where $a_i(p_i) \cap \Theta_{out} = \emptyset$. Intuitively, $X_i = 1$ denotes that action a_i is performed at point p_i.

Definition 84 (GBGOP-IP)**.** *Let set $R = \{(a_i, p_i) \in \mathcal{A} \times \mathcal{M} | a_i(p_i) \cap \Theta_{out} = \emptyset\}$.*

For each action-point pair $(a_i, p_i) \in R$, create variable $X_i \in \{0,1\}$.

$$\min \sum_{i=1}^{|R|} X_i \tag{7.1}$$

$$s.t. \sum_{a_j(p_j) | A_i \in a_j(p_j)} X_j \geq 1 \quad \forall A_i \in \Theta_{in} - s_0 \tag{7.2}$$

$$\sum_{(a_i,p_i) \in R} c_i \cdot X_i \leq \mathsf{c} \tag{7.3}$$

$$\sum_{(a_i,p_i) \in \Phi} X_i \leq 1 \quad \forall (\Phi \hookleftarrow \chi) \in IC_{s_0} \tag{7.4}$$

The objective function minimizes the total number of action-point pairs. Constraint (7.2) ensures that every ground atom in Θ_{in} (that does not appear in the initial state) is caused by at least one of the selected action-point pairs. Constraint (7.3) enforces the constraint on cost. Constraint (7.4) ensures that the integrity constraints are satisfied. Next we present our integer constraints for a BMGOP where the IP associates an integer-valued variable X_i with each action-point pair $(a_i, p_i) \in \mathcal{A} \times \mathcal{M}$, and an integer-valued variable Y_j with each ground atom $A_j \in B_{\mathcal{L}} - s_0$. The intuition for the X_i variables is the same as in GBGOP-IP.

Definition 85 (BMGOP-IP)**.** *For each action-point pair $(a_i, p_i) \in \mathcal{A} \times \mathcal{M}$, create variable $X_i \in \{0,1\}$. For each $A_i \in B_{\mathcal{L}} - s_0$ create variable $Y_i \in \{0,1\}$.*

$$\max \sum_{A_i \in s_0} b_i + \sum_{i=1}^{|B_{\mathcal{L}}|-|s_0|} b_i \cdot Y_i \tag{7.5}$$

$$s.t. \sum_{a_j(p_j) | A_i \in a_j(p_j)} X_j \geq Y_i \quad \forall A_i \in B_{\mathcal{L}} - s_0 \tag{7.6}$$

$$\sum_{(a_i,p_i) \in \mathcal{A} \times \mathcal{M}} X_i \leq k \tag{7.7}$$

$$\sum_{(a_i,p_i) \in \mathcal{A} \times \mathcal{M}} c_i \cdot X_i \leq \mathsf{c} \tag{7.8}$$

$$\sum_{(a_i,p_i) \in \Phi} X_i \leq 1 \quad \forall (\Phi \hookleftarrow \chi) \in IC_{s_o} \tag{7.9}$$

In the above IP, the objective function looks at each ground atom and sums the associated benefit if the associated Y_i variable is 1 - meaning that atom A_i is true after the actions are applied. Constraint (7.6) effectively sets a Y_i variable to 1 if an action that causes A_i to be true occurs. Constraint (7.7) enforces the cardinality requirement. Constraints 7.8-7.9 mirror constraints 7.3-7.4 of GBGOP-IP. The result below shows that a solution σ to the above IPs[4], when restricted to the X_i variables, provides an immediate solution to the GOP.

Proposition 65. *Suppose Γ is a GBGOP (resp. BMGOP) and $IP(\Gamma)$ is its corresponding integer program (GBGOP-IP, resp. BMGOP-IP). Then:*

1. *If SOL is a solution to Γ, then there is a solution σ of $IP(\Gamma)$ such that $\sigma \supseteq \{X_i = 1 \mid (a_i, p_i) \in SOL\}$.*

2. *If σ is a solution to $IP(\Gamma)$, then there is a solution SOL to Γ such that $\{X_i = 1 \mid (a_i, p_i) \in SOL\} \subseteq \sigma$.*

We note that for GBGOP-IP, the number of variables is fairly large – $O(|\{(a_i, p_i) \in \mathcal{A} \times \mathcal{M} | a_i(p_i) \cap \Theta_{out} = \emptyset\}|)$ variables and $O(|\Theta_{in} - s_0| + |IC_{s_0}| + 1)$ constraints. BMGOP-IP has even more variables - (though not exponential) - $O(|\mathcal{M}| \cdot (|\mathcal{A}| + |\mathcal{G}|))$ variables and $O(|\mathcal{M}| \cdot |\mathcal{G}| + |IC_{s_0}| + 2)$ constraints.

[4] A solution to GBGOP-IP or BMGOP-IP is an assignment of values to variables that optimizes the objective function. Thus, a solution can be described as a set of equations assigning values to the variables X_i, Y_j.

7.5 Correct Variable Reduction for GBGOP-IP

The set of integer constraints for GBGOP has $O(|R|)$ variables where $R \subseteq \mathcal{A} \times \mathcal{M}$. We show how to correctly reduce the number of variables by considering only a subset of R - thereby providing a smaller integer program. Our intuition is that an optimal solution SOL is an *irredundant* cover of Θ_{in} meaning there is no subset $SOL' \subset SOL$ that is also a solution. Hence, we can discard certain elements of R that cannot possibly be in an optimal solution. First, for a given GBGOP $\Gamma = (\mathcal{M}, s_0, \mathcal{A}, \mathsf{C}, IC, \mathbf{c}, \Theta_{in}, \Theta_{out})$, we introduce $Q^{\Gamma}_{(a,p)} = \{\Phi | (\Phi \hookleftarrow \chi) \in IC_{s_0} \wedge (a, p) \in \Phi\}$ and the set of ground atoms each action-point pair affects $\mathsf{Aff}^{\Gamma}_{(a,p)} = a_i(p_i) \cap (\Theta_{in} - (\Theta_{in} \cap s_0))$. We can now define a *reduced action-point set.*

Definition 86 (Reduced Action-Point Set)**.** *Given GBGOP*

$$\Gamma = (\mathcal{M}, s_0, \mathcal{A}, \mathsf{C}, IC, \mathbf{c}, \Theta_{in}, \Theta_{out})$$

*and set $R = \{(a_i, p_i) \in \mathcal{A} \times \mathcal{M} | a_i(p_i) \cap \Theta_{out} = \emptyset\}$, we define **reduced action-point set** $R^* = \{(a_i, p_i) \in R |\ \nexists (a_j, p_j) \in R$ s.t.*
$(c_j \leq c_i) \wedge (Q^{\Gamma}_{(a_j,p_j)} \subseteq Q^{\Gamma}_{(a_i,p_i)}) \wedge (\mathsf{Aff}^{\Gamma}_{(a_i,p_i)} \subseteq \mathsf{Aff}^{\Gamma}_{(a_j,p_j)})\}$

Example 7.5.1. *Consider the campaign scenario last discussed in Example 7.2.5. Suppose the candidate wants to optimize the following GBGOP:*

$$\Gamma = (\mathcal{M}_{cpgn}, s_{cpgn}, \mathcal{A}_{cpgn}, \mathsf{C}^{(s_{cpgn})}_{cpgn}, IC_{cpgn}, 4, \Theta^{cpgn}_{in}, \emptyset)$$

where each $A \in \Theta^{cpgn}_{in}$ has the form exposure(p) where p is a point in one of the two dashed rectangles in Figure 7.1. Note that as map $\mathcal{M}_{cpgn}$ contains 187 points,

$|\mathcal{A}| = 3$, *and* $\Theta_{out} = \emptyset$, *the cardinality of* R *is* 561. *By contrast, the set* R^* *consists of only* 7 *elements,* 1.2% *of the size of* R. *Here we have*

$$\begin{aligned} R^* \;&=\; \{(nor,(5,4)),(nor,(5,3)),(nor,(5,2)),(nor,(10,8)),\\ &\quad (nor,(10,7)),(nor,(10.6)),(appeal_1,(4,3))\} \end{aligned}$$

Intuitively, all elements in R^* are preferable for membership in an optimal solution over $R - R^*$ as they cost less, result in the same changes to the state, and occur in the same or fewer integrity constraints. Set R^* can be found in quadratic time with a naive algorithm - an operation that is likely dominated by solving or approximating GBGOP-IP. The next lemma says that R^* must contain an optimal solution.any optimal solution to a GBGOP. This can then be used to correctly reduce the number of variables in GBGOP-IP.

Lemma 22. *Given GBGOP* $\Gamma = (\mathcal{M}, s_0, \mathcal{A}, \mathcal{C}, IC, \mathbf{c}, \Theta_{in}, \Theta_{out})$, *for any optimal solution* $SOL \subseteq R$, *there is an optimal solution* $SOL' \subseteq R^*$.

Proof Sketch. We show this by proving that for any set $W = SOL \cap (R - R^*)$, there is some set $W' \subseteq R^* - (R^* \cap SOL)$ s.t. $(SOL - W) \cup W'$ is also a solution.

Proposition 66. *Suppose* Γ *is a GBGOP and* $IP(\Gamma)$ *is its corresponding integer program. We can create such a program with a variable for every element of* R^* *(instead of* R*) and Proposition 65 still holds true.*

7.6 The **BMGOP-Compute** Algorithm

While BMGOP-IP can solve a BMGOP exactly, doing so is computationally intractable. We now present an approximation algorithm that runs in PTIME but provides a lower approximation ratio than proved in Theorem 41. First, we show that a BMGOP reduces to an instance of submodular maximization problem[5] with respect to packing constraints. We then leverage some known methods [11] to solve such problems and develop a fast, deterministic algorithm to approximate BMGOP with an approximation bounds. Given BMGOP $\Gamma = (\mathcal{M}, s_0, \mathsf{B}, \mathcal{A}, \mathsf{C}, IC, k, \mathbf{c})$, consider the objective function in BMGOP-IP. We can write that function as a mapping from action-point pairs to reals. We denote this function (specific for BMGOP Γ) as $f_\Gamma : 2^{\mathcal{A}\times\mathcal{M}} \rightarrow \Re^+$, where $f_\Gamma(S) = \sum_{A_i \in \mathsf{appl}(S,s_0)} b_i$, which has certain properties.

Proposition 67. *For BMGOP Γ, function f_Γ is: (i) submodular, (ii) monotonic, i.e. $Z_1 \subseteq Z_2 \rightarrow f_\Gamma(Z_1) \leq f_\Gamma(Z_2)$ and (iii) under the condition $\forall A_i \in B_{\mathcal{L}}$, $b_i = 0$, we have $f_\Gamma(\emptyset) = 0$.*[6]

Proof Sketch. Consider $S \subseteq S' \subseteq \mathcal{A} \times \mathcal{M}$ and $(a,p) \notin S'$. We must show $f_\Gamma(S \cup \{(a,p)\}) - f_\Gamma(S) \geq f_\Gamma(S' \cup \{(a,p)\}) - f_\Gamma(S')$. Suppose, BWOC $f_\Gamma(S \cup \{(a,p)\}) - f_\Gamma(S) < f_\Gamma(S' \cup \{(a,p)\}) - f_\Gamma(S')$. Then we have $\sum_{A_i \in appl(S \cup \{(a,p)\}, s_0) - appl(S,s_0)} b_i < \sum_{A_i \in appl(S' \cup \{(a,p)\}, s_0) - appl(S',s_0)} b_i$. However, by the definition of *appl*, we have $appl(S \cup$

[5]Suppose Z is a set. A function $f : 2^Z \rightarrow \mathbb{R}$ is said to be *submodular* iff for all Z_1, Z_2 such that $Z_1 \subseteq Z_2$ and all $z \notin Z_2$, it is the case that $f(Z_1 \cup \{z\}) - f(Z_1) \geq f(Z_2 \cup \{z\}) - f(Z_2)$, i.e. the incremental value of adding z to the smaller set Z_1 exceeds the incremental value of adding it to the larger set Z_2. Here, $\mathcal{R}$ denotes the reals.

[6]Henceforth, we will assume this condition to be true.

$\{(a,p)\}, s_0) - appl(S, s_0) \supseteq appl(S' \cup \{(a,p)\}, s_0) - appl(S', s_0)$, which is a contradiction.$\Box$

As our objective function is submodular, and constraints 7.7-7.9 are linear packing constraints, any instance of a BMGOP can be viewed as maximization of a submodular function wrt linear packing constraints and hence, methods to solve such problems can be used here. The BMGOP-Compute algorithm leverages this idea and illustrated in Example 7.6.1.

Example 7.6.1. *Following Example 7.2.5. Suppose the candidate wants to optimize BMGOP:* $(\mathcal{M}_{cpgn}, s_{cpgn}, \mathsf{B}_{cpgn}, \mathcal{A}_{cpgn}, \mathsf{C}^{(s_{cpgn})}_{cpgn}, IC_{cpgn}, 3, 2)$. *In this case, we will set* $\delta = 0.001$. *He wishes to find a set of* 3 *action-point pairs to optimize his exposure. BMGOP-Compute sets* $\lambda = 22.14$, $w' = 0.33$, $w'' = 0.50$, *and* $w_1 = 0.50$ *in lines 1 and 2. In the first iteration of the loop at line 3, it finds the action-point pair that minimizes the quantity at line 3 is* $(appeal_1, (4,3))$ *- which has the associated value* 0.073. *Note, other action-point pairs with low values are* $(appeal_2, (10,7))$ *with* 0.083 *and* $(nor, (15,6))$ *also with* 0.083. *It then adds* $(appeal_1, (4,3))$ *to* SOL *and updates* $w' = 0.93$, $w'' = 1.09$, *and* $w_1 = 2.35$. *On the next iteration, the BMGOP-Compute picks* $(nor, (15,6))$, *which now has a value of* 0.164. *During this iteration, the value of* $(appeal_2, (10,7))$ *has increased substantially - to* 0.294, *so it is not selected. At the end of the iteration,* w' *is updated to* 2.611 *and* w'' *is updated to* 2.364. *As* $(nor, (15,6))$ *does not impact the lone integrity constraint, the value* w_1 *remains at* 2.354. *In the third iteration, BMGOP-Compute selects* $(nor, (15,9))$ *which has a value of* 0.421. *Again, the value of* $(appeal_2, (10,7))$ *has increased - but this time only*

```
BMGOP-Compute

INPUT: BMGOP $(\mathcal{M}, s_0, \mathsf{B}, \mathcal{A}, \mathsf{C}, IC, k, \mathbf{c})$

OUTPUT: $SOL \subseteq \mathcal{A} \times \mathcal{M}$

1. Set $SOL = \emptyset$, $\delta$ to be an infinitesimal,
   and set $\lambda = e^{2-\delta} \cdot (2 + |IC_{s_0}|)$.

2. Set $w' = 1/k$ and $w'' = 1/\mathbf{c}$. For each $(\Phi_i \hookleftarrow \chi_i) \in IC_{s_0}$, set $w_i = 1/(2-\delta)$.

3. While $k \cdot w' + \mathbf{c} \cdot w'' + (2-\delta) \cdot \sum_i w_i \leq \lambda$ and $SOL \neq \mathcal{A} \times \mathcal{M}$
    (a) Let $(a_j, p_j) \in \mathcal{A} \times \mathcal{M} - SOL$ have minimal
        $\frac{w' + w'' \cdot c_j + \sum_{i|(a_j,p_j) \in \Phi_i} w_i}{(\sum_{A_i \in appl(SOL \cup \{(a_j,p_j)\}, s_0)} b_i) - (\sum_{A_i \in appl(SOL, s_0)} b_i)}$
    (b) $SOL = SOL \cup \{(a_j, p_j)\}$
    (c) Set $w' = w' \cdot \lambda^{1/k}$, $w'' = w'' \cdot \lambda^{c_j/\mathbf{c}}$ and for each integrity constraint $i$ s.t.
        $(a_j, p_j) \in \Phi_i$, set
        $w_i = w_i \cdot \lambda^{1/(2-\delta)}$

4. If $SOL$ is not a valid solution then
    (a) If $\sum_{A_i \in appl(SOL - \{(a_j,p_j)\}, s_0)} b_i \geq$
        $\sum_{A_i \in appl(\{(a_j,p_j)\}, s_0)} b_i$,
        then $SOL = SOL - \{(a_j, p_j)\}$
    (b) Else $SOL = \{(a_j, p_j)\}$

5. Return $SOL$
```

to 0.472. *BMGOP-Compute re-calculates* $w' = 7.331$, $w'' = 5.128$ *and* w_1 *remains at* 2.354. *On the last iteration, BMGOP-Compute picks* $(appeal_2, (10, 7))$ *as it has the lowest value* – 0.942. *After this fourth iteration, it updates* $w' = 20.589$, $w'' = 11.124$, *and* $w_1 = 11.0861$ *- which now total to* 42.799 *– exceeding* λ *(22.14) – causing BMGOP-Compute to exit the outer loop. Now* SOL *has 4 elements, exceeding the cardinality constraint (as well as the integrity constraint). The checks done in line 4 remove* $(appeal_2, (10, 7))$ *from* SOL *- making the result feasible. BMGOP-Compute returns* $\{(appeal_1, (4, 3)), (nor, (15, 6)_, (nor, (15, 9))\}$ *which causes the benefit to be* 45.

Proposition 68. *Suppose* Γ *is a BMGOP and* SOL *is the set returned by BMGOP-Compute. Then* SOL *is a solution to* Γ.[7]

Proposition 69. *BMGOP-Compute runs in* $O(k \cdot |\mathcal{M}| \cdot |\mathcal{A}| \cdot |IC_{s_0}|)$ *time. Proof Sketch. Clearly, the outer loop can iterate no more than k times. The inner loop iterates for each element of* $\mathcal{A} \times \mathcal{M}$ *- hence requiring time* $O(|\mathcal{M}| \cdot |\mathcal{A}|)$. *The calculation at line 3a requires* $O(|IC_{s_0}|)$ *time.* □

The following important theorem states that BMGOP-Compute provides an approximation guarantee. Because of Theorem 41 and as BMGOP-Compute is polynomial, we know that this approximation guarantee cannot be as good as $\frac{e-1}{e} + \epsilon$. The result leverages Theorem 1.1 of [11] together with the above theorems. By this result, the approximation factor of BMGOP-Compute depends on $|IC_{s_0}|$. We

[7]Here, SOL is not necessarily an optimal solution.

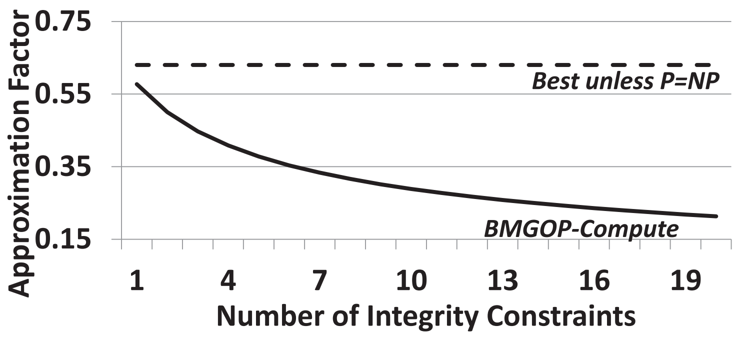


Figure 7.2: $|IC_{s_0}|$ vs. approximation ratio.

illustrate this relationship, in Figure 7.2. For our target applications, we envision $|IC_{s_0}| \leq 20$.

Theorem 46. *Under the assumption that $k, \mathsf{c} \geq 2 - \delta$, BMGOP-Compute provides a solution within a factor of $\frac{1}{(2+|IC_{s_0}|)^{1/(2-\delta)}}$ (where δ is an infinitesimal) of optimal.*

Proof Sketch. BMGOP-Compute follows from Algorithm 1 of [11] which optimizes a submodular function subject to m packing constraints within $\frac{1}{m^{1/W}}$ where W is the minimum width of the packing constraints - defined as the minimum of the size of the constraint divided by the cost of an element. For constraint 7.7, the $W = k$. For constraint 7.8, the $W \geq \mathsf{c}$. We can replace constraint 7.9 with: $\sum_{(a_i,p_i)\in\Phi_j} X_i \leq 2-\delta$ $\forall(\Phi_j \hookleftarrow \chi_j) \in IC_{s_o}$ which maintains correctness as two variables to set to 1 and exceeds $2-\delta$. The new constraint has width $2-\delta$, which, is the minimum. We then apply Theorem 1.1 of [11]. $\square$

7.7 Chapter 7 Related Work

Though spatial reasoning has been studied extensively in AI [7, 142, 103], many of the paradigms that have emerged for such reasoning are *qualitative* in nature. Such qualitative spatial reasoning efforts include the influential region connection calculus for qualitative reasoning about space. There has also been work on quantitative methods for reasoning about space [74] which contains articles on spatial reasoning in the presence of uncertainty using both logical and fuzzy methods. Spatial reasoning with quantitative information has been studied extensively in image processing [179, 163].

However, unlike this vast body of work, this chapter focuses on a different problem. Suppose we are dealing with a map $\mathcal{M}$, a cost function C, a set $\mathcal{A}$ of possible actions, a bound on the cost $\mathbf{c}$, and a bound on the number of actions we can take, what set of actions should be taken so as to optimize a given objective function. Two versions of this problem are studied in this chapter - GBGOP and BMGOP which differ in what they optimize. Both problems are proved to be NP-hard (NP-complete under realistic assumptions) and we further prove that the number of solutions to GBGOP is #P-complete. We also find limits on approximating an optimal solution to BMGOP and GBGOP (in PTIME) under accepted theoretical assumptions. We develop integer programming formulations of both problems and then present a way of simplifying the IP for GBGOP. We further present the BMGOP-Compute algorithm for BMGOP and show that it is polynomial and has a guaranteed approximation ratio (though not high enough to contract the

NP-hardness result).

7.8 Chapter Summary

In this chapter, we introduced "geospatial optimization problems" or GOPs that aide the user in taking certain actions over a geographic region. We showed these problems to be NP-hard and provided integer constraints. For the goal-based variant, we correctly reduce the number of variables. For the benefit-maximizing variant, we provide an approximation algorithm. In future work, we look to implement this framework and explore methods to achieve further scalability. In many applications, there also exists an underlying diffusion process (i.e. epidemiology). This is not accounted for with geospatial optimization. In the next chapter, we look at optimal selection of actions with respect to a diffusion process.

Chapter 8

Social Network Optimization Problems

While we look to optimize certain geospatial properties in the previous chapter, we note that for some real-world applications, such as many epidemiological situations, there is an underlying diffusion process that also affect geospatial proprieties. Given such a diffusion process and a network, we seek to find vertices of that network that optimize an aggregate and satisfy certain logical conditions. Here, we formalize the study of this type of agent behavior with the study of social network optimization problems (SNOPs).[1] Note that in this chapter, the acronym "GAPs" does not refer to the geospatial abduction problems of the past three chapters but rather the generalized annotated programs of [86].

[1]This chapter is based on [159] and [152], completed in cooperation with Maria Luisa Sapino, Matthias Broecheler and V.S. Subrahmanian.

8.1 Chapter Introduction

There is a rapid proliferation of different types of *graph data* in the world today. These include social network data (FaceBook, Flickr, YouTube, etc.), cell phone network data [126] collected by virtually all cell phone vendors, email network data (such as those derived from the Enron corpus[2]), as well as information on disease networks [45, 5]. There has been years of work on analyzing how various properties of nodes in such networks "diffuse" through the network - different techniques have been invented in different academic disciplines including economics [73, 150], infectious diseases [45], sociology [61] and computer science [81].

Past work on diffusion has several limitations. First, they largely assume that a social network is nothing but a set of vertices and edges [178, 29, 147]. In contrast, our notion of SNOPs allows a richer model where edges and vertices can both be labeled with properties. For instance, a political campaigner hoping to spread a positive message about a campaign needs to use demographics (e.g. sex, age group, educational level, group affiliations, etc.) for targeting a political message — a "one size fits all" message will not work. Second, past work on diffusion has no notion of "strength" associated with edges. It may well be the case, in many applications, that the degree of contact between two vertices (e.g. number of minutes person A spends on the cell phone with person B) is a proxy for the strength of the relationship between A and B, which in turn may have an impact of whether A can influence B or not. Third, these past frameworks [73, 150, 45, 61] usually reason about a single

[2]http://www.cs.cmu.edu/~enron/

diffusion model, rather than develop a framework for reasoning about a whole class of diffusion models.

In this chapter, we first argue that a class of the well-known Generalized Annotated Program (GAP) paradigm [86, 85, 168] and their variants [175, 88, 107, 109, 31] including *Linear GAPs* (introduced here) form a convenient method to express many diffusion models. Next, unlike most existing work in social networks which focus on learning diffusion models, we focus on reasoning with previously learned diffusion models (expressed via GAPs). In particular, we consider the problem of optimal decision making in social networks which have associated diffusion models expressible as Linear GAPs, though many of the results in the chapter apply to arbitrary GAPs as well. Here are two examples.

- **(Q1) Cell phone plans.** A cell phone company is promoting a new cell phone plan - as a promotion, it is giving away k free plans to existing customers.[3] Which set of k people should they pick so as to $\underline{\text{maximize}}$ the $\underline{\text{expected number}}$ of plan adoptees predicted by a cell phone plan adoption diffusion model they have learned from their past promotions?

- **(Q2) Medication distribution plan.** A government combating a disease spread by physical contact has limited stocks of free medication to give away. Based on a diffusion model of how the disease spreads (e.g. kids might be more susceptible than adults, those previously inoculated against the disease are safe,

[3]Our SNOP framework allows us to add additional constraints — for instance, that plans can only be given to customers satisfying certain conditions, e.g. not be employees of the cell phone company.

etc.), they want to find a set of k people who (jointly) maximally spread the disease (so that they can provide immediate treatment to these k people in an attempt to halt the disease's spread).[4]

Both the above problems are instances of a class of queries that we call SNOP queries. They differ from queries studied in the past in quantitative (both probabilistic and annotated) logic programming in two fundamental ways: (i) They are specialized to operate on graph data where the graph's vertices and edges are labeled with properties and where the edges can have associated weights, (ii) They optimize complex objective functions that can be specified by the user. Neither of these has been studied before by any kind of quantitative logic programming framework, though work on optimizing objective functions in the context of different types of semantics (minimal model and stable model semantics) has been studied before [99]. And of course, constraint logic programming [8] has also extensively studied optimization issues as well in logic programming - however, here, optimization and constraint solving is embedded in the constraint logic program, whereas in our case, they are part of the *query* over an annotated logic program.

This chapter is organized as follows. In Section 8.2, we provide an overview of GAPs (past work), define a social network, and explain how GAPs can represent some types of diffusion in SNs. Section 8.3 formally defines different types of

[4]Again, our SNOP framework allows us to add additional constraints — for instance, that medication can only be given to people satisfying certain conditions, e.g. be over a certain age, or be within a certain age range and not have any conditions that are contra-indicators for the medication in question.

social network optimization problems and provides results on their computational complexity and other properties. Section 8.4 shows how our framework can represent several existing diffusion models for social networks including economics and epidemiology. In Section 8.5 we present the exact SNOP-Mon algorithm to answer SNOP queries under certain assumptions of monotonicity. We then develop a greedy algorithm GREEDY-SNOP and show that under certain conditions, it is guaranteed to be an $(\frac{e}{e-1})$ approximation algorithm for SNOP queries — this is the best possible approximation guarantee. Last, but not least, we describe our prototype implementation and experiments in Section 8.7. Specifically, we tested our GREEDY-SNOP algorithm on a real-world social network data set consisting of over 7000 nodes and over 103,000 edges from Wikipedia logs. We show that we solve social network optimization problems over real data sets in acceptable times. We emphasize that much additional work is required on further enhancing scalability and that research on social network optimization problems is at its very infancy. Finally, in Section 8.8, we review related work.

8.2 Technical Preliminaries

In this section, we first formalize social networks, then briefly review generalized annotated logic programs (GAPs) [86] and then describe how GAPs can be used to represent concepts related to diffusion in SNs.

8.2.1 Social Networks Formalized

Throughout this chapter, we assume the existence of two arbitrary but fixed disjoint sets VP, EP of *vertex* and *edge predicate symbols* respectively. Each vertex predicate symbol has arity 1 and each edge predicate symbol has arity 2.

Definition 87. *A* ***social network*** *($\mathcal{S}$) is a 5-tuple* $(\mathbf{V}, \mathbf{E}, \ell_{vert}, \ell_{edge}, w)$ *where:*

1. $\mathbf{V}$ *is a set whose elements are called vertices.*
2. $\mathbf{E} \subseteq \mathbf{V} \times \mathbf{V}$ *is a* <u>*multi-set*</u> *whose elements are called edges.*
3. $\ell_{vert} : \mathbf{V} \to 2^{\mathsf{VP}}$ *is a function, called* vertex labeling function.
4. $\ell_{edge} : \mathbf{E} \to \mathsf{EP}$ *is a function, called* edge labeling function. [5]
5. $w : \mathbf{E} \times \mathsf{EP} \to [0, 1]$ *is a function, called* weight function.

We now present a brief example of an SN that will be used throughout this chapter.

Example 8.2.1. *Let us return to the cell phone example (query* ***(Q1)****). Figure 8.1 shows a toy SN the cell phone company might use. Here, we might have* $\mathsf{VP} = \{male, female, adopters, temp\_adopter, non\_adptr\}$ *denoting the sex and past adoption behavior of each vertex;* EP *might be the set* $\{phone, email, IM\}$ *denoting the types of interactions between vertices.* $w(v_1, v_2, ep)$ *denotes the percentage of communications of type* $ep \in \mathsf{EP}$ *initiated by* v_1 *that were with* v_2 *(measured either*

[5] Each edge $e \in \mathbf{E}$ is labeled by exactly one predicate symbol from EP. However, there can be multiple edges between two vertices labeled with different predicate symbols.

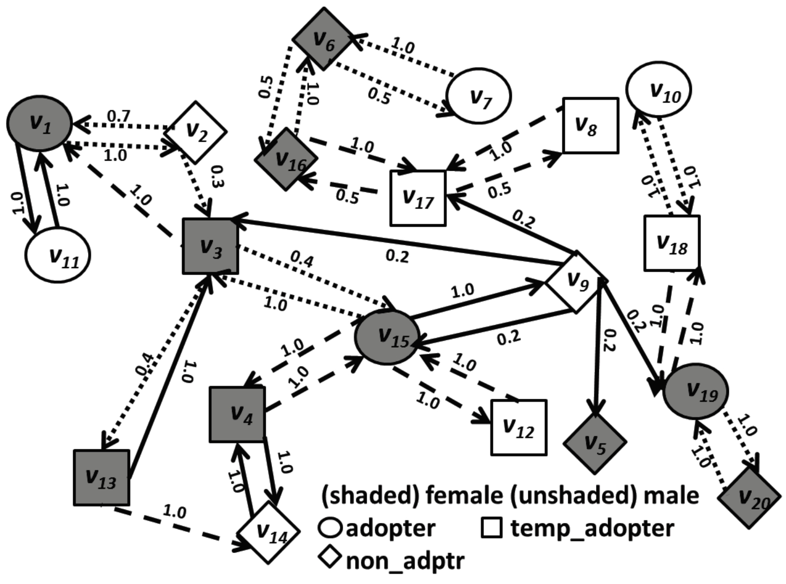


Figure 8.1: Example cellular network.

w.r.t. time or bytes). The function ℓ_{vert} is shown in figure 8.1 by the shape (denoting past adoption status) and shading (male/female). The type of edges (bold for phone, dashed for email, dotted for IM) is used to depict ℓ_{edge}.

It is important to note that our definition of social networks is much broader than that used by several researchers [5, 45, 73, 81] who often do not consider either ℓ_{edge} or ℓ_{vert} or edge weights through the function w — it is well-known in marketing that intrinsic properties of vertices (customers, patients) and the nature and strength of the relationships (edges) is critical for decision making in those fields.

Note. We assume that SNs satisfy various integrity constraints. In Example 8.2.1, it is clear that $\ell_{vert}(V)$ should include at most one of $male, female$ and at most one of $adopters$, *temp_adopter*,*non_adptr*. *We assume the existence of some integrity constraints to ensure this kind of semantic integrity* – they can be written in any reasonable syntax to express ICs – in the rest of this chapter, we assume that social networks have associated ICs and that they satisfy them. In our example, we will assume ICs ensuring that a vertex can be marked with at most one of $male/female$ and at most one of $adopters, temp\_adopter, non\_adptr$.

8.2.2 Generalized Annotated Programs: A Recap

We now recapitulate the definition of generalized annotated logic programs from [86]. We assume the existence of a set AVar of variable symbols ranging over the unit real interval $[0, 1]$ and a set $\mathcal{F}$ of function symbols each of which has an associated arity. We start by defining annotations.

Definition 88 (annotation term)**.** *(i) Any member of* $[0, 1] \cup \mathsf{AVar}$ *is an annotation. (ii) If* f *is an* n*-ary function symbol over* $[0, 1]$ *and* $t_1, \ldots, t_n$ *are annotations, then* $f(t_1, \ldots, t_n)$ *is an annotation.*

For instance, $0.5, 1, 3$ and X are all annotation terms. If $+, *, /$ are all binary function symbols, then $\frac{(X+1)*0.5}{3}$ is an annotation term.

We define a <u>separate</u> logical language whose constants are members of $\mathbf{V}$ and whose predicate symbols consist of $\mathsf{VP} \cup \mathsf{EP}$. We also assume the existence of a set $\mathcal{V}$ of variable symbols ranging over the constants (vertices). No function symbols are present. Terms and atoms are defined in the usual way (cf. [106]). If $A = p(t_1, \ldots, t_n)$ is an atom and $p \in \mathsf{VP}$ (resp. $p \in \mathsf{EP}$), then A is called a *vertex* (resp. *edge*) atom. We will use $\mathcal{A}_{\mathbf{V}}$ and $\mathcal{A}_{\mathbf{E}}$ to denote the sets of all ground vertex and edge atoms, respectively and $\mathcal{A} = \mathcal{A}_{\mathbf{V}} \cup \mathcal{A}_{\mathbf{E}}$. We note that $|\mathcal{A}_{\mathbf{V}}| = |\mathsf{VP}| \cdot |\mathbf{V}|$ and

$|\mathcal{A}_{\mathsf{E}}| = |\mathsf{EP}| \cdot |\mathsf{E}|$.

Definition 89 (annotated atom/GAP-rule/GAP). *If A is an atom and μ is an annotation, then $A : \mu$ is an* annotated atom. *If $A_0 : \mu_0, A_1 : \mu_1, \ldots, A_n : \mu_n$ are annotated atoms, then*

$$A_0 : \mu_0 \ \leftarrow \ A_1 : \mu_1 \wedge \ldots \wedge A_n : \mu_n$$

is called a GAP rule. *When $n = 0$, the above GAP-rule is called a* fact. *A GAP-rule is* ground *iff there are no occurrences of variables from either AVar or $\mathcal{V}$ in it. A generalized annotated program Π is a finite set of GAP rules.*

Every social network $SN = (\mathsf{V}, \mathsf{E}, \ell_{vert}, \ell_{edge}, w)$ can be represented by the GAP $\Pi_{SN} = \{q(v) : 1 \leftarrow \ \mid v \in \mathsf{V} \wedge q \in \ell_{vert}(v)\} \cup \{ep(V_1, V_2) : w(V_1, V_2, ep) \leftarrow \ \mid (V_1, V_2) \in \mathsf{E} \wedge \ell_{edge}(V_1, V_2) = ep\}$.

Definition 90 (embedded social network). *A social network SN is said to be* embedded *in a GAP Π iff $\Pi_{SN} \subseteq \Pi$.*

It is clear that all social networks can be represented as GAPs. When we augment Π_{SN} with other rules — such as rules describing how certain properties diffuse through the social network, we get a GAP $\Pi \supseteq \Pi_{SN}$ that captures both the structure of the SN and the diffusion principles. Here is a small example of such a GAP.

Example 8.2.2. *The GAP Π_{cell} might consist of Π_{SN} using the social network of Figure 8.1 plus the GAP-rules:*

1. *$will\_adopt(V) : 0.8 \times X + 0.2 \leftarrow adopter(V) : 1 \wedge male(V) : 1 \wedge$*

   $IM(V, V') : 0.3 \wedge female(V') : 1 \wedge will\_adopt(V') : X$.

2. *$will\_adopt(V) : 0.9 \times X + 0.1 \leftarrow adopter(V) : 1 \wedge male(V) : 1 \wedge$*

   $IM(V, V') : 0.3 \wedge male(V') : 1 \wedge will\_adopt(V') : X$.

3. *$will\_adopt(V) : 1 \leftarrow temp\_adopter(V) : 1 \wedge male(V) : 1 \wedge email(V', V) : 1 \wedge$*

   $female(V') : 1 \wedge will\_adopt(V') : 1$.

Rule (1) says that if V is a male adopter and V' is female and the weight of V's instant messages to V' is 0.3 or more, and we previously thought that V would be an adopter with confidence X, then we can infer that V will adopt the new plan with confidence $0.8 \times X + 0.2$. *The other rules may be similarly read.*

Suppose $\mathcal{S}$ is a social network and $\Pi \supseteq \Pi_{\mathcal{S}}$ is a GAP. In this case, we call the rules in $\Pi - \Pi_{\mathcal{S}}$ *diffusion rules.* We use $\mathcal{A}_{d-hd}$ to refer to the set of ground atoms in the heads of all diffusion rules of some fixed Π. Note that for the models presented in this chapter, $\mathcal{A}_{d-hd} \subseteq \mathcal{A}_{\mathsf{V}}$, meaning edge weights do not change as a result of the diffusion process. However, for the general case, it is possible for edge weights to change as a result of the diffusion process.

GAPs have a formal semantics that can be immediately used. An interpretation I is any mapping from the set of all grounds atoms to $[0, 1]$. The set $\mathcal{I}$ of all interpretations can be partially ordered via the ordering: $I_1 \preceq I_2$ iff for all ground atoms A, $I_1(A) \leq I_2(A)$. $\mathcal{I}$ forms a complete lattice under the $\preceq$ ordering.

Definition 91 (satisfaction/entailment). *An interpretation I* satisfies *a ground annotated atom $A : \mu$, denoted $I \models A : \mu$, iff $I(A) \geq \mu$. I satisfies the ground*

GAP-rule $AA_0 \leftarrow AA_1 \wedge \ldots \wedge AA_n$ *(denoted* $I \models AA_0 \leftarrow AA_1 \wedge \ldots \wedge AA_n$*) iff either (i)* I *satisfies* AA_0 *or (ii) there exists an* $1 \leq i \leq n$ *such that* I *does not satisfy* AA_i*.* I *satisfies a non-ground atom (rule) iff* I *satisfies all ground instances of it. GAP* Π entails AA*, denoted* $\Pi \models AA$*, iff every interpretation* I *that satisfies all rules in* Π *also satisfies* AA*.*

As shown by [86], we can associate a fixpoint operator with any GAP Π that maps interpretations to interpretations.

Definition 92. *Suppose* Π *is any GAP and* I *an interpretation. The mapping* $\mathbf{T}_\Pi$ *that maps interpretations to interpretations is defined as* $\mathbf{T}_\Pi(I)(A) = \mathbf{sup}\{\mu \mid A : \mu \leftarrow AA_1 \wedge \ldots \wedge AA_n$ *is a ground instance of a rule in* Π *and for all* $1 \leq i \leq n$*,* $I \models AA_i\}$*.*

[86] show that $\mathbf{T}_\Pi$ is monotonic and has a least fixpoint $lfp(\mathbf{T}_\Pi)$. Moreover, they show that Π entails $A : \mu$ iff $\mu \leq lfp(\mathbf{T}_\Pi)(A)$ and hence $lfp(\mathbf{T}_\Pi)$ precisely captures the ground atomic logical consequences of Π. They also define the *iteration* of $\mathbf{T}_\Pi$ as follows $\mathbf{T}_\Pi \uparrow 0$ is the interpretation that assigns 0 to all ground atoms. $\mathbf{T}_\Pi \uparrow (i+1) = \mathbf{T}_\Pi(\mathbf{T}_\Pi \uparrow i)$. This can be extended in the obvious way to limit ordinals.

Thus, we see that any social network $\mathcal{S}$ can be represented as a GAP $\Pi_\mathcal{S}$. We will show (in Section 8.4) that many existing diffusion models for a variety of phenomena can be expressed as a GAP $\Pi \supseteq \Pi_\mathcal{S}$ by adding some GAP-rules describing the diffusion process to $\Pi_\mathcal{S}$.

8.3 Social Network Optimization (SNOP) Queries

8.3.1 Basic SNOP Queries

In this section, we develop a formal syntax and semantics for optimization in social networks, taking advantage of the above embedding of SNs into GAPs. In particular, we formally define SNOP-queries, examples of which have been informally introduced earlier as **(Q1)** and **(Q2)**. We see from queries **(Q1)**,**(Q2)** that a SNOP-query looks for a set $\mathbf{V}'$ of vertices and has the following components: (i) an objective function expressed via an aggregate operator, (ii) an integer $k \geq 0$, (iii) a set of conditions that each vertex in $\mathbf{V}'$ must satisfy, and (iv) a *goal* atom $g(V)$ where g is a vertex predicate and V is a variable.

Aggregates. It is clear that in order to express queries like **(Q1)**,**(Q2)**, we need aggregate operators which are mappings $agg : \mathsf{FM}([0,1]) \rightarrow \mathbb{R}^+$ ($\mathbb{R}^+$ is the set of non-negative reals) where $\mathsf{FM}(X)$ denotes the set of all finite multisets that are subsets of X. Relational DB aggregates like SUM,COUNT,AVG,MIN,MAX are all aggregate operators which can take a finite multiset of reals as input and return a single positive real.

Vertex condition. A vertex condition VC is a conjunction of annotated vertex atoms containing exactly one variable.

Thus, in our example, $male(V) : 1 \wedge adopter(V) : 1$ is a conjunctive vertex condition, but $male(V) : 1 \wedge email(V, V') : 1$ is not. We are now ready to define a SNOP-query.

Definition 93 (SNOP-query)**.** *A* SNOP-query *is a* 4*-tuple* $(agg, VC, k, g(V))$ *where* agg *is an aggregate,* VC *is a vertex condition,* $k \geq 0$ *is an integer, and* $g(V)$ *is a goal atom.*

If we return to our cell phone example, we can set $agg = \mathsf{SUM}$, $k = 3$ (for example), $VC = true$ and the goal to be $adopter(V)$. Here, the goal is to find a set ANS of vertices v such that ANS's cardinality is 3 or less and such that $\mathsf{SUM}\{lfp(\mathbf{T}_\Pi)(adopter(v)) \mid v \in ANS\}$ is maximized. Here, the SUM is applied to a multiset rather than a set. Note that the diffusion model's impact is captured in this example via the $lfp(T_\Pi)(adopter(v))$ expression which, intuitively, tells us the confidence (according to the diffusion model) that vertex v will be an adopter. If we return to an extended version of our cell phone example and we want to ensure that the vertices in ANS are not employees of the company (let's call this company C), then we merely can set $VC = not\_employee(V) : 1$.[6] This query now asks us to find a set ANS of three or less vertices — none of which is an employee of the company C — such that the sum $\Sigma_{v \in ANS}\{lfp(\mathbf{T}_\Pi)(adopter(v)) \mid v \in ANS\}$ is maximized.

Our framework also allows the vertex condition VC to have annotations other than 1. So in our cell phone example, the company could explicitly exclude anyone whose "opinion" toward the company is negative. If opinion is quantified on a continuous $[0, 1]$ scale (such automated systems do exist [166]), then the vertex condition

[6]In this chapter, we do not consider non-monotonic negation and choose merely to represent *not_employee* as a predicate symbol. The extension of GAPs to non-monotonic negation has been studied [31] — future work can extend non-monotonic negation to the case of the type of social network optimization problems studied in this chapter.

might be restated as $VC = not\_employee(V) : 1 \wedge negative\_opinion(V)\_C : 0.7$ which says that the company wants to exclude anyone whose negativity about the company exceeds 0.7 according to an opinion scoring engine such as [166].

Definition 94 (pre-answer/value). *Suppose an SN $\mathcal{S} = (\mathbf{V}, \mathbf{E}, \ell_{vert}, \ell_{edge}, w)$ is embedded in a GAP Π. A* pre-answer *to the SNOP query $Q = (agg, VC, k, g(V))$ w.r.t. Π is any set $\mathbf{V}' \subseteq \mathbf{V}$ such that: (i) $|\mathbf{V}'| \leq k$, (ii) for all vertices $v' \in \mathbf{V}'$, $lfp(\mathbf{T}_{\{\Pi \cup \{g(v'):1 \leftarrow \mid v' \in \mathbf{V}'\}})\models VC[V/v']$. We use $\mathsf{pre\_ans}(Q)$ to denote the set of all pre-answers to query Q.*

The value*, $value(\mathbf{V}')$, of a pre-answer $\mathbf{V}'$ is $agg(\{lfp(\mathbf{T}_{\Pi \cup \{g(v'):1 \leftarrow \mid v' \in \mathbf{V}'\}})(g(V)) | V \in \mathbf{V}\})$ — here, the aggregate is applied to a multi-set rather than a set. We also note that we can define value as a mapping from interpretations to reals based on a SNOP query. We say $value(I) = agg(\{I(g(v)) \mid v \in \mathbf{V}\})$.*

If we return to our cell phone example, $\mathbf{V}'$ is the set of vertices to which the company is considering giving free plans. $(value(\mathbf{V}'))$ is computed as follows.

1. Find the least fixpoint of $T_{\Pi'_{cell}}$ where Π'_{cell} is Π_{cell} expanded with annotated atoms of the form $adopter(V') : 1$ for each vertex $V' \in \mathbf{V}'$.

2. For each vertex $V \in \mathbf{V}$ (the entire set of vertices, not just $\mathbf{V}'$ now), we now find the confidence assigned by the least fixpoint.

3. Summing up these confidences gives us a measure of the expected number of plan adoptees.

Definition 95 (answer). *Suppose an SN $\mathcal{S} = (\mathbf{V}, \mathbf{E}, \ell_{vert}, \ell_{edge}, w)$ is embedded in* *a GAP* Π *and* $Q = (agg, VC, k, g(V))$ *is a SNOP-query. A pre-answer* $\mathbf{V}'$ *is an* answer *to the SNOP-query* Q *iff the SNOP-query has no other pre-answer* $\mathbf{V}''$ *such that* $value(\mathbf{V}'') > value(\mathbf{V}')$.[7]

The answer set, $\mathsf{ans}(Q)$, *to the SNOP-query* $Q = (agg, VC, k, g(V))$ *w.r.t.* Π *is the set of all answers to* Q.

Example 8.3.1. *Consider the GAP* Π_{cell} *with the social network from Figure 8.1 embedded and the SNOP-query* $Q_{cell} = (SUM, true, 3, will\_adopt)$. *The sets* $\mathbf{V}'_1 = \{v_{15}, v_{19}, v_6\}$ *and* $\mathbf{V}'_2 = \{v_{15}, v_{18}, v_6\}$ *are both pre-answers. In the case of* $\mathbf{V}'_1$, *two applications of the* $\mathbf{T}_\Pi$ *operator yields a fixpoint where the vertex atoms formed with will_adopt in set* $\{v_{15}, v_{19}, v_6, v_{12}, v_{18}, v_7, v_{10}\}$ *are annotated with* 1. *For* $\mathbf{V}_2$, *only one application of* $\mathbf{T}_\Pi$ *is required to reach a fixpoint, and the corresponding set of vertices (where the vertex atom formed with will_adopt is annotated with* 1*) is* $\{v_{15}, v_6, v_{12}, v_{18}, v_7, v_{10}\}$. *As these are the only vertex atoms formed with will_adopt that have a non-zero annotation after reaching the fixed point, we know that* $value(\mathbf{V}'_1) = 7$ *and* $value(\mathbf{V}'_2) = 6$. *As* $value(\mathbf{V}'_1) > value(\mathbf{V}'_2)$, $\mathbf{V}'_1$ *is an answer to this SNOP-query.*

8.3.2 Special Cases of SNOP Queries

In this section, we examine several special cases of SNOP queries that still allow us to represent a wide variety of diffusion models. Table 8.1 illustrates the special cases discussed in this section while Table 8.2 illustrates the various proper-

[7]Throughout this chapter, we only treat maximization problems - there is no loss of generality in this because minimizing an objective function f is the same as maximizing $-f$.

Type	Special Case	Reference
Special cases of Π	Linear GAP	Definition 96
Special cases of agg	Monotonicity	Definition 97
	Positive linear	Definition 98
Special cases of $value$	Zero-starting	Definition 100
	A-priori VC	Definition 101

Table 8.1: Special cases of SNOP queries

ties.

Special Cases of the GAP. First, we present a class of GAPs called *linear* GAPs. Intuitively, a linear GAP is a GAP where the annotations in the rule head are linear functions and the annotations in the body are variables is the atom is a vertex atom and constant otherwise. It is important to note that a wide variety of diffusion models can be represented with GAPs that meet the requirements of this special case. We define it formally below.

Definition 96 (Linear GAP). *A GAP-rule r of the form*

$$H_0 : \mu_0 \leftarrow A_1 : \mu_1 \wedge \ldots \wedge A_n : \mu_n$$

is said to be linear *iff there exist constants $c_0, \ldots, c_i, \ldots, c_n$ where $\forall i$, $c_i \in [0, 1]$ and each ground instance $r\theta$ of r has the form*

$$H_0 : c_0 + c_1 \cdot X_1\theta + ... + c_n \cdot X_n\theta \leftarrow A_1 : X_1\theta \wedge \ldots \wedge A_n : X_n\theta$$

and $\Sigma_{i=1}^{n} c_i \in [0,1]$.

A GAP is linear iff each rule in it is linear.

Note that the linear GAP allows for a wide variety of models to be expressed. For example, suppose we have a diffusion model in which the edge atoms do not appear in any rule head (except the facts that embed the SN). In this case, edge weights can be treated as constants. Hence, we can allow rules where the annotation of a vertex atom is multiplied or divided by an edge weight (as they behave as constants) - provided the sum of all constants is in the interval $[0,1]$. Section 8.4 will show that several well-known network diffusion models can be embedded into our framework. Diffusion Models 8.4.2 and 8.4.4 are linear GAPs while Diffusion Models 8.4.1 and 8.4.3 are not.

Special Aggregates. We define two types of aggregates — *monotonic* aggregates and *positive linear* aggregates.

To define monotonicity, we first define a partial ordering $\sqsubseteq$ on multi-sets of numbers as follows. $X_1 \sqsubseteq X_2$ iff there exists an *injective* mapping $\beta : X_1 \to X_2$ such that $(\forall x_1 \in X_1) x_1 \leq \beta(x_1)$.

Definition 97 (Monotonic Aggregate)**.** *The aggregate agg is* ***monotonic (resp. anti-monotonic)*** *iff whenever* $X_1 \sqsubseteq X_2$*, it is the case that* $agg(X_1) \leq agg(X_2)$ *(resp.* $agg(X_2) \leq agg(X_1)$*).*

Definition 98 (Positive-Linear Aggregate)**.** *The aggregate agg, applied to the finite multiset* $\mathsf{FM}(X)$ *is* ***positive linear*** *iff it is of the form* $agg(\mathsf{FM}(X)) = c_0 + \Sigma_{x_i \in \mathsf{FM}(X)} c_i x_i$ *where (for* $n = |\mathsf{FM}(X)|$*)* $c_1, \ldots, c_n \geq 0$*. Note that* c_0 *can be positive,* *negative, or 0.*

Proposition 70. *If agg is positive-linear, then it is monotonic.*

It is important to note that in our definition of positive-linear, we only require that the coefficients associated with the elements of the multi-set be positive - we allow for an additive constant to be negative. One obvious example of a positive-linear aggregate is SUM. Any positive, weighted sum will also meet these requirements – an example is the fixed-subset average function given below.

Definition 99 (Fixed-Subset Average)**.** *For set X of reals, given a fixed subset $X_{subset} \subseteq X$, the* ***fixed-subset average*** *is the quantity:*

$$average(X_{subset}) = \frac{1}{card(X_{subset})} \sum_{x \in X_{subset}} (x)$$

Special cases of the query. We now describe two special cases of the query. In one case, we consider *zero-starting value* functions, while in a second case, we consider *a-priori* vertex conditions VC. Intuitively, zero-starting means that $value(\emptyset) = 0$. However, there are several ways in which this criteria may be met - for example, we can simply adjust the aggregate by subtracting a constant (which, for positive-linear aggregates, would still allow an aggregate to meet our definition of positive-linear). An *a-priori* VC is one where $lfp(\mathbf{T}_\Pi)$ satisfies VC iff VC was satisfied already by $\mathbf{T}_\Pi \uparrow 1$. Intuitively, an *a-prior* VC is like a "fact" in classical logic programming and where the application of the fixpoint operator makes no change to what was true originally. We present formal definitions below.

Definition 100 (Zero-starting)**.** *A SNOP-query is* ***zero-starting*** *(w.r.t. a given social network $\mathcal{S}$ and a GAP $\Pi \supseteq \Pi_{\mathcal{S}}$) iff $value(\emptyset) = 0$.*

Note that the function *value*() is *uniquely defined* by a social network, a SNOP-query, and a diffusion model Π and hence the above definition is well defined. The result below states that as long as we consider positive linear aggregates, we can always modify a non zero-starting aggregate to one that is.

Proposition 71. *If a SNOP-query is not zero-starting w.r.t. a social network $\mathcal{S}$ and a GAP $\Pi \supseteq \Pi_{\mathcal{S}}$, and the aggregate is positive-linear, it can be expressed as a zero-starting SNOP-query in linear time while still maintaining a positive-linear aggregate.*

Definition 101 (A-Priori VC)**.** *In an **a-priori** VC SNOP-query, for set $\mathbf{V}' \subseteq \mathbf{V}$, we modify the definition of value($\mathbf{V}'$) (Definition 94, part ii) as follows:*
For all vertices $v' \in \mathbf{V}'$, $g(v') : 1 \wedge \bigwedge_{pred \in \ell_{vert}(v')} pred(v') : 1 \models VC[V/v']$.

Note that both examples **(Q1)**,**(Q2)** we gave in the Introduction have a-priori VCs. If, in the cell phone example, we require that the free cell phones are given to non-employees, then this is an a-priori VC because being an employee is not determined by the diffusion process, but by whether a vertex in the social network had the associated non-employee property. Likewise, in the case of an a-priori VC in the medical example saying that an individual below 5 should not get the medicine, this boils down to a static labeling of each node's age (below 5 or not) which is not affected by the diffusion process.

Example 8.3.2. *Consider a painting company attempting to conduct a viral marketing strategy. Consider the simple social network depicted in Figure 8.2. White vertices represent individuals with whom the paint company has had prior business.*

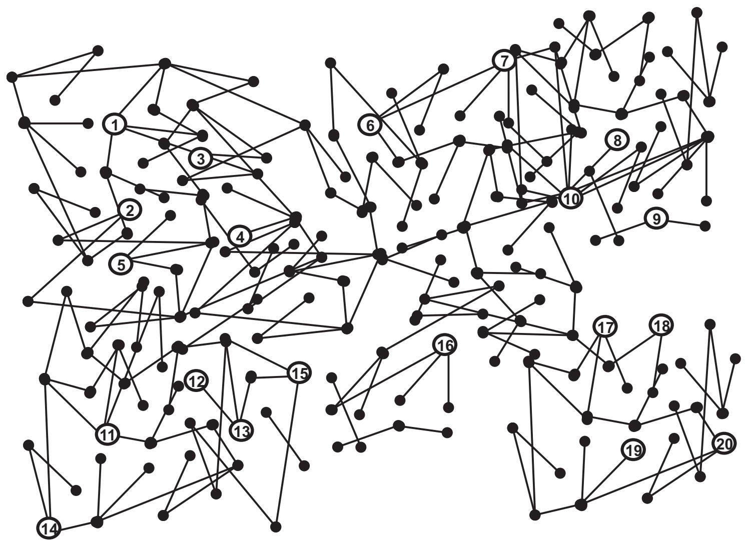


Figure 8.2: Social Network for the painting company.

Suppose the represent this with a predicate prior and vertex atoms formed with some white vertex v and prior are annotated with 1 *(i.e.* $prior(v) : 1$*) while the rest are annotated with* 0*. Based on local telemarketing legislation, the paint company can only contact individuals with which it had a prior business relationship. As the paint company intends to market to a set of high-payoff vertices in a short period of time, it is unreasonable to expect the number vertices where where a prior vertex atom is annotated with* 1 *to increase. Hence, they create a logic program such that the vertex condition* $VC(V) = prior(V) : 1$ *is* ***a-priori****.*

8.3.3 Properties of SNOPs

In this section, we will prove several usful properties of SNOP queries that use various combinations of the assumptions presented in the previous section. Later,

Propoerty	Assumptions
Monotonicity of *value* (Lemma 23)	Monotonicity
Multiset $\{\mathbf{V}' \subseteq \mathbf{V} \mid \mathbf{V}' \textit{ is a pre-answer}\}$ is a uniform matroid (Lemma 24)	A-priori VC
Submodularity (Theorem 47)	Linear GAP Positive linear agg A-priori VC

Table 8.2: Properties that can be proven given certain assumptions

we will leverage some of these properties in our algorithms. Table 8.2 summarizes the different properties that we prove in this section (as well as what assumptions we make to prove these properties). Table 8.3 shows how these properties are leveraged in the algorithms that we will present later in the chapter.

The first property we show is that the value function is monotonic. This follows directly from the monotonicity of the aggregate - hence we present the following easy

Algorithm	Property
Exact algorithm with pruning (Section 8.5.2)	Monotonicity of *value*
Approx. Ratio on Greedy Algorithm (Section 8.5.3)	Submodularity Zero-starting

Table 8.3: How the various properties are leveraged in the Algorithms

lemma.

Lemma 23. *Given SNOP query $Q = (agg, VC, k, g(V))$ (w.r.t. SN $\mathcal{S}$ and GAP $\Pi \supseteq \Pi_{\mathcal{S}}$), if agg is monotonic (Definition 97), then value (defined as per Q and Π) is monotonic.*

Next, we show that the multiset of pre-answers is a uniform matroid in the special case of an a-priori VC.

Lemma 24. *Given SNOP query $Q = (agg, VC, k, g(V))$ (w.r.t. SN $\mathcal{S}$ and GAP $\Pi \supseteq \Pi_{\mathcal{S}}$), if VC is applied a-priori (as per Definition 101), the set of pre-answers (to query Q) is a uniform matroid.*

An important property in social networks is *submodularity*. Intuitively,if X is a set, then a function $f : 2^X \to \mathbb{R}$ is *submodular* iff whenever $X_1 \subseteq X_2$ and $x \notin X_2$, $f(X_1 \cup \{x\}) - f(X_1) \geq f(X_2 \cup \{x\}) - f(X_2)$. The following result states that the *value*() function associated with a linear GAP with an a-priori vertex condition VC and a positive linear aggregate function is guaranteed to be submodular.

Theorem 47. *Given SNOP query $Q = (agg, VC, k, g(V))$ (w.r.t. SN $\mathcal{S}$ and GAP $\Pi \supseteq \Pi_{\mathcal{S}}$) if the following criteria are met:*

- Π *is a linear GAP*

- VC *is applied a-priori*

- *agg is positive linear,*

then value (defined as per Q and Π) is ***sub-modular***.

In other words, for $\mathbf{V}_{cond} \equiv \{v' | v' \in \mathbf{V}$ and $(g(v') : 1 \wedge \bigwedge_{pred \in \ell_{vert}(v')} pred(v') : 1 \models$

$VC[V/v'])\}$ *and sets* $\mathbf{V}_1 \subseteq \mathbf{V}_2 \subseteq \mathbf{V}_{cond}$ *and* $v \in \mathbf{V}_{cond}$*,* $v \notin \mathbf{V}_1 \cup \mathbf{V}_2$*, the following holds:*

$$value(\mathbf{V}_1 \cup \{v\}) - value(\mathbf{V}_1) \geq value(\mathbf{V}_2 \cup \{v\}) - value(\mathbf{V}_2)$$

Proof Sketch: *Consider a linear polynomial with a variable for each vertex in the set of vertices that meet the a-priori* VC*, where setting the variable to* 1 *corresponds to the vertex being picked and setting it to* 0 *indicates otherwise. For any subset of vertices meeting the a-priori* VC*, there is an associated polynomial of this form such that when the variables corresponding to the vertices are set to* 1 *(and the rest set to* 0*), the answer is equal to the corresponding value for that set. For a sets* $\mathbf{V}_1, \mathbf{V}_2$ *and vertex* v *(as per the statement), we show that submodualirty holds by manipulating such polynomials.*

Example 8.3.3. *We now show an example of a SNOP-query that is* ***not*** *submodular when a non-linear GAP is considered. Figure 8.3 shows a social network. This social network has one edge predicate, e, and all edges are weighted with* 1*. Nodes in the network are either susceptible to the disease (circles) or carriers (diamonds) - the associated predicates are suc and car respectively. Additionally, we have the predicates inf, exp denoting vertices that have been infected by or exposed to the disease.*

Let $\Pi_{disease}$ *be the embedding of this network plus the following diffusion rules.*

$$exp(V) : 1 \leftarrow inf(V) : 1$$

$$exp(V) : 1 \leftarrow e(V', V) : 1 \wedge inf(V') : 1 \wedge suc(V) : 1$$

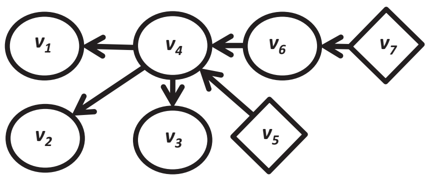


Figure 8.3: Social network corresponding with Example 8.5.1 concerning disease spread.

$$inf(V) : \lfloor \frac{\sum_i I_i}{\sum_i E_i} \rfloor \leftarrow exp(V) : 1 \wedge \bigwedge_{V_i | (V_i, V) \in \mathbf{E}} (edge(V_i, V) : E_i \wedge inf(V_i) : I_i)$$

Intuitively, the second rule says that a vertex becomes exposed if that vertex is susceptible and it has at least one incoming neighbor that is infected. The third rule states that a vertex becomes infected if it is exposed and all its neighbors are infected. Suppose, for illustrative purposes, that $inf(v_5), inf(v_7)$ *are annotated with* 1.

Consider the function value based on the SNOP query $(\Pi_{disease}, SUM, true, 2, inf(V))$. *Obviously, as the GAP is not linear, it does not meet the requirements of Theorem 47 to prove submodularity. We can actually show through counterexample, that this SNOP query is not submodular. Consider the following:*

$$value(\{v_1, v_5\}) - value(\{v_1\}) = 1$$

and

$$value(\{v_1, v_7, v_5\}) - value(\{v_1, v_7\}) = 5$$

This shows a clear violation of submodularity.

8.3.4 The Complexity of SNOP Queries

We now study the complexity of answering a SNOP query. First, we show that SNOP-query answering is NP-hard by a reduction from max k-cover [46]. We show that the problem is NP-hard even when many of the special cases hold.

Theorem 48. *Finding an answer to SNOP query $Q = (agg, VC, k, g(V))$ (w.r.t. SN $\mathcal{S}$ and GAP $\Pi \supseteq \Pi_{\mathcal{S}}$) is NP-hard (even if Π is a linear GAP, $VC = \emptyset$, $agg = SUM$ and value is zero-starting).*

Proof Sketch: *The known NP-hard problem of MAX-K-COVER [46] is defined as follows.*

INPUT: Set of elements, S and a family of subsets of S, $\mathcal{H} \equiv \{H_1, \ldots, H_{max}\}$, and positive integer K.

OUTPUT: Less than or equal to K subsets from $\mathcal{H}$ such that the union of the subsets covers a maximal number of elements in S.

We show that MAX-K-COVER can be embedded into a social network and that the corresponding SNOP-query gives an optimal answer to MAX-K-COVER. The embedding is done by creating a social network resembling a bipartite graph, where vertices represent either the elements or the subsets from the input of MAX-K-COVER. For every vertex pair representing a set and an element of that set, there is an edge from the set vertex to the element vertex. A single vertex and edge predicate are used - vertex and edge. A single non-ground diffusion rule is added to the GAP: $vertex(V) : X \leftarrow vertex(V') : X \wedge edge(V', V) : 1$. The aggregate is simply the sum of the annotations associated with the vertex atoms. We show that the picked *vertices that maximize the aggregate correspond with picked subsets that maximize output of the problem. Also, as we do not use VC, the GAP is linear, and the aggregate is positive-linear, we know that the value function is submodular.*

Under some reasonable conditions, the problem of answering SNOP-queries is also in NP.

Theorem 49. *Finding an answer to a decision problem associated with SNOP query $Q = (agg, VC, k, g(V))$ (w.r.t. SN $\mathcal{S}$ and GAP $\Pi \supseteq \Pi_{\mathcal{S}}$) where agg and the functions in $\mathcal{F}$ are polynomially computable is in-NP.*

Most common aggregate functions like SUM, AVERAGE, Weighted average, MIN, MAX, COUNT are all polynomially computable. Moreover, the assumption that the functions in $\mathcal{F}$ are polynomially computable is also reasonable.

Later in this chapter, we shall address the problem of answering a SNOP-query using an approximation algorithm. We re-state the definition of approximation below (see [54]).

Definition 102 (Approximation)**.** *For a given instance I of a maximization problem with optimal solution $OPT(I)$, an α-approximation algorithm A is an algorithm such that for any instance I*

$$OPT(I) \leq \alpha \cdot A(I)$$

Based on the above definition, we shall say that $\mathbf{V'}$ is an $\frac{1}{\alpha}$-approximation to a SNOP query if $value(\mathbf{V'}_{opt}) \leq \alpha \cdot value(\mathbf{V'})$ (where $\mathbf{V}_{opt}$ is the answer to the SNOP query). Likewise, the algorithm that produces $\mathbf{V'}$ in this case is an α-approximation algorithm. We note that due to the nature of the reduction from MAX-K-COVER that we used to prove NP-hardness, we can now show that unless $\mathbf{P} = \mathbf{NP}$, there is no PTIME-approximation of the SNOP-query answering problem that can guarantee that the approximate answer is better than 0.63 of the optimal value. This gives us an idea of the limits of approximation possible for a SNOP-query with a polynomial-time algorithm. Later, we will develop a greedy algorithm that precisely matches this approximation ratio.

Theorem 50. *Answering a SNOP query $Q = (agg, VC, k, g(V))$ (w.r.t. SN $\mathcal{S}$ and GAP $\Pi \supseteq \Pi_{\mathcal{S}}$), cannot be approximated in PTIME within a ratio of $\frac{e-1}{e} + \epsilon$ for some $\epsilon > 0$ (where e is the inverse of the natural log) unless $\mathbf{P} = \mathbf{NP}$ – even if Π is a linear GAP, $VC = \emptyset$, $agg = SUM$ and value is zero-starting.*

(That is, there is no polynomial-time algorithm that can approximate value within a factor of about 0.63 under standard assumptions.)

8.3.5 Counting Complexity of SNOP-Queries

In this section, we ask the question: how many answers are there to a SNOP-query $(agg, VC, k, g(V))$? In the case of the cell phone example, this corresponds to asking "How many sets ANS of people are there in the the network such that ANS has k or fewer people and ANS optimizes the aggregate, subject to the vertex condition VC?" If there are m such sets $ANS_1, \ldots, ANS_m$, this means the cell phone company can give the free cell phone plan to eithe all members of ANS_1 or to all members of ANS_2, and so forth. The "counting complexity" problem of determining m is is $\#P$-complete.

Theorem 51. *Counting the number of answers to SNOP query $Q = (agg, VC, k, g(V))$ (w.r.t. SN $\mathcal{S}$ and GAP $\Pi \supseteq \Pi_{\mathcal{S}}$) is $\#P$-complete.*

8.3.6 The SNOP-ALL Problem

Though the cell phone company may not want to give free calling plans to all possible members of $ANS_1, \ldots, ANS_m$, in the case of the epidemiology example where a government wants to check the spread of a disease, the government may reason as follows. It has only k units of medicine to hand out now, and hence it needs to choose to give those medicines to all members of exactly one of the ANS_i's. However, the government wants to know how many people are in all of the ANS_i's so as to determine how to plan for the future (e.g. placing future orders).

Although the counting version of the query is $\#P$-hard, finding the *union* of all answers to a SNOP query is no harder than a SNOP query (w.r.t. PTIME reductions). We shall refer to this problem as *SNOP-ALL* - and it reduces both to and from a regular SNOP query in PTIME.

We first prove NP-hardness, showing by showing we can answer a SNOP query in PTIME with an oracle to SNOP-ALL.

Theorem 52. *Given SNOP query $Q = (agg, VC, k, g(V))$ (w.r.t. SN $\mathcal{S}$ and GAP $\Pi \supseteq \Pi_{\mathcal{S}}$), finding $\bigcup_{V'_{ans} \in \mathsf{ans}(Q)} V'_{ans}$ is NP-hard.*

Proof Sketch: *We show NP-hardness by the embedding of a SNOP-query in a SNOP-ALL query via the following informal algorithm (*$\mathsf{FIND\text{-}SET}$*) that takes an* *instance of SNOP-ALL (Q) and some vertex set V^*, $|V^*| \leq k$.*

1. *If $|V^*| = k$, return V^**

2. *Else, solve SNOP-ALL(V^*), returning set V''.*

    (a) *If $V'' - V^* \equiv \emptyset$, return V^**

    (b) *Else, pick $v \in V'' - V^*$ and return FIND-SET($Q, V^* \cup v$)*

The theorem below shows that SNOP-ALL can be answered in PTIME with an oracle to a SNOP-query.

Theorem 53. *Given SNOP query $Q = (agg, VC, k, g(V))$ (w.r.t. SN $\mathcal{S}$ and GAP $\Pi \supseteq \Pi_{\mathcal{S}}$), finding $\bigcup_{V'_{ans} \in \mathsf{ans}(Q)} V'_{ans}$ reduces to $|V| + 1$ SNOP-queries.*

Proof Sketch: *Using an oracle that correctly answers SNOP-queries, we can answer a SNOP-ALL query by setting up $|V|$ SNOP-queries as follows:*

- *Let k_{all} be the k value for the SNOP-ALL query and for each SNOP-query i, let k_i be the k for that query. For each query i, set $k_i = k_{all} - 1$.*

- *Number each element of $v_i \in V$ such that $g(v_i)$ and $VC(v_i)$ are true. For the ith SNOP-query, let v_i be the corresponding element of V*

- *Let Π_i refer to the GAP associated with the ith SNOP-query and Π_{all} be the program for SNOP-ALL. For each program Π_i, add fact $g(v_i) : 1$*

- *For each SNOP-query i, the remainder of the input is the same as for SNOP-ALL.*

After the construction, do the following:

1. *We shall refer to a SNOP-query that has the same input as SNOP-ALL as the "primary query." Let* ${V'_{ans}}^{(pri)}$ *be an answer to this query and* $value({V'_{ans}}^{(pri)})$ *be the associated value.*

2. *For each SNOP-query* i*, let* ${V'_{ans}}^{(i)}$ *be an answer and* $value({V'_{ans}}^{(i)})$ *be the associated value.*

3. *Let* V''*, the solution to SNOP-ALL be initialized as* $\emptyset$*.*

4. *For each SNOP-query* i*, if* $value({V'_{ans}}^{(i)}) = value({V'_{ans}}^{(pri)})$*, then add vertex* v_i *to* V''*.*

8.4 Applying SNOPs to Real Diffusion Problems

In this section, we show how SNOPs can be applied to real-word diffusion problems. We look at three categories of diffusion models – **tipping** models (Section 8.4.1), where a given vertex adopts a behavior based on the ratio of how many of its neighbors previously adopted the behavior, **cascading** models (Section 8.4.2), where a property passes from vertex to vertex solely based on the strength of the relationship between the vertices, and **homophilic** models (Section 8.4.3), where vertices with similar properties tend to adopt the same behavior – irrespective (or in addition to) of network relationships. *None of these approaches solves SNOP-queries — they merely specify diffusion models rather than performing the kinds of optimizations that we perform in SNOP-queries.*

8.4.1 Tipping Diffusion Models

Tipping models [150, 61] have been studied extensively in economics and sociology to understand diffusion phenomena. In tipping models, a vertex changes a property based on the cumulative effect of its neighbors. In this section, we present the tipping model of Jackon-Yariv [73], which generalizes many existing tipping models.

The Jackson-Yariv Diffusion Model [73]. In this framework, the social network is just a directed graph $\mathsf{G}' = (\mathsf{V}', \mathsf{E}')$ consisting of a set of agents (e.g. people). Each agent has a default behavior (A) and a new behavior (B). Suppose d_i denotes the degree of a vertex v_i. [73] use a function $\gamma : \{0, \ldots, |\mathsf{V}| - 1\} \rightarrow [0, 1]$ to describe how the number of neighbors of v affects the benefits to v for adopting behavior B. For instance, $\gamma(3)$ specifies the benefits (in adopting behavior B) that accrue to an arbitrary vertex $v \in \mathsf{V}'$ that has three neighbors. Let π_i denote the fraction of neighbors of v_i that have adopted behavior B. Let constants b_i and ρ_i be the benefit and cost respectively for vertex v_i to adopt behavior B, respectively. [73] state that node v_i switches to behavior B iff $\frac{b_i}{\rho_i} \cdot \gamma(d_i) \cdot \pi_i \geq 1$.

Returning to our cell-phone example, one could potentially use this model to describe the spread of the new plan. In this case, behavior A would be adherence to the current plan the user subscribes to, while B would be the use of the new plan. The associated SNOP-query would find a set of nodes which, if given a free plan, would jointly maximize the expected number of adoptees of the plan. Cost and benefit could be computed from factors such as income, time invested in switching plans, etc. Below is a straight-forward embedding of this model into our framework.

Diffusion Model 8.4.1 (Jackson-Yariv model)**.** *Given a Jackson-Yariv model consisting of $\mathbf{G}' = (\mathbf{V}', \mathbf{E}')$ and g, we can set up an SN $(\mathbf{V}', \mathbf{E}'', \ell_{vert}, \ell_{edge}, w)$ as follows. We define $\mathbf{E}'' = \{(x,y),(y,x) \mid (x,y) \in \mathbf{E}'\}$. We have a single edge predicate symbol edge and ℓ_{edge} assigns 1 to all edges in $\mathbf{E}''$. Our associated GAP Π_{JY} now consists of Π_{SN} plus the single rule:*

$$B(V_i) : \lfloor \frac{b_i}{\rho_i} \cdot \gamma(\sum_j E_j) \cdot \frac{\sum_j X_j}{\sum_j E_j} \rfloor \leftarrow \bigwedge_{V_j \mid (V_j,V_i) \in \mathbf{E}''} (edge(V_j,V_i) : E_j \wedge B(V_j) : X_j)$$

It is easy to see that this rule (when applied in conjunction with Π_{SN} for a social network SN) precisely encodes the Jackson-Yariv semantics. Note that $\Pi_{disease}$ from Example 8.3.3 on page 359 is a special case of this model.

We notice right away that the above GAP is not linear. However, the good news is the non-linearity is only due to the floor function. If we eliminate the floor-function, we can represent a variant of this model where the annotation would represent an "expected likelihood" that an agent adopts behavior B. This new embedding of the Jackson-Yariv models is a linear GAP under the following conditions (forall V_i).

$$|\{V_j | (V_j,V_i) \in \mathbf{E}''\}| \cdot \frac{b_i}{\rho_i} \cdot \gamma(\sum_{V_q|(V_q,V_i)\in\mathbf{E}''} w(V_q,V_i,edge)) \cdot \frac{1}{\sum_{V_q|(V_q,V_i)\in\mathbf{E}''} w(V_q,V_i,edge)} \leq 1$$

$$\frac{b_i}{\rho_i} \cdot \gamma(\sum_{V_q|(V_q,V_i)\in\mathbf{E}''} w(V_q,V_i,edge)) \cdot \frac{1}{\sum_{V_q|(V_q,V_i)\in\mathbf{E}''} w(V_q,V_i,edge)} \in [0,1]$$

As the Jackson-Yariv model does not cause edge weights to change, they can be treated as constants upon grounding (hence, annotations of edge atoms can be multiplied or divided by the annotations of vertex atoms in the heads of the diffusion rules). This allows us to easily create a linear version of the Jackson-Yariv model below.

Diffusion Model 8.4.2 (Linear Jackson-Yariv model)**.**

$$B(V_i) : \frac{b_i}{\rho_i} \cdot \gamma(\sum_j E_j) \cdot \frac{\sum_j X_j}{\sum_j E_j} \leftarrow \bigwedge_{V_j | (V_j, V_i) \in \mathbf{E}''} (edge(V_j, V_i) : E_j \wedge B(V_j) : X_j)$$

If we consider the above model in terms of Definition 96 (Linear GAPs), for each ground diffusion rule, the annotated atom in the head, formed with $B(V_i)$ is annotated with a linear expression of the form

$$c_0 + c_1 \cdot X_1 + \ldots + c_{|\{V_j|(V_j,V_i)\in\mathbf{E}''\}|} \cdot X_{|\{V_j|(V_j,V_i)\in\mathbf{E}''\}|}$$

Here, $c_0 = 0$, and for all $j > 0$,

$$c_j = \frac{b_i}{\rho_i} \cdot \gamma(\sum_{V_q|(V_q,V_i)\in\mathbf{E}''} w(V_q, V_i, edge)) \cdot \frac{1}{\sum_{V_q|(V_q,V_i)\in\mathbf{E}''} w(V_q, V_i, edge)}$$

where each j is an index of an incoming vertex to V_i. Note that we can directly use edge weights from the original social network (as expressed by the function w) because the E_j annotations are for edge atoms and do not change in the diffusion process (as edge weights do not appear in the heads of any diffusion rules in the model). Clearly, under our stated assumption, linearity holds.

Example 8.4.1. *Figure 8.4 illustrates a social network of individuals who share photographs. Edges are directional formed with a predicate share and weighted* 1*. Vertex predicates include* $\{buys\_camera, pro\}$*. If the vertex is shaded, the vertex atom formed with pro is annotated with* 1*. All other vertex atoms are annotated with zero.*

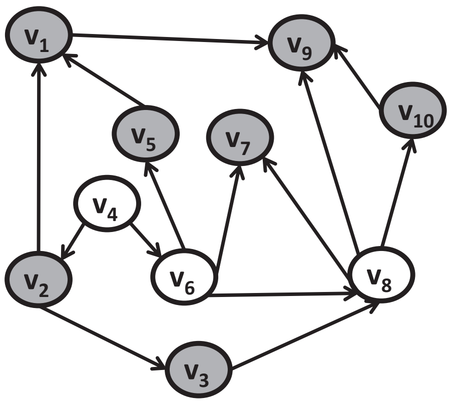


Figure 8.4: Social network of individuals sharing photographs. Shaded vertices are professional photographers. All edges are directional *share* edges.

A vendor wishes to sell a camera and wants to see how the popularity of the camera will spread in the network. He wants to use a Jackson-Yariv style diffusion model. Consider the social network embedded into a logic program, Π *along with following Jackon-Yariv style tipping diffusion rule:*

$$buys\_camera(V) : \lfloor \frac{\sum_j X_j \cdot E_j}{\sum_j E_j} \rfloor \leftarrow \bigwedge_{V_j | (V_j, V) \in \mathbf{E}} (shares(V_j, V) : E_j \wedge buys\_camera(V_j) : X_j)$$

We will call the logic program with the above diffusion rule Π_{sfwfd}*. Alternatively, we could have a linear version of it as follows (again, linearity follows by the fact that we can treat the edge weights as constants upon grounding):*

$$buys\_camera(V) : \frac{\sum_j X_j \cdot E_j}{\sum_j E_j} \leftarrow \bigwedge_{V_j | (V_j, V) \in \mathbf{E}} (shares(V_j, V) : E_j \wedge buys\_camera(V_j) : X_j)$$

We will call the logic program formed with that diffusion rule (no floor function) Π_{lin}*. In this case, the grounded diffusion rules have a head formed with the atom* *buys_camera*(V) *annotated with the linear expression*

$$c_o + c_1 \cdot X_1 + \ldots + c_{|\{V_j|(V_j,V)\in\mathbf{E}''\}|} \cdot X_{|\{V_j|(V_j,V)\in\mathbf{E}\}|}$$

Here, $c_0 = 0$ *and for all* $j > 0$ *we have,*

$$c_j = \frac{w(V_j, V, shares)}{\sum_{V_q|(V_q,V)\in\mathbf{E}} w(V_q, V, edge)}$$

where each j *is an index of an incoming vertex to* V*. Clearly, each* $c_j \in [0,1]$ *and the sum of all these constants is* 1*, which gives us linearity in accordance with Definition 96. Table 8.4 shows the least fixed point for the two different GAPs (original JY model and the linear version) that arise when we assign vertex atom* *buys_camera*(v_2) *an annotation of* 1 *— it also shows as well as the sum of the annotations.*

8.4.2 Cascading Diffusion Models

In a **cascading** model, a vertex obtains a property from one of its neighbors, typically based on the strength of its relationship with the neighbor. These models were introduced in the epidemiology literature in the early 20th century, but gained increased notice with the seminal work of [5]. Recently, cascading diffusion models have been applied to other domains as well. For example, [20] (diffusion of photos in Flickr) and [167] (diffusion of bookmarks in FaceBook) both look at diffusion process in social networks as "social cascades" of this type. In this section, we present an encoding of the popular SIR model of disease spread in our framework.

Vertex Atom	Annotation Assigned by $lfp(\mathbf{T}_{\Pi_{sfwd} \cup \{buys\_camera(v_2):1 \leftarrow\}})$	Annotation Assigned by $lfp(\mathbf{T}_{\Pi_{lin} \cup \{buys\_camera(v_2):1 \leftarrow\}})$
$buys\_camera(v_1)$	0.0	0.5
$buys\_camera(v_2)$	1.0	1.0
$buys\_camera(v_3)$	1.0	1.0
$buys\_camera(v_4)$	0.0	0.0
$buys\_camera(v_5)$	0.0	0.0
$buys\_camera(v_6)$	0.0	0.0
$buys\_camera(v_7)$	0.0	0.25
$buys\_camera(v_8)$	0.0	0.5
$buys\_camera(v_9)$	0.0	0.5
$buys\_camera(v_{10})$	0.0	0.5
SUM	2	4.25

Table 8.4: Comparison between straightforward and linear Jackson-Yariv Models

The SIR Model of Disease Spread. The SIR (*susceptible, infectious, removed*) model of disease spread [5] is a classic disease model which labels each vertex in a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ (of humans) with *susceptible* if it has not had the disease but can receive it from one of its neighbors, *infectious* if it has caught the disease and t_{rec} units of time have not expired, and *removed* where the vertex can no longer catch or transmit the disease. The SIR model assumes that a vertex v that is infected can transmit the disease to any of its neighbors v' with a probability $p_{v,v'}$ for t_{rec} units of time. We would like to "find a set of k vertices that would maximize the expected number of vertices that become infected". These are obviously good candidates to treat with appropriate medications.

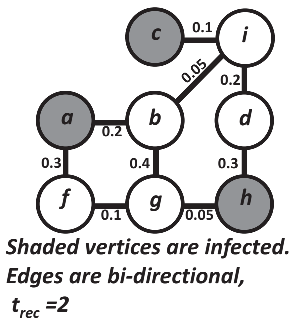
Diffusion Model 8.4.3 (SIR model)**.** *Let $\mathcal{S} = (\mathbf{V}, \mathbf{E}, \ell_{vert}, \ell_{edge}, w)$ be an SN where each edge is labeled with the predicate symbol e and $w(v, v', e) = p_{v,v'}$. We use the predicate inf to designate the initially infected vertices. In order to create a GAP Π_{SIR} capturing the SIR model of disease spread, we first define t_{rec} predicate symbols $rec_1, \ldots, rec_{t_{rec}}$ where $rec_i(v)$ intuitively means that node v was infected i days ago. Hence, $rec_{t_{rec}}(v)$ means that v is "removed." We embed $\mathcal{S}$ into GAP Π_{SIR} by adding the following diffusion rules. If $t_{rec} > 1$, we add a non-ground rule for each $i = \{2, \ldots, t_{rec}\}$ - starting with t_{rec}:*

$$\begin{aligned} rec_i(V) : R &\leftarrow rec_{i-1}(V) : R \\ rec_1(V) : R &\leftarrow inf(V) : R \\ inf(V) : (1-R) \cdot P_{V',V} \cdot (P_{V'} - R') &\leftarrow rec_{t_{rec}}(V) : R \wedge e(V', V) : P_{V',V} \wedge \\ &\qquad inf(V') : P_{V'} \wedge rec_{t_{rec}}(V') : R'. \end{aligned}$$

The first rule says that if a vertex is in its $(i-1)$'th day of recovery with certainty R in the j'th iteration of the $\mathbf{T}_{\Pi_{SIR}}$ operator, then the vertex is i days into recovery (with the same certainty) in the $j+1$'th iteration of the operator. Likewise, the second rule intuitively encodes the fact that if a vertex became infected with certainty R in the j'th iteration of the $\mathbf{T}_{\Pi_{SIR}}$ operator, then the vertex is one day into recovery in the $j+1$'th iteration of the operator with the same certainty. The last rule says that if a vertex V' was infected with probability $P_{V'}$ and there is an edge from V' to V in the social network (weighted with probability $P_{V',V}$), and the vertex V' has recovered with certainty R', given the probability $1-R$ that V is not already recovered, (and hence, cannot be re-infected), then the certainty that the vertex V gets infected is $(1-R)\cdot P_{V',V}\cdot(P_{V'}-R')$. Here, $P_{V'}-R'$ is one way of measuring the certainty that V' has recovered (difference of the probability that it was infected and the probability it has recovered) and $P_{V',V}$ is the probability of infecting the neighbor.

To see how this GAP works, we execute a few iterations of the $\mathbf{T}_{\Pi_{SIR}}$ operator and show the fixpoint that it reaches on the small graph shown in Figure 8.5. In this graph, the initial infected vertices are those shown as a shaded circle. The transmission probabilities weight the edges in the graph.

The SNOP-query $(SUM, true, k, inf)$ can be used to compute the *expected number* of infected vertices in the least fixpoint of T_{Π}. This query says "find the k vertices in the social network which, if infected, would cause the maximal number of vertices to become infected in the future." However, the above set of rules can be easily used to express other things. For instance, an epidemiologist may not be



1	*inf(a):1, inf(c):1, inf(d):1*
2	*rec_1(a):1, rec_1(c):1, rec_1(d):1, inf(b):0.2, inf(d):0.3, inf(f):0.3, inf(g):0.05, inf(i):0.1*
3	*rec_2(a):1, rec_2(c):1, rec_2(d):1, rec_1(b):0.2, rec_1(d):0.3, rec_1(f):0.3, rec_1(g):0.05, rec_1(i):0.1 inf(g):0.08*
4	*rec_2(b):0.2, rec_2(d):0.3, rec_2(f):0.3, rec_2(g):0.05, rec_2(i):0.1, rec_1(g):0.08*
5	*rec_2(g):0.08*

Figure 8.5: Left: Sample network for disease spread. Right: annotated atoms entailed after each application of $\mathbf{T}_{\Pi_{SIR}}$ (maximum, non-zero annotations only).

satisfied with only one set of k vertices that can cause the disease to spread to the maximum extent - as there may be another, disjoint set of k vertices that could cause the same effect. The epidemiologist may want to find all members of the population, that if in a group of size k could spread the disease to a maximum extent. This can be answered using a *SNOP-ALL* query, described in Section 8.3.

The SIS Model of Disease Spread. The SIS (Susceptible-Infectious-Susceptible) model [67] is a variant of the SIR model. In SIS, an an individual becomes susceptible to disease after recovering (as opposed to SIR, where an individual acquires permanent immunity). SIS can be easily represented by a modification to the construction given above.

Diffusion Model 8.4.4 (SIS model)**.** *Take Diffusion Model 8.4.3 and change the third rule to*

$$\mathit{inf}(V) : P_{V',V} \cdot (P_{V'} - R') \;\;\leftarrow\;\; e(V',V) : P_{V',V} \wedge \mathit{inf}(V') : P_{V'} \wedge \mathit{rec}_{t_{rec}}(V') : R'.$$

Here, we do not consider the probability that vertex V is immune – hence this probability of recovery does not change the probability of becoming infected.

Diffusion in the Flickr Photo Sharing Network. The Flickr social network is designed for sharing of digital photographs. Users create a list of "favorite" photos that can be viewed by other users on the network. In [20], the authors studied how photographs spread to the favorite lists of different users using a variant of the SIS model. The key difference is that they do not consider a node "recovered" – i.e. once a photo was placed on a favorite list, it was relatively permanent (the study was conducted over about 100 days). They also found that photos lower on a favorite list (as the result of a user marking a large number of photos as "favorite") for a given user could still spread through the network. Hence, we present a GAP that captures the intuition of how Flickr photos spread according to [20].

Diffusion Model 8.4.5 (Flickr Photo Diffusion)**.**

$$\mathit{photo}_i(V) : \mathit{const}_i \cdot X_i \quad \leftarrow \quad \mathit{connected\_to}(V', V) : 1 \wedge \mathit{photo}_i(V') : X_i$$

In Diffusion Model 8.4.5, the annotation of the vertex atom $\mathit{photo}_i(V)$ is the likelihood that vertex V has marked photo i as one of its favorites. The predicate *connected_to* is the sole edge label representing that there is a connection from vertex V' to V (users select other users on this network). Additionally, the value const_i is a number in $[0, 1]$ that determines the likelihood that a given photo spreads in the network. As the edge weights do not change in this model, upon grounding, we can eliminate the annotated atom $\mathit{connected\_to}(V', V) : 1$ from the body (as for each vertex V, we would only need to ground out a diffusion rule for each incoming edge to V). Therefore, as $const_i \in [0,1]$, linearity follows. We note that for all of these models, the annotation functions reflect one interpretation of the likelihood that a vertex is infected or recovered – others are possible in our framework.

8.4.3 Homophilic Diffusion Models

Recently, [9] studied the spread of mobile application use on a global instant-messaging network of over 27 million vertices. They found that network-based diffusion could overestimate the spread of a mobile application and, for this scenario, over 50% of the adopted use of the applications was due to **homophily** - vertices with similar properties adopting similar applications.

This result should not be surprising – the basic idea behind web-search advertising is that two users with a similar property (search term) will be interested in the same advertised item. In fact, [20] explicitly pre-processed their Flickr data set with a heuristic to *eliminate* properties attained to vertices that could not be accounted for with a diffusion process. We can easily represent homophilic diffusion in a GAP with the following type of diffusion rule:

Diffusion Model 8.4.6 (Homophilic Diffusion of a Product)**.**

$$\textit{buys\_product}(V) : 0.5 \times X \quad \leftarrow \quad \textit{property}(V) : 1 \wedge \textit{exposed\_to\_product}(V) : X$$

In Diffusion Model 8.4.6, if a vertex is exposed to product (i.e. through mass advertising) and has a certain property, then the person associated with the vertex purchases the product with a likelihood of 0.5. For this rule, there are no network effects. Note that if the predicate *property* does not appear in any other rule heads, then the GAP is linear.

[177], the authors propose a "big seed" marketing approach that combines both homophilic and network effects. They outline a strategy of advertising to a large group of individuals who are likely to spread the advertisement further through network effects. We now describe a GAP that captures the ideas underlying big seed marketing. Suppose we have a set of attribute labels $\mathsf{AL} \subseteq \mathsf{VP}$. These attributes may be certain demographic characteristics - anything from education level to race to level of physical fitness. Suppose we want to advertise to k groups with one of these attributes to maximize an aggregate with respect to a goal predicate g. Consider the following construction.

Diffusion Model 8.4.7 (Big Seed Marketing)**.** *The GAP includes an embedding of the social network, as well as the network diffusion model of the user's choice, and the following additions.*

1. *Add vertex label attrib to VP.*

2. *For each attribute label lbl $\in \mathsf{AL}$, add vertex v_{lbl} to $\mathbf{V}$. Set $\ell_{vert}(v_{lbl}) = \{attrib\}$.*

3. *For each attribute label lbl $\in \mathsf{AL}$, let eff_{lbl} be a constant in $[0,1]$. This corresponds to the confidence that, if advertised to, a vertex v with label lbl obtains an annotation of 1 on $g(v)$.*

4. *This construction uses an a-priori $VC = attrib(V) : 1$.*

5. *Subtract k from the aggregate – this discounts the vertices created in part 2.*

6. *For each $lbl \in \mathsf{AL}$, add the following non-ground rule:*

$$g(V) : \mathit{eff}_{lbl} \times X \quad \leftarrow \quad lbl(V) : 1 \wedge g(v_{lbl}) : X$$

Note that in the above diffusion model, the v_{lbl} vertices correspond to advertisements directed toward different vertex properties. The VC condition forces the query to only return v_{lbl} vertices. The diffusion rule, added per label, ensures that the mass advertisement is received and that the vertex acts accordingly (hence the eff_{lbl} constants). Also, it is important to note that this construction is linear if no vertex atom formed with a predicate in AL appears in the head of a diffusion rule.

We close this section with a note that while all diffusion models mentioned here have been developed by others and have been shown above to be representable via GAPs, none of these papers has developed algorithms to solve SNOP-queries. We emphasize that not only do we give algorithms to answer SNOP-queries in the next section, our algorithms take any arbitrary diffusion model that can be expressed as a GAP, and an objective function as input. In addition, our notion of a social network is much more general than that of many of these extant approaches.

8.5 Algorithms

In this section we study how to solve SNOP problems algorithmically.

8.5.1 Naive Algorithm

The naive algorithm for solving the SNOPS query $(agg, VC, k, g(V))$ is to first find all pre-answers to the query. Then compute the value for each pre-answer and find the best. This is obviously an extremely expensive algorithm that is unlikely to terminate in a reasonable amount of time.

An execution strategy that first finds all vertices in a social network $\mathcal{S}$ that satisfy the vertex condition and then somehow restricts interest to those vertices in the above computation (where $\mathcal{S}$ is embedded in a GAP Π) would not be correct for two reasons. First, $lfp(\mathbf{T}_\Pi)$ assigns a truth value to each ground vertex atom A that might be different from what is initially assigned within the social network. Second, when we add a new ground vertex atom A to Π (e.g. in our cell phone example, when we consider the possibility of assigning a free calling plan to a vertex v), it might be the case that vertices that previously did not satisfy the vertex condition VC do so after the addition of A to Π.

8.5.2 A Non-Ground Algorithm in the Monotonic Case

There are three major problems with the Naive algorithm. The first problem is that the aggregate function is very general and has no properties that we can take advantage of. Hence, we can show that the entire search space might need to be explored if an arbitrary aggregate function is used. The second problem is that it works on the "ground" instantiation of Π. The third problem is that the $\mathbf{T}_\Pi$ operator maps all *ground* atoms to the $[0, 1]$ interval and there can be a very large number of ground atoms to consider. For instance, if we have a very small social network with just 1000 vertices and a rule with 3 variables in it, that rule has 10^9 possible ground instances - an enormous number. Likewise, if there is a ternary predicate symbol in the language of Π, then there are 10^9 ground atoms to consider. *All these problems are further aggravated by the fact that fixpoints might have to be computed several times.*

In this section, we provide an algorithm to compute answers to a SNOP-query *under the assumption* that our aggregate function is *monotonic* and under the assumption[8] that all rules in a GAP have the form $A : f(X_1, \ldots, X_n) \leftarrow B_1 : X_1, \wedge \cdots, B_n : X_n$.

In this case, we define a *non-ground* interpretation and a *non-ground* fixpoint operator $\mathbf{S}_\Pi$. This leverages existing work on non-ground logic programming initially pioneered by [114] and later adapted to different logic programming extensions by [59, 39, 165]. We start by defining a non-ground interpretation.

Definition 103. *A* non-ground interpretation *is a* partial *mapping $NG : \mathcal{A} \rightarrow [0, 1]$. Every non-ground interpretation NG* represents *an interpretation $grd(NG)$ defined as follows: $grd(NG)(A) = \mathsf{max}\{NG(A') \,|\, A \text{ is a ground instance of } A'\}$. When there is no atom A' which has A as a ground instance and for which $NG(A')$ is defined,*

[8]This latter assumption does not cause any loss of generality for all practical purposes if we also make the reasonable assumption that any constant annotations in a rule body can be translated into constraints. So if $B_i : 0.5$ occurs in the body of a rule, it can be replaced by $B_i : V_i \wedge V_i \geq 0.5$. [86] show that allowing such constraints involving annotated constraints can be easily accommodated by a simple extension to the semantics of GAPs.

then we set $grd(NG)(A) = 0$.

Thus, in a language with just three constants a, b, c and one predicate symbol p, the non-ground interpretation that maps $p(X, a)$ to 0.5 and everything else to 0 corresponds to the interpretation that assigns 0.5 to $p(a, a), p(b, a)$ and $p(c, a)$ and 0 to every other ground atom. Non-ground interpretations are succinct representations of ordinary interpretations - they only keep track of assignments to non-ground atoms (not necessarily all ground atoms) and they do not need to worry about atoms assigned 0. In the worst case, the number of non-ground atoms that NG keeps track of is no worse than a ground interpretation. We now define a fixpoint operator that maps non-ground interpretations to non-ground interpretations.

Definition 104 (operator $\mathbf{S}_\Pi$)**.** *The operator $\boldsymbol{S}_\Pi$ associated with a GAP Π maps a non-ground interpretation NG to the non-ground interpretation $\boldsymbol{S}_\Pi(NG)$ where $\boldsymbol{S}_\Pi(NG)(A) = \mathsf{max}\{f(X_1, \ldots, X_n) \mid A : f(\mu_1, \ldots, \mu_n) \leftarrow B_1 : \mu_1 \wedge \ldots \wedge B_n : \mu_n$ is a rule in Π such that for all $1 \leq i \leq n$, there exists an atom B'_i such that $(B_1, \ldots, B_n)$ and $(B'_1, \ldots, B'_n)$ are simultaneously unifiable via a most general unifier θ and (i) if μ_i is a constant, then $NG(B_i\theta) \geq \mu_i$, and (ii) if μ_i is a variable, then $NG(B_i\theta) = X_i\}$. (In this definition, without loss of generality, we assume the variables occurring in rules in Π are mutually standardized apart and are also different from those in NG).*

The fixpoint operator $\mathbf{S}_\Pi$ delays grounding to the maximal extent possible by (i) only looking at the rules in Π directly rather than ground instances of rules in Π (which is what $\mathbf{T}_\Pi$ does), and (ii) by trying to assign values to non-ground atoms rather than ground instances - unless there is no other way around it. The following example shows how $\mathbf{S}_\Pi$ works.

Example 8.5.1. *Some specific diffusion models focus on certain features in a graph that encourage the diffusion process. For example, [105] describes a diffusion process that is augmented by the presence of "funnels" in the graph. In this example, concerning disease spread, we take advantage of such features computationally by leveraging the operator $\boldsymbol{S}_\Pi$.*

Consider Example 8.3.3 from page 359 where we present a social network and some diffusion rules for disease spread embedded in program $\Pi_{disease}$.

Let us apply $\boldsymbol{S}_{\Pi_{disease}}$ till we reach a fixed point. With the first application, we entail annotated atoms $\{exp(v_4) : 1, exp(v_5) : 1, exp(v_6) : 1, exp(v_7) : 1, \}$. With the next application, $\{inf(v_4) : 1, inf(v_6) : 1\}$ are entailed. Then, with the next application, the non-ground annotated atom $exp(V) : 1$ is entailed. With the final application of the operator, the non-ground annotated atom $inf(V) : 1$ is entailed.

Consider the ordering $\preceq$ defined as follows on non-ground interpretations: $NG_1 \preceq NG_2$ iff $grd(NG_1) \leq grd(NG_2)$. In this case, it it easy to see that:

Proposition 72. *Suppose Π is any GAP. Then:*

1. *$\boldsymbol{S}_\Pi$ is monotonic.*

2. *$\boldsymbol{S}_\Pi$ has a least fixpoint $lfp(\boldsymbol{S}_\Pi)$ and $lfp(\mathbf{T}_\Pi) = grd(lfp(\boldsymbol{S}_\Pi))$. That is, $lfp(\boldsymbol{S}_\Pi)$ is a non-ground representation of the (ground) least fixpoint operator $\mathbf{T}_\Pi$.*

In short, $\mathbf{S}_\Pi$ is a version of $\mathbf{T}_\Pi$ that tries to work in a non-ground manner as much as possible. We now present the SNOP-Mon algorithm to compute answers to a

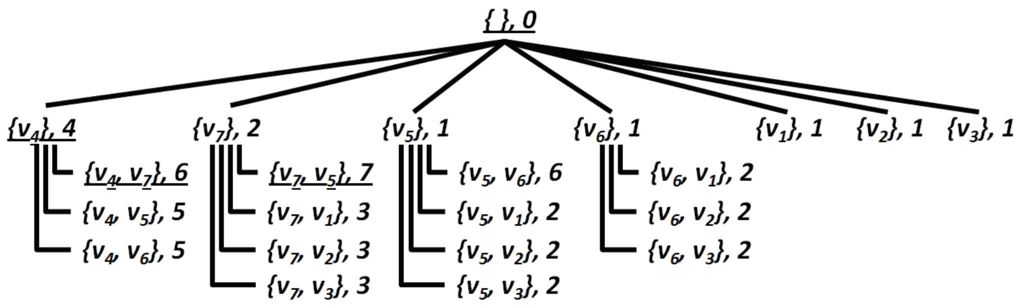


Figure 8.6: Search tree for Example 8.5.2.

SNOP-query $(agg, VC, k, g(V))$ when agg is monotonic. The SNOP-Mon algorithm uses the following notation: $value(NG)$ is the same as $value(grd(NG))$ and NG satisfies a formula iff $grd(NG)$ satisfies it.

The following example shows how the SNOP-Mon algorithm works.

Example 8.5.2. *Consider the program $\Pi_{disease}$ from Example 8.5.1. Suppose, we want to answer a SNOP query $(\Pi_{disease}, SUM, true, 2, inf(V))$. The search-tree in Figure 8.6 illustrates how SNOP-Mon searches for an optimal solution to the query. In the figure, we labeled each node with the set of vertices and a real number. The vertices correspond to the vertex atoms (annotated with 1) formed with inf added to GAP in step 4(c)i. The real number corresponds to the value resulting from this addition. Underlined nodes in the search tree represent potential solutions where $bestVal$ and $bestSOL$ are updated. Notice, that, for example, the set $\{v_4, v_1\}$ is never considered. This is because $inf(v_1)$ is entailed anytime a candidate solution includes v_4. The optimal solution is found to be $\{v_7, v_5\}$. In this example, the algorithm considers solutions in the following order: $\{\}, \{v_4\}, \{v_4, v_7\}, \{v_4, v_5\}, \{v_4, v_6\}, \{v_7\}, \{v_7, v_5\}, \{v_7, v_1\}$, $\{v_7, v_2\}, \{v_7, v_3\}, \{v_5\}, \{v_5, v_6\}, \{v_5, v_1\}, \{v_5, v_2\}, \{v_5, v_3\}, \{v_6\}, \{v_6, v_1\}, \{v_6, v_2\}$,*

```
SNOP-Mon($\Pi, agg, VC, k, g(V)$)

1. The variable $Curr$ is a tuple consisting of a GAP and natural number. We initialize
   $Curr.Prog = \Pi$; $Curr.Count = 0$.
2. $Todo$ is a set of tuples described in step 1. We initialize $Todo \equiv \{Curr\}$
3. Initialize the real number $bestVal = 0$ and GAP $bestSOL = NIL$
4. while $Todo \neq \emptyset$ do
   (a) $Cand$ = first member of $Todo$; $Todo = Todo - \{Cand\}$
   (b) if $value(lfp(\mathbf{S}_{Cand.Prog})) \geq bestVal \wedge lfp(\mathbf{S}_{Cand.Prog}) \models VC$ then
       i. $bestVal = value(lfp(\mathbf{S}_{Cand.Prog})$; $bestSOL = Cand$
   (c) if $Cand.Count < k$ then
       i. For each ground atom $g(V)\theta$, s.t. $\nexists OtherCand \in Todo$ where
          $OtherCand.Prog \supseteq Cand.Prog$,
          $|OtherCand.Prog| \leq |Cand.Prog| + 1$,
          and $lfp(\mathbf{S}_{OtherCand.Prog}) \models g(V)\theta : 1$, do the following:
          A. Create new tuple $NewCand$.
             Set $NewCand.Prog = Cand.Prog \cup \{g(V)\theta : 1 \leftarrow\}$.
             Set $New.Count = Cand.Count + 1$)
          B. Insert $NewCand$ into $Todo$
       ii. Sort the elements of $Element \in Todo$ in descending order of
           $value(Element.Prog)$, where the first element, $Top \in Todo$, has the
           greatest such value (i.e. there does not exist another element $Top'$ s.t.
           $value(Top'.Prog) > value(Top.Prog)$)
5. if $bestSOL \neq NIL$ then return ($bestSOL.Prog - \Pi$) else return NIL.
```

$\{v_6, v_3\}, \{v_1\}, \{v_2\}, \{v_3\}$.

The following result states that the SNOP-Mon algorithm is correct.

Theorem 54. *Given SNOP query $Q = (agg, VC, k, g(V))$ (w.r.t. SN $\mathcal{S}$ and GAP $\Pi \supseteq \Pi_{\mathcal{S}}$), if agg is monotonic then:*

- *There is an answer to the SNOP-query Q w.r.t. the GAP Π iff SNOP-Mon$(\Pi, agg, VC, k, g(V))$ does not return NIL.*
- *If SNOP-Mon$(\Pi, agg, VC, k, g(V))$ returns any result other than NIL, then that result is an answer to the SNOP-query Q w.r.t. the GAP Π.*

8.5.3 Approximation Algorithms: GREEDY-SNOP

Even though SNOP-Mon offers advantages such as pruning of the search tree and leverages non-ground operations to increase efficiency over the naive algorithm, it is still intractable in the worst case. Regretfully, Theorem 48 precludes an exact solution in PTIME and Theorem 50 precludes a PTIME α-approximation algorithm where $\alpha < \frac{e}{e-1}$. Both of these results hold for the restricted case of linear-GAPs and positive linear aggregate functions.

The good news is that we were able to show that (i) for linear-GAPs and positive-linear aggregates, the *value* function is *submodular* (Theorem 47). (ii) Under these conditions, we can reduce the problem to the maximization of a submodular function over a uniform matroid (the uniformity of the matroid is proved in Lemma 24 when VC is applied a-priori). (iii) We can leverage the work of [127] that admits a greedy $\frac{e}{e-1}$ approximation algorithm. In this section, we develop a greedy algorithm for SNOP-queries that leverages the above three results.

The GREEDY-SNOP algorithm shown below assumes a linear GAP, a positive-linear aggregate, and a zero-starting *value* function. The algorithm provides $\frac{e}{e-1}$ approximation to the SNOP-query problem. As this matches the upper bound of Theorem 50, we cannot do better in terms of an approximation guarantee.

```
GREEDY-SNOP($\Pi, agg, VC, k, g(V)$) returns $SOL \subseteq \mathbf{V}$

  1. Initialize $SOL = \emptyset$ and $REM = \{v \in \mathbf{V} | \left(g(v) : 1 \wedge \bigwedge_{pred \in \ell_{vert}(v)} pred(v) : 1\right) \models VC[V/v]\}$
  2. While $|SOL| < k$ and $REM \neq \emptyset$
     (a) $v_{best} = \mathsf{null}$, $val = value(SOL)$, $inc^{(alg)} = 0$
     (b) For each $v \in REM$, do the following
         i. Let $inc^{(alg)}_{new} = value(SOL \cup \{v\}) - val$
         ii. If $inc^{(alg)}_{new} \geq inc^{(alg)}$ then $inc^{(alg)} = inc^{(alg)}_{new}$ and $v_{best} = v$
     (c) $SOL = SOL \cup \{v_{best}\}$, $REM = REM - \{v_{best}\}$
  3. Return $SOL$
```

We now analyze the time complexity of GREEDY-SNOP.

Proposition 73. *Given SNOP query $Q = (agg, VC, k, g(V))$ (w.r.t. SN $\mathcal{S}$ and GAP $\Pi \supseteq \Pi_{\mathcal{S}}$), the complexity of GREEDY-SNOP is $O(k \cdot |\mathbf{V}| \cdot F(|\mathbf{V}|))$ where $F(|\mathbf{V}|)$ is the time complexity to compute value(V') for some set $V' \subseteq \mathbf{V}$ of size k.*

We note that most likely, the most expensive operation is the computation of *value* at line 2(b)i. One obvious way to address this issue is by using a non-ground version of the fixed-point. We address this issue later.

Theorem 55. *If SNOP query $Q = (agg, VC, k, g(V))$ (w.r.t. SN $\mathcal{S}$ and GAP $\Pi \supseteq \Pi_{\mathcal{S}}$) meets the following criteria:*

- Π *is a linear GAP*
- VC *is applied a-priori*
- *agg is positive linear*
- *value is zero-starting.*

Then GREEDY-SNOP *is an* $(\frac{e}{e-1})$*-approximation algorithm for the query.*

Example 8.5.3. *Consider Example 8.4.1 and program Π_{lin} from page 369. Consider the SNOP-query where $agg =$ SUM, $VC(V) = pro(V)$, $k = 2$, and $g(V) = buys\_camera(V)$. On the first iteration of* GREEDY-SNOP*, the algorithm computes the value for all vertices in the set $REMAINING$ which are $v_1, v_2, v_3, v_5, v_7, v_9, v_{10}$. The resulting annotations of the fixed points and aggregates are shown in Table 8.5.*

As $value(\emptyset) = 0$, the incremental increase afforded by v_2 is 4.25 – and clearly the greatest of all the vertices considered. GREEDY-SNOP *adds v_2 to SOL, removes it from REM and proceeds to the next iteration. Table 8.6 shows the incremental increases for the second iteration. As v_5 provides the greatest increase, it is picked, and the resulting solution is $\{v_2, v_5\}$.*

Vertex Atom	v_1	v_2	v_3	v_5	v_7	v_9	v_{10}
$buys\_camera(v_1)$	1.0	0.5	0.0	0.5	0.0	0.0	0.0
$buys\_camera(v_2)$	0.0	1.0	0.0	0.0	0.0	0.0	0.0
$buys\_camera(v_3)$	0.0	1.0	1.0	0.0	0.0	0.0	0.0
$buys\_camera(v_4)$	0.0	0.0	0.0	0.0	0.0	0.0	0.0
$buys\_camera(v_5)$	0.0	0.0	0.0	1.0	0.0	0.0	0.0
$buys\_camera(v_6)$	0.0	0.0	0.0	0.0	0.0	0.0	0.0
$buys\_camera(v_7)$	0.0	0.25	0.25	0.0	1.0	0.0	0.0
$buys\_camera(v_8)$	0.0	0.5	0.5	0.0	0.0	0.0	0.0
$buys\_camera(v_9)$	0.33	0.5	0.33	0.17	0.0	1.0	0.33
$buys\_camera(v_{10})$	0.0	0.5	0.5	0.0	0.0	0.0	1.0
SUM	1.33	4.25	2.58	1.67	1.0	1.0	1.33

Table 8.5: First iteration of the greedy algorithm.

Vertex	Incremental Increase on First Iteration	Incremental Increase on Second Iteration
v_1	1.33	0.67
v_2	4.25	NA
v_3	2.58	0.0
v_5	1.67	1.67
v_7	1.0	0.75
v_9	1.0	0.5
v_{10}	1.33	0.67

Table 8.6: Incremental Increases for Both Iterations of GREEDY-SNOP.

8.6 Scaling GREEDY-SNOP

This section is dedicated to providing improvements to GREEDY-SNOP in order to increase speed and/or enhance scalability. In this section, we will present an approach that does not necessarily select the same vertices as GREEDY-SNOP called GREEDY-SNOP2. We shall use the term "the greedy algorithm" to describe one of the two algorithms - noting when it makes a difference.

Most of the notation in this section will specify an iteration of the greedy algorithm (i.e. for GREEDY-SNOP, this would refer to an iterations of the outer loop at line 2). Our first piece of notation will be SOL_i which specifies the solution after i iterations of the greedy algorithm. So, for GREEDY-SNOP, $SOL_0 \equiv \emptyset$ and SOL_i refers to SOL after i executions of line 2c. Likewise, we shall define REM_i as the set of vertices (satisfying some a-priori VC) not picked at the end of iteration i.

The next piece of notation is for the GAP itself, Π. We define Π_i as $\Pi \cup \bigcup_{v \in SOL_i} \{g(v) : 1 \leftarrow\}$. This allows us to define $I_i^{(alg)} = lfp(\mathbf{S}_{\Pi_i})$ which is an interpretation that corresponds to the fixed point at each iteration.[9] We will also specify $I_i(v) = lfp(\mathbf{S}_{\Pi_{i-1} \cup \{g(v):1\leftarrow\}})$ which is an interpretation at each iteration *if* the greedy algorithm picks some vertex v written. We define $I_i(v)$ only for $i > 0$. We will also define a special mapping that tells us the *increase in annotation* if vertex v is selected by the greedy algorithm at iteration i. We will often treat this mapping as an interpretation and define it for each ground atom A. Formally,

[9]We can substitute the **S** operator for the **T** operator if we wished to, but throughout this section, we shall assume the use of the **S** operator as it would most likely yield an improvement in performance.

$INC_i(v)(A) = I_i(v)(A) - I_{i-1}^{(alg)}(A)$. Unless specified otherwise, we will only be concerned about ground atoms formed with the goal predicate ($g(V)$). Hence, we can most likely reduce storage requirements for $I_i(v)$ and $INC_i(v)$ in practice. For linear GAPs, we have the following proposition concerning the increase in annotation.

Proposition 74. *For all ground atoms A and vertices v, $INC_{i-1}(v)(A) \geq INC_i(v)(A)$.*

Now we will show that by saving $I_i(v)$ at each iteration, we can potentially increase the speed at which subsequent fixed points are calculated. First, we consider the GAP formed from some GAP Π at its least fixed point.

Definition 105. $PROG(\Pi) = \Pi \cup \{A : \mu | A : \mu$ *is a non-ground annotated atom in* $lfp(\boldsymbol{S}_\Pi)\}$

From this, we have the following two lemmas.

Lemma 25. *For all programs Π and any atom A, $lfp(S_{PROG(\Pi)})(A) = lfp(S_\Pi)(A)$*

Lemma 26. *If $\Pi_3 \equiv \Pi_1 \cup \Pi_2$, then for any atom A,*

$$lfp(\boldsymbol{S}_{\Pi_3})(A) = lfp(\boldsymbol{S}_{PROG(\Pi_1) \cup PROG(\Pi_2)})(A)$$

This leads us to the following proposition.

Proposition 75. *If $\Pi_3 \equiv \Pi_1 \cup \Pi_2$, then for any atom A,*

$$lfp(\boldsymbol{S}_{\Pi_3})(A) = lfp(\boldsymbol{S}_{PROG(PROG(\Pi_1) \cup PROG(\Pi_2))})(A)$$

So, suppose GREEDY-SNOP is on iteration $i-1$ and considers some vertex v which it does not select. As it calculated the fixed-point, we can save $I_{i-1}(v)$ and easily create $PROG(\Pi_{i-2} \cup \{g(v) : 1 \leftarrow\})$ using this information. At the end of iteration $i-1$ we can also have $PROG(\Pi_{i-1})$ easily stored as well. Now suppose vertex v is being considered again on iteration i. Rather than computing the fixed point of $\mathbf{S}_{\Pi_{i-1}\cup\{g(v):1\leftarrow\}}$ in the straight-forward manner, we can use Proposition 75 and compute the least fixed point of $S_{PROG(\Pi_{i-1})\cup PROG(\Pi_{i-2}\cup\{g(v):1\leftarrow\})}$, which will likely converge faster. We will use the notation $PROG_i(v)$ to refer to the program $PROG(\Pi_{i-1}\cup\{g(v):1\leftarrow\})$.

Example 8.6.1. *Consider Example 8.5.3. Consider what happens when GREEDY-SNOP computes value when considering vertex v_3 on the second iteration. A quick look at Table 8.5 reveals that, as vertex atom buys_camera(v_3) is annotated with* 1 *after the first iteration when v_2 was considered. This means that the annotations assigned when v_3 is added after v_2 will remain the same (hence, there was no incremental increase when v_2 was added in the second iteration). By calculating the fixed point using $S_{PROG(\Pi_1)\cup PROG(\Pi_0\cup\{g(v_3):1\leftarrow\})}$ will cause $\boldsymbol{S}$ to converge after a single iteration in this case – as the maximum annotations are already assigned by rules in program $PROG(\Pi_1)$.*

With the $I_i^{(alg)}$ defined, we can specify *value* at each iteration:

$$val_i = agg(\{I_i^{(alg)}(g(V))|V\in\mathbf{V}\})$$

Next, we specify a notation to refer to how much the *value* has increased after i iterations of the greedy algorithm – the *incremental increase* – or $inc_i^{(alg)}$ (defined for $i>0$). Formally, $inc_i^{(alg)} = val_i - val_{i-1}$. Note that we use the superscript (alg) to signify that this corresponds with the incremental increase based on the vertex (or vertices) selected by the greedy algorithm at iteration i. The *optimal incremental* *increase* – $inc_i^{(opt)}$ – refers to the incremental increase if the greedy algorithm selects a single vertex that causes the greatest possible incremental increase to $value$. Note that GREEDY-SNOP always picks a vertex at each iteration such that $inc_i^{(alg)} = inc_i^{(opt)}$. Consider the following proposition.

Proposition 76. $inc_i^{(opt)} \leq inc_{i-1}^{(opt)}$.

We will now define the incremental increase *if* the algorithm selects a specific vertex v at iteration i - written $inc_i(v)$. So, if the greedy algorithm selects vertex v, then $inc_i^{(alg)} = inc_i(v)$. Formally, $inc_i(v) = value(SOL_{i-1} \cup \{v\}) - val_{i-1}$. As with $I_i(v)$, $inc_i(v)$ is only defined for $i > 0$. Also based on the definition of submodularity, we have the following corollary to Proposition 76.

Corollary 14. $inc_i(v) \leq inc_{i-1}(v)$.

Corollary 14 allows for a possible speed-up. For example, consider GREEDY-SNOP on some iteration i. Suppose, while considering vertex v_1, it computes $inc_i^{(opt)}(v_1)$. Now, it proceeds to the next vertex, v_2. If, on some previous iteration $j \leq i$, we saved $inc_j^{(opt)}(v_2)$ and this value is less than or equal to $inc_i^{(opt)}(v_1)$, then we need not consider v_2 as the incremental increase it can provide cannot possibly be greater than v_1. Such a technique is also leveraged in [101] on a different problem that reduces to the maximization of a submodular function over a uniform matroid. In that work, this type of improvement led to an increase in speed by a factor of 700.

However, the storage of the last incremental increase for a given vertex may still need to be re-calculated after several iterations. One way to avoid this is to obtain an upper bound on $inc_i(v)$. We can do this with the special interpretation $INC_i(v)$ that we already defined. Consider the following observation for positive-linear aggregates.

Fact 1. $inc_i(v) = agg(\{INC_i(v)(g(v'))|v' \in \mathbf{V}\})$

Based on Observation 1 and Proposition 74, we can use the following result to obtain an upper bound for $inc_i(v)$.

Proposition 77. *For $j \leq i$, we have:*

$$inc_i(v) \leq agg\left(\{\min\left(1, INC_j(v)(g(v')) + I_{i-1}^{(alg)}(g(v'))\right) - I_{i-1}^{(alg)}((g(v')))|v' \in \mathbf{V}\}\right)$$

We shall refer to the quantity

$$agg\left(\{\min\left(1, INC_j(v)(g(v')) + I_{i-1}^{(alg)}(g(v'))\right) - I_{i-1}^{(alg)}((g(v')))|v' \in \mathbf{V}\}\right)$$

where the j is the annotation increase for vertex v was stored, as $inc_i^{(up)}(v)$. Consider the following example.

Example 8.6.2. *Consider Example 8.5.3. Suppose, at the start of the second iteration, the algorithm computes an $inc_2^{(up)}(v)$ for all $v \in REM_1$. It would simply use the fixed point computations of the first iteration to create $INC_1(v)$ for each (see Table 8.5). In Table 8.7, we show values assigned by $I_1^{(alg)}$ (interpretation after the first iteration) and $INC_1(v_5)$ (incremental increase for each vertex atom from v_5).*

Using the information in Table 8.7, we can easily compute $inc_2^{(up)}(v_5)$ to be 1.67*. In this case, this is a very tight upper bound, matching the actual incremental increase as depicted in Table 8.6.*

Vertex Atom	$I_1^{(alg)}$	$INC_1(v_5)$
$buys\_camera(v_1)$	0.5	0.5
$buys\_camera(v_2)$	1.0	0.0
$buys\_camera(v_3)$	1.0	0.0
$buys\_camera(v_4)$	0.0	0.0
$buys\_camera(v_5)$	0.0	1.0
$buys\_camera(v_6)$	0.0	0.0
$buys\_camera(v_7)$	0.25	0.0
$buys\_camera(v_8)$	0.5	0.0
$buys\_camera(v_9)$	0.5	0.17
$buys\_camera(v_{10})$	0.5	0.0

Table 8.7: Calculating $inc_2^{(up)}(v_5)$.

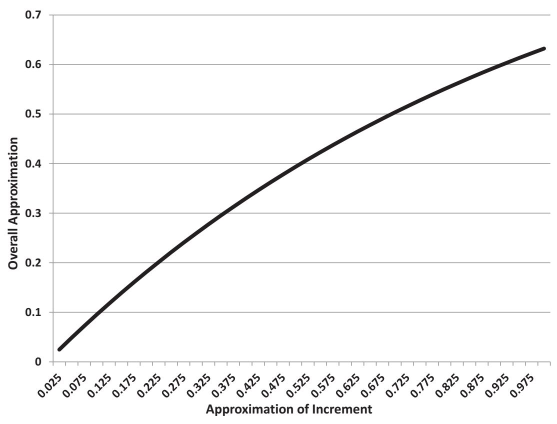


Figure 8.7: Effect on overall approximation given an incremental approximation factor.

We can also use upper bounds on $inc_i(v)$ to obtain an upper bound on $inc_i^{(opt)}$ for a given iteration. We present the following observation.

Fact 2. $inc_i^{(opt)} \leq \min\left(inc_{i-1}^{(opt)}, \max_{v \in REM_{i-1}}(inc_i^{(up)}(v))\right)$

We shall refer to the quantity $\min\left(inc_{i-1}^{(opt)}, \max_{v \in REM_{i-1}}(inc_i^{(up)}(v))\right)$ as $inc^{(opt,up)}$. We can use this information to select vertices that cause an incremental increase within ϵ of optimal. Consider the following result of [60] (Theorem 1).

Theorem 56. *Consider the greedy algorithm of [127]. If at each step of the greedy algorithm, the incremental improvement is approximated within a factor of ϵ, then the greedy algorithm is an $\frac{e^\epsilon}{e^\epsilon - 1}$ approximation algorithm (i.e. obtains a solution within $\frac{e^\epsilon - 1}{e^\epsilon}$ of optimal).*

We plot the relationship between the approximation of the incremental improvement vs. overall approximation in Figure 8.7.

So, suppose the user specifies an additional parameter ϵ in the input of the greedy algorithm that corresponds to the ϵ in Theorem 56. One way to leverage this approximation is to compute the aggregate after each application of the $\mathbf{S}$ operator and halt computation once the aggregate is within $\epsilon \cdot inc^{(opt,up)}$. We introduce new notation for each vertex v that takes this partial fixed point computation into account – $PROG_i^{(\epsilon)}(v), inc_i^{(\epsilon)}(v)$, and $INC_i^{(\epsilon)}(v)$, which correspond with the previously described $PROG_i(v), inc_i(v)$, and $INC_i(v)$ respectively. The algorithm APPROX-VALUE computes these items for the current iteration.

Theorem 56 can be leveraged in another way that allows for the selection of multiple vertices in a single iteration of the greedy algorithm. First, we define the notion of *vertex spread* which intuitively refers to all other vertices that increase their annotation when vertex v is added at iteration i. For vertex v at iteration i, with parameter ϵ, we define $spread_i^{(\epsilon)}(v) = \{v' \in \mathbf{V} | INC_i^{(\epsilon)}(v)(g(v')) > 0\}$. Using this information, for a set of vertices, $\mathbf{V}'$, we can now specify a *spread-graph*.

Definition 106 (Spread Graph)**.** *For a given iteration, i, set of vertices $\mathbf{V}'$, and parameter ϵ, we define the* ***spread-graph*** $GS_i^{(\epsilon)}(\mathbf{V}') = (V_{spread}, E_{spread})$ *as a graph where:*

1. *For each $v_p \in \mathbf{V}'$, there is a corresponding node $v'_p \in V_{spread}$ and no other nodes in V_{spread}.*

2. *There is an undirected edge $(v'_p, v'_q) \in E_{spread}$ iff for corresponding vertices $v_p, v_q \in \mathbf{V}'$, $spread_i^{(\epsilon)}(v_p) \cap spread_i^{(\epsilon)}(v_q) \not\equiv \emptyset$*

Returning to the notion of selecting vertices that where $inc_i(v)$ or $inc_i^{(\epsilon)}(v)$

```
APPROX-VALUE($v, PROG_j^{(\epsilon)}(v), PROG(\Pi_{i-1}), agg, inc^{(opt,up)}, val_{i-1}, I_{i-1}^{(alg)}, \epsilon$) ($j < i$)
returns real number $inc_i^{(\epsilon)}(v)$, function $INC_i^{(\epsilon)}(v)$, program $PROG_i^{(\epsilon)}(v)$, and Boolean $flag$.

  1. $inc_i^{(\epsilon)}(v) = 0$, $INC_i^{(\epsilon)}(v)$ and $I_{temp}$ assign all atoms 0, $\Pi_{temp} = PROG_j^{(\epsilon)}(v) \cup \Pi_{i-1}$,
     $flag =$ false.
  2. While $inc_i^{(\epsilon)}(v) < \epsilon \cdot inc^{(opt,up)}$ and $\neg flag$
     (a) $I_{prev} = I_{temp}$
     (b) Let $I_{temp}$ be $\mathbf{S}_{\Pi_{temp}}$ applied to $I_{temp}$.
     (c) $flag = (I_{prev} == I_{temp})$
     (d) $inc_i^{(\epsilon)}(v) = agg(\{I_{temp}(g(V)) | V \in \mathbf{V}\}) - val_{i-1}$
  3. For all $A \in \mathcal{A}$, set $INC_i^{(\epsilon)}(v)(A) = \max(0, I_{temp}(A) - I_{i-1}^{(alg)}(A))$
  4. Set $PROG_i^{(\epsilon)}(v) = PROG(\Pi_{i-1}) \cup \{A : I_{temp}(A) \leftarrow | A \in \mathcal{A}\}$
  5. Return $inc_i^{(\epsilon)}(v)$, $INC_i^{(\epsilon)}(v)$, $PROG_i^{(\epsilon)}(v)$, $flag$.
```

are greater than or equal to $\epsilon \cdot inc^{(opt,up)}$, let us define a set $cand^{(\epsilon)}{}_i = \{v \in REM_{i-1} | inc_i^{(\epsilon)}(v) \geq \epsilon \cdot inc^{(opt,up)}\}$. Let $cand^{(\epsilon)}{}_i{}'$ be a subset of $cand^{(\epsilon)}{}_i$. We have the following theorem.

Theorem 57. *If the nodes in $GS_i^{(\epsilon)}(cand^{(\epsilon)}{}_i)$ corresponding with elements of $cand^{(\epsilon)}{}_i{}'$ are an independent set of $GS_i^{(\epsilon)}(cand^{(\epsilon)}{}_i)$, then the greedy algorithm can select all vertices in $cand^{(\epsilon)}{}_i{}'$ and still obtain a solution within $\frac{e^\epsilon - 1}{e^\epsilon}$ of optimal.*

So, Theorem 57 allows the greedy algorithm to select more than one vertex during a given iteration. Further, as the value of ϵ increases, the cardinality of an independent set of $GS_i^{(\epsilon)}(cand^{(\epsilon)}{}_i)$ should also increase, meaning that the user can use ϵ as a way to trade accuracy for performance.

Although the problem of finding a maximal independent set is NP-hard, several polynomial approximation algorithms have been studied [69]. Where n is the number of vertices, a simple greedy approach illustrated in [69] runs in $O(n^2)$ time and ensures finding an independent set of at least $\frac{n}{\delta+1}$ where δ is the average degree of the graph. Note that for our application $n = |cand^{(\epsilon)}{}_i|$, and we expect $|cand^{(\epsilon)}{}_i| << |\mathbf{V}|$. We present this algorithm, GREEDY-INDEP-SET, below.

There are several modifications that can be made to GREEDY-INDEP-SET. For example, we can leverage the Fibonacci heap of [49] to obtain a $O(n \lg n)$ run time. Another easy modification to GREEDY-INDEP-SET that may provide better approximations in practice would be to select vertices that not only have a low degree, but also where the incremental increase is greater. In Example 8.6.3, we describe such a heuristic. Additionally, in [69], the authors also present a more advanced approx-

```
GREEDY-INDEP-SET($G = (V, E)$) returns $V' \subset V$
  1. $V' = \emptyset$
  2. While $V \not\equiv \emptyset$ do the following.
      (a) Let $v$ be the vertex in $V$ with the smallest degree. Add $v$ to set $V'$. Remove
          $v$ and all its neighbors (and adjacent edges) from $G$
  3. Return $V'$
```

imation algorithm that provides an indecent set within $\frac{2}{\delta+1}$ of optimal. However, it is important to note that we need not solve this problem exactly, and we do not want this to become a dominating operation in the overall algorithm.

So far, we have illustrated a variety of ways to scale GREEDY-SNOP and still provide an approximation guarantee. We combine the techniques we have described thus far in GREEDY-SNOP2, illustrated in the following example.

Example 8.6.3. *Consider Example 8.4.1 and program Π_{lin} from page 369. Consider the SNOP-query where agg = SUM, $VC(V) = pro(V)$, $k = 3$, and $g(V) = buys\_camera(V)$ along with the parameter $\epsilon = 0.4$. On the first iteration of GREEDY-SNOP2, the algorithm computes the value for all vertices in the set $REMAINING$ which are $v_1, v_2, v_3, v_5, v_7, v_9, v_{10}$. Note that due to step 3 of GREEDY-SNOP2, the algorithm computes the complete fixed point, just as it did in Example 8.5.3 when we used GREEDY-SNOP. Refer to Table 8.5 on page 389 for the resulting interpretations. As $inc^{(opt,up)}$ is 4.25 for the first iteration, the set $cand^{(\epsilon)}$ for this iteration includes all vertices where the incremental increase is greater than or equal to $0.4 \cdot 4.25 = 1.7$.*

```
GREEDY-SNOP2($\Pi, agg, VC, k, g(V), \epsilon$) returns $SOL \subset \mathbf{V}$

1. Initialize $SOL = \emptyset$ and $REM = \{v \in \mathbf{V} | g(v) : 1 \wedge \bigwedge_{pred \in \ell_{vert}(v)} pred(v) : 1 \models VC[V/v]\}$
2. Compute $PROG^{(\epsilon)}(\Pi_0)$ and $I_0^{(alg)}$.
3. For each $v \in REM$, compute $PROG_0^{(\epsilon)}(v)$, and $INC_1(v)$.
4. While $|SOL| \leq k$ and $REM \neq \emptyset$
    (a) Set $v_{best} = \mathsf{null}$, $cand^{(\epsilon)} = \emptyset$, $inc^{(alg)} = 0$, $val = value(SOL)$
    (b) For each $v \in REM$, calculate $inc^{(up)}(v)$ as per Proposition 77 using the last saved $INC_j(v)$ that was calculated
    (c) Calculate $inc^{(opt,up)}$ as per Observation 2.
    (d) Sort the elements of $REM$ from greatest to least by $inc^{(up)}(v)$
    (e) For each $v \in REM$ where $inc^{(up)}(v) > \min(inc_{best}^{(alg)}, \epsilon \cdot inc^{(opt,up)})$, do the following
        i. $\langle inc^{(\epsilon)}(v), INC^{(\epsilon)}(v), PROG^{(\epsilon)}(v), flag \rangle =$
           APPROX-VALUE($v, PROG^{(\epsilon)}(v), PROG(\Pi), agg, inc^{(opt,up)}, val, I^{(alg)}, \epsilon$)
        ii. If $inc^{(\epsilon)}(v) \geq inc^{(alg)}$ then $inc^{(alg)} = inc^{(\epsilon)}(v)$ and $v_{best} = v$
        iii. If $\neg flag$ then add $v$ to $cand^{(\epsilon)}$
        iv. If $flag$ then set $INC(v) = INC^{(\epsilon)}(v)$
    (f) If $cand^{(\epsilon)} \equiv \emptyset$ then $SOL = SOL \cup \{v_{best}\}$, $REM = REM - \{v_{best}\}$
    (g) Else do the following:
        i. Create $GS^{(\epsilon)}(cand^{(\epsilon)})$
        ii. Let $cand^{(\epsilon)'}$ be the subset of $REM$ corresponding with the nodes of an independent set in $GS^{(\epsilon)}(cand^{(\epsilon)})$
        iii. $SOL = SOL \cup cand^{(\epsilon)'}$, $REM = REM - cand^{(\epsilon)'}$
5. Return $SOL$
```

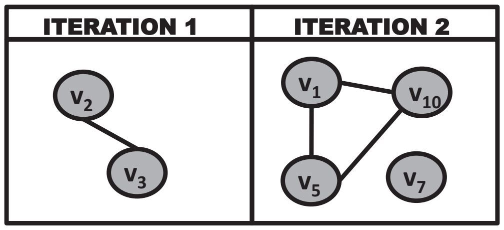


Figure 8.8: Left: spread graph after iteration 1. Right: spread graph after iteration 2.

Hence, cand$^{(\epsilon)} = \{v_2, v_3\}$*. In Figure 8.8, we show the spread graph created with this set. As the singletons* v_2, v_3 *are both independent sets, the algorithm can pick either. Although we do not specify a "tie-breaker" in* GREEDY-SNOP2*, a reasonable heuristic would be to select the vertex with the greatest incremental increase, which would be* v_2*, so we add* v_2 *to* SOL*.*

In the second iteration, we calculate the upper bound using the special interpretation $INC_i(v)$ *for each vertex* v*. In Example 8.6.2 on page 395, we show how to do this for vertex* v_5 *and this is found to be* 1.67*. This also happens to be the greatest upper bound for any vertex in REM. Therefore, after computing the fixed points, we select all vertices whose incremental increase is greater than or equal to* $0.4 \cdot 1.67 = 0.67$*. So, for this iteration cand*$^{(\epsilon)} = \{v_1, v_5, v_7, v_{10}\}$*. The spread graph is also shown in Figure 8.8. Using the heuristic we described for the first iteration, the algorithm would select* $\{v_5, v_7\}$ *and add them to* SOL*. Hence the algorithm returns* $\{v_2, v_5, v_7\}$*. Note that the algorithm was able to totally avoid a third iteration, even though it was required to (and does) return a set of three vertices.*

Proposition 78. *The complexity of* GREEDY-SNOP2 *is* $O(k \cdot |\mathbf{V}| \cdot F(|\mathbf{V}|))$ *where*

$F(|\mathbf{V}|)$ is the time complexity to compute $value(V')$ for some set $V' \subseteq \mathbf{V}$ of size k.

Proposition 79. *Given a SNOP-query meeting the following criteria:*

- *Π is a linear GAP*
- *VC is applied a-priori*
- *agg is positive linear*
- *value is zero-starting*

Then GREEDY-SNOP2 *is an $\frac{e^\epsilon}{e^\epsilon - 1}$-approximation algorithm for the query.*

Now we will show a way to possibly further improve scalability while preserving the approximation guarantee of Proposition 79. Our intuition is to use a spread-graph on all vertices in REM_0 to partition the problem, and then run the greedy algorithm on each sub-problem. Consider the following definition:

Definition 107 (Disjoint Node Set)**.** *Given an un-directed, un-weighted graph $G = (V, E)$, we say sets $V_1, V_2 \subseteq V$ are* ***disjoint node sets*** *iff*

1. *There is no edge from any node in V_1 to a node in V_2*
2. *(or equivalently) Any set of two nodes where one is picked from V_1 and one is picked from V_2 is an independent set of G*

We provide a simple algorithm for finding disjoint node sets in Appendix G.2.15 (page 609). Now we present GREEDY-SNOP-DIV that uses disjoint node set to partition the problem and still maintain the approximation guarantee of Proposition 79.

```
GREEDY-SNOP-DIV($\Pi, agg, VC, k, g(V), \epsilon$) returns $SOL \subset \mathbf{V}$

    1. Initialize $SOL = \emptyset$ and $REM_0 = \{v \in \mathbf{V} | g(v) : 1 \wedge \bigwedge_{pred \in \ell_{vert}(v)} pred(v) : 1 \models VC[V/v]\}$
    2. For each $v \in REM_0$, calculate set $spread_1^{(\epsilon)}(v)$
    3. Create graph $GS_1^{(\epsilon)}(REM_0) = (V_{spread}, E_{spread})$.
    4. Let $DNS_1, \ldots, DNS_n$ be the disjoint node sets of $V_{spread}$
    5. Create predicates $set_1, \ldots, set_n$. For all $v \in \mathbf{V}$ set the weight $set_i(v)$ to 1 iff the corresponding node in $V_{spread}$ is in $DNS_i$ and 0 otherwise.
    6. Create $n$ new SNOP queries where for query $i$, the input is $\Pi, agg, VC(V) \wedge set_i(V), \min(k, |DNS_i|), g(V), \epsilon$.
    7. Let $SOL^{(1)}, \ldots, SOL^{(n)}$ be the solutions to each SNOP query as returned by GREEDY-SNOP or GREEDY-SNOP2. Let $SOL_{all}$ be the union of all these sets. With each vertex $v \in SOL_{all}$, let $inc(v)$ be the incremental increase caused by that vertex in its SNOP-query.
    8. Sort $SOL_{all}$ by $inc(v)$ from greatest to least
    9. Return the top $k$ elements of $SOL_{all}$.
```

In the below example, we use the disjoint node sets of $GS_0^{(\epsilon)}(REM_0)$ to partition the problem.

Example 8.6.4. *Consider Example 8.3.2 on page 355. Recall, that in this problem,* 20 *vertices,* $v_1, \ldots, v_{20}$ *meet an a-priori* VC *and thus comprise the set* REM_0 *(see Figure 8.2). Suppose, for each* $v_i \in REM_0$*, we find the set* $spread_1^{(\epsilon)}(v_i)$ *and the results are shown by the shaded ovals in Figure 8.6.4.*

Using Figure 8.9(top), we can easily see the intersection of two sets of vertices corresponding with vertex spreads. For example, there are 4 *vertices in the set* $spread_1^{(\epsilon)}(v_{18}) \cap spread_1^{(\epsilon)}(v_{20})$*, while there are* 8 *vertices in* $spread_1^{(\epsilon)}(v_1) \cap spread_1^{(\epsilon)}(v_2)$*. Based on these intersections, we can obtain the spread graph* $GS_1^{(\epsilon)}(REM_0)$ *– shown in Figure 8.9(bottom).*

Based on the spread graph of Figure 8.9(bottom), there are 5 *disjoint node sets - this is how* GREEDY-SNOP-DIV *will partition the problem:*

$$\begin{aligned}
DNS_1 &= \{ \quad v_1, v_2, v_3, v_4, v_5 \quad \} \\
DNS_2 &= \{ \quad v_6, v_7, v_8, v_9, v_{10} \quad \} \\
DNS_3 &= \{ \quad v_{11}, v_{12}, v_{13}, v_{14}, v_{15} \quad \} \\
DNS_4 &= \{ \quad v_{16}, v_{17} \quad \} \\
DNS_5 &= \{ \quad v_{18}, v_{19}, v_{20} \quad \}
\end{aligned}$$

So, GREEDY-SNOP-DIV *would then create predicates* $set_1, set_2, set_3, set_4, set_5$ *for each of the disjoint node sets above. The vertex atoms formed with these predicates are assigned* 1 *iff the vertex is in the corresponding disjoint node set. For*

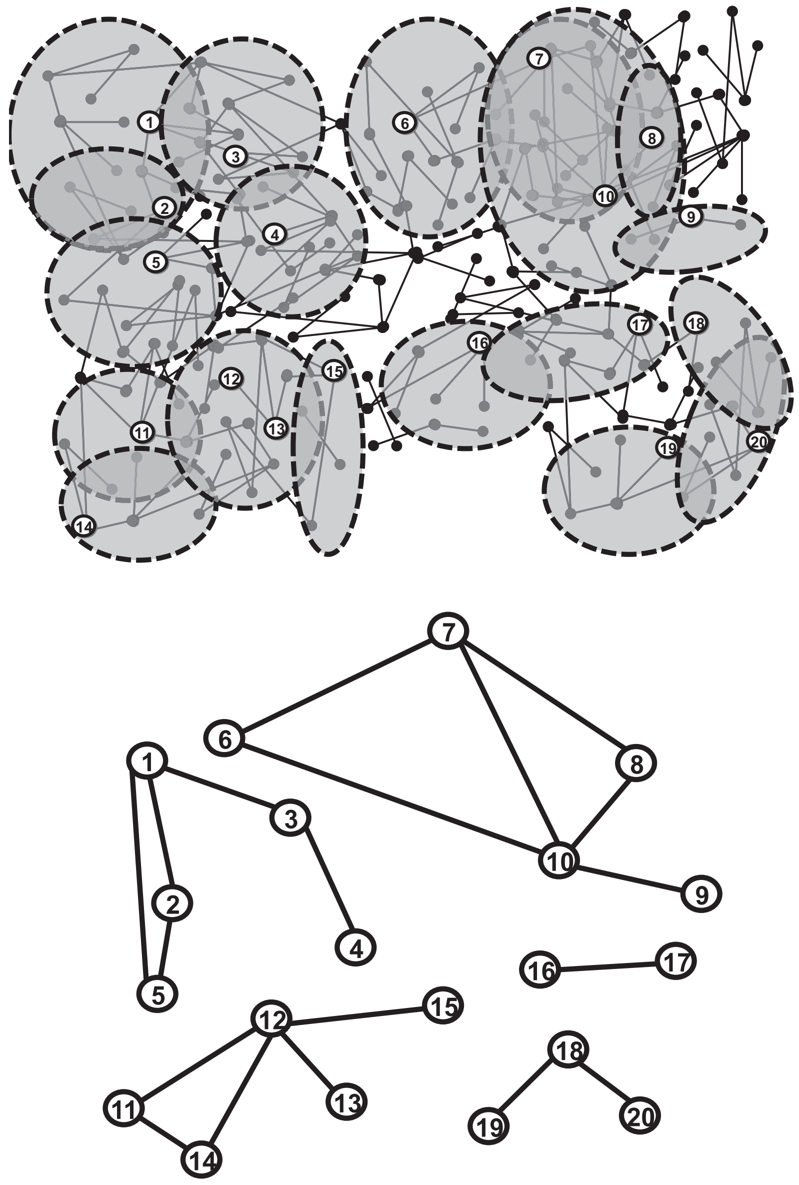


Figure 8.9: Top: Social Network for the painting company with vertex spread shown as shaded ovals. Bottom: Spread graph $GS_1^{(\epsilon)}(REM_0)$ for the painting company example.

example, $set_1(v_1)$ *and* $set_4(v_{16})$ *are annotated with* 1 *while* $set_4(v_1)$ *and* $set_1(v_{16})$ *are annotated with* 0*.*

Recall that the original VC *was* $prior(V)$ *(see Example 8.3.2). We now create* 5 *new SNOP queries with the following a-priori vertex conditions (i.e.* VC_i *is the vertex condition for query* i*).*

$$\begin{aligned}
\text{Query 1:}\quad VC_1 &= prior(V):1 \wedge set_1(V):1 \\
\text{Query 2:}\quad VC_2 &= prior(V):1 \wedge set_2(V):1 \\
\text{Query 3:}\quad VC_3 &= prior(V):1 \wedge set_3(V):1 \\
\text{Query 4:}\quad VC_4 &= prior(V):1 \wedge set_4(V):1 \\
\text{Query 5:}\quad VC_5 &= prior(V):1 \wedge set_5(V):1
\end{aligned}$$

The algorithm would then take the results of the 5 *queries obtained from runs of* GREEDY-SNOP2 *and order the union of all solutions by the incremental increase.* GREEDY-SNOP-DIV *would then return the top* k *vertices.*

Proposition 80. *Given a SNOP-query meeting the following criteria:*

- Π *is a linear GAP*
- VC *is applied a-priori*
- agg *is positive linear*
- $value$ *is zero-starting*

Then GREEDY-SNOP-DIV *is an* $\frac{e^{\epsilon}}{e^{\epsilon}-1}$*-approximation algorithm for the query.*

We notice that GREEDY-SNOP-DIV allows us to partiiton the problem in way where for each of the n disjoint node sets can be handled by an instance of GREEDY-SNOP2 on a different machine. Further, we can maintain a “master process” where each of the n instances of GREEDY-SNOP2 can report their latest vertices added to the solution and the corresponding incremental increase. This can allow the master process to terminate the instances of GREEDY-SNOP2 early (i.e. once the incremental increase of a vertex picked by an instance of GREEDY-SNOP2 is to low to be added to the final solution).

8.7 Implementation and Experiments

We have implemented the GREEDY-SNOP algorithm in 660 lines of Java code by re-using and extending the diffusion modeling Java library of [17] (approx 35K lines of code). Our implementation uses multiple threads in the inner loop of the GREEDY-SNOP algorithm to increase efficiency. All experiments were executed on the same machine with a dedicated 4-core 2.4GHz processor and 22GB of main memory. Times were measured to millisecond precision and are reported in seconds.

8.7.1 Experimental Setting

Data set. In order to evaluate GREEDY-SNOP, we used a real world dataset based on a social network of Wikipedia administrators and authors. Wikipedia is an online encyclopedia collaboratively edited by many contributors from all over the world. Selected contributors are given privileged administrative access rights to help maintain and control the collection of articles with additional technical features. A vote by existing administrators and ordinary authors determines whether an individual is granted administrative privileges. These votes are publicly recorded. [100] crawled 2794 elections from the inception of Wikipedia until January 2008. The votes casted in these elections give rise to a social network among Wikipedia administrators and authors by representing a vote of user i for user j as a directed edge from node i to j. In total, the dataset contains $103,663$ votes (edges) connecting more than 7000 Wikipedia users (vertices). Hence, the network is large and densely connected.

SNOP-Query. In our experiments, we consider the hypothetical problem of finding the most influential administrators in the Wikipedia social network described above. We treat votes as a proxy for the inverse of influence. In other words, if user i voted for user j, we assume user j (intentionally through lobbying or unintentionally through the force of his contributions to Wikipedia) influenced user i to vote for him. All edges are assigned a weight of 1. Our SNOP-queries are designed as per the following definition.

Definition 108 (Wikipedia SNOP-Query)**.** *Given some natural number $k > 1$, a Wikipedia SNOP query, $WQ(k)$ is specified as follows:*

- $agg = \mathsf{SUM}$ – *the intuition is that the aggregate provides us an expected number of vertices that are influenced.*
- $VC = \mathsf{true}$ – *we do not use a vertex condition in our experiments*
- k *as specified by the input*

- $goal(V) = influenced(V)$

Diffusion Models Used. We represented the diffusion process with two different models: one tipping and one cascading.

- **Cascading diffusion model.** We used the Flickr Diffusion Model (Diffusion Model 8.4.5 on page 376) described in Section 8.4.2. In this model, a constant parameter α represents the "strength" or "likelihood" of influence. The larger the parameter α the higher the influence of a user on those who voted for her.

- **Tipping diffusion model.** [21] shows that there is a relationship between the likelihood of a vertex marking a photo as a favorite and the percentage of their neighbors that also marked that photo as a favorite. This implies a tipping-model (as in Section 8.4.1). We apply the Jackson-Yariv mode (i.e. Diffusion Model 8.4.2) with B equated to $influence$. The predicate e corresponds to *vote*. For each vertex $V_j \in \mathbf{V}$, we set the benefit to cost ratio $(\frac{b_j}{c_j})$ to 1. Finally, the function γ defined in the Jackson Yariv model is the constant-valued function (for all values of x):

  $$\gamma(x) = \alpha.$$

  This says that irrespective of the number of neighbors that a vertex has, the benefit to adopting strategy B (i.e. $influence$) is α. Therefore, the resulting diffusion rule for the linear Jackson-Yariv model is:

  $$influence(V) : \alpha \cdot \frac{\sum_j X_j}{\sum_j E_j} \leftarrow \bigwedge_{V_j | (V_j, V) \in \mathbf{E}} (vote(V_j, V) : E_j \wedge influence(V_j) : X_j)$$

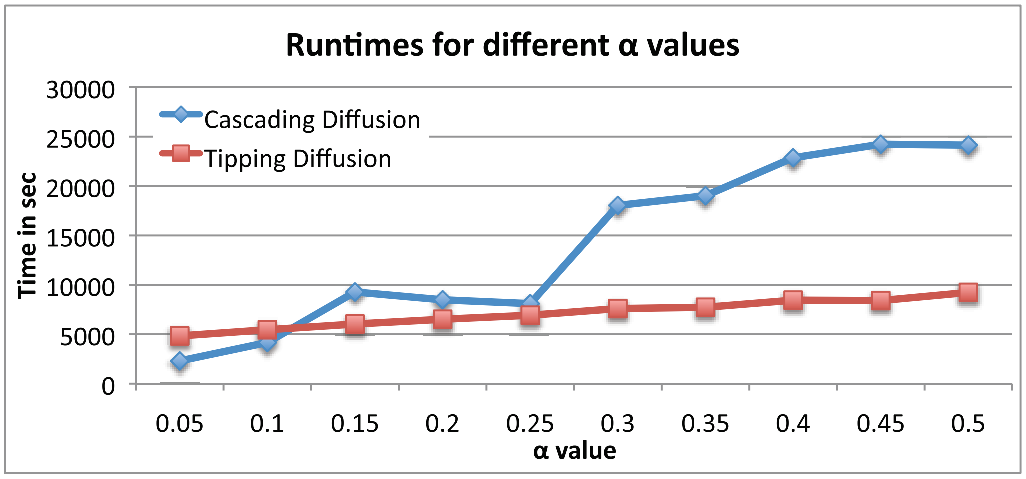


Figure 8.10: Runtimes of GREEDY-SNOP for different values of α and $k = 5$ in both diffusion models

For both models, we derive a unique logic program for each setting of the parameter α. The parameter α depends on the application and can be learned from ground truth data. In our experiments, we varied α to avoid introducing bias.

8.7.2 Experimental Results

Run-time of GREEDY-SNOP with varying α and different diffusion models. Figure 8.10 shows the total runtime of GREEDY-SNOP in seconds to find the set of $k = 5$ most influential users in the Wikipedia voting network for different values of the strength of influence parameter α. We varied α from 0.05 (very low level of influence) to 0.5 (very high level of influence) for both the cascading and tipping diffusion model. We observe that higher values of α lead to higher runtimes as expected since the scope of influence of any individual in the network is larger. Furthermore, we observe that the runtimes for the tipping diffusion model increase

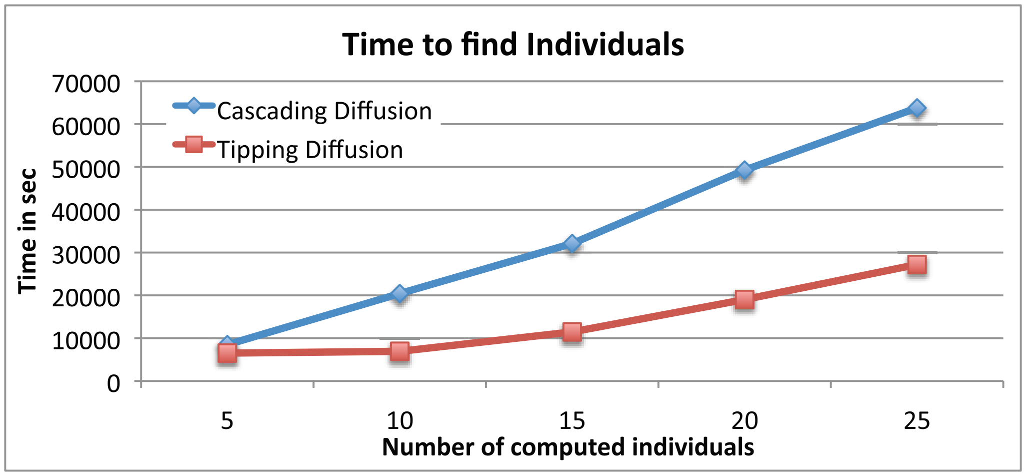


Figure 8.11: Runtimes of GREEDY-SNOP for different values of k and $\alpha = 0.2$ in both diffusion models

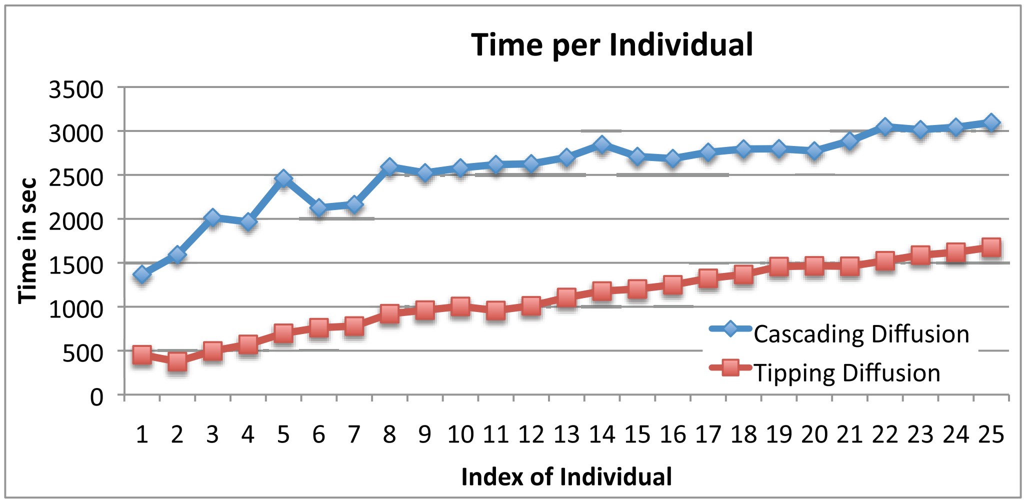


Figure 8.12: Time per iteration of GREEDY-SNOP for $\alpha = 0.2$ in both diffusion models

more slowly with α compared to the cascading model.

Run-time of GREEDY-SNOP with varying k. For the next set of experiments, we keep the strength of influence fixed to $\alpha = 0.2$ and varied k which governs the size of the set of influencers. Figure 8.11 reports the runtime of GREEDY-SNOP for the query $WQ(k)$ with $k = 5, 10, 15, 20, 25$. For the cascading model, the runtime is approximately linear in k a curve-fitting analysis using Excel showed a slight superlinear trend (even though the figure itself looks linear at first sight). Figure 8.12 shows the time taken to execute each of the 25 iterations of the outer loop for the query $WQ(25)$ with $\alpha = 0.2$. Note that each subsequent iteration is more expensive than the previous one since the size of the logic programs to consider increases with the addition of each ground atom $influence(V_i)$. However, we also implemented the practical improvement of "lazy evaluation" of the submodular function as described in [101]. This improvement, which maintains correctness of the algorithms, stores previous improvements in total score and prunes the greedy search for the highest scoring vertex as discussed. We found that this technique also reduced the runtime of subsequent iterations.

Our experimental results show that we can answer SNOP queries on large social networks. For example, computing the set of five most influential Wikipedia users in the voting network required approximately 2 hours averaged over the different values of α in the tipping diffusion model.

8.8 Chapter 8 Related Work

There has been extensive work in reasoning about diffusion in social networks. However, to our knowledge, there is no work on the relationship between logic programming and social networks. Moreover, there is no general framework to solve social network optimization problems that can take a broad class of diffusion models as input. We believe this work represents the first deterministic framework for representing generalized diffusion models that allows for different properties and weights on vertices and edges. Previously, the authors presented the framework of SNOPs in [159]. However, this brief technical communication did not include either our exact or approximate algorithms, an implementation, experiments, the SNOP-ALL problem, many of the complexity results, or many of the constructions seen in this chapter (such as the homophilic diffusion models and big-seed marketing).

8.8.1 Related Work in Logic Programming

We first compare our work with annotated logic programming [86, 85, 168] and its many extensions and variants [175, 88, 107, 109, 31, 82, 110]. There has been much work on annotated logic programming and we have built on the syntax and semantics of annotated LP. However, we are not aware of any work on solving optimization queries (queries that seek to optimize an aggregate function) w.r.t. annotated logic programming.

There are a few papers on solving optimization problems in logic programming. The best of these is constraint logic programming [172] which can embed numerical computations within a logic program. However, CLP does not try to find solutions to optimization problems involving semantics structures of the program itself. Important examples of constraint logic programming include [51, 117] where annotated LP is used for temporal reasoning, [99] assumes the existence of a cost function on models. They present an analysis of the complexity and algorithms to compute an optimal (w.r.t. the cost function) model of a disjunctive logic program in 3 cases: when all models of the disjunctive logic program are considered, when only minimal models of the disjunctive logic program is considered, and when stable models of the disjunctive logic program are considered. In contrast, in this chapter, there are two differences. First, we are considering GAPs. Second, we are not looking for models of a GAP that optimize an objective function - rather, we are trying to find models of a GAP *together with some additional information* (namely some vertices in the social network for which a goal atom $g(v) : 1$ is added to the GAP) which is constrained (at most k additional atoms) so that the resulting least fixpoint has an optimal value w.r.t. an arbitrary *value* function. In this regard, it has some connections with abduction in logic programs[41], but there is no work on abduction in annotated logic programs that we are aware of or work that optimizes an arbitrary objective function.

Our chapter builds on many techniques in logic programming. It builds upon non-ground fixpoint computation algorithms proposed by [114] and later extended for stable models semantics [59, 39], and extends these non-ground fixpoint algorithms to GAPs and hen applies the result to define the SNOP-Mon algorithm to find answers to SNOP-queries which, to the best of our knowledge, have not been considered before.

8.8.2 Work in Social Networks

[81] is one of the classic works in this area where a generalized diffusion framework for social networks is proposed. This work presents two basic diffusion models – the linear threshold and independent cascade models. Both models utilize random variables to specify how the diffusion propagates. These models roughly resemble non-deterministic versions of the tipping and cascading models presented in Section 8.4 of this chapter. Neither model allows for a straightforward representation of multiple vertex or edge labels as this work does. Additionally, unlike this chapter, where we use a fixed-point operator to calculate how the diffusion process unfolds, the diffusion models of [81] utilize random variables to define the diffusion process and compute the expected number of vertices that have a given property. The authors of [81] only approximate this expected value and leave the exact computation of it as an open question. Further, they provide no evidence that their approximation has theoretical guarantees.

The more recent work of [23] showed this computation to be #P-hard by a reduction from S-T connectivity, which has no known approximation algorithm. This suggests that a reasonable approximation of the diffusion process of [81] may not be possible. This contrasts sharply with the fixed-point operator of [86], which can be solved in PTIME under reasonable assumptions (which are present in this chapter). [81] focus on the problem of finding the "most influential nodes" in the graph – which is similar in intuition to a SNOP query. However, this problem only looks to maximize the the expected number of vertices with a given property, not a complex aggregate as a SNOP query does. Further, the approximation guarantee presented for the "most influential node" problem is contingent on an approximation of the expected number of vertices with a certain property, which is not shown (and, as stated earlier, was shown by [23] to be a #P-hard problem).

In short, the frameworks of [23] and [81] cannot handle arbitrary aggregates nor vertex conditions nor edge and vertex predicates nor edge weights as we do. Nor can they define an objective function using a mix of the aggregate and the $g(-)$ predicate specified in the definition of a SNOP-query.

8.9 Chapter Summary

Social networks are proliferating rapidly and have led to a wave of research on diffusion of phenomena in social networks. In this chapter, we introduce the class of *Social Network Optimization Problems* (SNOPs for short) which try to find a set of vertices (where each vertex specifies some user specified vertex condition) that have cardinality k or less (for a user-specified $k > 0$) and that optimize an objective function specified by the user in accordance with a diffusion model represented via the well-known Generalized Annotated Program (GAP) framework. We have used specific examples of SNOP-queries drawn from product adoption (cell phone example) and epidemiology.

The major contributions of this chapter include the following:

- We showed that the complexity of answering SNOP-queries as NP-Complete and identified the complexity classes associated with related problems (under various restrictions). We showed that the complexity of counting the number of solutions to SNOP-queries is #P-complete.

- We proved important results showing that there is no polynomial-time algorithm that computes an α-approximation to a SNOP-query when $\alpha \geq \frac{e}{e-1}$.

- We described how various well-known classes of diffusion models (cascading, tipping, homophilic) from economics, product adoption and marketing, and epidemiology can be embedded into GAPs.

- We presented an exact-algorithm for solving SNOP-queries under the assumption of a monotonic aggregate function.

- We proved that SNOP-queries are guaranteed to be submodular when the GAP representing the diffusion model is linear and the aggregate is positive-linear. We were able to leverage this result to develop the GREEDY-SNOP algorithm that runs in polynomial-time and that achieves the best possible approximation ratio of $\frac{e}{e-1}$ for solving SNOPs.

- We develop the first implementation for solving SNOP-queries and showed it could scale to a social network with over 7000 vertices and over 103,000 edges. Our experiments also show that SNOP-queries over tipping models can generally be solved more quickly than SNOP-queries over cascading models.

Much work remains to be done and this chapter merely represents a first step towards the solution of SNOP-queries. Clearly, we would like to scale SNOP-queries further for social networks consisting of millions of vertices and billions of edges. This will require some major advances and represents a big challenge.

Chapter 9

Future Work

There are many interesting questions that remain to be studied regarding spatio-temporal aspects of an agent's behavior. In this section, we briefly outline some important open questions relating to the work presented in this dissertation.

First, let us discuss extensions to reasoning about time using APT logic. Based on the framework presented in Chapter 2, we devised a fixpoint operator in Chapter 3 that provides sound, but incomplete solutions to consistency and entailment problems. Given a propositional formula F at time t (together forming a time formula $F : t$) and an APT program $\mathcal{K}$, our operator was able to produce a probability bound $[\ell, u]$ such that $\mathcal{K}$ entails $F : t : [\ell, u]$. As our operator is only sound, we do not guarantee that the bounds $[\ell, u]$ are the tightest possible. Further, even if we could obtain the tightest probability bounds possible, there is no guarantee on how close ℓ is to u. Hence, if we obtain the probability bounds $[0.4, 0.7]$, what can we say about the likelihood of F occurring at time t? The work of [18] considers a novel approach for dealing with the problem of the action-probabilistic programs of

[83][1]. Using random walks over the space of solutions to the query, they were able to produce a histogram of the the semantic structures for the query formula. As it is easy to compute the probability associated with each semantic structure, the authors of that paper were able to create a histogram of number of semantic structures with a given probability. Therefore, if a query returned a bounds of $[0.4, 0.7]$, they may also know that 80% of the semantic structures had a probability in the range $[0.67, 0.7]$, for example. A key issue encountered in [18] was the dimensionality of the space, which was exponenetial in the number of ground atoms. With APT logic, the problem is greatly increased, as the number of dimensions would be exponential in the product of number of ground atoms and time points. Most likely, heuristic methods for dimensionality reduction (perhaps by leveraging the FELC or WELC constraints of Chapter 2) would have to be employed in such work.

As far as the geospatial abduction problems of Chapters 4-6, an important direction would be to consider the case where the observations were caused by more than one agent. For example, in our experiments described in Chapter 4-5, we considered attacks and caches from Iranian-sponsored militants in Iraq. We implicitly assumed that these groups conducting attacks would operate in a similar manner and would share areas used for caches. As the results of our experiments were generally encouraging, this was most likely a valid assumption. However, suppose we have a set of attacks that could come from a variety of groups, which may not all operate in a similar manner and may not cooperate with each other. In such a case, it may not be appropriate to apply the algorithms of those chapters

[1]Time is not considered in [83] or [18].

as-is. There are two approaches to this variant: (1) cluster the attacks beforehand and solve an GAP for each cluster or (2) extend GAPs to a probabilistic and/or multi-agent case. Both raise interesting technical and practical issues.

Our work on optimally selecting agent actions in Chapters 7-8 looked at picking a set of agenst action with respect to some structure that maximizes an aggregate function. For the geospatial optimization problems of Chapter 7, although we were able to devise a polynomial-time approximation algorithm, it still has a runtime linearly proportional to the size of the map. For a large map, or a map of fine granularity, this may not be practical. Therefore, an obvious direction in future work is to devise a method to scale algorithms for answering geospatial optimization queries. For queries where we considered a diffusion process (the SNOP queries of Chapter 8), we did provide some methods to increase scalability. However, there is another issue with SNOPs. Although we could show embeddings for a wide variety of diffusion processes, these diffusion processes were monotonic in nature. For example, the confidence that vertex v has some property increases with each application of the fixpoint operator. This monotonic nature is what allowed us to leverage the generalized annotated programs of [86]. However, there are diffusion process such as voter models in physics [162] and evolutionary graph theory in biology [105] that are not monotonic in nature - *hence the confidence vertex v has some property may increase* ***or*** *decrease at each time step.* Computing the outcome of such process is difficult - in [105], the authors show evolutionary graph problems to be NP-hard. Currently, most work in this area relies on simulation. Hence, the first challenge with *non-monotonic diffusion* is to develop an efficient algorithm to determine the outcome of the diffusion process. One way to do this would be to introduce negation into the logic program. However, in this case we will most likely lose some computational properties that allow us to approximate SNOPs. Another is to adopt the "competitive" diffusion framework of [17]. However, it is unclear if voter model and/or evolutionary graph problems can be embedded into this framework. The second challenge is to answer a SNOP-query with respect to these problems. This most likely will add an additional layer of complexity. There are other aspects of SNOPs that can be explored as well. We note that our framework can be used to solve problems where "homophilic" [9] or non-network effects – even at the same time as network diffusion. Although the algorithms of this dissertation can be applied to these problems in a straight-forward manner, it remains an open question to create a tailored, efficient approach to these type of problems. Such an approach would aide greatly in "big-seed" marketing [177] that combines both viral-marketing along with mass-marketing.

Hence, although this paper explored many aspects of spatio-temporal reasoning about agent behavior, there are still some interesting open questions that should be explored.

Chapter 10

Conclusion

In this dissertation, we examined several aspects of reasoning regarding spatio-temporal agent behavior. These included determining the probability that an agent takes a given action at a certain time, abducing geospatial phenomenon, and optimizing the selection of an agents actions.

To determine the probability of an agent taking a given action at a given time, we have introduced a new framework for temporal-probabilistic reasoning called Annotated Probabilistic Temporal (APT) Logic. This logic-programming based approach allows one to create and/or automatically learn models of agent behavior based on past actions and determine the probability of some action at a certain time by performing an entailment query. Notably, unlike other formalisms for reasoning about time and probability together, APT-logic does not make Markov or independence assumptions. Despite not making these assumptions, we have designed and implemented an approximation technique based on a sound, but incomplete fixpoint operator (we resort to approximation techniques as we show answering such a query is NP-complete). Although incomplete, in our experiments, the implementation of this operator was shown to find tight bounds on entailment formualae. The calculation runs in approximately linear time in the size of the model, which is a significant improvement over exact methods for solving these queries which require solving a linear program with an exponential number of variables.

To reason about the spatial aspects of an agent's behavior, we looked at observed manifestations of the behavior (called "observations") that must have been caused by some other, unobserved, geospatial phenomenon (called "partners"). Finding a set of partners corresponding to the observations is an instance of a problem known as "geospatial abduction." In this dissertation, we have created a framework for this scenario and explored the problem of finding a set of partners given a set of observations and constraints on the relationship between the two. Unfortunately, as we show many such problems to be NP-complete, we again resorted to approximation techniques. In addition to showing reductions from well-known problems, we created a novel greedy algorithm, that while maintaining an approximation guarantee, allows for the use of additional heuristics. We implemented this algorithm in a software package called "SCARE" and showed that it significantly out-performed naive techniques for locating weapons caches associated with attack sites using a counter-insurgency data-set. We then explored a variant of a geospatial abduction problem that requires the solution to return regions rather than pin-point locations. Again, as this problem was NP-complete, we had to resort to approximation techniques. We note that a special case of this problem actually reduces to circle-covering, for which there are known approximation techniques. We also introduced an approximation technique for a more general case, implemented it, and showed it to provide viable results on real-world data.

As a geospatial abduction problem, like many other abduction problems, can have multiple solutions, a natural question is "how does one solve such problems when the adversary has knowledge of your algorithm?" We explore this situation where the adversary has a probability distribution of the solutions to a geospatial abduction problem and can position his partners ahead of time in a manner to avoid discovery by the agent. This problem, again, is NP-complete. However, we show it can reduce to a mixed linear-integer program that can be made more tractable by significantly reducing the number of variables and using parallelization. We show the viability of this approach by implementing this algorithm using a linear-integer program solver. A natural complement to this problem that we explored is how an agent should select partners given a probability distribution of how the adversary selected locations. We show that this problem reduces to the maximization of a submodular function over a uniform matroid and can be solved using several well-known approximation techniques. We also presented an implementation.

We then studied optimal selection of agent actions. The first problem of this type was a "geospatial optimization" problem. Here the agent has a set of actions that modify attributes of a geospatial region and he wishes to select a limited number of such actions (with respect to some budget) in a manner that either satisfies some goal (goal-based geospatial optimization) and/or maximizes a benefit function (benefit-maximizing geospatial optimization). Additionally, there are certain combinations of actions that cannot be performed together. We proved that both goal-based and benefit-maximizing geospatial optimization problems are NP-complete under reasonable assumptions and proved theoretical limits on their approximation. We then develop algorithms for solving such problems - either exactly or within a certain factor of optimal.

We also look at *optimally selecting agent actions in the presence of a diffusion process under the structure of a social network.* To address this topic, we presented an annotated-program based framework for studying social network optimization problems - that is given a social network (a weighted, directed graph with vertex and edge labels), and a diffusion process, can we identify the vertices of the network that cause a given phenomenon to spread to the maximum extent possible. This generalized framework allows great flexibility is expressing several well-known diffusion models in the areas of marketing, information spread, and disease. We show queries relating to this problem to be strongly NP-complete as it can encode the max-k-cover problem. However, we also show that if the annotated program representing the diffusion process is "linear" then the value oracle associated with this query is submodular - which allows us to leverage a greedy approach that provides the best approximation guarantee for such a query unless P=NP. Using a variety of techniques, we show that this algorithm can be scaled to large networks and we provide an implementation as well.

Appendix A

Appendix for Chapter 2

A.1 Additional Results

A.1.1 Frequency Equivalence under the PCD Restriction

While obtaining a noticeable speedup for constrained programs using FELC, we were still required to conduct an operation exponential in the product of atoms and time points. In this subsubsection we leverage the PCD restrictions from Definition 3.5 to ensure that there are no empty frequency-equivalence classes. First, we present some notation to describe sets of threads associated with each rule. Then, we show how the PCD restrictions allow us to leverage FELC without preprocessing. Finally, as the PCD requirements do not permit annotated formulas, we present some methods to allow for annotated formulas as well.

Our key intuition is noticing that the result of Lemma 3.6 uses the axioms to ensure that we can create threads where the frequency function for each rule equals 0 or 1. Then, we use the *one-tailed* restriction that PCD's provide us in order to ensure that $\beta = 1$ (it is not hard to prove a similar theorem where α is set to 0). With this *one-tailed* restriction, threads with a frequency function of 0 are outside of $[\alpha, \beta]$ and threads with a frequency of 1 will be inside this range.

To help us in our discussion of how FELC can leverage PCD programs, we present some notation used in describing classes of frequency equivalent threads; associated thread subsets allow us to formalize the notion of a frequency-equivalent class of threads. We provide the definition below.

Definition 109 (Associated Thread Subsets (ATS)). *For a given constrained rule, $r_i = F \overset{\mathsf{fr}}{\hookrightarrow} G : [\Delta t, \ell, u, \alpha, \beta]$, the Associated Thread Subsets (ATS) are the subsets of the set of threads considered in the satisfaction of r_i:*

- *ATS_i is the set of threads $\{Th \in \mathcal{T} \mid \alpha \leq \mathsf{fr}(Th, F, G, \Delta t) \leq \beta\}$.*
- *$\overline{\mathsf{ATS}}_i$ is the set of threads $\{Th \in \mathcal{T} \mid Th \notin \mathsf{ATS}_i\}$.*

Intuitively, a thread is in a rule's associated thread subset (ATS) if its frequency function with respect to that particular rule falls within that rule's $[\alpha, \beta]$ frequency function bounds. Threads not meeting this criteria are said to be in the complement associated thread subset ($\overline{\mathsf{ATS}}$) for that rule.

Normally, in this subsubsection, we refer to a given frequency equivalence class as cl_s where $s \in [0,1]^m$ - where m is the number of rules in the APT-program. With this notation, the ATS of every rule (r_i) where $s_i = 1$ is intersected with the $\overline{\mathsf{ATS}}$ of every rule where $s_i = 0$. Formally,

$$cl_s = \left\{ \bigcap_{s_i=1} \mathsf{ATS}_i \right\} \cap \left\{ \bigcap_{s_i=0} \overline{\mathsf{ATS}}_i \right\}$$

One can easily see that a frequency equivalence class is empty if the intersub-subsection of any two subsets described above are empty. Consider the following example.

Example A.1.1. *Consider the discussion on the two rules from $\mathcal{K}_{stock}$ in Example 3.11.*
Let $r_2 = \textsf{\textit{sec\_rumor}} \wedge \textsf{\textit{earn\_incr(10\%)}} \overset{pfr}{\hookrightarrow} \textsf{\textit{stock\_decr(10\%)}} : [2, 0.65, 0.97, 0.7, 1.0]$ *and* $r_3 = \textsf{\textit{sec\_rumor}} \wedge \textsf{\textit{earn\_incr(10\%)}} \overset{pfr}{\hookrightarrow} \textsf{\textit{stock\_decr(10\%)}} \wedge \textsf{\textit{cfo\_resigns}} : [2, 0.68, 0.95, 0.7, 0.8]$. *Recall that in Example 3.11 we determined that for any given thread, it was not possible for the pfr associated with r_3 to exceed the pfr associated with r_2. Therefore, as $\beta_2 = 1$, we can conclude that $\overline{\textsf{\textit{ATS}}_2} \cap \textsf{\textit{ATS}}_3 \equiv \emptyset$.*

We can now examine how to utilize the PCD restriction to ensure that all frequency equivalence classes are non-empty. First, using the one-tailed restriction described earlier, consider rule $r_i = F_i \overset{\mathsf{fr}_i}{\hookrightarrow} G_i : [\Delta t_i, \ell_i, u_i, \alpha_i, 1]$ (where $\alpha_i > 0$). For thread Th, if $\mathsf{fr}_i(Th, F_i, G_i, \Delta t_i) = 1$ then $Th \in \mathsf{ATS}$. If $\mathsf{fr}_i(Th, F_i, G_i, \Delta t_i) = 0$ then $Th \in \overline{\mathsf{ATS}}$. However, we then must add most of the other PCD restrictions to ensure that the thread construction of the 0 and 1 threads can occur for each rule. In fact, the only PCD restriction not needed is the sixth restriction that sets $u = 1$ for all rules.

Two of the PCD restrictions are particularly limiting. One requires that for each rule pre-condition there exists a unique world that only satisfies that pre-condition. The second is similar: there exists a unique world that does not satisfy any rule's pre or post-conditions. It may, however, be possible to specify such restrictions as part of a procedure to obtain such rules. Regardless, once one has an APT-logic program satisfying the PCD restrictions, one can be guaranteed the below property, which we will shortly see to be useful.

Theorem 58. *Suppose APT-Logic program, $\mathcal{K} = \bigcup_{i=1}^{m} \{r_i\}$ where $r_i = F_i \overset{fr_i}{\hookrightarrow} G_i : [\Delta t_i, \ell_i, u_i, \alpha_i, \beta_i]$ meets PCD restrictions 1-6 of Definition 3.5. Then, for all binary numbers $s \in [0,1]^m$, frequency equivalence classes $cl_s = \{\bigcap_{s_i=1} \mathsf{ATS}_i\} \cap \{\bigcap_{s_i=0} \overline{\mathsf{ATS}}_i\}$ contains at least one thread.*

Before investigating the utility of the above theorem, we notice that restriction 4 creates an issue if we decide to include annotated formulas. Recall that, by Theorem 2.20, to create a constrained rule that is equivalent to an annotated formula, we must set the pre-condition to TRUE. Additionally, PCD programs do not allow qfr. Clearly, this causes the above Theorem to be not applicable under that circumstance. However, we can provide a similar set of restrictions that will allow annotated formulas. The intuition for the proof is simple, we simply extend t_{max} to be as long as the maximum t value in any of the annotated formulas in question, and add some restrictions about the existence of satisfying worlds which resemble those of the last theorem extended to annotated formulas.

Corollary 15. *For a constrained APT-Logic program, $\mathcal{K} = \bigcup_{i=1}^{m} \{r_i\}$ where $r_i = F_i \overset{fr_i}{\hookrightarrow} G_i : [\Delta t_i, \ell_i, u_i, \alpha_i, \beta_i]$ and set of annotated formulas, $FACTS = \bigcup_{j=1}^{k} af_j$ where $af_j = Q_j : [t'_j, \ell'_j, u'_j]$ with the following restrictions,*

1. *$\mathcal{K}$ meets PCD restrictions 1-4 of Definition 3.5*

2. *$t_{max} \geq max_j(t'_j) + |\mathcal{K}| \cdot max_i(\Delta t_i)$*

3. *$\exists$ world $w_\emptyset$ such that $\forall i \in [0,m]$ $w_\emptyset \not\models F_i$, $w_\emptyset \not\models G_i$, and $\forall j \in [0,k]$ $w_\emptyset \not\models Q_j$*

4. *$\forall j \in [0,k]$, there exists world qw_j such that $qw_j \models Q_j$ and $\forall i \in [0,m]$ $qw_j \not\models F_i$.*

5. *No two annotated formulas in $FACTS$ are at the same timepoint.*

Then for all $s \in [0,1]^{m+k}$, frequency equivalence classes $cl_s = \left\{\bigcap_{s_i=1} \mathsf{ATS}_i\right\} \cap \left\{\bigcap_{s_i=0} \overline{\mathsf{ATS}}_i\right\}$ contains at least one thread.

Again, we have a potentially problematic restriction in that we do not allow multiple annotated formulas in the same time point (restriction 5). We can relax this restriction with the following corollary to determine if a given frequency equivalence class exists or not. Essentially, the non-emptiness of a frequency equivalence class that contains intersubsubsections of ATS or $\overline{\mathsf{ATS}}$ sets for annotated formulas is determined by the existence of worlds satisfied by the non-annotated portion of the annotated formulas that share the same time point.

Corollary 16. *Assume we have the following:*

- *Constrained APT-Logic program, $\mathcal{K} = \bigcup_{i=1}^{m} \{r_i\}$ where $r_i = F_i \overset{\mathsf{fr}_i}{\hookrightarrow} G_i : [\Delta t_i, \ell_i, u_i, \alpha_i, \beta_i]$*

- *Set of annotated formulas, $FACTS = \bigcup_{j=1}^{k} af_j$ where $af_j = Q_j : [t_j, \ell_j, u_j]$*

- *$\forall j \in [1,k]$, we define $\mathsf{ATS}_j^{(q)}$ and $\overline{\mathsf{ATS}}_j^{(q)}$ to be the ATS and $\overline{\mathsf{ATS}}$ for rule $r = \mathsf{TRUE} \overset{qfr}{\hookrightarrow} Q_j : [t, \ell_j, u_j, 1, 1]$ created from annotated formula $af_j \in FACTS$ using qfr*

- *Restrictions 1-5 from Corollary 15*

Then, for all subsets of $SAMETIME \subseteq FACTS$, where for all $af_1, af_2 \in SAMETIME$, $t_1 = t_2$; for all strings $s2 \in [0,1]^{|SAMETIME|}$ all frequency equivalence classes, cl that intersect $\{\bigcap_{s2_i=1} \mathsf{ATS}_i^{(q)}\} \cap \{\bigcap_{s2_i=0} \overline{\mathsf{ATS}_i^{(q)}}\}$ are non-empty iff:
$\exists$ world w_p such that for all $af_i \in SAMETIME$ where $s2_i = 1$, $w_p \models Q_i$ and for all $af_j \in SAMETIME$ where $s2_j = 0$, $w_p \not\models Q_j$.

Theorem 58, and Corollaries 15 and 16 provide restrictions that force all frequency-equivalence classes to be non-empty. This allows us to leverage FELC without any type of pre-processing algorithm, which we have shown to be expensive. As stated earlier, PCD restrictions could be specified by a tool used to learn the rules from a given data-set.

A.1.2 The **ALC-ENT** Algorithm for Entailment

We present an algorithm for alternate linear constraints, ALC-ENT, that computes entailment using linear constraints other than SLC.

Proposition 81. *If $\mathcal{K}$ entails r, then ALC-ENT returns ENTAILS. If $\mathcal{K}$ does not entail r, then ALC-ENT returns NOT ENTAILS.*

In the worst case, to solve ALC-ENT requires constructing the linear constraints once and solving the linear program twice.

Example A.1.2. *Recall that in Example A.1.1 we presented rules*
$r_2 = \mathit{sec\_rumor} \wedge \mathit{earn\_incr(10\%)} \overset{pfr}{\hookrightarrow} \mathit{stock\_decr(10\%)} : [2, 0.65, 0.97, 0.7, 1.0]$ *and*
$r_3 = \mathit{sec\_rumor} \wedge \mathit{earn\_incr(10\%)} \overset{pfr}{\hookrightarrow} \mathit{stock\_decr(10\%)} \wedge \mathit{cfo\_resigns} : [2, 0.68, 0.95, 0.7, 0.8]$.

Algorithm 24 Alt. Linear Constraints for Entailment of Rule r by Program $\mathcal{K}$

```
ALC-ENT(APT-Program K)

1. Create the set of alternate linear constraints for $\mathsf{WELC}(\mathcal{K} \cup r)$ or $\mathsf{FELC}(\mathcal{K} \cup r)$.

2. If $r$ is unconstrained, ($r = F \overset{\mathsf{fr}}{\rightsquigarrow} G : [\Delta t, \ell, u]$), create rule $r' = F \overset{\mathsf{fr}}{\rightsquigarrow} G : [\Delta t, \ell', u']$ where $\ell', u'$ are variables. Note that unconstrained entailment can only be checked if the constraints used are WELC in this algorithm.

3. If $r$ is constrained, ($r = F \overset{\mathsf{fr}}{\hookrightarrow} G : [\Delta t, \ell, u, \alpha, \beta]$) create rule $r' = F \overset{\mathsf{fr}}{\hookrightarrow} G : [\Delta t, \ell', u', \alpha, \beta]$ where $\ell', u'$ are variables.

4. Create set of linear constraints $\mathsf{WELC}(\mathcal{K} \cup \{r'\})$ or $\mathsf{FELC}(\mathcal{K} \cup \{r'\})$.

5. Let $\bar{\ell}'$ be the minimization of $\ell'$ subject to the linear constraints.

6. Let $\bar{u}'$ be the maximization of $u'$ subject to the linear constraints.

7. If $[\bar{\ell}', \bar{u}'] \subseteq [\ell, u]$ return ENTAILS otherwise return NOT ENTAILS.
```

Let $\mathcal{K}_{\textit{ent-ex}} = \{r_2\}$. Suppose we want to determine if $\mathcal{K}_{\textit{ent-ex}}$ entails r_3 using ALC-ENT, employing the FELC constraints. We create rule $r_3' = \mathsf{sec\_rumor} \wedge \mathsf{earn\_incr}(10\%) \overset{pfr}{\hookrightarrow} \mathsf{stock\_decr}(10\%) \wedge cfo\_resigns : [2, \ell', u', 0.7, 0.8]$. Based on Example A.1.1, we know there are 3 frequency equivalence classes for $\mathcal{K}_{\textit{ent-ex}} \cup \{r_3'\}$ as $\overline{\mathsf{ATS}_2} \cap \mathsf{ATS}_3 \equiv \emptyset$. Hence, we have 3 variables, $\bar{v}_{00}, \bar{v}_{01}, \bar{v}_{11}$. Therefore, we set up the following constraints:

- *For rule r_2:* $0.65 \leq \bar{v}_{01} + \bar{v}_{11} \leq 0.97$
- *For rule r_3':* $\ell' \leq \bar{v}_{11} \leq u'$

- $\bar{v}_{00} + \bar{v}_{01} + \bar{v}_{11} = 1$

As it turns out, the minimization of ℓ' *is* 0 *and the maximization of* u' *is* 0.97. *Since* $[0, 0.97] \not\subset [0.68, 0.95]$, *we can say that* $\mathcal{K}_{\textit{ent-ex}}$ *does* ***not*** *entail* r_3.

A.1.3 An Example Comparing PCTL to APT-rules

In this appendix, we provide a small example of a Markov Process which for some t_{max} can be expressed as an APT-program. We then show that there is a "lead-to" PCTL formula that is satisfied by the Markov Process and also show an APT-rule that is similar, though not entailed by $\mathcal{K}$, to the corresponding APT-program.

Given ground atoms $B_{\mathcal{L}}$, consider a Markov Process, M defined as follows:

- Set of states $S = \{S_1, S_2, S_3\}$, where each s_i has a unique label based on ground atoms in $B_{\mathcal{L}}$

- Transition probability P defined as follows:

  $P(S_1, S_2) = 0.3$,

  $P(S_1, S_3) = 0.7$,

  $P(S_2, S_1) = 0.2$,

  $P(S_2, S_3) = 0.8$,

  $P(S_3, S_1) = 0.1$,

  $P(S_3, S_2) = 0.9$,

  and 0 for all other transitions.

- Initial state S_3

We will define formulas $F(S_1)$, $F(S_2)$, $F(S_3)$ to be propositional formulas satisfied by exactly S_1, S_2, S_3 respectively. For $t_{max} = 5$, let APT program $\mathcal{K}$ be the APT-program corresponding to M constructed using MAKE-APT (Algorithm 7).

Let us consider the following PCTL formula:

$$G_{\geq 1}^{\leq t_{max}} F(S_1) \curvearrowright_{\geq 0.75}^{\leq 2} F(S_1)$$

This formula intuitively says: "for all sequences starting at time 1 and ending at time t_{max} (*i.e.*, the first t_{max} states), the probability that formula $F(S_1)$ is followed by itself in less than 2 time units is greater than 0.75."

So, let us consider all sequences of states that start in S_1 and end in S_1 of length 3 or less (obviously, $S_1 \models F(S_1)$).

- $\langle S_1, S_2, S_1 \rangle$
- $\langle S_1, S_3, S_1 \rangle$

By the multiplication of transition probabilities, the first sequence has a probability of 0.06 and the second has a probability of 0.07 – hence, as they sum to 0.13, we see that the PCTL formula is **not** satisfied by M.

$$M \not\models G_{\geq 1}^{\leq t_{max}} F(S_1) \curvearrowright_{\geq 0.75}^{\leq 2} F(S_1)$$

So, now let us consider an "analogous" APT-rule:

$$F(S_1) \overset{efr}{\rightsquigarrow} F(S_1) : [2, 0.75, 1]$$

By the results of Section 6.1, we know there must be exactly one satisfying interpretation for $\mathcal{K}$ – as there is exactly one interpretation associated with M; lets call this interpretation I. Consider the following list of sequences of states, which correspond to threads assigned a non-zero probability by I. Below we list the sequences, the probability, and the frequency – that is for the associated thread, Th, the value $\mathit{efr}(F(S_1), F(S_1), 2, \mathit{Th})$.

Sequence	*Probability*	*Frequency*
S_3, S_1, S_2, S_1, S_2	0.0018	1
S_3, S_1, S_2, S_1, S_3	0.0042	1
S_3, S_1, S_2, S_3, S_1	0.0024	0
S_3, S_1, S_2, S_3, S_2	0.0216	0
S_3, S_1, S_3, S_1, S_2	0.0021	1
S_3, S_1, S_3, S_1, S_3	0.0049	1
S_3, S_1, S_3, S_2, S_1	0.0126	0
S_3, S_1, S_3, S_2, S_3	0.0504	0
S_3, S_2, S_1, S_2, S_1	0.0108	1
S_3, S_2, S_1, S_2, S_3	0.0432	0
S_3, S_2, S_1, S_3, S_1	0.0126	1
S_3, S_2, S_1, S_3, S_2	0.1134	0
S_3, S_2, S_3, S_1, S_2	0.0216	1
S_3, S_2, S_3, S_1, S_3	0.0504	1
S_3, S_2, S_3, S_2, S_1	0.1296	1
S_3, S_2, S_3, S_2, S_3	0.5184	1

We note that the probability mass associated with the APT-rule sums to 0.7564; hence, the APT-rule is entailed by the program and satisfied by I, while the PCTL formula is not.

It is also important to notice other differences. As we can create APT-programs that do not have corresponding MDP's, there is a corresponding expressiveness in the APT-rules that cannot be replicated with PCTL. Conversely, APT-logic does not handle infinite temporal sequences; problems with this requirement may be better suited for PCTL.

A.2 Proofs

A.2.1 Proof of Lemmas 2.12 and 2.14

pfr satisfies Axioms FF1-FF4 (2.12).
efr satisfies Axioms FF1-FF4 (2.14).

Proof. .
Axioms FF1,FF2,FF3 follow directly from the definitions of *pfr* and *efr*.
Axiom FF4: We construct Th_0 such that $pfr(Th_0, F, G, \Delta t) = 0$ as follows (one can verify that the same Th_0 causes $efr(Th_0, F, G, \Delta t) = 0$). Either $F \wedge \neg G$ has a solution or it does not. Proceeding by cases. When $F \wedge \neg G$ has a solution, let $Th_0(i) \models F \wedge \neg G$ for all i. Notice that in this case, $pfr(Th_0, F, G, \Delta t) = 0$. When $F \wedge \neg G$ has no solution, then let $w_0 \models F$ (possible since F is not a contradiction) and $w_1 \models \neg G$ (possible since G is not a tautology), and set $Th_0(0) = w_0$, $Th_0(i > 0) = w_1$. Note that w_1 does not satisfy F (otherwise $F \wedge \neg G$ would have a solution), and that therefore $pfr(Th_0, F, G, \Delta t) = 0$. In all cases $pfr(Th_0, F, G, \Delta t) = 0$.

We now construct Th_1 such that $pfr(Th_1, F, G, \Delta t) = 1$ as follows (one can again verify that the same Th_1 causes $efr(Th_1, F, G, \Delta t) = 1$). When F is not a tautology, there is a world w that does not satisfy F. Assign $Th_1(i) = w$ for all i. Note that $pfr(Th_1, F, G, \Delta t) = 1$. When F is a tautology then we know G is not a contradiction. Thus there is w that satisfies G. Assign $Th_1(i) = w$, and note that $pfr(Th_1, F, G, \Delta t) = 1$. □

A.2.2 Proof of *pfr* Property 5

Below we prove *pfr* property 5.

Proof. CASE 1: $a + b \leq 1$

As $pfr(Th, F_1, F_2 \wedge F_3, \Delta t) \geq 0$ by the definition of *pfr*, this is trivial.

CASE 2: $a + b > 1$

Let $W_1, W_2, W_3, W_{2,3}$ be sets of worlds that satisfy $F_1, F_2, F_3, F_2 \wedge F_3$ respectively.

As $a + b > 1$, then we know that $W_2 \cap W_3 \neq \emptyset$. Let W^* be this set.

As all $w^* \in W^*$ satisfy F_2 and F_3, then $W^* \subseteq W_{2,3}$.

Let F^* be a formula that $\forall w^* \in W^*, w^* \models F^*$.

Hence, $pfr(Th, F_1, F^*, \Delta t) \geq a + b - 1$.

As $W^* \subseteq W_{2,3}$, the statement follows. □

A.2.3 Proof of Proposition 2.15

Part (1): Let Th be a thread, F and G be formulas, and Δt_1 and Δt_2 be two temporal intervals. If $\Delta t_1 \leq \Delta t_2$, we have that $pfr(Th, F, G, \Delta t_1) \leq efr(Th, F, G, \Delta t_2)$.

Proof. CLAIM 1: $pfr(Th, F, G, \Delta t_1) \leq efr(Th, F, G, \Delta t_2)$ for $efn(Th, F, G, \Delta t_2, t_{max}-$

$\Delta t_2, t_{max}) = 0$.

(1) By the definition of efn, we have: $|\{t : Th(t) \models F \wedge Th(t+\Delta t_1) \models G\}| \leq |\{t : Th(t) \models F \wedge \exists t' \in [t, t+\Delta t_2]\ s.t.\ Th(t') \models G\}|$

(2) Hence, $|\{t : Th(t) \models F \wedge Th(t+\Delta t_1) \models G\}| \leq efn(Th, F, G, \Delta t_2, 0, t_{max})$. The claim follows.

CLAIM 2: $pfr(Th, F, G, \Delta t_1) \leq efr(Th, F, G, \Delta t_2)$

(1) Let $x = efn(Th, F, G, \Delta t_2, t_{max} - \Delta t_2, t_{max}) > 0$

(2) Let $\frac{a}{b} = pfr(Th, F, G, \Delta t_1)$

(3) Let $\frac{c}{d} = efr(Th, F, G, \Delta t_2)$

(4) By claim 1, we have $\frac{a}{b} \leq \frac{c-x}{d-x}$.

(5) By the definition of pfr and efr, we know that $d - x = b$, therefore $ab \leq bc - bx$.

(6) By the definition of pfr, we know that $a \leq b$, therefore $a(b+x) \leq bc$ which is equivalent to $ad \leq bc$. (7) Hence, $ad \leq bc$ which gives us $pfr(Th, F, G, \Delta t_1) \leq efr(Th, F, G, \Delta t_2)$. $\square$

Part 2: Let Th be a thread, F and G be formulas, and Δt be a temporal interval. The following inequality always holds:

$$efr(Th, F, G, \Delta t) \leq \sum_{i=1}^{\Delta t} pfr(Th, F, G, i)$$

CLAIM 1: If $efr(Th, F, G, \Delta t) - efr(Th, F, G, \Delta t - 1) = efr(Th, F, G, \Delta t)$ then $efn(Th, F, G, \Delta t, t_{max} - \Delta t, t_{max}) = 0$

(Claim 1). Suppose, $\exists efn(Th, F, G, \Delta t, t_{max} - \Delta t, t_{max}) \neq 0$ then, as there can be

no two worlds in Th that satisfy F and G with exactly Δt time periods between, $\mathit{efr}(Th, F, G, \Delta t - 1) \neq 0$ as the two such worlds would satisfy F and G within $\leq \Delta t - 1$ time intervals.

Therefore, we have a contradiction as $\mathit{efr}(Th, F, G, \Delta t) - \mathit{efr}(Th, F, G, \Delta t - 1) = \mathit{efr}(Th, F, G, \Delta t)$ forces $\mathit{efr}(Th, F, G, \Delta t - 1) = 0$.

□

CLAIM 2: $\mathit{efr}(Th, F, G, \Delta t) \leq \mathit{efr}(Th, F, G, \Delta t - 1) + \mathit{pfr}(Th, F, G, \Delta t)$

Intuition: If we consider an *efr* with Δt and subtract the same *efr* with $\Delta t - 1$, we are essentially only considering G at exactly Δt, hence the *pfr* for Δt. Claim 1 allows us to ignore the effect of efn in for the worlds between $t_{max} - \Delta t, t_{max}$.

(Claim 2). By the definition of *efr*, we know $\mathit{efr}(Th, F, G, \Delta t) \geq \mathit{efr}(Th, F, G, \Delta t - 1)$.

Therefore, $0 \leq \mathit{efr}(Th, F, G, \Delta t) - \mathit{efr}(Th, F, G, \Delta t - 1) \leq \mathit{efr}(Th, F, G, \Delta t)$.

By claim 1, if $\mathit{efr}(Th, F, G, \Delta t) - \mathit{efr}(Th, F, G, \Delta t - 1)$ is maximized (hence $\mathit{efr}(Th, F, G, \Delta t) - \mathit{efr}(Th, F, G, \Delta t - 1) = \mathit{efr}(Th, F, G, \Delta t)$) then $efn(Th, F, G, \Delta t, t_{max} - \Delta t, t_{max}) = 0$. Therefore, by the definitions of *pfr* and *efr* we have $\mathit{efr}(Th, F, G, \Delta t) - \mathit{efr}(Th, F, G, \Delta t - 1) \leq \mathit{pfr}(Th, F, G, \Delta t)$. Therefore, we know if $\mathit{efr}(Th, F, G, \Delta t) - \mathit{efr}(Th, F, G, \Delta t - 1) \leq \mathit{pfr}(Th, F, G, \Delta t)$. The claim follows.

□

(Proposition 2). By induction on Δt.

BASE CASE: By the definition of *pfr* and *efr* we have $efr(Th, F, G, 1) = pfr(Th, F, G, 1)$.

INDUCTIVE HYPOTHESIS: Assume $efr(Th, F, G, \Delta t{-}1) \leq \sum_{i=1}^{\Delta t-1} pfr(Th, F, G, i)$ is true.

INDUCTIVE STEP: By the inductive hypothesis. $efr(Th, F, G, \Delta t{-}1) \leq \sum_{i=1}^{\Delta t-1} pfr(Th, F, G, i)$. We add $pfr(Th, F, G, \Delta t)$ to both sides. $efr(Th, F, G, \Delta t - 1) + pfr(Th, F, G, \Delta t) \leq \sum_{i=1}^{\Delta t} pfr(Th, F, G, i)$. By claim 2, we have $efr(Th, F, G, \Delta t) \leq \sum_{i=1}^{\Delta t} pfr(Th, F, G, i)$. We can now apply the inductive hypothesis and are finished. $\square$

A.2.4 Proof of Proposition 2.17

Part (1): Let I be a temporal interpretation, F and G be formulas, and Δt be a temporal interval. If $I \models \bigcup_{i=1}^{\Delta t}\{F \overset{pfr}{\rightsquigarrow} G : [i, \ell_i, u_i]\}$ then $I \models F \overset{efr}{\rightsquigarrow} G : [\Delta t, max(\ell_i), min(\sum_{i=1}^{\Delta t} u_i, 1)]$

Proof. By Proposition 2.15, we have the following: $max(\sum_{Th \in \mathcal{T}} I(Th) \cdot pfr(Th, F, G, i)) \leq \sum_{Th \in \mathcal{T}} I(Th) \cdot efr(Th, F, G, \Delta t)$. By Proposition 2, we have the following: $\sum_{Th \in \mathcal{T}} I(Th) \cdot efr(Th, F, G, \Delta t) \leq \sum_{Th \in \mathcal{T}} I(Th) \cdot min(\sum_{i=1}^{\Delta t} pfr(Th, F, G, i))$

The statement immediately follows. $\square$

Part (2): If $I \models F \overset{\mathsf{fr}}{\hookrightarrow} G : [\Delta t, \ell_p, u_p, a, b]$ then $\forall a_\ell, b_\ell, a_u, b_u$ such that $a_\ell \leq a \leq a_u$ and $b_\ell \leq b \leq b_u$ we have $I \models F \overset{\mathsf{fr}}{\hookrightarrow} G : [\Delta t, \ell_p, 1, a_\ell, b_u]$ and $I \models F \overset{\mathsf{fr}}{\hookrightarrow} G : [\Delta t, 0, u_p, a_u, b_\ell]$.

Proof. PART 1: $I \models F \overset{\mathsf{fr}}{\hookrightarrow} G : [\Delta t, \ell, u, a, b]$ then $I \models F \overset{\mathsf{fr}}{\hookrightarrow} G : [\Delta t, \ell, 1, a_\ell, b_u]$

We know that $\{Th : a \leq \mathsf{fr}(Th, F, G, \Delta t) \leq b\} \subseteq \{Th : a_\ell \leq \mathsf{fr}(Th, F, G, \Delta t) \leq b_u\}$

Hence, if $\ell_1 \leq \sum_{Th \in \mathcal{T}, a_\ell \leq \mathsf{fr}(Th,F,G,\Delta t) \leq b_u} I(Th) \leq u_1$ then $\ell \leq \ell_1$ and $u_1 \leq 1$. The statement follows.

PART 2: $I \models F \overset{\mathsf{fr}}{\hookrightarrow} G : [\Delta t, \ell, u, a, b]$ then $I \models F \overset{\mathsf{fr}}{\hookrightarrow} G : [\Delta t, 0, u, a_u, b_\ell]$

We know that $\{Th : a_u \leq \mathsf{fr}(Th, F, G, \Delta t) \leq b_\ell\} \subseteq \{Th : a \leq \mathsf{fr}(Th, F, G, \Delta t) \leq b\}$

Hence, if $\ell_2 \leq \sum_{Th \in \mathcal{T}, a_u \leq \mathsf{fr}(Th,F,G,\Delta t) \leq b_\ell} I(Th) \leq u_2$ then $0 \leq \ell_2$ and $u_2 \leq u$. The statement follows.

$\square$

A.2.5 Proof of Lemma 2.19

The *qfr* satisfies Axioms FF1-FF4.

Proof. *qfr* satisfies axioms FF1-FF3 by definition. Axiom FF4 is satisfied with the following thread constructions: Create thread, Th_1 such that $Th_1(1) \models F$ and $Th_1(\Delta t) \models G$. By the definition of *qfr*, $qfr(Th_1, F, G, \Delta t) = 1$. Create thread, Th_0 such that $Th_0(1) \not\models F$ and $Th_0(\Delta t) \not\models G$. By the definition of *qfr*, $qfr(Th_0, F, G, \Delta t) = 0$.

$\square$

A.2.6 Proof of Theorem 2.20

Part (1): Let $q = Q : [t, \ell, u]$ be an annotated formula, $r = \mathsf{TRUE} \overset{qfr}{\hookrightarrow} Q : [t, \ell, u, 1, 1]$ be a constrained rule, and I be a tp-interpretation. Then, $I \models q$ iff $I \models r$.

Proof. By the definition of *qfr*, $\forall Th_i$ *such that* $qfr(Th_i, \mathsf{TRUE}, Q, t) = 1$, $Th_i(t) \models$

Q. Hence, the set of threads where $\ell \leq \mathit{qfr}(Th_i, \mathsf{TRUE}, Q, t) \leq u$ equivalent to the set of threads where $Th_i(t) \models Q$. By the definitions of satisfaction for annotated formulae and constrained rules, the statement follows. □

Part (2): Let $q = Q : [t, \ell, u]$ be an annotated formula, $r = \mathsf{TRUE} \overset{\mathit{qfr}}{\rightsquigarrow} Q : [t, \ell, u]$ be a unconstrained rule, and I be a tp-interpretation. Then, $I \models q$ iff $I \models r$.

Proof. By the definition of qfr, $\mathit{qfr}(Th, \mathsf{TRUE}, Q, t) = 0$ iff $Th(t) \not\models Q$ and $\mathit{qfr}(Th, \mathsf{TRUE}, Q, t) = 1$ iff $Th(t) \models Q$.
Hence, for all interpretations, $\sum_{Th \in \mathcal{T}, Th(t) \models Q} I(Th) = \sum_{Th \in \mathcal{T}} I(Th)\mathit{qfr}(Th, \mathsf{TRUE}, Q, t)$.
By the definitions of satisfaction for annotated formulae and constrained rules, the statement follows. □

A.2.7 Proof of Lemma 3.1

Consider the APT-Program consisting of $\{r\}$ where $r = F \overset{\mathsf{fr}}{\rightsquigarrow} G : [\Delta t, \ell, u]$.

1. If G is a tautology, then $\{r\}$ is consistent iff $u = 1$.
2. If F is a tautology and G is a contradiction, then $\{r\}$ is consistent iff $\ell = 0$.
3. If F is a contradiction, then $\{r\}$ is consistent iff $u = 1$.
4. If F is not a contradiction, G is not a tautology, and either F is not a tautology or G is not a contradiction then $\{r\}$ is consistent.

Proof. The items follow directly from Axioms FF1-FF4 respectively.

1. Suppose G is a tautology.

   By FF1, $\mathsf{fr}(Th, F, G, \Delta t)$ is 1 for all Th, Δt and F. Thus $\sum_{Th \in \mathcal{T}} I(Th)\mathsf{fr}(Th, F, G, \Delta t) = \sum_{Th \in \mathcal{T}} I(Th) = 1$. Therefore $\{r\}$ is consistent iff $u = 1$.

2. Suppose F is a tautology and G is a contradiction.

   By FF2, $\mathsf{fr}(Th, F, G, \Delta t)$ is 0 for all Th, and Δt. Thus $\sum_{Th \in \mathcal{T}} I(Th)\mathsf{fr}(Th, F, G, \Delta t) = 0$. Therefore r is consistent iff $\ell = 0$.

3. Suppose F is a contradiction.

   By FF3, $\mathsf{fr}(Th, F, G, \Delta t)$ is 1 for all Th, Δt and G. Thus $\sum_{Th \in \mathcal{T}} I(Th)\mathsf{fr}(Th, F, G, \Delta t) = \sum_{Th \in \mathcal{T}} I(Th) = 1$. Therefore $\{r\}$ is consistent iff $u = 1$.

4. Suppose F is not a contradiction, G is not a tautology, and either F is not a tautology or G is not a contradiction

   By FF4, we have Th_0 and Th_1 such that $\mathsf{fr}(Th_0, F, G, \Delta t) = 0$ and $\mathsf{fr}(Th_1, F, G, \Delta t) = 1$. Let I be the interpretation assigning probability ℓ to Th_1 and probability $1 - \ell$ to interpretation Th_0. I fr-satisfies r, thus $\{r\}$ is consistent.

□

A.2.8 Proof of Theorem 3.2

Deciding the consistency of an APT-logic program containing a single unconstrained APT-rule is NP-Complete.

Proof. **In NP**: Lemma 3.1 covers all possible cases where a single unconstrained rule $r = F \overset{\mathsf{fr}}{\rightsquigarrow} G : [\Delta t, \ell, u]$ may be consistent. In each case, there is a different witness:

1. If $\ell = 0$ and $u = 1$ then no witness is needed (such rules are always consistent).

2. If $\ell = 0$ and $u < 1$ then we need two worlds w_0 and w_1 as a witness. w_0 does not satisfy G and proves G is not a tautology (keeping part one of Lemma 3.1 from applying). w_1 satisfies F and proves F is not a contradiction. (keeping part three of Lemma 3.1 from applying). Note that either part two or part four of the lemma apply (depending on if F is a contradiction and G is a tautology or not), and in either case $\{r\}$ is consistent.

3. If $\ell > 0$ and $u = 1$ we need a world, w, which does not satisfy F or does satisfy G (keeping part two of Lemma 3.1 from applying). Note that with these assumptions, exactly one of the other parts of the lemma applies and in all cases $\{r\}$ is consistent.

4. In all other cases we have that $\ell > 0$ and $u < 1$. Here we need three worlds as the witness, w_0 which does not satisfy G, w_1 which satisfies F, and w_3 which either does not satisfy F or satisfies G. When such worlds do not exist, one of the other cases applies and enforces that $\{r\}$ is not consistent – thus such worlds always exist when $\{r\}$ is consistent. The worlds allow the application of part four of Lemma 3.1 to prove consistency.

NP-hard: By reduction from SAT. Take SAT formula F, and create annotated rule $r = F \overset{\mathsf{fr}}{\rightsquigarrow} \mathsf{FALSE} : [1, 0, 0]$. c is consistent iff F has a satisfying assignment.

($\Rightarrow$) Suppose $\{r\}$ is consistent and F has no satisfying assignment. Thus F is a contradiction and by part three of lemma 3.1, u must be 1 in order for $\{r\}$ to be consistent. But u is not 1 (it is 0), so there is a contradiction and F has a satisfying assignment.

($\Leftarrow$) Suppose F has a satisfying assignment. Then either F is a tautology, which gives that $\{r\}$ is consistent by part two of lemma 3.1; or not, in which case part three of lemma 3.1 implies $\{r\}$ is consistent. $\square$

A.2.9 Proof of Lemma 3.3

Let $\mathcal{K} = \{r = F \overset{\mathsf{fr}}{\hookrightarrow} G : [\Delta t, \ell, u, \alpha, \beta]\}$ be a constrained APT-Program consisting of a single rule. $\mathcal{K}$ is consistent iff at least one of the following conditions hold.

- $u = 1$ **and** there exists Th_{in} such that $\alpha \leq \mathsf{fr}(Th_{in}, F, G, \Delta t) \leq \beta$.
- $\ell = 0$ **and** there exists Th_{out} such that $\alpha > \mathsf{fr}(Th_{out}, F, G, \Delta t)$ or $\beta < \mathsf{fr}(Th_{out}, F, G, \Delta t)$.
- There exists both Th_{in} and Th_{out} as described above.

Proof. (1) Let Th_{in} be a thread such that $\alpha \leq \mathsf{fr}(Th_{in}, F, G, \Delta t)\beta$. Let Th_{out} be a thread such that $\alpha > \mathsf{fr}(Th_{out}, F, G, \Delta t)$ or $\beta < \mathsf{fr}(Th_{out}, F, G, \Delta t)$.

(2) By the axioms FF1-FF4 and the pigeon-hole principle, there must exist at least one of th_{in}, Th_{out}.

We have three cases: CASE 1: th_{in}, Th_{out} both exist.

Consider interpretation I such that $I(Th_{in}) = 1$. By the definition of satisfaction, $I \models r$. Therefore $\{r\}$ is consistent.

CASE 2: Only th_{in} exists.

Then, for all threads, Th, $\alpha \leq \mathsf{fr}(Th, F, G, \Delta t) \leq \beta$. So, for any satisfying interpretation, the sum of all threads Th where $\alpha \leq \mathsf{fr}(Th, F, G, \Delta t) \leq \beta$ is 1. Hence, u must equal 1, or I is not a satisfying interpretation.

CASE 3: Only th_{out} exists.

Then, for all threads, Th, $\alpha > \mathsf{fr}(Th, F, G, \Delta t)$ or $\beta < \mathsf{fr}(Th, F, G, \Delta t)$. So, for any satisfying interpretation, the sum of all threads Th where $\alpha \leq \mathsf{fr}(Th, F, G, \Delta t) \leq \beta$ is 0. Hence, ℓ must equal 0, or I is not a satisfying interpretation.

The statement follows directly from the above cases. □

A.2.10 Proof of Theorem 3.4

Deciding the consistency of an APT-logic program containing a single constrained APT-rule is NP-Complete.

Proof. **In NP:** Lemma 3.3 covers all cases where a single constrained rule $r = F \overset{\mathsf{fr}}{\hookrightarrow} G : [\Delta t, \ell, u, \alpha, \beta]$ may be consistent. In each case, there is a different witness:

- There exists Th_{in} such that $\alpha \leq \mathsf{fr}(Th_{in}, F, G, \Delta t)\beta$ and $u = 1$. Here Th_{in} is the witness.

- There exists Th_{out} such that $\alpha > \mathsf{fr}(Th_{out}, F, G, \Delta t)$ or $\beta < \mathsf{fr}(Th_{out}, F, G, \Delta t)$ and $\ell = 1$. Here Th_{out} is the witness.

- There exists both Th_{in} and Th_{out} as described above. Here, the witnesses are Th_{in} and Th_{out}.

NP-hard: By reduction from SAT. Take SAT formula F, and create annotated rule $r = F \overset{\mathsf{fr}}{\hookrightarrow} \mathsf{FALSE} : [1, 1, 1, 0, 0]$. r is consistent iff F has a satisfying assignment.

($\Rightarrow$) Suppose $\{r\}$ is consistent and F has no satisfying assignment. Thus F is a contradiction. Thus, by FF3, for all Th, $\mathsf{fr}(Th, F, \mathsf{FALSE}, \Delta t) = 1$. This is outside of the range $[\alpha, \beta]$ for the rule. Hence, Th_{in}, as described in Lemma 3.3 cannot possibly exist. Although Th_{out} does exist, as $\ell \neq 0$, $\{r\}$ is not consistent by Lemma 3.3. Therefore, we have a contradiction and F must have a satisfying assignment.

($\Leftarrow$) Suppose F has a satisfying assignment. If F is not a tautology, then we can apply FF4 and create Th_0 such that $\mathsf{fr}(Th_0, F, \mathsf{FALSE}, 1) = 0$. If F is a tautology, then all threads are Th_0 as described earlier. As $0 \in [\alpha, \beta]$ and $u = 1$, then by Lemma 3.3, we know that $\{r\}$ is consistent. $\square$

A.2.11 Proof of Lemma 3.6

If an APT-Program, $\mathcal{K} = \{r_1, \ldots, r_i, \ldots, r_n\}$, is PCD, then for any disjoint partition of rules, $\mathcal{K}_1$, $\mathcal{K}_2$, there exists a thread Th such that for all rules $r_1 \in \mathcal{K}_1$, $\mathsf{fr}_1(Th, F_1, G_1, \Delta t_1) = 1$ and for all rules $r_2 \in \mathcal{K}_2$, $\mathsf{fr}_2(Th, F_2, G_2, \Delta t_2) = 0$.

Proof. We will use the worlds w_i and $w_\emptyset$ specified in the definition of PCD. Let $max(\Delta t)$ be the maximum Δt of any rule in $\mathcal{K}$. For each rule r_i in $\mathcal{K}_2$, we set world $Th(max(\Delta t_i) \cdot (i - 1)) = w_i$. Set all other worlds in Th to $w_\emptyset$. Note that by the axioms, for all rules $r_1 \in \mathcal{K}_1$, $\mathsf{fr}_1(Th, F_1, G_1, \Delta t_1) = 1$ and for all rules $r_2 \in \mathcal{K}_2$, $\mathsf{fr}_2(Th, F_2, G_2, \Delta t_2) = 0$. $\square$

A.2.12 Proof of Theorem 3.7

For a mixed PCD APT-Program $\mathcal{K} = \{r_1, \ldots, r_i, \ldots, r_n\}$, if for all r_i, $\ell_i \leq \dfrac{|\mathcal{K}|-1}{|\mathcal{K}|}$ then $\mathcal{K}$ is consistent.

Proof. (1) For every rule $r_i \in \mathcal{K}$, let thread Th_i such $\mathsf{fr}_i(Th_i, F_i, G_i, \Delta t_i) = 0$ and $\forall j \neq i$, $\mathsf{fr}_j(Th, F_j, G_j, \Delta t_j) = 1$. These threads exists by Lemma 3.6.

(2) Let thread $Th_\emptyset$ is a thread where every world is $w_\emptyset$. This thread exists by Lemma 3.6.

(3) Let $max(\ell_i)$ be the maximum lower probability bound of all rules in $\mathcal{K}$.

(4) We create interpretation I as follows: $\forall Th_i$, $I(th_i) = \dfrac{1}{|\mathcal{K}|-1} \cdot max(\ell_i)$ and $I(Th_\emptyset) = 1 - \sum_{i=1}^{|\mathcal{K}|} I(th_i)$. For any other thread, Th, $I(Th) = 0$.

CLAIM 1: $\sum_{Th \in \mathcal{T}} = 1$

(5) By (4), the only threads that must have a non-zero probability by I are $Th_1, \ldots Th_i, \ldots, Th_{|\mathcal{K}|}$.

(6) By (4), $\sum_{i=1}^{|\mathcal{K}|} I(Th) = |\mathcal{K}| \cdot \dfrac{1}{|\mathcal{K}|-1} \cdot max(\ell_i)$.

(7) Then, by the requirement on ℓ_i in the theorem statement, $\sum_{i=1}^{|\mathcal{K}|} I(Th) \leq |\mathcal{K}| \cdot \dfrac{1}{|\mathcal{K}|-1} \cdot \dfrac{|\mathcal{K}|-1}{|\mathcal{K}|}$.

(8) Hence, $\sum_{i=1}^{|\mathcal{K}|} I(Th) \leq 1$.

(9) By (8) and (4), the claim follows.

CLAIM 2: Interpretation I satisfies all unconstrained rules in $\mathcal{K}$

(10) We will consider $r_i \in \mathcal{K}$. As $u_i = 1$, we have to show only that $\ell_i \leq$

$\sum_{Th\in\mathcal{T}} I(Th)\mathsf{fr}_i(Th, F^{(i)}, G^{(i)}, \Delta t^{(i)})$.

(11) Based on (1-2), $\sum_{j=1}^{|\mathcal{K}|} I(th_j) - I(th_i) \leq \sum_{Th\in\mathcal{T}} I(Th)\mathsf{fr}_i(Th, F^{(i)}, G^{(i)}, \Delta t^{(i)})$.

(12) Hence, by (4), $(|\mathcal{K}|-1)\cdot(\frac{1}{|\mathcal{K}|-1}\cdot max(\ell_i)) \leq \sum_{Th\in\mathcal{T}} I(Th)\mathsf{fr}_i(Th, F^{(i)}, G^{(i)}, \Delta t^{(i)})$.

(13) By (3), for all rules, $r_i \in \mathcal{K}$, $\ell_i \leq \sum_{Th\in\mathcal{T}} I(Th)\mathsf{fr}_i(Th, F^{(i)}, G^{(i)}, \Delta t^{(i)})$

(14) Therefore, by (13) and the definition of satisfaction, all unconstrained rules in $\mathcal{K}$ are satisfied by I.

CLAIM 3: Interpretation I satisfies all unconstrained rules in $\mathcal{K}$

We have two cases:

CASE 1: $\alpha = 0$

Then, $\sum_{Th\in\mathcal{T}0\leq\mathsf{fr}(Th,F,G,\Delta t)\leq 1} I(Th) = 1$ and as $u = 1$, I satisfies constrained rule r

CASE 2: $\alpha \neq 0$

Notice that for all threads, Th that I assigns a non-zero probability to, that $\mathsf{fr}(Th, F, G, \Delta t)$ is either zero or one. Hence, for all rules, $\sum_{Th\in\mathcal{T}} I(Th)\mathsf{fr}(Th, F, G, \Delta t) = \sum_{Th\in\mathcal{T}\alpha\leq\mathsf{fr}(Th,F,G,\Delta t)\leq 1} I(Th) = 1$. By the first claim, we know that for all rules $\ell_i \leq \sum_{Th\in\mathcal{T}} I(Th)\mathsf{fr}(Th, F, G, \Delta t)$, therefore, by the definition of satisfaction, and that fact that $\beta_i = 1$ for all constrained rules, we know that I satisfies all constrained rules.

$\square$

A.2.13 Proof of Proposition 3.9

For mixed APT-Logic Program $\mathcal{K}$, $\mathcal{K}$ is consistent iff $\mathsf{SLC}(\mathcal{K})$ has a solution.

Proof. ($\Rightarrow$): Let I be an interpretation satisfying $\mathcal{K}$. For each thread, Th_j, set variable $v_j = I(Th_j)$. Based on the definitions of interpretation and satisfaction, we know that for the first m lines of the linear program provide a valid solution (i.e. substituting $I(Th_j)$ for v_j for a given unconstrained rule gives $\ell_i \leq \sum_{j=1}^{n} \mathsf{fr}(Th_j, F_i, G_i, \Delta t_i) \cdot I(Th_j) \leq u_i$ which is the definition of satisfaction, substituting $I(Th_j)$ for v_j for a given constrained rule gives
$\ell_i \leq \sum_{Th_j \in \mathcal{T}_{\alpha \leq \mathsf{fr}(Th_j, F_i, G_i, \Delta t_i) \leq \beta}} I(Th_j) \leq u_i$ which is also definition of satisfaction). Based on the definition of an interpretation, we know that $\sum_{j=1}^{n} I(Th_j) = 1$, which is equivalent to the last line of the linear program.
($\Leftarrow$): Let $v_1, \ldots, v_n$ be a solution to the linear program. Let I be an interpretation where $I(Th_j) = v_j$. Based on the definitions of satisfaction, interpretation, and the lines of the linear program, I is a valid interpretation for $\mathcal{K}$. $\square$

A.2.14 Proof of Lemma 3.13

For APT-logic program $\mathcal{K}$, and $\equiv_{\mathcal{K}}$-partitioning $P_1, \cdots, P_m$ of $\mathcal{T}$, for all threads $Th, Th' \in P_i$, all $F, G \in formula(\mathcal{K})$, and all Δt

- $qfr(Th, F, G, \Delta t) = qfr(Th', F, G, \Delta t)$
- $efr(Th, F, G, \Delta t) = efr(Th', F, G, \Delta t)$
- $qfr(Th, F, G, \Delta t) = qfr(Th', F, G, \Delta t)$

Proof. In both pfr and efr, the numerator and denominator depend only on the worlds in the threads satisfied by F and G. Since F and G are in $formula(\mathcal{K})$, we know that at all time points Th and Th' either both satisfy F or both do not satisfy F (and likewise for G). Therefore exactly the same time points will be counted in the numerator and denominator of pfr and efr for both Th and Th', so the values $qfr(Th, F, G, \Delta t)$ and $qfr(Th', F, G, \Delta t)$ will be equivalent (and likewise for efr).

For qfr, we notice that other than circumstances where the value of qfr reflects the axioms, this frequency function returns 1 if F is satisfied at $Th(1)$ and G is satisfied at $Th(\Delta t)$. As F and G are also in $formula(\mathcal{K})$, we know that worlds $Th(1)$ and $Th(\Delta t)$ either satisfy or do not satisfy F and G respectively. Therefore, $qfr(Th, F, G, \Delta t) = qfr(Th', F, G, \Delta t)$. $\square$

A.2.15 Proof of Proposition 3.15

For any APT-program $\mathcal{K}$, $\mathsf{WELC}(\mathcal{K})$ is solvable iff $\mathcal{K}$ is consistent.

Proof. $\Rightarrow$ Suppose $\mathsf{WELC}(\mathcal{K})$ is solvable to show that $\mathcal{K}$ is consistent. Define interpretation I as follows: For each partition P_i, pick one $Th \in P_i$ and set $I(Th) = \hat{v}_{label(P_i)}$. For all other Th, set $I(Th) = 0$. Because of conditions 4 and 5, $\sum_{Th} I(Th) = 1$. Because of the first constraints for constrained rules, unconstrained rules, and annotated formula, I satisfies $\mathcal{K}$.

$\Leftarrow$ Suppose $\mathcal{K}$ is consistent. Let I be a satisfying interpretation. Assign a solution to $\mathsf{WELC}(\mathcal{K})$ as follows: For each P_i and lbl where $lbl = label(P_i)$, $\hat{v}_{lbl} =$

$\sum_{Th \in P_i} I(Th)$, and for any lbl where there is no P_i s.t. $label(P_i) = lbl$, $\hat{v}_{lbl} = 0$. Since I is consistent, this variable assignment will satisfy conditions relating to constrained rules, unconstrained rules, and annotated formula. The other constraints are clearly met.

□

A.2.16 Proof of Theorem 3.17

For APT-Logic Program, $\mathcal{K}$, determining the existence of an equivalence class is NP-Complete.

Proof. **NP-Hard**: Let F be a sat formula. Create program $\mathcal{K}$ consisting of annotated formula $F : [1, 1, 1]$ and let $t_{max} = 1$. There is equivalence class P_i such that $label(P_i) = 1$ iff there is a satisfying assignment for F. $\Rightarrow$: Suppose there is an equivalence class P_i, then for $Th \in P_i$, $Th(1) \models F$ and $Th(1)$ is a satisfying assignment for F. $\Leftarrow$: Suppose there is a satisfying assignment w for F, then the thread Th where $Th(1)$ has label 1.

In NP: The existence of equivalence class P_i such that $label(P_i) = lbl$ can be guaranteed by a thread Th such that $label(Th) = lbl$. This thread is the witness. □

A.2.17 Proof of Proposition 3.19

For any constrained APT-logic program $\mathcal{K}$, $\sim^{\mathcal{K}}$ is reflexive, symmetric, and transitive.

Proof. Straightforward. □

A.2.18 Proof of Theorem 3.21

For constrained APT-Logic Program $\mathcal{K}$, $\mathcal{K}$ is consistent iff there is a solution to $\mathsf{FELC}(\mathcal{K})$.

Proof. ($\Rightarrow$): Let I be an interpretation satisfying $\mathcal{K}$. Create an assignment θ where for each $E \in \mathcal{T}[\sim^{\mathcal{K}} \mathsf{fr}]$, assign $\bar{v}_{str(E)}\theta = \sum_{Th \in E} I(Th)$. That this assignment θ satisfies constraint 1 follows from the fact that $\sum_{Th \in \mathcal{T}} I(Th) = 1$ by simple algebra:

$$1 = \sum_{Th \in \mathcal{T}} I(Th) = \sum_{E \in \mathcal{T}[\sim^{\mathcal{K}} \mathsf{fr}]} \sum_{Th \in E} I(Th) = \sum_{E \in \mathcal{T}[\sim^{\mathcal{K}} \mathsf{fr}]} \bar{v}_{str(E)}$$

That this assignment θ satisfies constraint 2 follows directly form the definition of θ ($\bar{v}_{str(E)}\theta$ is zero when E is empty).

That this assignment θ satisfies constraint 3 follows from the fact that for all i,

$$\ell_i \leq \sum_{Th \in \mathcal{T}, \alpha_i \leq \mathsf{fr}(Th, F_i, G_i, \Delta t_i) \leq \beta_i} I(Th) \leq u_i$$

. Consider that due to the definition of $str(E)$:

$$\sum_{Th \in \mathcal{T}, \alpha_i \leq \mathsf{fr}(Th, F_i, G_i, \Delta t_i) \leq \beta_i} I(Th) = \sum_{E \in \mathcal{T}[\sim^{\mathcal{K}} \mathsf{fr}], str(E)_i = 1} \sum_{Th \in E} I(Th) = \sum_{E \in \mathcal{T}[\sim^{\mathcal{K}} \mathsf{fr}], str(E)_i = 1} \bar{v}_{str(E)}\theta.$$

By direct substitution, we now have that θ satisfies the last, and final, constraint.

($\Leftarrow$): Let θ be a solution to $\mathsf{FELC}(\mathcal{K})$. Construct an interpretation I where for each $E \in \mathcal{T}[\sim^{\mathcal{K}} \mathsf{fr}]$, we pick one $Th \in E$ and assign $I(Th) = \bar{v}_{str(E)}$ and all other $I(Th)$ are set to 0 (due to constraint 2, this construction is well-defined). That $\sum_{Th \in \mathcal{T}} I(Th) = 1$ follows from constraint 1. That $I \models \mathcal{K}$ follows from constraint 3 algebraically similar to the above. $\square$

A.2.19 Proof of Proposition 3.23

If a given equivalence class is empty, BFECA returns EMPTY. If there is a thread in a given equivalence class, BFECA returns OK.

Proof. CLAIM 1: If a given equivalence class is empty, BFECA returns EMPTY. Suppose by way of contradiction, that for a class, cl_s reported EMPTY by BFECA actually contains thread Th. The class cl_s is defined as follows:

$$cl_s = \left\{ \bigcap_{s_i=1} \mathsf{ATS}_i \right\} \cap \left\{ \bigcap_{s_i=0} \overline{\mathsf{ATS}}_i \right\}$$

Then, for all ATS_i such that $s_i = 1$, $Th \in \mathsf{ATS}_i$ and all $\overline{\mathsf{ATS}}_i$ such that $s_i = 0$, $Th \in \overline{\mathsf{ATS}}_i$. If such a thread existed, it would have been found in steps 1-3 of BFECA, hence a contradiction.

CLAIM 2: If there is a thread in a given equivalence class, BFECA returns OK.

Suppose by way of contradiction, that for a class, cl_s reported OK by BFECA actually does not contain a thread. The class cl_s is defined as follows:

$$cl_s = \left\{ \bigcap_{s_i=1} \mathsf{ATS}_i \right\} \cap \left\{ \bigcap_{s_i=0} \overline{\mathsf{ATS}}_i \right\}$$

Hence, there does not exist a thread, Th such that for all ATS_i such that $s_i = 1$, $Th \in \mathsf{ATS}_i$ and all $\overline{\mathsf{ATS}}_i$ such that $s_i = 0$, $Th \in \overline{\mathsf{ATS}}_i$. However, by steps 1-3 of of BFECA, at least one such thread was identified. Hence a contradiction.

The statement follows directly from claims 1-2. □

A.2.20 Proof of Theorem 58

Suppose APT-Logic program, $\mathcal{K} = \bigcup_{i=1}^{m} \{r_i\}$ where $r_i = F_i \overset{\mathsf{fr}_i}{\hookrightarrow} G_i : [\Delta t_i, \ell_i, u_i, \alpha_i, \beta_i]$ meets PCD restrictions 1-6 of Definition 3.5. Then, for all binary numbers $s \in [0,1]^m$, frequency equivalence classes $cl_s = \{\bigcap_{s_i=1} \mathsf{ATS}_i\} \cap \{\bigcap_{s_i=0} \overline{\mathsf{ATS}}_i\}$ contains at least one thread.

Proof. CLAIM: There exists at least one thread in any cl_s

Follows directly from lemma 3.6. Note that PCD restriction 7 is not used in this lemma. By the definition of the associated thread subsets, class cl_s contains at least one thread.

The statement of the theorem follows from the above claim. □

A.2.21 Proof of Corollary 15

For a constrained APT-Logic program, $\mathcal{K} = \bigcup_{i=1}^{m} \{r_i\}$ where $r_i = F_i \overset{\mathsf{fr}_i}{\hookrightarrow} G_i : [\Delta t_i, \ell_i, u_i, \alpha_i, \beta_i]$ and set of annotated formulas, $FACTS = \bigcup_{j=1}^{k} af_j$ where $af_j = Q_j : [t_j, \ell_j, u_j]$ with the following restrictions,

1. $\mathcal{K}$ meets PCD restrictions 1-4 of Definition 3.5
2. $t_{max} \geq max(t) + |\mathcal{K}| \cdot max(\Delta t_i)$
3. $\exists$ world $w_\emptyset$ such that $\forall i \in [0,m]$ and $\forall j \in [0,k]$ $w_\emptyset \not\models F_i$, $w_\emptyset \not\models G_i$, and $w_\emptyset \not\models Q_j$

4. $\forall j \in [0,k]$, there exists world qw_j such that $qw_j \models Q_j$ and $\forall i \in [0,m]$ $qw_j \not\models F_i$.

5. No two annotated formulas in $FACTS$ are at the same timepoint.

Then for all $s \in [0,1]^{m+k}$, frequency equivalence classes $cl_s = \{\bigcap_{s_i=1} \mathsf{ATS}_i\} \cap \{\bigcap_{s_i=0} \overline{\mathsf{ATS}}_i\}$ contains at least one thread.

Proof. $\forall j \in [1,k]$, we define $\mathsf{ATS}_j^{(q)}$ and $\overline{\mathsf{ATS}}_j^{(q)}$ to be the ATS and $\overline{\mathsf{ATS}}$ for rule $r = \mathsf{TRUE} \overset{qfr}{\hookrightarrow} Q_j : [t, \ell_j, u_j, 1, 1]$ created from annotated formula af_j using qfr.

CLAIM: There exists at least one thread in any cl_s

For the string $s \in [0,1]^{m+k}$, let the first m digits correspond with the m constrained rules and the last k digits correspond with the k annotated formulae.

Create a thread, Th where for any rule, r_i if $s_i = 0$, set world $Th(max(t_j) + max(\Delta t_i) \cdot (i-1)) = w_i$.

For any formulae, af_j where $s_j = 1$, set the world $Th(t) = qw_j$. Set all other worlds in Th to $w_\emptyset$. Note that by the axioms, $\forall i$, $\mathsf{fr}_i(Th, F_i, G_i, \Delta t_i) = 0$ and $\forall j \neq i$, $\mathsf{fr}_j(Th, F_j, G_j, \Delta t_j) = 1$. Further, for all annotated formulae where qw_j is at time t_j, there the thread is in $\mathsf{ATS}_j^{(q)}$. For all annotated formula qw_j where $w_\emptyset$ is at $Th(t)$, the thread is in $\overline{\mathsf{ATS}}_j^{(q)}$.

By the definition of the associated thread subsets, class cl_s contains at least one thread.

□

A.2.22 Proof of Corollary 16

Let:

- Constrained APT-Logic program, $\mathcal{K} = \bigcup_{i=1}^{m} \{r_i\}$ where $r_i = F_i \overset{\mathsf{fr}_i}{\hookrightarrow} G_i : [\Delta t_i, \ell_i, u_i, \alpha_i, \beta_i]$

- Set of annotated formulas, $FACTS = \bigcup_{j=1}^{k} af_j$ where $af_j = Q_j : [t_j, \ell_j, u_j]$

- $\forall j \in [1, k]$, we define $\mathsf{ATS}_j^{(q)}$ and $\overline{\mathsf{ATS}}_j^{(q)}$ to be the ATS and $\overline{\mathsf{ATS}}$ for rule $r = \mathsf{TRUE} \overset{qfr}{\hookrightarrow} Q_j : [t, \ell_j, u_j, 1, 1]$ created from annotated formula af_j using *qfr*

- Restrictions 1-5 from Corollary 15

Then, for all subsets of $SAMETIME \subseteq FACTS$, where for all $af_1, af_2 \in SAMETIME$, $t_1 = t_2$; for all strings $s2 \in [0, 1]^{|SAMETIME|}$ all frequency equivalence classes, *cl* that intersect $\{\bigcap_{s2_i=1} \mathsf{ATS}_i^{(q)}\} \cap \{\bigcap_{s2_i=0} \overline{\mathsf{ATS}}_i^{(q)}\}$ are non-empty iff:
$\exists$ world w_p such that for all $af_i \in SAMETIME$ where $s2_i = 1$, $w_p \models Q_i$ and for all $af_j \in SAMETIME$ where $s2_j = 0$, $w_p \not\models Q_j$.

Proof. ($\Leftarrow$) By the definition of associated thread subsets, we can create a thread *Th* where the world at time t is w_p. Hence, for all $af_i \in SAMETIME$ where $s2_i = 1$, $qfr(Th, \mathsf{TRUE}, Q_i, t_i) = 1$ and for all $af_j \in SAMETIME$ where $s2_j = 0$, $qfr(Th, \mathsf{TRUE}, Q_i, t_i) = 0$. As per Corollary 15, all other annotated formulae with different values for t and constrained rules, this thread will have the appropriate value for the corresponding frequency function. Hence, for all subsets of $SAMETIME \subseteq FACTS$, where for all $aF, aG \in SAMETIME$, $t_1 = t_2$ for all strings $s2 \in [0, 1]^{|SAMETIME|}$ all equivalence classes, *cl* that intersect

$\{\bigcap_{s2_i=1} \mathsf{ATS}_i^{(q)}\} \cap \{\bigcap_{s2_i=0} \overline{\mathsf{ATS}}_i^{(q)}\}$ are not empty.

($\Rightarrow$) Suppose by way of contradiction, we can have a thread in the equivalence class cl that intersects $\{\bigcap_{s2_i=1} \mathsf{ATS}_i^{(q)}\} \cap \{\bigcap_{s2_i=0} \overline{\mathsf{ATS}}_i^{(q)}\}$ and there does not exists a world w_p such that for all $af_i \in SAMETIME$ where $s2_i = 1$, $w_p \models Q_i$ and for all $af_j \in SAMETIME$ where $s2_j = 0$, $w_p \not\models Q_j$. Hence, a thread in such a class must have one of the following characteristics in the below three cases:

CASE 1: There exists world, w'_p where there exists $af_i \in SAMETIME$ where $s2_i = 1$, $w'_p \not\models Q_i$ and for all $af_j \in SAMETIME$ where $s2_j = 0$, $w'_p \not\models Q_j$ and for thread Th_1, $Th_1(t_i) = w'_p$.
Thread Th_1 cannot possibly be in cl as $qfr(Th_1, \mathsf{TRUE}, Q_i, t_i) = 0$ - it would have to be 1 to be in cl by the definition of associated thread subsets.

CASE 2: There exists world, w'_p such that for all $af_i \in SAMETIME$ where $s2_i = 1$, $w'_p \models Q_i$ and there exists $af_j \in SAMETIME$ where $s2_j = 0$, $w'_p \models Q_j$ and for thread Th_2, $Th_2(t_i) = w'_p$.
Thread Th_2 cannot possibly be in cl as $qfr(Th_2, \mathsf{TRUE}, Q_j, t_j) = 1$ - it would have to be 0 to be in cl by the definition of associated thread subsets.

CASE 3: There exists world, w'_p where there exists $af_i \in SAMETIME$ where $s2_i = 1$, $w'_p \not\models Q_i$ and there exists $af_j \in SAMETIME$ where $s2_j = 0$, $w'_p \models Q_j$

and for thread Th_3, $Th_3(t_i) = w'_p$.

Thread Th_2 cannot possibly be in cl for reasons described in cases 1-2.

Hence, we have a contradiction and there cannot exist a thread in class cl. The statement follows. □

A.2.23 Proof of Proposition 3.25

If a given frequency equivalence class is empty, WEFE returns EMPTY. If there is a thread in a given frequency equivalence class, WEFE returns OK.

Proof. CLAIM 1: If a given frequency equivalence class is empty, WEFE returns EMPTY.

Suppose by way of contradiction, that for a class, cl_s reported EMPTY by BFECA actually contains thread Th. The class cl_s is defined as follows:

$$cl_s = \left\{ \bigcap_{s_i=1} \mathsf{ATS}_i \right\} \cap \left\{ \bigcap_{s_i=0} \overline{\mathsf{ATS}_i} \right\}$$

By Lemma 3.13, a world-equivalence based thread partition has the same frequency function as all threads in that partition. Hence, by the definition of the set $PCLASS_s$ in steps 4-5 of WEFE, there must be a partition in set $PCLASS_s$ corresponding to a thread in class cl_s. However, that set is empty and we have a contradiction.

CLAIM 2: If there is a thread in a given frequency equivalence class, WFE returns OK.

Suppose by way of contradiction, that for a class, cl_s reported OK by BFECA actually does not contain a thread. The class cl_s is defined as follows:

$$cl_s = \left\{ \bigcap_{s_i=1} \mathsf{ATS}_i \right\} \cap \left\{ \bigcap_{s_i=0} \overline{\mathsf{ATS}}_i \right\}$$

Hence, there does not exist a thread, Th such that for all ATS_i such that $s_i = 1$, $Th \in \mathsf{ATS}_i$ and all $\overline{\mathsf{ATS}}_i$ such that $s_i = 0$, $Th \in \overline{\mathsf{ATS}}_i$. Therefore, by Lemma 3.13 and steps 4-5 of WEFE, $PCLASS_i$ must be empty. However, by the result of WEFE it is not, so we have a contradiction.

The statement follows directly from claims 1-2. $\square$

A.2.24 Proof of Theorem 4.2

Given an APT-logic program $\mathcal{K}$ and an annotated formula, af, deciding if $\mathcal{K}$ entails af is coNP-Hard.

Proof. Intuition: The proof of the above result is by a reduction from SAT.

Let $\mathcal{K}_\emptyset = \emptyset$ be an APT-logic program. Take SAT formula F and create an annotated formula $af = \neg F : [1, 1, 1]$. We say that $\mathcal{K}_\emptyset$ entails af iff F is not satisfiable.

($\Rightarrow$) Suppose BWOC, F is not satisfiable and $\mathcal{K}_\emptyset$ does not entail af. Then, there exists an interpretation I s.t. $I \models \mathcal{K}_\emptyset$ and $I \not\models af$. As F is not satisfiable, we know that for all worlds $w \in 2^{B_{\mathcal{L}}}$, $w \not\models F$. Hence, for any valid interpretation, $\sum_{Th | Th(1) \models \neg F} I(Th) = 1$. By the definition of satisfaction, interpretation $I \models af$ – which is a contradiction.

($\Leftarrow$) Suppose BWOC, F is satisfiable and $\mathcal{K}_\emptyset$ does entails af. By the definition of $\mathcal{K}_\emptyset$, $\forall I$, $\sum_{Th \in \mathcal{T}} I(th) = 1$. Based on the definition of entailment, $\forall I$, $\sum_{Th \in \mathcal{T} : th(1) \models \neg F} = 1$.

Therefore, $\forall th \rightarrow th(1) \models \neg F$ and $\forall th \rightarrow th(1) \not\models F$ hence, F is not satisfiable - a contradiction. $\square$

A.2.25 Proof of Proposition 4.3

For unconstrained rule $r = F \overset{\mathsf{fr}}{\rightsquigarrow} G : [\Delta t, \ell, u]$ or constrained rule $r = F \overset{\mathsf{fr}}{\hookrightarrow} G : [\Delta t, \ell, u, \alpha, \beta]$ and program $\mathcal{K}$, $\mathsf{SLC\text{-}ENT}$ return ENTAILS iff $\mathcal{K}$ entails r and returns NOT ENTAILS iff $\mathcal{K}$ does not entail r

Proof. To show this, we show that $\mathcal{K}$ entails r iff $[\bar{\ell'}, \bar{u'}] \subseteq [\ell, u]$.

CLAIM 1: If $\mathcal{K}$ entails r then $[\bar{\ell'}, \bar{u'}] \subseteq [\ell, u]$.

Suppose, BWOC, that $[\bar{\ell'}, \bar{u'}] \not\subseteq [\ell, u]$. Then either $\bar{\ell'}$ or $\bar{u'}$ is not in $[\ell, u]$. However, clearly there is a solution to the linear program that assigns a constraint associated with r a set of variables that sums to either $\bar{\ell'}$ or $\bar{u'}$. Hence, there is an interpretation that would assign the non-probabilistic portion of the rule r one of those numbers as a probability. Such an interpretation would not satisfy r, which would be a contradiction.

CLAIM 2: If $[\bar{\ell'}, \bar{u'}] \subseteq [\ell, u]$ then $\mathcal{K}$ entails r.

Suppose, BWOC, that $\mathcal{K}$ does not entail r. Then, there must be some interpretation that satisfies the program but not r. However, by the solution of the linear program, any probability a satisfying interpretation assigns r would fall in $[\bar{\ell'}, \bar{u'}]$ – a contradiction. $\square$

A.2.26 Proof of Proposition 81

If $\mathcal{K}$ entails r, then $\mathsf{ALC\text{-}ENT}$ returns ENTAILS. If $\mathcal{K}$ does not entail r, then $\mathsf{ALC\text{-}ENT}$ returns NOT ENTAILS.

Proof. CLAIM 1: If $\mathcal{K}$ entails r, then $\mathsf{ALC\text{-}ENT}$ returns ENTAILS.

Suppose, by way of contradiction, that there exists interpretation I that satisfies program $\mathcal{K}$ but does not satisfy rule r. If r is constrained, then $\sum_{Th \in \mathcal{T}} I(Th) \cdot \mathsf{fr}(Th, F, G, \Delta t)$ is either less than ℓ or greater than u. If r is constrained, then $\sum_{Th \in \mathcal{T}, \alpha \leq \mathsf{fr}(Th,F,G,\Delta t) \leq \beta}$ is either less than ℓ or greater than u. However, by Theorem 3.15 for WELC and Theorem 3.21 for FELC, such an interpretation cannot exist as $[\bar{\ell}, \bar{u}] \subseteq [\ell, u]$ when $\mathsf{ALC\text{-}ENT}$ returns ENTAILS. Therefore, we have a contradiction and the statement of the claim follows.

CLAIM 2: If $\mathcal{K}$ does not entail r, then $\mathsf{ALC\text{-}ENT}$ returns NOT ENTAILS.

Suppose, by way of contradiction, that all interpretations that satisfy program $\mathcal{K}$ also r. However, as there is a solution to either WELC or FELC such that either $\bar{\ell} < \ell$ or $\bar{u} > u$, then we know by Theorem 3.15 for WELC and Theorem 3.21 for FELC, that there is an interpretation that assigns the quantity $\sum_{Th \in \mathcal{T}} I(Th) \cdot \mathsf{fr}(Th, F, G, \Delta t)$ (or $\sum_{Th \in \mathcal{T}, \alpha \leq \mathsf{fr}(Th,F,G,\Delta t) \leq \beta} I(Th)$ if r is constrained) a value either less than ℓ or greater than u. Therefore, by the definition of satisfaction, there exists an interpretation that satisfies $\mathcal{K}$ and does not satisfy r. Hence, we have a contradiction and the statement of the claim follows. $\square$

A.2.27 Proof of Theorem 6.5

Theorem 59. *If an interpretation I satisfies MDP L with set of policies L, then it satisfies* APT*-Program $\mathcal{K}$ generated from* MAKE-APT*.*

Proof. Suppose, by way of contradiction, that there exists an interpretation I that satisfies (L, POL) that does not satisfy $\mathcal{K}$. Therefore, I must not satisfy one of the annotated formulas in $\mathcal{K}$. As s_1 is the initial state, obviously all I satisfying the Markov Process satisfy $F(s_1) : [1, 1, 1]$. Therefore, for some state s and time point t, $I \not\models F(s_i) : [t, min(SPM_{L,\pi}(s_t, t)), max(SPM_{L,\pi}(s_t, t))]$. Then, by the definition of satisfaction, $\sum_{Th(t) \models F(s_i)} I(Th) > min(SPM_{L,\pi}(s_i, t))$ or $\sum_{Th(t) \models F(s_i)} I(Th) > max(SPM_{L,\pi}(s_i, t))$. However, we notice that $s_1 \rightarrow^{t-1} s_i$ is the set of all prefixes for all sequences that include state s_i after $t-1$ time points. Hence, the sum of probabilities for all sequences in $s_1 \rightarrow^{t-1} s_i$ is equal to the sum of all probabilities of all sequences that include s_i after $t-1$ time points. Therefore, $\sum_{Th(t) \models F(s_i)} I(Th)$ must fall within the bounds $[min(SPM_{L,\pi}(s_i, t)), max(SPM_{L,\pi}(s_i, t))]$, which is a contradiction. The claim follows. □

A.2.28 Proof of Corollary 6.6

Corollary 17. *An interpretation I satisfies MDP L with policy π, iff it satisfies* APT*-Program $\mathcal{K}$ generated from* MAKE-APT*.*

Proof. ($\Rightarrow$): Follows directly from Theorem 6.5.

($\Leftarrow$): By the definition of satisfaction of an MDP and single policy, there exists only one I such that $I \models (L, \pi)$. We claim that there is exactly one interpretation for the constructed APT-Program, $\mathcal{K}$, and then use the pigeon hole principle to show that it is the same interpretation that satisfies (L, π). We prove this by induction on t_{max}.

Base case: If $t_{max} = 1$, then the only rule in $\mathcal{K}$ is $F(s_1) : [1, 1, 1]$ is necessary (all other annotated formulas have a t greater than t_{max}. As $F(s_1)$ is satisfied by exactly one world, and the probability bounds are both 1, and there is only one time-point, there can be only one possible interpretation.

Inductive hypothesis: Assume that $\mathcal{K}$ has only one interpretation for $t_{max} - 1$.

Inductive step: As $\mathcal{K}$ has only one interpretation for $t_{max} - 1$, only the annotated formulas where $t \leq t_{max} - 1$ are included. Let I be the interpretation that satisfies $\mathcal{K}$ for $t_{max} - 1$. Let T be the set of threads for $t_{max} - 1$.

We add all possible annotated formulas where $t = t_{max}$. Let us say that there are n such annotated formulas. We note that the regular formula in each annotated formulas is satisfied by exactly one world, and all of the formulas are satisfied by a different world. Let W be this set of worlds. Therefore, the new set of threads can have one of n possible worlds at time point t_{max}. Let T' be the new set of threads. Therefore, for each $Th \in T$, there are n number of threads in T'.

For $w \in W$, let $p(w)$ be the probability of w being the world at time t_{max}. As all annotated formulas in $\mathcal{K}$ have $\ell = u$, then there is only one possible value for $p(w)$. Note that as the ℓ value for all annotated formulas where $t = t_{max}$ is 1, then $\sum_{w \in W} p(w) = 1$. Suppose by way of contradiction that there is a thread $Th' \in T'$ that can be assigned more than one probability. However, there can be only one probability for the first $t_{max} - 1$ worlds of Th'. We shall call this initial sequence of worlds thread Th. This is interpretation I (we know this is the only possible interpretation for the first $t_{max} - 1$ worlds of Th' by the inductive hypothesis). We know the probability of a given w at time t_{max} is $p(w)$. Hence, the only probability for the thread Th' is $I(Th) \cdot p(w)$. Further, as the sum of the probabilities for all threads in T equal to 1 (based on I), and as $\sum_{w \in W} p(w) = 1$, then the sum of the probabilities for all threads in T' is 1. So, we have a contradiction, and exactly one interpretation for $\mathcal{K}$.

Therefore, as both $\mathcal{K}$ and (L, π) have exactly one satisfying interpretation, then we know by the pigeon-hole principle and Theorem 6.5 that if $I \models \mathcal{K}$ then $I \models (L, \pi)$. $\square$

Appendix B

Appendix for Chapter 3

B.1 Complexity Proofs (Section 3.3)

B.1.1 Small-Model Lemma for APT-Logic

The following lemmas are not part of the main text, but are needed to prove some of the theorems.

Let us define the "size" of a rational number $\frac{a}{b}$ (where a, b are relatively prime) as the number of bits it takes to represent a and b. As stated earlier, for both the probaiblity bound of rules, as well as the values returned by frequency functions, we assume that this is a fixed quantity. In [44], the authors provide another result we can leverage to ensure that there is a solution to a linear program where the solution can be represented with a polynomial number of bits.

Lemma 27. *If a system of r linear inequalities and/or equalities with integer coefficients of length at most l has a nonnegetive solution, then it has a nonnegetive solution with at most r entries positive, and where the size of each solution is*

$O(r \cdot l + r \cdot \log(l))$. (Lemma 2.7 in [44]).

Lemma 28. *APT-program $\mathcal{K}$ is consistent iff it has an interpretation that only assigns non-zero probabilities to at most $2 \cdot |\mathcal{K}| + 1$ threads and the probability assigned to each thread can be represented with $O(|\mathcal{K}| \cdot size + |\mathcal{K}| \cdot \log(size))$ bits (where size is the maximum number of bits required to represent the result of a freuqency function of probability bounds of a rule).*

Proof. By Proposition 3.9 of [155], an APT-program is consistent iff there is a solution to the SLC constraints. By Remark 3.10 of [155], there are $2 \cdot |\mathcal{K}| + 1$ constraints in SLC. Hence, by Theorem 9, if there is a solution to the SLC constraints, then there exists a solution where only $2 \cdot |\mathcal{K}| + 1$ are given positive values. The second part of the satement follows directly from Lemma 27. The statement of the theorem follows. □

B.1.2 Proof of Theorem 10

Deciding if APT-program $\mathcal{K}$ is consistent is NP-Complete if $|\mathcal{K}|$ is a polynomial in terms of $|B_{\mathcal{L}}|$.

Proof. NP-Hardness by Theorem 3.4 of [155]. By Lemma 28, every consistent APT-program must be associated with a set $\mathcal{T}'$ of threads, where $|\mathcal{T}'| \leq 2 \cdot |\mathcal{K}| + 1$ and that there exists an interpretation I' which only assigns non-zero probabilities to threads in $\mathcal{T}'$ and satisfies $\mathcal{K}$. Hence, we use $\mathcal{T}'$ as a witness. We can check the witness in polynomial time by setting up SLC constraints using only threads in $\mathcal{T}'$ rather than $\mathcal{T}$. By the statement, such a linear program will have a polynomial number of variables. Hence, $\mathcal{K}$ is consistent iff there is a solution to this linear program (which can be checked in PTIME). The statement follows. $\square$

B.1.3 Proof of Theorem 11

Deciding if APT-rule r is entailed by APT-program $\mathcal{K}$ is coNP-Complete if $|\mathcal{K}|$ is a polynomial in terms of $|B_{\mathcal{L}}|$.

Proof. coNP-hardness by Theorem 4.2 of [155]. Let $[\ell, u]$ be the probability bounds associated with r. Let $num \in [0, 1]$ be a real number that is outside of $[\ell, u]$. Create new rule r' that is the same as r except the probability bounds are $[num, num]$. Create APT-program $\mathcal{K}' = \mathcal{K} \cup \{r'\}$. Note that if $\mathcal{K}'$ is consistent, then r is not entailed. Hence, we can check the consistency of $\mathcal{K}'$ using a witness $\mathcal{T}'$ as described in Theorem 10 as well as num. Note that this check can still be performed in PTIME. The statement follows. $\square$

B.1.4 Proof of Theorem 12

Given APT-program $\mathcal{K}$, interpretation I, and ptf ϕ, determining the maximum ℓ and minimum u such that $\phi : [\ell, u]$ is entailed by $\mathcal{K}$ **and** is satisfied by I is $\#P$-hard. Further, for constant $\epsilon > 0$, approximating either the maximum ℓ and/or minimum u within $2^{|B_{\mathcal{L}}|^{1-\epsilon}}$ is NP-Hard.

For ease of readability, we divide the above theorem into three leammas. The statement of the theorem follows directly from Lemmas 29 and 30. Throughout the proof, we shall define the problem **APT-OPT-ENT** as follows:

APT-OPT-ENT

INPUT: APT-program $\mathcal{K}$, interpretation I, and ptf ϕ

OUTPUT: maximum ℓ and minimum u such that $\phi : [\ell, u]$ is entailed by $\mathcal{K}$ **and** is satisfied by I.

Lemma 29. ***APT-OPT-ENT*** *is #P-hard.*

Proof. Intuition Given an instance of #SAT (known to be #P-complete), we can an instance of **APT-ENT-OPT** and such that #SAT $\leq_p$ **APT-ENT-OPT**.

Definition of #SAT:

INPUT: Set of atoms $B_{\mathcal{L}}$, formula f.

OUTPUT: Number of worlds in $2^{B_{\mathcal{L}}}$ that satisfy f.

CONSTRUCTION:

1. Set F to be f.

2. Set $t = 1$.

3. For each $a \in B_{\mathcal{L}}$, add $a : [1, 0.5, 0.5]$ to $\mathcal{K}$.

4. Set $t_{max} = 1$.

5. We will consider $B_{\mathcal{L}}$ (the set of atoms from the input of #SAT) as the set of atoms used for the

6. input of **APT-ENT-OPT**.

7. Set $IC \equiv \emptyset$.

8. Interpretation $I_{uniform}$ sets each thread in $\mathcal{T}$ a probability of $\frac{1}{|\mathcal{T}|}$

For this construction, we shall denote the set of all threads formed with $t_{max} = 1$ on set of atoms $B_{\mathcal{L}}$ as $\mathcal{T}$.

As step 3 is the only step of the construction that cannot be done in constant time, but requires $O(|B_{\mathcal{L}}|)$ time, so the construction is polynomial.

CLAIM 1: Interpretation $I_{uniform}$ satisfies $\mathcal{K}$.

Each thread in $\mathcal{T}$ consists of only one world. For some atom $a \in B_{\mathcal{L}}$, half of all possible worlds satisfy a. Hence, as $I_{uniform}$ is a uniform probability distribution among threads, the sum of probabilities for all threads that satisfy a in the first (and only) time point is 0.5. By the construction of $\mathcal{K}$ in step 3 in the construction, the claim follows.

CLAIM 2: For any annotated formula $F : [t, \ell, u]$ that is entailed by $\mathcal{K}$ and satisfied by $I_{uniform}$, ℓ must equal u.

As $\mathcal{K}$ is satisfied by exactly one interpretation, $I_{uniform}$, the sum of probabilities for all threads that satisfy F at time t is bounded above and below by the same number.

CLAIM 3: If f is satisfied by exactly m worlds, then $f : [1, \frac{m}{2^{|B_{\mathcal{L}}|}}, \frac{m}{2^{|B_{\mathcal{L}}|}}]$ is entailed by $\mathcal{K}$.

Let $W_1, \ldots, W_m$ be the worlds that satisfy f. Let $Th_1, \ldots, Th_m$ be all the threads in $\mathcal{T}$ where $Th_i \equiv W_i$ (W_i is the ith world that satisfies f). As we have only one time point, and our threads are created using $B_{\mathcal{L}}$, we know that the following holds:

$$\sum_{i=1}^{m} I_{uniform}(Th_i) = \frac{m}{2^{|B_{\mathcal{L}}|}}$$

This is equivalent to the following:

$$\sum_{\substack{Th \in \mathcal{T} \\ Th(1) \models f}} I_{uniform}(Th)$$

Hence, by claims 1-2 and the definition of satisfaction, the claim follows.

CLAIM 4: If $f : [1, \frac{m}{2^{|B_{\mathcal{L}}|}}, \frac{m}{2^{|B_{\mathcal{L}}|}}]$ is entailed by $\mathcal{K}$, then f is satisfied by exactly m worlds.

By claims 1-3 and the definition of satisfaction, there are exactly m threads that satisfy f in the first time point. As there is only one time point per threads, there are also m worlds that satisfy f. Since $B_{\mathcal{L}}$ is the set of atoms for both the instance of #SAT and **APT-ENT-OPT**, the statement follows.

The proof of the theorem follows directly from claims 3-4. $\square$

Lemma 30. *For constant $\epsilon > 0$, approximating* ***APT-ENT-OPT*** *(i.e. approximating outputs ℓ and/or u) within $2^{|B_{\mathcal{L}}|^{1-\epsilon}}$ is NP-Hard.*

Proof. Suppose, by way of contradiction, that approximating a solution within $2^{|B_{\mathcal{L}}|^{1-\epsilon}}$ is easier than NP-Hard. Then, using the construction from the proof of Theorem 29, we could approximate #SAT within $2^{|B_{\mathcal{L}}|^{1-\epsilon}}$. However, by [145] (Theorem 3.2), approximating #2MONCNF, a more restricted version of #SAT, within $2^{|B_{\mathcal{L}}|^{1-\epsilon}}$ is NP-hard. The statement follows. $\square$

B.2 Supplementary Information for Section 3.4

B.2.1 Proof of Proposition 3.4.1

If $F_1 : t_1 \wedge \ldots \wedge F_n : t_n \wedge F_{n+1} : t'_1 \wedge \ldots \wedge F_{n+m} : t'_m$ and $G_1 : t_1 \wedge \ldots \wedge G_n : t_n \wedge G_{n+1} : t''_1 \wedge \ldots \wedge G_{n+m} : t''_m$ are time conjunctions, then

$$(F_1 \wedge G_1) : t_1 \wedge \ldots \wedge (F_n \wedge G_n) : t_n \wedge F_{n+1} : t'_1 \wedge \ldots \wedge F_{n+m} : t'_m \wedge G_{n+1} : t''_1 \wedge \ldots \wedge G_{n+m} : t''_m$$

is also a time conjunction.

Proof. Straightforward from the definitions of satisfaction and time conjunction. □

B.2.2 Proof of Proposition 14

For formulas F, G, time Δt, and time conjunction ϕ,

$EFR(F, G, \Delta t, \phi) \subseteq$

$$\left[\frac{cnt(\phi, F, G, \Delta t) + end(\phi, F, G, \Delta t)}{denom(\phi, F, G, \Delta t) + end(\phi, F, G, \Delta t)}, \frac{poss(\phi, F, G, \Delta t) + endposs(\phi, F, G, \Delta t)}{denom(\phi, F, G, \Delta t) + endposs(\phi, F, G, \Delta t)}\right]$$

Proof. Straightforward from definitions. □

B.2.3 Proof of Theorem 8

1. If $I \models \phi : [\ell, u]$ and $\rho : [\ell', u']$, then $I \models \phi \wedge \rho : [\max(0, \ell + \ell' - 1), \min(u, u')]$
2. If $I \models \phi : [\ell, u]$ and $\rho : [\ell', u']$, then $I \models \phi \vee \rho : [\max(\ell, \ell), \min(1, u + u')]$
3. If $I \models \phi : [\ell, u]$ and $\phi \Rightarrow \rho$ then $I \models \rho : [\ell, 1]$
4. If $I \models \phi : [\ell, u]$ and $\rho \Rightarrow \phi$ then $I \models \rho : [0, u]$

5. If $I \models \phi : [\ell, u]$ then $I \models \neg\phi : [1-u, 1-\ell]$

Proof. Adapted from Theorem 1 of [128] and Definition 32, except case 5:

Suppose, BWOC, $I \models \phi : [\ell, u]$ and $I \not\models \neg\phi : [1-u, 1-\ell]$. By the definition of satisfaction:

$$\ell \leq \sum_{\substack{Th \in \mathcal{T} \\ Th \models \phi}} I(Th) \leq u$$

By the definitoin of negation, we know that:

$$\sum_{\substack{Th \in \mathcal{T} \\ Th \models \neg\phi}} I(Th) = 1 - \sum_{\substack{Th \in \mathcal{T} \\ Th \models \phi}} I(Th)$$

Hence,

$$\ell \leq \sum_{\substack{Th \in \mathcal{T} \\ Th \models \neg\phi}} I(Th) \leq u$$

Which, by the definition of satisfaction, gives a contradiction. $\square$

B.2.4 Proof of Theorem 13

If interpretation $I \models \phi : [1, 1]$ where $EFR(F, G, \Delta t, \phi) \subseteq [\alpha, \beta]$, $I \models F \overset{efr}{\rightsquigarrow} G : [\Delta t, \alpha, \beta]$.

Proof. **CLAIM 1:** If interpreataion I satisfies $\phi : [\ell, u]$ and $EFR(F, G, \Delta t, \phi) \subseteq [\alpha, \beta]$, then $I \models F \overset{efr}{\hookrightarrow} G : [\Delta t, \ell, 1, \alpha, \beta]$.

Suppose, BWOC, there exists interpreation I s.t. $I \models \phi : [\ell, u]$ and $I \not\models F \overset{efr}{\hookrightarrow} G : [\Delta t, \ell, 1, \alpha, \beta]$. By the definition of satisfaction, we know that:

$$\ell \leq \sum_{\substack{Th \in \mathcal{T} \\ Th \models \phi}} I(Th) \leq u$$

As $EFR(F, G, \Delta t, \phi) \subseteq [\alpha, \beta]$, we know that:

$$\sum_{\substack{Th \in \mathcal{T} \\ Th \models \phi}} I(Th) \leq \sum_{\substack{Th \in \mathcal{T} \\ efr(Th,F,G,\Delta t) \in [\alpha,\beta]}} I(Th)$$

Hence,

$$\ell \leq \sum_{\substack{Th \in \mathcal{T} \\ efr(Th,F,G,\Delta t) \in [\alpha,\beta]}} I(Th) \leq 1$$

So, by the definition of satisfaction, $I \models F \overset{efr}{\hookrightarrow} G : [\Delta t, \ell, 1, \alpha, \beta]$ – a contradiction.

CLAIM 1.1: If $I \models \phi[1, 1]$, then $I \models F \overset{efr}{\hookrightarrow} G : [\Delta t, 1, 1, \alpha, \beta]$ (directly from claim 1).

CLAIM 2: If interpretation I satisfies $F \overset{efr}{\hookrightarrow} G : [\Delta t, \ell, u, \alpha, \beta]$, then $I \models F \overset{efr}{\rightsquigarrow} G : [\Delta t, \alpha \cdot \ell, 1]$. Suppose, BWOC, there exists interpreation I s.t. $I \models F \overset{efr}{\hookrightarrow} G : [\Delta t, \ell, u, \alpha, \beta]$ and $I \not\models F \overset{efr}{\rightsquigarrow} G : [\Delta t, \alpha \cdot \ell, 1]$. By the definition of satisfaction,

$$\ell \leq \sum_{\substack{Th \in \mathcal{T} \\ efr(Th,F,G,\Delta t) \in [\alpha,\beta]}} I(Th) \leq u$$

We multiply through by α:

$$\alpha \cdot \ell \leq \sum_{\substack{Th \in \mathcal{T} \\ efr(Th,F,G,\Delta t) \in [\alpha,\beta]}} \alpha \cdot I(Th)$$

It follows that:

$$\alpha \cdot \ell \leq \sum_{\substack{Th \in \mathcal{T} \\ efr(Th,F,G,\Delta t) \in [\alpha,\beta]}} \alpha \cdot I(Th) + \sum_{\substack{Th \in \mathcal{T} \\ efr(Th,F,G,\Delta t) \notin [\alpha,\beta]}} efr(Th, F, G, \Delta t) \cdot I(Th)$$

and

$$\sum_{\substack{Th \in \mathcal{T} \\ efr(Th,F,G,\Delta t) \in [\alpha,\beta]}} \alpha \cdot I(Th) \leq \sum_{\substack{Th \in \mathcal{T} \\ efr(Th,F,G,\Delta t) \in [\alpha,\beta]}} efr(Th, F, G, \Delta t) \cdot I(Th)$$

Hence, it follows that:

$$\alpha \cdot \ell \leq \sum_{Th \in \mathcal{T}} efr(Th, F, G, \Delta t) \cdot I(Th) \leq 1$$

So, by the definition of satisfaction, $I \models F \overset{efr}{\rightsquigarrow} G : [\Delta t, \alpha \cdot \ell, 1]$ – which is a contradiction. **CLAIM 2.1:** If $I \models \phi[1, 1]$, then $I \models F \overset{efr}{\rightsquigarrow} G : [\Delta t, \alpha, 1]$. (follows directly from claims 1.1 and 2).

CLAIM 3: If interpretation I satisfies $F \overset{efr}{\hookrightarrow} G : [\Delta t, 1, 1, \alpha, \beta]$, then $I \models F \overset{efr}{\rightsquigarrow} G : [\Delta t, 0, \beta]$. Suppose, BWOC, $I \models \overset{efr}{\hookrightarrow} G : [\Delta t, 1, 1, \alpha, \beta]$ and $I \not\models F \overset{efr}{\rightsquigarrow} G : [\Delta t, 0, \beta]$. By the definiton of satisfaction:

$$\sum_{\substack{Th \in \mathcal{T} \\ efr(Th, F, G, \Delta t) \in [\alpha, \beta]}} I(Th) = \sum_{Th \in \mathcal{T}} I(Th) = 1$$

Hence,

$$\sum_{Th \in \mathcal{T}} \beta \cdot I(Th) = \beta$$

We know that:

$$0 \leq \sum_{Th \in \mathcal{T}} efr(Th, F, G, \Delta t) \cdot I(Th) \leq \sum_{Th \in \mathcal{T}} \beta \cdot I(Th)$$

Which leads to:

$$0 \leq \sum_{Th \in \mathcal{T}} efr(Th, F, G, \Delta t) \cdot I(Th) \leq \beta$$

Which, by the definition of satisfaction, gives us a contradiction.

PROOF OF THEOREM: Follows directly from claims 2.1 and 3. $\square$

B.2.5 Proof of Corollary 2

If interpretation $I \models \phi : [\ell, u]$ where $EFR(F, G, \Delta t, \phi) \subseteq [\alpha, \beta]$, $I \models F \overset{efr}{\rightsquigarrow} G : [\Delta t, \alpha \cdot \ell, 1]$.

Proof. Follows directly from the first two claims of Theorem 13. □

B.2.6 Proof of Theorem 14

Given time formulas ϕ, ρ s.t. $EFR(F, G, \Delta t, \phi) \subseteq [\alpha_1, \beta_1]$ and $EFR(F, G, \Delta t, \phi \wedge \rho) \subseteq [\alpha_2, \beta_2]$ and interpretation I that satisfies $\phi : [1, 1]$ (see note[1]) and $F \overset{efr}{\rightsquigarrow} G : [\Delta t, \ell, u]$:

1. If $\beta_2 < \beta_1$, then $I \models \rho : [0, \min(\frac{\ell - \beta_1}{\beta_2 - \beta_1}, 1)]$
2. If $\alpha_2 > \alpha_1$, then $I \models \rho : [0, \min(\frac{u - \alpha_1}{\alpha_2 - \alpha_1}, 1)]$

Proof. **CLAIM 1:** Given time formulas ϕ, ρ s.t. $EFR(F, G, \Delta t, \phi) \subseteq [\alpha_1, \beta_1]$ and $EFR(F, G, \Delta t, \phi \wedge \rho) \subseteq [\alpha_2, \beta_2]$ (where $\beta_2 < \beta_1$) and interpretation I that satisfies $\phi : [1, 1]$ and $F \overset{efr}{\rightsquigarrow} G : [\Delta t, \ell, u]$ ($\ell \leq \beta_1$), $I \models \rho : [0, \min(\frac{\ell - \beta_1}{\beta_2 - \beta_1}, 1)]$.

Assume, BWOC, $I \not\models \rho : [0, \frac{\ell - \beta_1}{\beta_2 - \beta_1}]$. By the definition of satisfaction, we know that:

$$\ell \leq \sum_{Th \in \mathcal{T}} efr(Th, F, G, \Delta t) \cdot I(Th)$$

As $I \models \phi : [1, 1]$ and $EFR(F, G, \Delta t, \phi) \subseteq [\alpha_1, \beta_1]$, we have:

$$\ell \leq \sum_{\substack{Th \in \mathcal{T} \\ efr(Th, F, G, \Delta t) \in [\alpha_1, \beta_1]}} efr(Th, F, G, \Delta t) \cdot I(Th)$$

[1]Note that Theorem 13 requires $\ell \leq \beta_1$ and $\alpha_1 \leq u$

We note that all threads either satisfy ρ or not. Hence, we have:

$$\sum_{\substack{Th \in \mathcal{T} \\ efr(Th,F,G,\Delta t) \in [\alpha_1,\beta_1] \\ Th \models \rho}} I(Th) + \sum_{\substack{Th \in \mathcal{T} \\ efr(Th,F,G,\Delta t) \in [\alpha_1,\beta_1] \\ Th \not\models \rho}} I(Th) = 1$$

Therefore:

$$\ell \leq \sum_{\substack{Th \in \mathcal{T} \\ efr(Th,F,G,\Delta t) \in [\alpha_1,\beta_1] \\ Th \models \rho}} \beta_2 \cdot I(Th) + \sum_{\substack{Th \in \mathcal{T} \\ efr(Th,F,G,\Delta t) \in [\alpha_1,\beta_1] \\ Th \not\models \rho}} \beta_1 \cdot I(Th)$$

and:

$$\ell \leq \beta_2 \cdot \sum_{\substack{Th \in \mathcal{T} \\ efr(Th,F,G,\Delta t) \in [\alpha_1,\beta_1] \\ Th \models \rho}} I(Th) + \beta_1 \cdot (1 - \sum_{\substack{Th \in \mathcal{T} \\ efr(Th,F,G,\Delta t) \in [\alpha_1,\beta_1] \\ Th \models \rho}} I(Th))$$

$$\ell - \beta_1 \leq \beta_2 \cdot \sum_{\substack{Th \in \mathcal{T} \\ efr(Th,F,G,\Delta t) \in [\alpha_1,\beta_1] \\ Th \models \rho}} I(Th) - \beta_1 \cdot \sum_{\substack{Th \in \mathcal{T} \\ efr(Th,F,G,\Delta t) \in [\alpha_1,\beta_1] \\ Th \models \rho}} I(Th))$$

Notice that $\ell - \beta_1 \leq 0$ as $\ell \leq \beta_1$ by the statement. Also, we know that $\beta_2 < \beta_1$, the quantity $\beta_2 - \beta_1$ is negative. We have the following:

$$\frac{\ell - \beta_1}{\beta_2 - \beta_1} \geq \sum_{\substack{Th \in \mathcal{T} \\ Th \models \rho}} I(Th)$$

By the definition of satisfaction, this gives us a contradiction.

CLAIM 2: Given time formulas ϕ, ρ s.t. $EFR(F, G, \Delta t, \phi) \subseteq [\alpha_1, \beta_1]$ and $EFR(F, G, \Delta t, \phi \wedge \rho) \subseteq [\alpha_2, \beta_2]$ $(\alpha_2 > \alpha_1)$ and interpretation I that satisfies $\phi : [1, 1]$ and $F \overset{efr}{\rightsquigarrow} G : [\Delta t, \ell, u]$ $(\alpha_1 \leq u$ or inconsistent), $I \models \rho : [0, \min(\frac{u-\alpha_1}{\alpha_2-\alpha_1}, 1)]$.

Assume, BWOC, $I \not\models \rho : [0, \frac{u-\alpha_1}{\alpha_2-\alpha_1}]$. By the definition of satisfaction, we know that:

$$\sum_{Th \in \mathcal{T}} efr(Th, F, G, \Delta t) \cdot I(Th) \leq u$$

Hence, as all threads either satisfy ρ or not, and as $I \models \phi : [1,1]$, we know that all threads must also have a α_1 lower bound for the frequency function, and that the threads satisfying ρ must have α_2 as a lower bound. So, we have the following:

$$\sum_{\substack{Th \in \mathcal{T} \\ Th \models \rho}} \alpha_2 \cdot I(Th) + \sum_{\substack{Th \in \mathcal{T} \\ Th \not\models \rho}} \alpha_1 \cdot I(Th) \leq u$$

As we know the sum of all theads must be 1, we have the following:

$$\alpha_2 \cdot \sum_{\substack{Th \in \mathcal{T} \\ Th \models \rho}} I(Th) + \alpha_1 \cdot (1 - \sum_{\substack{Th \in \mathcal{T} \\ Th \models \rho}} I(Th)) \leq u$$

$$(\alpha_2 - \alpha_1) \cdot \sum_{\substack{Th \in \mathcal{T} \\ Th \models \rho}} I(Th) \leq u - \alpha_1$$

As, by the statement, we know the quantities $\alpha_2 - \alpha_1$ and $u - \alpha_1$ are positive, we have the following:

$$\sum_{\substack{Th \in \mathcal{T} \\ Th \models \rho}} I(Th) \leq \frac{u - \alpha_1}{\alpha_2 - \alpha_1}$$

Which, by the definition of satisfaction, gives us a contradiction.

Proof of theorem: Follows directly from claims 1-2. □

B.2.7 Proof of Proposition 15

If for atoms A_i and program $\mathcal{K}$, if $\mathsf{BLK}(A_i) :< \mathsf{blk}_i \in \mathcal{K}$ and if there exists a ptf $\phi : [1,1] \in \mathcal{K}$ such that $\phi \Rightarrow A_i : t - \mathsf{blk}_i + 1 \wedge A_i : t - \mathsf{blk}_i + 2 \wedge \ldots \wedge A_i : t - 1$ then $\mathcal{K}$ entails $A : t : [0,0]$.

Proof. Suppose, BWOC, there exists interpretation I s.t. $I \models \mathcal{K}$ and $I \not\models A : t : [0, 0]$. As $I \models \mathcal{K}$, we know $I \models \mathsf{BLK}(A_i) :< \mathsf{blk}_i$. Hence, for all therads s.t. $I(Th) \neq 0$, there does not exist a series of blk_i or more consecutive worlds in Th satisfying atom A_i. We note that as $I \models \phi : [1, 1]$, then $I \models A_i : t - \mathsf{blk}_i + 1 \wedge A_i : t - \mathsf{blk}_i + 2 \wedge \ldots \wedge A_i : t - 1 : [1, 1]$ by the statement. Hence, there is a sequence of $\mathsf{blk}_i - 1$ consecutive worlds satisfying A_i in every thread assigned a non-zero probability by I. So, by the definition of satisfaction, we have a contradiction. $\square$

B.2.8 Proof of Proposition 16

If for atoms A_i and program $\mathcal{K}$, if $\mathsf{OCC}(A_i) : [\mathsf{lo}_i, \mathsf{up}_i] \in \mathcal{K}$ and if there exists a ptf $\phi : [1, 1] \in \mathcal{K}$ such that there are numbers $t_1, \ldots, t_{\mathsf{up}_i} \in \{1, \ldots, t_{max}\}$ where $\phi \Rightarrow A_i : t_1 \wedge \ldots \wedge A_i : t_{\mathsf{up}_i}$ then for any $t \notin \{t_1, \ldots, t_{\mathsf{up}_i}\}$ $\mathcal{K}$ entails $A : t : [0, 0]$.

Proof. Suppose, BWOC, there exists interpretation I s.t. $I \models \mathcal{K}$ and $I \not\models A : t : [0, 0]$. As $I \models \mathcal{K}$, we know $I \models \mathsf{OCC}(A_i) : [\mathsf{lo}_i, \mathsf{up}_i]$. Hence, for all therads s.t. $I(Th) \neq 0$, there does not exist more than up_i worlds in Th satisfying atom A_i. We note that as $I \models \phi : [1, 1]$, then $I \models A_i : t_1 \wedge \ldots \wedge A_i : t_{\mathsf{up}_i} : [1, 1]$ by the statement. Hence, there are up_i worlds satisfying A_i in every thread assigned a non-zero probability by I. So, by the definition of satisfaction, we have a contradiction.

$\square$

B.2.9 Proof of Proposition 17

Given APT-program $\mathcal{K}$, the following are true:

- $\forall I$ s.t. $I \models \mathcal{K}$, $I \models \Gamma(\mathcal{K})$
- $\forall I$ s.t. $I \models \Gamma(\mathcal{K})$, $I \models \mathcal{K}$

Proof. Follows directly from Theorems 13-14 and Corollary 2. □

B.2.10 Proof of Proposition 18

One iteration of Γ can be performed in time complexity $O(|\mathcal{K}|^2 \cdot CHK)$ where CHK is the bound on the time it takes to check (for arbitrary time formulas ϕ, ρ if $\phi \models \rho$ is true.

Proof. To compare a given element of $\mathcal{K}$ with every other element (not conjuncts of elements) - we obviously need $O(|\mathcal{K}| \cdot CHK)$ time. As we do this for every element in $\mathcal{K}$, the statement follows. □

B.2.11 Proof of Lemma 9

Given $\bot \equiv \{\}$ and $\top \equiv$ *inconsistent*, then $\langle PROG_{B_{\mathcal{L}},t_{max}}, \sqsubseteq \rangle$ is a complete lattice.

Proof. We must show that for any subset $PROG'$ of $PROG_{B_{\mathcal{L}},t_{max}}$, that $inf(PROG')$ and *sup*$(PROG')$ exist in $PROG_{B_{\mathcal{L}},t_{max}}$. We show this for $PROG_{B_{\mathcal{L}},t_{max}}$ as a set of APT-programs, and the result obviously extends for $PROG_{B_{\mathcal{L}},t_{max}}$ as a set of equivalence classes of APT-programs.

CLAIM 1: For a set $PROG'$ of APT-programs, $inf(PROG')$ exists and is in $PROG_{B_{\mathcal{L}},t_{max}}$.

Let $PROG' = \{\mathcal{K}_1, \ldots, \mathcal{K}_i, \ldots, \mathcal{K}_n\}$. We create $\mathcal{K}' \equiv inf(PROG')$ as follows. Consider all ϕ such that $\phi : [\ell_i, u_i]$ appears in each $\mathcal{K}_i$. Add $\phi : [\min(\ell_i), \max(u_i)]$ to $\mathcal{K}'$. Next, consider all $F, G, \Delta t$ s.t. $F \overset{efr}{\rightsquigarrow} G : [\Delta t, \ell_i, u_i]$ appears in all $\mathcal{K}_i$. Add $F, G, \Delta t$ s.t. $F \overset{efr}{\rightsquigarrow} G : [\Delta t, \min(\ell_i), \max(u_i)]$ to $\mathcal{K}'$. Clearly, for each element in $\mathcal{K}'$, there is an element in every $\mathcal{K}_i$ with the same or tighter probability bounds. It is also obvious tha $\mathcal{K}' \in PROG_{B_{\mathcal{L}}, t_{max}}$. Assume that there is a $\mathcal{K}''$ (not equivalent to $\mathcal{K}'$) that is below each $\mathcal{K}_i$ but above $\mathcal{K}'$. Then, for all elements in $\mathcal{K}'$, there must be a corresponding element (with tighter probability bounds) in $\mathcal{K}''$ s.t. the probability bounds is looser than any $\mathcal{K}_i$. However, by the construction, this is clearly not possible unless $\mathcal{K}' \equiv \mathcal{K}''$, so we have a contradiction.

CLAIM 2: For a set $PROG'$ of APT-programs, $sup(PROG')$ exists and is in $PROG_{B_{\mathcal{L}}, t_{max}}$.

Let $PROG' = \{\mathcal{K}_1, \ldots, \mathcal{K}_i, \ldots, \mathcal{K}_n\}$. Let $\mathcal{K}' = \bigcup_i \{\mathcal{K}_i\}$. Clearly, by the definition of $\sqsubseteq$, this is a least upper bound of $PROG'$. We must show that $\mathcal{K}'$ is in $PROG_{B_{\mathcal{L}}, t_{max}}$. We have two cases. (1) If $\mathcal{K}'$ is inconsistent, then it is equivalent to $\top$ and in $PROG_{B_{\mathcal{L}}, t_{max}}$. (2) If $\mathcal{K}'$ is consistent, then it is also in $PROG_{B_{\mathcal{L}}, t_{max}}$. $\square$

B.2.12 Proof of Lemma 10

$\mathcal{K} \sqsubseteq \Gamma(\mathcal{K})$.

Proof. Follows directly from the definition of Γ - all rules and ptf's in $\mathcal{K}$ are in $\Gamma(\mathcal{K})$ with equivalent or tighter probability bounds. All IC's in $\mathcal{K}$ remain in $\Gamma(\mathcal{K})$. $\square$

B.2.13 Proof of Lemma 11

Γ is monotonic.

Proof. Given $\mathcal{K}_1 \sqsubseteq \mathcal{K}_2$, we must show $\Gamma(\mathcal{K}_1) \sqsubseteq \Gamma(\mathcal{K}_2)$. Suppose, BWOC, there exists $\phi : [\ell, u] \in \Gamma(\mathcal{K}_1)$ (see note [2]) s.t. there does not exist $\phi : [\ell', u'] \in \Gamma(\mathcal{K}_2)$ where $[\ell', u'] \subseteq [\ell, u]$. Therefore, there must exist a set of ptf's and/or rules (call this set $\mathcal{K}_1'$) in $\mathcal{K}_1$ s.t. for each element in $\mathcal{K}_1'$, there does not exist a an element in $\mathcal{K}_2$ s.t. the probability bounds are tighter. However, as $\mathcal{K}_1 \sqsubseteq \mathcal{K}_2$, this cannot be possible, and we have a contradiction. $\square$

B.2.14 Proof of Theorem 15

Γ has a least fixed point.

Proof. Follows directly from Lemma 10 and Lemma 11. $\square$

B.2.15 Proof of Lemma 12

If APT-logic program $\mathcal{K}$ entails rule $F \overset{efr}{\rightsquigarrow} G : [\Delta t, \ell, u]$ or $\phi : [\ell, u]$ such that one of the following is true:

- $\ell > u$
- $\ell < 0$ or $\ell > 1$
- $u < 0$ or $u > 1$

[2]Resp. $F \overset{efr}{\rightsquigarrow} G : [\Delta t, \ell, u] \in \Gamma(\mathcal{K}_1)$, we note that the proof can easily be mirrored for rules, we only show with ptfs here.

Then $\mathcal{K}$ is **inconsistent** - i.e. there exists no interpretation I such that $I \models \mathcal{K}$.

Proof. Following directly from the definitions of satisfaciton and entailment, if $\mathcal{K}$ entails such a rule or ptf, there can be no satisfying interpreation. □

B.2.16 Proof of Theorem 4

For APT-logic program $\mathcal{K}$, if there exists natural number i such that $\Gamma(\mathcal{K}) \uparrow i$ that contains rule $F \overset{efr}{\rightsquigarrow} G : [\Delta t, \ell, u]$ or $\phi : [\ell, u]$ such that one of the following is true:

- $\ell > u$
- $\ell < 0$ or $\ell > 1$
- $u < 0$ or $u > 1$

Then $\mathcal{K}$ is **inconsistent**.

Proof. We know by Propositions 17 that any number of applications of Γ result in an APT-program entailed by $\mathcal{K}$. Therefore, all of the elemenets of that program must be entailed by $\mathcal{K}$. By Lemma 12, the statement follows. □

B.2.17 Proof of Proposition 19

If there does not exist at least one thread that satisfies all integrity constraints in an APT-logic program, then that program is inconsistent.

Proof. For an APT-logic program to be consistent, then there must exist a satisfying interpretation such that the sum of the probabilities assigned to all threads is 1.

However, if there is no thread that satisfies all integrity constraints, then the sum of the probabilities of all threads in a satisfying interpretation is 0 – a contradiction. $\square$

B.2.18 Proof of Proposition 20

If $\mathsf{lo}_i > \left\lceil \frac{(\mathsf{blk}_i - 1) \cdot t_{max}}{\mathsf{blk}_i} \right\rceil$ then there does not exist a partial thread for ground atom A_i such that the single block-size and occurrence IC associated with A_i hold.

Follows directly from the following Proposition:

Proposition 82. *For atom a_i, block size blk_i and t_{max}, if more than $\left\lceil \frac{(\mathsf{blk}_i - 1) \cdot t_{max}}{\mathsf{blk}_i} \right\rceil$ worlds must be true, then all partial threads will have a block of size blk_i.*

Proof. CLAIM 1: If we require less than (or equal) $\left\lceil \frac{(\mathsf{blk}_i - 1) \cdot t_{max}}{\mathsf{blk}_i} \right\rceil$ worlds to satisfy the atom, there exists at least one partial thread that does not contain a block.
Simply consider $\mathsf{blk}_i - 2$ sub-sequences of $\mathsf{blk}_i - 1$ worlds, and one sub-sequence of $\leq \mathsf{blk}_i - 1$ worlds satisfying atom a_i - each separated by a world that does not satisfy a_i. Obviously, this partial thread does not contain a block.

CLAIM 2: If we require more than $\left\lceil \frac{(\mathsf{blk}_i - 1) \cdot t_{max}}{\mathsf{blk}_i} \right\rceil$ worlds to satisfy the atom, there can be no sequence of two consecutive worlds that do not satisfy a_i, or there exists a block.
This follows from the pigeon hole principle - if two consecutive worlds satisfy $\neg a_i$, then there must exists a sequence of at least blk_i worlds that satisfy a_i.

PROOF OF PROPOSITION: Suppose we have a partial thread with $\left\lceil \frac{(\mathsf{blk}_i - 1) \cdot t_{max}}{\mathsf{blk}_i} \right\rceil$

worlds satisfying the atom, and require one additional world to satisfy a_i. By claim 2, this world must be between two sub-sequences, as there are no more than two non-satisfying worlds, hence the statement of the proposition follows. $\square$

B.2.19 Proof of Propositon 21

For ground atom A_i with (with associated ICs), if $\mathsf{up}_i > \left\lceil \frac{(\mathsf{blk}_i - 1)\cdot t_{max}}{\mathsf{blk}_i} \right\rceil$ we know that for numbers of worlds satisfying A_i cannot be in the range $\left[\left\lceil \frac{(\mathsf{blk}_i - 1)\cdot t_{max}}{\mathsf{blk}_i} \right\rceil, up_i\right]$.

Proof. As, in this case, $\mathsf{up}_i > \left\lceil \frac{(\mathsf{blk}_i - 1)\cdot t_{max}}{\mathsf{blk}_i} \right\rceil$, lowering the value of up_i will not cause an inconsistency unless Proposition 20 applies. We note that by Proposition 82, we cannot have threads with more than this amount of worlds satisfying a_i. $\square$

B.2.20 Proof of Proposition 22

ThEX can be solved in $O(1)$.

Proof. As the check in Proposition 20 can be performed in $O(1)$ time, the statement follows. $\square$

B.3 Proofs for Section 3.5

B.3.1 Proof of Lemma 13

Given non-ground formulas F_{ng}, G_{ng}, time Δt, and non-ground time formula ϕ_{ng}. Let $(\alpha_{in}, \beta_{in}) = EFR\_IN(F_{ng}, G_{ng}, \Delta t, \phi_{ng})$ and $[\alpha_{out}, \beta_{out}] = EFR\_OUT(F_{ng}, G_{ng}, \Delta t, \phi_{ng})$. Then the following holds true:

1. If $Th \models \phi_{ng}$, then for all ground instances F, G of F_{ng}, G_{ng} we have $\mathit{efr}(F, G, \Delta t, Th) \in [\alpha_{out}, \beta_{out}]$
2. If $Th \models \phi_{ng}$, then there exists ground instances F, G of F_{ng}, G_{ng} we have $\mathit{efr}(F, G, \Delta t, Th) \geq \alpha_{in}$
3. If $Th \models \phi_{ng}$, then there exists ground instances F, G of F_{ng}, G_{ng} we have $\mathit{efr}(F, G, \Delta t, Th) \leq \beta_{in}$

Proof. CLAIM 1: Part 1 is true..

Suppose, BWOC, there is some thread, $Th \models \phi_{ng}$ s.t. there are ground instances F, G of F_{ng}, G_{ng} s.t. $\mathit{efr}(F, G, \Delta t, Th) \notin [\alpha_{out}, \beta_{out}]$. This directly contradicts Definition 48.

CLAIM 2: Part 2 is true.

This directly contradicts Definition 48.

CLAIM 3: Part 3 is true.

This directly contradicts Definition 48. □

B.3.2 Proof of Theorem 16

Given non-ground APT-program $\mathcal{K}^{(ng)}$ that contains the following:

$$\textbf{Non-ground rule:} \quad F_{ng} \overset{efr}{\rightsquigarrow} G_{ng} : [\Delta t, \ell, u]$$

$$\textbf{Non-ground ptf:} \quad \phi_{ng} : [1, 1]$$

Let $(\alpha_{in}, \beta_{in}) = EFR\_IN(F_{ng}, G_{ng}, \Delta t, \phi_{ng})$. If we are given $\alpha_{in}^{-} \leq \alpha_{in}$ and $\beta_{in}^{+} \geq \beta_{in}$, then, $\mathcal{K}^{(ng)}$ is not consistent if one (or both) of the following is true:

1. $\alpha_{in}^{-} > u$

2. $\beta_{in}^{+} < \ell$

Proof. CLAIM 1: If $\alpha_{in}^{-} > u$, then $\mathcal{K}^{(ng)}$ is not consistent.

Suppose, BWOC that $\alpha_{in}^{-} > u$ and $\mathcal{K}^{(ng)}$ is consistent. Then, by Lemma 13 there exists ground instances F, G of F_{ng}, G_{ng} s.t. $\mathit{EFR}(F, G, \Delta t, gnd(\phi_{ng})) \subseteq [\alpha_{in}^{-}, 1]$. Therefore, by Theorem 13, $\mathcal{K}^{(ng)}$ entails $F \overset{efr}{\rightsquigarrow} G : [\Delta t, \alpha_{in}^{-}, 1]$. However, as $\mathcal{K}^{(ng)}$ includes $F_{ng} \overset{efr}{\rightsquigarrow} G_{ng} : [\Delta t, \ell, u]$, then $\mathcal{K}^{(ng)}$ also entails $F \overset{efr}{\rightsquigarrow} G : [\Delta t, \ell, u]$. As $[\alpha_{in}^{-}, 1] \cap [\ell, u] = \emptyset$, we know that $\mathcal{K}^{(ng)}$ cannot be consistent (by Lemma 12) – a contradiction.

CLAIM 2: If $\beta_{in}^{+} < \ell$, then $\mathcal{K}^{(ng)}$ is not consistent.

Suppose, BWOC, that $\beta_{in}^{+} < \ell$ and $\mathcal{K}^{(ng)}$ is consistent. Then, by Lemma 13 there exists ground instances F, G of F_{ng}, G_{ng} s.t. $\mathit{EFR}(F, G, \Delta t, gnd(\phi_{ng})) \subseteq [0, \beta_{in}^{+}]$. Therefore, by Theorem 13, $\mathcal{K}^{(ng)}$ entails $F \overset{efr}{\rightsquigarrow} G : [\Delta t, 0, \beta_{in}^{+}]$. However, as $\mathcal{K}^{(ng)}$ includes $F_{ng} \overset{efr}{\rightsquigarrow} G_{ng} : [\Delta t, \ell, u]$, then $\mathcal{K}^{(ng)}$ also entails $F \overset{efr}{\rightsquigarrow} G : [\Delta t, \ell, u]$. As $[0, \beta_{in}^{+}] \cap [\ell, u] = \emptyset$, we know that $\mathcal{K}^{(ng)}$ cannot be consistent (by Lemma 12) – a contradiction.

□

B.3.3 Proof of Corollary 5

Given non-ground APT-program $\mathcal{K}^{(ng)}$ that contains the following:

$$\textbf{Non-ground rule:} \quad F_{ng} \overset{efr}{\leadsto} G_{ng} : [\Delta t, \ell, u]$$

$$\textbf{Non-ground ptf:} \quad \phi_{ng} : [\ell', u']$$

Let $(\alpha_{in}, \beta_{in}) = EFR\_IN(F_{ng}, G_{ng}, \Delta t, \phi_{ng})$. If we are given $\alpha_{in}^{-} \leq \alpha_{in}$ and $\beta_{in}^{+} \geq \beta_{in}$, then, $\mathcal{K}^{(ng)}$ is not consistent if $\alpha_{in}^{-} \cdot \ell' > u$.

Proof. Suppose, BWOC, $\alpha_{in}^{-} \cdot \ell' > u$ and $\mathcal{K}^{(ng)}$ is consistent. Then, by Lemma 13 there exists ground instances F, G of F_{ng}, G_{ng} s.t. $EFR(F, G, \Delta t, gnd(\phi_{ng})) \subseteq [\alpha_{in}^{-}, 1]$. Therefore, by Corollary 2, $\mathcal{K}^{(ng)}$ entails $F \overset{efr}{\leadsto} G : [\Delta t, \alpha_{in}^{-} \cdot \ell', 1]$. However, as $\mathcal{K}^{(ng)}$ includes $F_{ng} \overset{efr}{\leadsto} G_{ng} : [\Delta t, \ell, u]$, then $\mathcal{K}^{(ng)}$ also entails $F \overset{efr}{\leadsto} G : [\Delta t, \ell, u]$. As $[\alpha_{in}^{-} \cdot \ell', 1] \cap [\ell, u] = \emptyset$, we know that $\mathcal{K}^{(ng)}$ cannot be consistent (by Lemma 12) – a contradiction. $\square$

B.3.4 Proof of Proposition 23

If the list returned by NG-INCONSIST-CHK contains any elements, then $\mathcal{K}^{(ng)}$ is not consistent.

Proof. Follows directly from Theorem 16 and Corollary 5. $\square$

B.3.5 Proof of Proposition 24

NG-INCONSIST-CHK performs $O(|\mathcal{K}^{(ng)}|^2)$ comparisons.

Proof. The algorithm consists of two nested loops. The outer loop considers all ptf's in the program – requiring $O(|\mathcal{K}^{(ng)}|)$ time, while the inner loop considers all rules in the program – also requiring $O(|\mathcal{K}^{(ng)}|)$ time. The statement follows. □

B.3.6 Proof of Lemma 14

$\mathcal{K} \subseteq \Lambda_{\mathcal{K}^{(ng)}}(\mathcal{K})$ wrt $\langle PROG_{B_{\mathcal{L}},t_{max}}, \sqsubseteq \rangle$

Proof. Follows directly from Definition 49. □

B.3.7 Proof of Lemma 15

$\Lambda_{\mathcal{K}^{(ng)}}$ is monotonic.

Proof. Given $\mathcal{K}_1 \sqsubseteq \mathcal{K}_2$ (both ground), we must show $\Lambda_{\mathcal{K}^{(ng)}}(\mathcal{K}_1) \sqsubseteq \Lambda_{\mathcal{K}^{(ng)}}(\mathcal{K}_2)$. Suppose, BWOC, there is an element (rule, ptf, or IC) of $\Lambda_{\mathcal{K}^{(ng)}}(\mathcal{K}_1)$ that either has a tighter probability bound than a corresponding element in $\Lambda_{\mathcal{K}^{(ng)}}(\mathcal{K}_2)$ or not in $\Lambda_{\mathcal{K}^{(ng)}}(\mathcal{K}_2)$. However, this is a contradiction as all elements in $\mathcal{K}_1$ are in $\mathcal{K}_2$ – or in $\mathcal{K}_2$ with a tighter probability bound. Therefore, such an element would be in $\Lambda_{\mathcal{K}^{(ng)}}(\mathcal{K}_2)$ – a contradiction. □

B.3.8 Proof of Theorem 17

$\Lambda_{\mathcal{K}^{(ng)}}$ has a least fixed point.

Proof. Follows directly from Lemma 14 and Lemma 15. □

B.3.9 Proof of Lemma 16

Given non-ground program $\mathcal{K}^{(ng)}$, and ground program $\mathcal{K}$, $lfp(\Lambda_{\mathcal{K}^{(ng)}}(\mathcal{K})) \subseteq ground(\mathcal{K}^{(ng)}) \cup \mathcal{K}$.

Proof. Suppose, BWOC, that $lfp(\Lambda_{\mathcal{K}^{(ng)}}(\mathcal{K})) \not\subseteq ground(\mathcal{K}^{(ng)}) \cup \mathcal{K}$. Then, there must exist a ground rule, ptf, or IC in element in $lfp(\Lambda_{\mathcal{K}^{(ng)}}(\mathcal{K}))$ that is not in $ground(\mathcal{K}^{(ng)}) \cup \mathcal{K}$. However, all elements in $lfp(\Lambda_{\mathcal{K}^{(ng)}}(\mathcal{K}))$ are either elements of $\mathcal{K}$ or ground instances of elements in $\mathcal{K}^{(ng)}$ – hence a contradiction. □

B.3.10 Proof of Theorem 18

Definition 110 (Tightening)**.** *For APT-rule $F \overset{efr}{\rightsquigarrow} G : [\Delta t, \ell, u]$ or ptf $\phi : [\ell, u]$, for any $[\ell', u'] \subseteq [\ell, u]$,*

1. *$F \overset{efr}{\rightsquigarrow} G : [\Delta t, \ell', u']$ is a **tightening** of $F \overset{efr}{\rightsquigarrow} G : [\Delta t, \ell, u]$*
2. *$\phi : [\ell, u]$ is a **tightening** of $\phi : [\ell', u']$*

Definition 111 (Update)**.** *Given ground APT-program $\mathcal{K}$, ground rule $r = F \overset{efr}{\rightsquigarrow} G : [\Delta t, \ell_1, u_1]$, and ground ptf $p = \phi : [\ell_2, u_2]$, any tightening to the bounds of r or p causes by an application of the operator Γ is an **update**.*

Definition 112 (Update Widget)**.** *Given ground APT-program $\mathcal{K}$, ground rule $r = F \overset{efr}{\rightsquigarrow} G : [\Delta t, \ell_1, u_1]$, and ground ptf $p = \phi : [\ell_2, u_2]$, ground atomic time formula $A : t$, we define the following **update widgets**.*

1. *Let the ground rule $r' = F \overset{efr}{\rightsquigarrow} G : [\Delta t, \ell', u']$ be a tightening of r where $\ell' = \mathbf{l\_bnd}(F, G, \Delta t, \mathcal{K})$ or $u' = \mathbf{u\_bnd}(F, G, \Delta t, \mathcal{K})$. Then an **update widget***

*consists of a graph of a vertex $v_{r'}$ for r' (called a **top vertex**) and set V of vertices - one vertex for each ground rule and ptf in $\mathcal{K}$ that led to the tightening (as per Definition 41) (called **bottom vertices**) and directed edges from all elements in V to $v_{r'}$.*

2. *Let the ground ptf $p' = \phi : [\ell', u']$ be a tightening of $\phi : [\ell_2, u_2]$ where $\ell' \in \{\textsf{l\_bnd}(\phi, \mathcal{K}), 1 - \textsf{u\_bnd}(\neg\phi, \mathcal{K})\}$ or $u' \in \{\textsf{u\_bnd}(\phi, \mathcal{K}), 1 - \textsf{l\_bnd}(\neg\phi, \mathcal{K})\}$. Then an **update widget** consists of a graph of a vertex $v_{p'}$ for p' (called a **top vertex**) and set V of vertices - one vertex for each ground rule and ptf in $\mathcal{K}$ that led to the tightening (as per Definition 41) (called **bottom vertices**) and directed edges from all elements in V to $v_{p'}$.*

3. *If $\mathcal{K}$ entails $A : t : [0, 0]$ due to the presence of ptf's and IC's (as per Propositions 15-16), then Then an **update widget** consists of a graph of a vertex $v_{A:t:[0,0]}$ for $A : t : [0, 0]$ (called a **top vertex**) and set V of vertices - one for each IC and ptf in $\mathcal{K}$ that led to the entailment of $A : t : [0, 0]$ (called **bottom vertices**) and directed edges from all elements in V to $v_{r'}$.*

Definition 113 (Deduction Tree)**.** *A series of update widgets with the top vertices of all but one widgets are the bottom vertices for another widget is called a **deduction tree**. A vertex that is not a bottom vertex for any widget in the tree is a **root** and a vertex that is not top vertex for any widget in the tree is a **leaf**. For a given deduction tree, T, let $leaf(T)$ be the set of ptf's or rules corresponding with leaf nodes in the tree.*

Definition 114 (Corresponding Deduction Tree)**.** *Given ground* APT*-program* $\mathcal{K}$*, for ground ptf* $p = \phi : [\ell_2, u_2]$*, s.t.* $p \in lfp(\Gamma(\mathcal{K}))$*, then the* ***corresponding deduction tree*** *is a deduction tree, rooted in a node representing* p *s.t. for each update performed by* Γ*, there is a corresponding update widget in the tree. For program* $\mathcal{K}$ *and ptf* p*, let* $T_{\mathcal{K},p}$ *be the corresponding deduction tree.*

Lemma 31. *If* $\phi : [\ell, u] \in lfp(\Gamma(\mathcal{K} \cup \{\phi : [0,1]\})$ *then there exists* $\phi : [\ell', u'] \in lfp(\Gamma(leaf(T_{\mathcal{K},\phi:[\ell,u]}) \cup \{\phi : [0,1]\})$ *s.t.* $[\ell', u'] \subseteq [\ell, u]$*.*

Proof. Suppose, BWOC, that $[\ell', u'] \not\subseteq [\ell, u]$. Then, there must exist an update performed by Γ that uses some ptf or rule *other* $\in \mathcal{K}$ s.t. *other* $\notin leaf(T_{\mathcal{K},\phi:[\ell,u]})$. However, by the Definition 114 this is not possible as $T_{\mathcal{K},\phi:[\ell,u]}$ accounts for all updates performed by Γ. □

Theorem 18

Given non-ground program $\mathcal{K}^{(ng)}$

$$\phi : [\ell, u] \in lfp(\Gamma(lfp(\Lambda_{\mathcal{K}^{(ng)}}(\{\phi : [0,1]\}))))$$

iff

$$\phi : [\ell, u] \in lfp(\Gamma(ground(\mathcal{K}^{(ng)}) \cup \{\phi : [0,1]\}))$$

Proof. CLAIM 1: If $\phi : [\ell, u] \in lfp(\Gamma(lfp(\Lambda_{\mathcal{K}^{(ng)}}(\{\phi : [0,1]\}))))$ then for some $[\ell', u'] \subseteq [\ell, u]$, $\phi : [\ell', u'] \in lfp(\Gamma(ground(\mathcal{K}^{(ng)}) \cup \{\phi : [0,1]\}))$.

By Lemma 16, we know that $lfp(\Lambda_{\mathcal{K}^{(ng)}}(\{\phi : [0,1]\})) \subseteq ground(\mathcal{K}^{(ng)}) \cup \{\phi : [0,1]\}$, so the claim follows.

CLAIM 2: If $\phi : [\ell, u] \in lfp(\Gamma(ground(\mathcal{K}^{(ng)}) \cup \{\phi : [0,1]\}))$ then for some $[\ell', u'] \subseteq [\ell, u]$, $\phi : [\ell, u] \in lfp(\Gamma(lfp(\Lambda_{\mathcal{K}^{(ng)}}(\{\phi : [0,1]\}))))$.

By Definition 49 and Definition 114, $leaf(T_{\mathcal{K},\phi:[\ell,u]}) \cup \{\phi : [0,1]\} \subseteq lfp(\Lambda_{\mathcal{K}^{(ng)}}(\{\phi : [0,1]\}))$. Hence, we can apply Lemma 31 and the claim follows.

The statement of the theorem follows directly from claims 1-2. $\square$

B.4 Supplemental Information for Section 3.6

B.4.1 Proof of Proposition 25

OC-EXTRACT runs in time $O((n - t_{max}) \cdot t_{max})$.

Proof. This follows directly from the two for loops in the algorithm - the first iterating $(n - t_{max})$ time and a nested loop iterating t_{max} times. $\square$

B.4.2 Proof of Proposition 26

There are no historical threads such that atom a_i is satisfied by less than lo_i or more than up_i worlds when $\mathsf{lo}_i, \mathsf{up}_i$ are produced by OC-EXTRACT.

Proof. Suppose, by way of contradiction, that there exists a historical thread that does not meet the constraints. As we examine all possible historical threads in OC-EXTRACT and take the minimum and maximum number of times a_i is satisfied over all these threads, we have a contradiction. $\square$

B.4.3 Proof of Proposition 27

BLOCK-EXTRACT runs in time $O(n)$.

Proof. Follows directly from the for loop in the algorithm - which iterates n times.

$\square$

B.4.4 Proof of Proposition 28

Given blk_i as returned by BLOCK-EXTRACT, there is no sequence of blk_i or more consecutive historical worlds that satisfy atom a_i.

Proof. Suppose there is a sequence of at least blk_i or more. However, the algorithm maintains the variable *best* which is the greatest number of consecutive time points in the historical data where a_i is true – this is a contradiction. $\square$

Appendix C

Appendix for Chapter 4

C.1 Proofs

C.1.1 Proof of Theorem 19

k-**SEP** is NP-Complete.

Proof. **Geometric Covering by Discs.** (GCD)

INPUT: A set P of integer-coordinate points in a Euclidean plane, positive integers $b > 0$ and $k < |P|$.

OUTPUT: "Yes" if there exist k discs of diameter b centered on points in P such that there is a disc covering each point in P — "no" otherwise.

CLAIM 1: k-**SEP** is in the complexity class NP.

Suppose a non-deterministic algorithm can guess a set $\mathcal{E}$ that is a k-sized simple (α, β) explanation for $\mathcal{O}$. We can check the feasibility of every element in $\mathcal{E}$ in $O(|\mathcal{E}|)$ time and compare every element of $\mathcal{E}$ to every element of $\mathcal{O}$ in $O(|\mathcal{O}|^2)$ time. Hence, k-**SEP** is in the complexity class NP as we can check the solution in poly-

Algorithm 25 (GCD-TO-KSEP)

INPUT: Instance of GCD $\langle S, P, b, k\rangle$

OUTPUT: Instance of k-**SEP** $\langle \mathcal{S}, \mathcal{O}, \mathsf{feas}, \alpha, \beta, k'\rangle$

```
1. Set $\mathcal{S}$ to be a set of lattice points in the Euclidean plane that include all points in $P$
2. Set $\mathcal{O} = P$
3. Let $feas(x) = \mathsf{TRUE}$ iff $x \in P$
4. Set $\alpha = 0$
5. Set $\beta = b/2$
6. Set $k' = k$
```

nomial time.

CLAIM 2: k-**SEP** is NP-Hard.

We use the polynomial algorithm GCD-TO-KSEP to take an instance of GCD and create an instance of k-**SEP**.

CLAIM 2.1: If there is a k'-sized simple (α, β) explanation for $\mathcal{O}$, then there are k discs, each centered on a point in P of diameter b that cover all points in P.

Let $\mathcal{E}$ be the k'-sized simple (α, β) explanation for $\mathcal{O}$. Suppose by way of contradiction, that there are not k discs, each centered on a point in P of diameter b that cover all points in P. As $k' = k$, and all elements of $\mathcal{E}$ must be in P by the definition of feas, let us consider the k discs of diameter b centered on each element of $\mathcal{E}$. So, for these discs to not cover all elements of P, there must exist an element of P, that is not covered by a disc. As $P \equiv \mathcal{O}$, then there must exist an element of $\mathcal{O}$ outside of one of the discs. Note that all elements of $\mathcal{O}$ are within a distance β of an element of $\mathcal{E}$ by the definition of a k'-sized simple (α, β) explanation (as $\alpha = 0$). As $\beta = b/2$, each element of $\mathcal{O}$ falls inside a disc of diameter b centered on an element of $\mathcal{E}$, thus falling within a disc and we have a contradiction.

CLAIM 2.2: If there are k discs, each centered on a point in P of diameter b that cover all points in P then there is a k'-sized simple (α, β) explanation for $\mathcal{O}$.

Let set E be the set of points that are centers of the k discs. We note that $E \subseteq P$. Assume by way of contradiction, that there is no k'-sized simple (α, β) explanation for $\mathcal{O}$. Let us consider if E is a k'-sized simple (α, β) explanation for $\mathcal{O}$. As $k = k'$, $\alpha = 0$, and all points of E are feasible, there must be some $o \in \mathcal{O}$ such that $\forall e \in E$, $d(e, o) > \beta$. As $\mathcal{O} \equiv P$, we know that all points in $\mathcal{O}$ fall in a disc centered on a point in E, hence each $o \in \mathcal{O}$ must be a distance of $b/2$ or less from a point in E. As $\beta = b/2$, we have a contradiction. □

C.1.2 Proof of Corollary 6

Cost-based Explanation is NP-Complete.

Proof. CLAIM 1: Cost-based Explanation is in the complexity class NP.

This follows directly from Theorem 19, instead of checking the size of $\mathcal{E}$, we only need to apply the function χ to the $\mathcal{E}$ produced by the non-deterministic algorithm to ensure that $\chi(\mathcal{E}) \leq v$.

CLAIM 2: Cost-based Explanation is NP-Hard.

We show k-**SEP**$\leq_p$ CBE. Given an instance of k-**SEP**, we transform it into an instance of CBE in polynomial time where $\chi(\mathcal{E}) = |\mathcal{E}|$ and $v = k$.

CLAIM 2.1: If there is a set $\mathcal{E}$ such that $\chi(\mathcal{E}) \leq v$ then $|\mathcal{E}| \leq k$.

Straightforward.

CLAIM 2.2: If there is a set $\mathcal{E}$ of size k or less then $\chi(\mathcal{E}) \leq v$

Straightforward. □

C.1.3 Proof of Corollary 7

WT-SEP is NP-Complete.

Proof. Membership in the complexity class NP follows directly from Theorem 19, instead of checking the size of $\mathcal{E}$, we check if $\sum_{p\in\mathcal{E}} c(p) \leq v$. We also note that the construction for cost-based explanation in Theorem 19 is also an instance of WT-SEP, hence NP-hardness follows immediately.

□

C.1.4 Proof of Theroem 20

TD-SEP is NP-Complete.

Proof. CLAIM 1: TD-SEP is in the complexity class NP.

Given a set $\mathcal{E}$, we can easily determine in polynomial time that it meets the standards of the output specified in the problem statement.

CLAIM 2: TD-SEP is NP-hard.

Consider Euclidean k-Median Problem, as presented and shown to be NP-Complete in [134], defined as follows:

INPUT: A set P of integer-coordinate points in a Euclidean plane, positive integer $k' < |P|$, real number $v' > 0$.

OUTPUT: "Yes" if there is a set of points, $S \subseteq P$ such that $|S| = k'$ and $\sum_{x_i\in X} \min_{s_j\in S} d(x_i, s_j) \leq v'$ — "no" otherwise.

Given an instance of the Euclidean k-Median Problem, we create an instance of TD-SEP as follows:

- Set $\mathcal{S}$ to be a set of lattice points in the Euclidean plane that include all points in P
- Set $\mathcal{O} = P$
- Let $feas(x) = \mathsf{TRUE}$ iff $x \in P$
- Set $\alpha = 0$
- Set β greater than the diagonal of $\mathcal{S}'$
- Set $k = k'$
- Set $v = v'$

CLAIM 2.1: If there is $\mathcal{E}$, a k-sized explanation for $\mathcal{O}$ such that $\sum_{o_i \in \mathcal{O}} \min_{p_j \in \mathcal{E}} d(o_i, p_j) \leq v$, then there is a set $S \subseteq P$ such that $|S| = k'$ and $\sum_{x_i \in P} \min_{s_j \in S} d(x_i, s_j) \leq v'$.

Because of how we set feas and $\mathcal{O}$, $\mathcal{E} \subseteq P$. As α and β do not affect $\mathcal{E}$, the only real restrictions on $\mathcal{E}$ is that its cardinality is k and that $\sum_{o_i \in \mathcal{O}} \min_{p_j \in \mathcal{E}} d(o_i, p_j) \leq v$. Because of how we set k and v, we can see that $\mathcal{E}$ meets all the conditions to be a solution to the Euclidean k-Median problem, hence the claim follows.

CLAIM 2.2: If there is set $S \subseteq P$ such that $|S| = k'$ and $\sum_{x_i \in P} \min_{s_j \in S} d(x_i, s_j) \leq v'$, then there is set $\mathcal{E}$, a k-sized explanation for $\mathcal{O}$ such that $\sum_{o_i \in \mathcal{O}} \min_{p_j \in \mathcal{E}} d(o_i, p_j) \leq v$.

In the construction, the arguments α, β and feas allow any element of a solution to the k-Median problem to be a partner for any observation in $\mathcal{O}$. By how we set k and v, we can easily see that S is a valid solution to TD-SEP. The claim follows.

The statement of the theorem follows directly from claims 1-2. □

C.1.5 Proof of Proposition 29

If there is a k-sized simple (α, β) explanation for $\mathcal{O}$, then NAIVE-KSEP-EXACT returns an explanation. Otherwise, it returns NO.

Proof. CLAIM 1: If there is a k-sized simple (α, β) explanation for $\mathcal{O}$, then NAIVE-KSEP-EXACT returns an explantion.

Suppose, by way of contradiction, that there is a k-sized simple (α, β) explanation for $\mathcal{O}$ and NAIVE-KSEP-EXACT returns NO. Then there does not exist k bit strings such that for all o_i, $\sum_{j=1}^{k}(\ell_j(i)) \geq 1$. As each bit string is associated with a point in $\mathcal{S}$, then by the construction of the bit strings, there are not k points in $\mathcal{S}$ such that each point is feasible and falls no closer than α and no further than β distance away from each point in $\mathcal{O}$. This is a contradiction.

CLAIM 2: If there is no k-sized simple (α, β) explanation for $\mathcal{O}$, then NAIVE-KSEP-EXACT returns NO.

Suppose, by way of contradiction, that there is no k-sized simple (α, β) explanation for $\mathcal{O}$ and NAIVE-KSEP-EXACT returns an explanation. Then there must exist k

bit strings such that $\bigvee_{j=1}^{k}(\ell_j(i)) = 1$. As each bit string is associated with a point in $\mathcal{S}$, then by the construction of the bit strings, there must exist k points in $\mathcal{S}$ such that each point is feasible and falls no closer than α and no further than β distance away from each point in $\mathcal{O}$. This is a contradiction.

□

C.1.6 Proof of Proposition 30

The complexity of NAIVE-KSEP-EXACT is $O(\frac{1}{(k-1)!}(\pi(\beta^2 - \alpha^2)|\mathcal{O}|)^{(k+1)})$.

Proof. Note that as all pointers in M are initially null, thus there is no need to iterate through every element in M - rather lists in M can only be initialized as needed. Hence, the cost to set-up M in $O(1)$ and not the size of the matrix.

As each $o \in \mathcal{O}$ has, at most $\pi(\beta^2 - \alpha^2)$ partners, the total complexity of the inner loop is $\pi(\beta^2 - \alpha^2)|\mathcal{O}|$.

As we have, at most, $\pi(\beta^2 - \alpha^2)|\mathcal{O}|$ elements in L (recall that L is the subset of $\mathcal{S}$ that can be partnered with elements in $\mathcal{O}$), then there are $\binom{\pi(\beta^2-\alpha^2)|\mathcal{O}|}{k}$ iterations taking place in step 5. Each iteration costs $k \cdot |\mathcal{O}|$ as we must compare the $|\mathcal{O}|$ bits of each k bit string. So,

$$\binom{\pi(\beta^2 - \alpha^2)|\mathcal{O}|}{k} \cdot k \cdot |\mathcal{O}|$$
$$= \frac{(\pi(\beta^2 - \alpha^2)|\mathcal{O}|) \cdot (\pi(\beta^2 - \alpha^2)|\mathcal{O}| - 1) \cdot \ldots \cdot (\pi(\beta^2 - \alpha^2)|\mathcal{O}| - (k-1))}{k!} \cdot k \cdot |\mathcal{O}|$$
$$< O(\frac{1}{(k-1)!}(\pi(\beta^2 - \alpha^2)|\mathcal{O}|)^{(k+1)})$$

As this term dominates the complexity of the inner loop, the statement follows. □

C.1.7 Proof of Theorem 21

k-**SEP**$\leq_p$ SET_COVER

Proof. We employ the first four steps of NAIVE-KSEP-EXACT. We view the bit-strings in list L as subsets of $\mathcal{O}$ where if the ith bit of the string is 1, o_i of $\mathcal{O}$ is in the set.

CLAIM 1: If there are k subsets of L that cover $\mathcal{O}$, then there is a k-sized simple (α, β) explanation for $\mathcal{O}$.
Suppose, by way of contradiction, that there are k subsets of L that cover $\mathcal{O}$ and there is no k-sized simple (α, β) explanation for $\mathcal{O}$. Then, by Proposition 29, for every combination of k bit strings, there is some bit i such that $\bigvee_{j=1}^{k}(\ell_j(i)) = 1$ does not hold. Hence, by the reduction, a set cover with k sets from L would be impossible. This is a contradiction.

CLAIM 2: If there there is a k-sized simple (α, β) explanation for $\mathcal{O}$, then there are k subsets of L that cover $\mathcal{O}$.
Suppose, by way of contradiction, there is a k-sized simple (α, β) explanation for $\mathcal{O}$ and there are not k subsets of L that cover $\mathcal{O}$. Then, for any combination of k subsets of L, there is at least one element of $\mathcal{O}$ not included. Hence, for any bit-string representation of an element in L, for some bit i, $\bigvee_{j=1}^{k}(\ell_j(i)) = 1$ does not hold. However, by Proposition 29, this must hold or there is no k-sized simple (α, β) explanation for $\mathcal{O}$. This is a contradiction. □

C.1.8 Proof of Proposition 31

NAIVE-KSEP-SC has a complexity of $O(\Delta \cdot f \cdot |\mathcal{O}|^2)$ and an approximation ratio of $1 + ln(f)$.

Proof. CLAIM 1: NAIVE-KSEP-SC has a complexity of $O(\Delta \cdot f \cdot |\mathcal{O}|^2)$.

The loop at line 3, which reduces the problem to set-covering, takes $O(\Delta \cdot |\mathcal{O}|)$ time.

The loop at line 4 iterates, at most, $|\mathcal{O}|$ times.

The first nested loop at line 4b iterates, at most, $\Delta \cdot |\mathcal{O}|$ times.

The second nested loop at line 4(b)ii iterates, at most, f times.

The updating procedure at line 4d, which is still inside the loop at line 4, iterates, at most, f times.

Hence, by the above statements, the total complexity of NAIVE-KSEP-SC is $O(|\mathcal{O}| \cdot (\Delta \cdot |\mathcal{O}| \cdot f + f) + \Delta \cdot |\mathcal{O}|)$, hence the statement follows.

CLAIM 2: NAIVE-KSEP-SC has an approximation ratio of $1 + ln(f)$.

Viewing list L as a family of subsets, each subset is the set of observations associated with a potential partner, hence the size of the subsets is bounded by f. The approximation ratio follows directly from the analysis of the set-covering problem. □

C.1.9 Proof of Proposition 32

A solution $\mathcal{E}$ to NAIVE-KSEP-SC provides a partner to every observation in $\mathcal{O}$ if a partner exists.

Proof. Follows directly from Theorem 21. □

C.1.10 Proof of Proposition 33

The complexity of KSEP-TO-DOMSET is $O(\Delta \cdot |\mathcal{O}|)$.

Proof. Notice that the number of points in $\mathcal{S}$ considered for each $o \in \mathcal{O}$ examined in the inner loop is bounded by $O(\Delta)$. As the outer loop is bounded by the size of $\mathcal{O}$, the complexity of KSEP-TO-DOMSET is $O(|\mathcal{O}|)$. □

C.1.11 Proof of Theorem 22

k-**SEP**$\leq_p$ **DomSet**.

Proof. We can run KSEP-TO-DOMSET that creates graph $G_{\mathcal{O}} = (V_{\mathcal{O}}, E_{\mathcal{O}})$ based on the set of observations. We show that $G_{\mathcal{O}}$ has a dominating set of size k iff there is a k-sized simple (α, β) explanation for $\mathcal{O}$.

CLAIM 1: If $G_{\mathcal{O}}$ has a dominating set of size k or less, then there is a k-sized (or less) simple (α, β) explanation for $\mathcal{O}$.
Suppose, by way of contradiction, that $G_{\mathcal{O}}$ has a dominating set of size k and there is not a k-sized simple (α, β) explanation for $\mathcal{O}$. Then, there has to be at least one element $o_i \in \mathcal{O}$ such that there is no feasible $p \in \mathcal{S}$ where $\alpha \leq d(o_i, p) \leq \beta$. Consider the nodes V_i from the inner loop of KSEP-TO-DOMSET that are associated with o_i. Note that these nodes form a complete subgraph. As each node in V_i is associated with o_i, no node in V_i can be in the dominating set of $G_{\mathcal{O}}$ (if one were, then we would have a contradiction). However, note that half of the nodes in V_i only have edges to other nodes in V_i, so there must be an element of V_i in the dominating set.

This is a contradiction.

CLAIM 2: If there is a k-sized simple (α, β) explanation for $\mathcal{O}$, then $G_{\mathcal{O}}$ has a dominating set of size k or less.

Suppose, by way of contradiction, that there is a k-sized simple (α, β) explanation for $\mathcal{O}$, and $G_{\mathcal{O}}$ has does not have a dominating set of size k or less. Let $\mathcal{E}$ be a k-sized simple (α, β) explanation for $\mathcal{O}$. Let this also be a subset of the nodes in $G_{\mathcal{O}}$. By the KSEP-TO-DOMSET, in each set of nodes V_i, there must be at least one element of $\mathcal{E}$. As each set of vertices V_i is a complete graph, then we have a dominating set of size k. Hence, a contradiction.

$\square$

C.1.12 Proof of Proposition 34

Solving k-**SEP** by a reduction to **DomSet** utilizing a straight-forward greedy approach has time-complexity $O(\Delta^3 \cdot f \cdot |\mathcal{O}|^2)$ and an approximation ratio bounded by $O(1 + \ln(2 \cdot f \cdot \Delta))$.

Proof. This is done by a well-known reduction of an instance of **DomSet** into an instance of SET_COVER. In the reduction, each node is an element, and the subsets are formed by each node and its neighbors. The Table C.1 shows the quantities:

Hence, the total time complexity of the algorithm is $O(8 \cdot \Delta^3 \cdot f \cdot |\mathcal{O}|^2)$ and the complexity part of the statement follows. As the maximum number of elements per subset, the approximation ratio $O(1 + \ln(2 \cdot f \cdot \Delta))$ follows by the well-known analysis of the greedy set-covering algorithm. $\square$

Item	Quantity
Number of elements to be covered (number of nodes in $G_{\mathcal{O}}$)	$2 \cdot \Delta \cdot \lvert\mathcal{O}\rvert$
Number of subsets (number of nodes in $G_{\mathcal{O}}$)	$2 \cdot \Delta \cdot \lvert\mathcal{O}\rvert$
Number of elements per subset (Maximum degree of nodes in $G_{\mathcal{O}}$ determined by the produce of partners per observation and observations per partner	$2 \cdot \Delta \cdot f$

Table C.1: Quantities for the Greedy-Approach in the **DomSet** reduction.

C.1.13 Proof of Proposition 35

Solving k-**SEP** by a reduction to **DomSet** utilizing the distributed, randomized algorithm presented in [75] has a time complexity $O(\Delta \cdot |\mathcal{O}| + ln(2 \cdot \Delta \cdot |\mathcal{O}|) \cdot ln(2 \cdot \Delta \cdot f))$ with high probability and approximation ratio of $O(1 + \ln(2 \cdot f \cdot \Delta))$.

Proof. By Proposition 33, the complexity of KSEP-TO-DOMSET is $O(\Delta \cdot |\mathcal{O}|))$. The graph $G_{\mathcal{O}}$ has $O(2 \cdot \Delta \cdot |\mathcal{O}|)$ nodes, and the maximum degree of each node is bounded $2 \cdot \Delta \cdot f$ as per Proposition 34. As the algorithm in [75] has a complexity of $O(lg(n) \cdot lg(d))$ (with high probability) where n is the number of nodes and d is the maximum degree, the complexity of this approach requires $O(\Delta \cdot |\mathcal{O}| + ln(2 \cdot \Delta \cdot |\mathcal{O}|) \cdot ln(2 \cdot \Delta \cdot f))$ with high probability (the statement follows).

As the approach in [75] is greedy, it maintains the $O(1+\ln(2 \cdot f \cdot \Delta))$ (Proposition 34) (the approximation ratio in this case being a factor of the optimal in expectation). □

C.1.14 Proof of Proposition 36

OPT-KSEP-IPC consists of $O(|\mathcal{O}|\pi(\beta^2 - \alpha^2))$ variables and $1+|\mathcal{O}|$ constraints.

Proof. Follows directly from Definition 56. □

C.1.15 Proof of Proposition 37

For a given instance of the optimization version k-**SEP**, if OPT-KSEP-IPC is solved, then $\bigcup_{p_j \in L_{x_j=1}} p_j$ is an optimal solution to k-**SEP**.

Proof. Suppose, by way of contradiction, that $\bigcup_{p_j \in L_{x_j=1}} p_j$ is not an optimal solution to k-**SEP**. By the constraint, $\forall o_i \in \mathcal{O}$, $\sum_{p_j \in L} x_j \cdot str(p_j)_i \geq 0$, we are ensured that for each observation, there is a partner p_j such that $x_j = 1$. Further, if we associate x_j with the selected parter p_j for any solution $\mathcal{E}$ to k-**SEP**, then this constraint must hold. Hence, $\bigcup_{p_j \in L_{x_j=1}} p_j$ is a valid explanation. Therefore, the optimal solution to the instance of k-**SEP**, we shall call $\mathcal{E}_{OPT}$, must be smaller than $\bigcup_{p_j \in L_{x_j=1}} p_j$. As the minimization of $\sum_{p_j \in L} x_j$ ensures that the cardinality of $\bigcup_{p_j \in L_{x_j=1}} p_j$ is minimized. Therefore, $|\mathcal{E}_{OPT}|$ cannot be smaller than $|\bigcup_{p_j \in L_{x_j=1}} p_j|$, as the constraint $\forall o_i \in \mathcal{O}$, $\sum_{p_j \in L} x_j \cdot str(p_j)_i \geq 0$ holds for any solution to k-**SEP**. This is a contradiction. □

C.1.16 Proof of Proposition 38

NAIVE-KSEP-ROUND returns an explanation for $\mathcal{O}$ that is within a factor f of optimal, where f is the maximum number of possible partners associated with any observation.

Proof. [68] shows that the solution to the relaxation of the integer program representation of set-cover approximates the optimal solution within a factor of f, which is the greatest number of sets an element can be found in. For k-**SEP**, this would be the greatest number of partners for any given observation, which is bounded by $O(\pi(\beta^2 - \alpha^2))$, but may be much lower in practice. As OPT-KSEP-IPC employs this technique, the statement follows directly. □

C.1.17 Proof of Proposition 39

GREEDY-KSEP-OPT1 has a complexity of $O(\Delta \cdot f \cdot |\mathcal{O}|)$ and an approximation ratio of $1 + ln(f)$.

Proof. CLAIM 1: GREEDY-KSEP-OPT1 has a complexity of $O(\Delta \cdot f \cdot |\mathcal{O}|)$.
This follows the same analysis of NAIVE-KSEP-SC in Proposition 31, except that line 4 iterates only Δ times rather than $\Delta \cdot |\mathcal{O}|$ times. Hence, the total complexity is $O(|\mathcal{O}| \cdot (\Delta \cdot f + f) + \Delta \cdot |\mathcal{O}|)$ and the statement follows.

CLAIM 2: GREEDY-KSEP-OPT1 has an approximation ratio of $1 + ln(f)$.
The proof of this claim resembles the approximation proof of the standard greedy algorithm for set-cover (i.e. see [28] page 1036).

Let $p_1, \ldots, p_i, \ldots, p_n$ be the elements of $\mathcal{E}$, the solution to GREEDY-KSEP-OPT1, numbered by the order in which they were selected. For each iteration, let set COV_i be the subset of observations that are partnered for the first time with point p_i. Note that each element of $\mathcal{O}$ is in exactly one COV_i. For each $o_j \in \mathcal{O}$, we define $cost_j$ to be $\frac{1}{|COV_i|}$ where $o_j \in COV_i$.

CLAIM 2.1: $\sum_{p_i \in \mathcal{E}^*} \sum_{o_j \in \mathcal{O}_{p_i, o_j \text{ are partners}}} cost_j \geq |\mathcal{E}|$

By the definition of $cost_j$, exactly one unit of cost is assigned every time a point is picked for the solution $\mathcal{E}$. Hence,

$$COST(\mathcal{E}) = |\mathcal{E}| = \sum_{o_j \in \mathcal{O}} cost_j$$

The statement of the claim follows.

CLAIM 2.2: For some point $p \in L$, $\sum_{o_j \in \mathcal{O}_{p, o_j \text{ are partners}}} cost_j \leq 1 + \ln(f)$.

Let P be the subset of $\mathcal{O}$ that can be partners with p. At each iteration i of the algorithm, let $uncov_i$ be the number of elements in P that do not have a partner. Let $last$ be the smallest number such that $uncov_{last} = 0$. Let $\mathcal{E}_P = \{p_i \in \mathcal{E} | (i \leq last) \wedge (COV_i \cap P \not\equiv \emptyset)\}$. From here on, we shall renumber each element in $\mathcal{E}_P$ as $p_1, \ldots, p_{|\mathcal{E}_P|}$ by the order they are picked in the algorithm (i.e. if an element is picked that cannot partner with anything in P, we ignore it and continue numbering with the next available number, we will use this new numbering for COV_i and the iterations of the algorithm as well, but do not re-define the set based on the new numbering).

We note that for each iteration i, the number of items in P that are partnered is equal to $uncov_{i-1} - uncov_i$. Hence,

$$\sum_{\substack{o_j \in \mathcal{O} \\ p, o_j \text{ are partners}}} cost_j = \sum_{i=1}^{last} \frac{uncov_{i-1} - uncov_i}{|COV_i|}$$

At each iteration of the algorithm, let $PCOV_i$ be the subset of observations that are covered for the first time if point p is picked instead of point p_i. We note, that for all iterations in $1, \ldots, last$, the point p is considered by the algorithm as one of its options for greedy selection. Therefore, as p is not chosen, we know that $|COV_i| \geq |PCOV_i|$. Also, by the definition of $ucov_i$, we know that $|PCOV_i| = ucov_{i-1}$. This gives us:

$$\sum_{\substack{o_j \in \mathcal{O} \\ p, o_j \text{ are partners}}} cost_j \leq \sum_{i=1}^{last} \frac{uncov_{i-1} - uncov_i}{ucov_{i-1}}$$

Using the algebraic manipulations of [28] (page 1037), we get the following:

$$\sum_{\substack{o_j \in \mathcal{O} \\ p, o_j \text{ are partners}}} cost_j \leq H_{|P|}$$

Where H_j is the jth harmonic number. By definition of the symbol f (maximum number of observations supported by a single partner), we obtain the statement of the claim.

(Proof of claim 2): Combining claims 1-2, we get $|\mathcal{E}| \leq \sum_{p_i \in \mathcal{E}^*} (1 + \ln(f))$, which gives us the claim. $\square$

C.1.18 Proof of Proposition 40

GREEDY-KSEP-OPT1 returns a $|\mathcal{E}|$-sized (α, β) explanation for $\mathcal{O}$.
GREEDY-KSEP-OPT1 returns IMPOSSIBLE if there is no explanation for $\mathcal{O}$.

Proof. Suppose by way of contradiction that there exists and element $o \in \mathcal{O}$ such that there is no in $\mathcal{E}$. We note that set $\mathcal{O}'$ contains all elements of $\mathcal{O}$ and the only way for an element to be removed from $\mathcal{O}'$ is if a partner for that element is added to $\mathcal{E}$. Hence, if the program returns a set $\mathcal{E}$, we are guaranteed that each $o \in \mathcal{O}$ has a partner in $\mathcal{E}$.

Suppose by way of contradiction that GREEDY-KSEP-OPT1 returns IMPOSSIBLE and there exists a set $\mathcal{E}$ that is a valid (α, β) explanation for $\mathcal{O}$. Then, for every element of $\mathcal{O}$, there exists a valid partner. However, this contradicts line 3b of NAIVE-KSEP-SC (called by line 4b of GREEDY-KSEP-OPT1) which causes the program to return IMPOSSIBLE only if an element of $\mathcal{O}$ is found without any possible partner. □

C.1.19 Proof of Theorem 23

GREEDY-KSEP-OPT2 has a complexity of $O(\Delta \cdot f^2 \cdot |\mathcal{O}| + |\mathcal{O}| \cdot \ln(|\mathcal{O}|))$ and an approximation ratio of $1 + \ln(f)$.

Proof. CLAIM 1: GREEDY-KSEP-OPT2 has a complexity of $O(\Delta \cdot f^2 \cdot |\mathcal{O}| + |\mathcal{O}| \cdot \ln(|\mathcal{O}|))$.
Line 1 takes $O(\Delta \cdot |\mathcal{O}|)$ time.

The loop starting at line 4 iterates $|\mathcal{O}|$ times.

The nested loop at line 4a iterates Δ times.

The second nested loop at line 4(a)i iterates f times. The inner body of this loop can be accomplished in constant time.

In line 5, initializing the Fibonacci heap takes constant time, as does inserting elements, hence this line takes only $O(|\mathcal{O}|)$ time.

The loop at line 6 iterates, at most, $|\mathcal{O}|$ times.

Viewing the minimum of a Fibonacci heap, as in line 6a can be done in constant time.

As per the analysis of GREEDY-KSEP-OPT1, line 6b takes $\Delta \cdot f$ iterations. The updating procedure starts with line 6c which iterates f times.

The removal of an elements in line 6(c)ii from a Fibonacci heap costs $O(\ln(|\mathcal{O})$ amortized time. However, we perform this operation no more than $|\mathcal{O}|$ times, hence we can add $|\mathcal{O}| \cdot \ln(|\mathcal{O}|))$ to the complexity.

Note that the size of a list pointed to by REL_OBS$[o']$ is bounded by $\Delta \cdot f$ - f observations associated with each of Δ partners - hence line 6(c)iii iterates, at most, $\Delta \cdot f$ times.

We note that decreasing the key of an item in the Fibonacci heap (in line 6(c)iii) takes constant time (amortized).

Therefore, by the above statements, the complexity of GREEDY-KSEP-OPT2 is $O(|\mathcal{O}| \cdot (\Delta \cdot f + \Delta \cdot f^2) + |\mathcal{O}| \cdot \ln(|\mathcal{O}|) + \Delta \cdot f \cdot |\mathcal{O}| + \Delta \cdot |\mathcal{O}|)$ and the statement follows.

CLAIM 2: GREEDY-KSEP-OPT2 has an approximation ratio of $1 + \ln(f)$.

Follows directly from Proposition 39. $\square$

C.1.20 Proof of Proposition 41

GREEDY-KSEP-OPT2 returns a $|\mathcal{E}|$-sized (α, β) explanation for $\mathcal{O}$.

GREEDY-KSEP-OPT2 returns a IMPOSSIBLE if there is no explanation for $\mathcal{O}$.

Proof. Mirrors that of Proposition 40. □

Appendix D

Appendix for Chapter 5

D.1 Proofs

D.1.1 Proof of Lemma 17

Given observations $\mathcal{O}$ and the set of regions $R_{\mathcal{O}}$, then a region $r \in R_{\mathcal{O}}$ sub-explains an observation $o \in \mathcal{O}$ iff is super-explain o.

Proof. CLAIM 1: Any point in a region $r \in R_{\mathcal{O}}$ is either within distance $[\alpha, \beta]$ or outside the distance $[\alpha, \beta]$ from each $o \in \mathcal{O}$.
As $R_{\mathcal{O}}$ is created by drawing circles of radii α, β around each observation, the statement follows by the definition of $R_{\mathcal{O}}$.

CLAIM 2: ($\Leftarrow$) There is no $r \in R_{\mathcal{O}}$ that super-explains some $o \in \mathcal{O}$ but does not sub-explain the observation.
Suppose, BWOC, there is some $r \in R_{\mathcal{O}}$ that super-explains some $o \in \mathcal{O}$ but does not sub-explain it. Then, there must be at least one point in r that can be partnered

with $\mathcal{O}$ and at least one point in r that cannot be partnered with o. However, by claim 1, this is not possible, hence a contradiction.

CLAIM 3: ($\Rightarrow$) There is no $r \in R_{\mathcal{O}}$ that sub-explains some $o \in \mathcal{O}$ but does not super-explain the observation.

Follows directly from Observation 5.2.1. $\square$

D.1.2 Proof of Theorem 24

I-REP $\leq_p$ AC-Sup-REP.

AC-Sup-REP $\leq_p$ Sup-REP.

Proof. CLAIM 1: I-REP $\leq_p$ AC-Sup-REP.

Set up an instance of AC-Sup-REP with the input for I-REP plus the parameter $A = \pi \cdot (\beta^2 - \alpha^2)$. For direction $\Leftarrow$, note that a solution to this instance of I-REP is also a solution to AC-Sup-REP, as any region that sub-explain an observation also super-explains it for the set of region $R_{\mathcal{O}}$ (Lemma 17) and the fact that, by definition, all regions in the set $R_{\mathcal{O}}$ must have an area less than A. For direction $\Rightarrow$, we know that only regions that can be partnered with observations are considered by the area restriction, and by Lemma 17, the all regions in the solution are also super-explanations for their corresponding observation.

CLAIM 2: AC-Sup-REP $\leq_p$ Sup-REP.

Consider the set R from AC-Sup-REP and let set $R' = \{r \in R| \textit{ the area of } r \textit{ is less than or equal to}$

Set up an instance of Sup-REP where the set of regions is R' and the rest is the input from AC-Sup-REP. or direction $\Leftarrow$, it is obvious that any solution to AC-Sup-REP is also a solution to Sup-REP, as $R - R'$ are all regions that cannot possibly be in the solution to the instance of AC-Sup-REP. Going the other direction ($\Rightarrow$), we observe that by the definition of R', all regions in the result of the instance of Sup-REP meet all the requirements of the AC-Sup-REP problem. $\square$

D.1.3 Proof of Theorem 25

I-REP is NP-Complete.

Proof. CLAIM 1: I-REP is in-NP.

Given a set of regions, $R' \subseteq R_{\mathcal{O}}$ we can easily check in polynomial time that for each $o \in \mathcal{O}$ there is an $r \in R$ that is a partner for o. Simply check if each r falls within the distance $[\alpha, \beta]$ for a given $o \in \mathcal{O}$. The operation will take time $O(|\mathcal{O}| \cdot |R'|)$ - which is polynomial.

CLAIM 2: I-REP is strongly NP-hard.

We show that for an instance of the known strongly NP-complete problem, circle covering (CC), $CC \leq_p I - REP$ by the following transformation.

- Set $\mathcal{S} = \mathcal{S}'$
- Set $\mathcal{O} = P$
- Set $\beta = \beta'$
- Set $\alpha = 0$

- Set $k = k'$

This transformation obviously takes polynomial time. We prove correctness with the following two sub-claims.

CLAIM 2.1: If there is a k-sized solution R' for I-REP, then there is a corresponding k'-sized solution for CC.

Consider some $r \in R'$. Let $\mathcal{O}'$ be the subset of $\mathcal{O}$ (also of P) such that all points in $\mathcal{O}'$ are partnered with r. By definition, all points enclosed by r are of distance β or less away from each point in $\mathcal{O}'$. Hence, we can pick some point enclosed by r and we have the center of a circle that covers all elements in $\mathcal{O}'$. The statement follows.

CLAIM 2.2: If there is a k'-sized solution Q for CC, then there is a corresponding k-sized set solution for I-REP.

Consider some point $q \in Q$. Let P' be the subset of P (also of $\mathcal{O}$) such that all points in P' are of distance β' from q. As p is within β of an element of $\mathcal{O}$, it is in some region of the set $R_{\mathcal{O}}$. Hence, the region that contains p is a partner region for all elements of P'. The statement follows. □

D.1.4 Proof of Corollary 8

I-REP-MC cannot be approximated by a fully polynomial-time approximation scheme (FPTAS) unless $P == NP$.

Proof. Follows directly from [125] and Theorem 25. □

D.1.5 Proof of Corollary 9

1. Sub-REP and Sup-REP are NP-Complete.
2. Sub-REP-MC, Sup-REP-MC, I-REP-MC, Sub-REP-ME, Sup-REP-ME, and I-REP-ME are NP-Hard.
3. Sub-REP-MC, Sup-REP-MC cannot be approximated by a FPTAS unless $P == NP$.

Proof. All follow directly from Lemma 17, Theorem 25, and Corollary 8. □

D.1.6 Proof of Theorem 26

Sub/Sup-REP-MC $\leq_p$ Set-Cover

Sub/Sup-REP-ME $\leq_p$ Max-k-Cover

Proof. CLAIM 1: Sub/Sup-REP-MC $\leq_p$ Set-Cover

Consider the instance of set-cover $\langle \mathcal{O}, \bigcup_{r \in R}\{\mathcal{O}_r\}\rangle$ obtained from REDUCE-TO-COVERING$(\mathcal{O}, R)$.

Let $\mathcal{H}'$ be a solution to this instance of set-cover. ($\Leftarrow$) If R' is a solution to the instance of Sub/Sup-REP-MC, then the set $\bigcup_{r \in R'}\{\mathcal{O}_r\}$ is a solution to set-cover. Obviously, it must cover all elements of $\mathcal{O}$ and a smaller solution to set-cover would indicate a smaller R' – a contradiction. ($\Rightarrow$) Given set $\mathcal{H}'$, let $R'' \equiv \{r \in R | \mathcal{O}_r \in \mathcal{H}'\}$. Obviously, R'' provides a partner for all observations in $\mathcal{O}$. Further, a smaller solution to Sub/Sup-REP-MC would indicate a smaller $\mathcal{H}'$ is possible – also a contradiction.

CLAIM 2: Sub/Sup-REP-ME $\leq_p$ Max-k-Cover

Consider the instance of max-k-cover $\langle \mathcal{O}, \bigcup_{r\in R}\{\mathcal{O}_r\}, k\rangle$ obtained from REDUCE-TO-COVERING$(\mathcal{O}, R, k)$. Let $\mathcal{H}'$ be a solution to this instance of max-k-cover. ($\Leftarrow$) If R' is a solution to the instance of Sub/Sup-REP-ME, then the set $\bigcup_{r\in R'}\{\mathcal{O}_r\}$ is a solution to max-k-cover. Obviously, both have the same cardinality requirement. Further, if there is a solution to max-k-cover that covers more elements in $\mathcal{O}$, this would imply a set of regions that can be partnered with more observations in $\mathcal{O}$ - which would be a contradiction. ($\Rightarrow$) Given set $\mathcal{H}'$, let $R'' \equiv \{r \in R | \mathcal{O}_r \in \mathcal{H}'\}$. Obviously, R'' meets the cardinality requirement of k. Further, a solution to Sub/Sup-REP-ME that allows more observations in $\mathcal{O}$ to be partnered with a region would indicate a more optimal solution to max-k-cover – a contradiction. □

D.1.7 Proof of Proposition 42

REDUCE-TO-COVERING requires $O(|\mathcal{O}| \cdot |R|)$ time.

Proof. Follows directly from Line 1. □

D.1.8 Proof of Proposition 43

GREEDY-REP-ME requires $O(k \cdot |R| \cdot f)$ time and returns a solution whose where the number of observations in $\mathcal{O}$ that have a partner region in R' is within a factor $\left(\frac{e}{e-1}\right)$ of optimal.

Proof. The complexity proof mirrors that of Proposition 44 while the approximation guarantee follows directly from the results of [127]. □

D.1.9 Proof of Proposition 44

GREEDY-REP-ME requires $O(|\mathcal{O}| \cdot |R| \cdot f)$ time and returns a solution whose cardinality is within a factor of $1 + \ln(f)$ of optimal.

Proof. The outer loop of the algorithm iterates no more than $|\mathcal{O}|$ times, while the inner loop iterates no more than $|R|$ times. The time to compare the number of elements in a set $\mathcal{O}_r$ is $O(f)$.

The approximation factor of $1 + \ln(f)$ follows directly from [136]. □

D.1.10 Proof of Proposition 45

GREEDY-REP-MC2 runs in $O(\Delta \cdot f^2 \cdot |\mathcal{O}| + |\mathcal{O}| \cdot \ln(|\mathcal{O}|)$ time and returns a solution whose cardinality is within a factor of $1 + \ln(f)$ of optimal.

Proof. CLAIM 1: GREEDY-REP-MC2 runs in $O(\Delta \cdot f^2 \cdot |\mathcal{O}| + |\mathcal{O}| \cdot \ln(|\mathcal{O}|)$ time. The pre-processing in lines 1-4 can be accomplished in $O(\Delta + \Delta \cdot f)$ as the size of each GRP_o is bound by Δ and the size of each REL_o is bound by $\Delta \cdot f$.

The outer loop of the algorithm iterates $\mathcal{O}$ times. In each loop, the selection of the minimal element (line 5a) can be accomplished in constant time by use of a Fibonacci heap [49] (i.e. storing observations in $\mathcal{O}'$ organized by the value key_o). The next lines of the inner loop (lines 5b-5c) can be accomplished in $O(\Delta)$ time.

The next line, line 5d requires $O(\ln(|\mathcal{O}|)$ time per observation using a Fibonacci heap, as observations partnered with . However, we can be assured that, during the entire run of the algorithm, this operation is only performed $|\mathcal{O}|$ times (hence, we add an $|\mathcal{O}| \cdot \ln(|\mathcal{O}|)$). The final loop at line 5e occurs after the inner loop and iterates, at most f times. At each iteration, it considers, at most $f \cdot \Delta$ elements. Hence, the overall complexity is:

$$O(|\mathcal{O}| \cdot \big(\Delta + f^2 \cdot \Delta\big) + |\mathcal{O}| \cdot \ln(|\mathcal{O}|))$$

The statement of the claim follows.

CLAIM 2: GREEDY-REP-MC2 returns a solution whose cardinality is within a factor of $1 + \ln(f)$ of optimal.

The proof of this claim resembles the approximation proof of the standard greedy algorithm for set-cover (i.e. see [28] page 1036).

Let $r_1, \ldots, r_i, \ldots, r_n$ be the elements of R', the solution to GREEDY-REP-MC2, numbered by the order in which they were selected. For each iteration (of the outer loop), let set COV_i be the subset of observations that are partnered for the first time with region r_i. Note that each element of $\mathcal{O}$ is in exactly one COV_i. For each $o_j \in \mathcal{O}$, we define $cost_j$ to be $\frac{1}{|COV_i|}$ where $o_j \in COV_i$. Let R^* be an optimal solution to the instance of Sub/Sup-REP-MC.

CLAIM 2.1: $\sum_{r_i \in R^*} \sum_{o_j \in \mathcal{O}_{r_i}} cost_j \geq |R|$

By the definition of $cost_j$, exactly one unit of cost is assigned every time a region is picked for the solution R. Hence,

$$COST(R) = |R| = \sum_{o_j \in \mathcal{O}} cost_j$$

The statement of the claim follows.

CLAIM 2.2: For some region $r \in R$, $\sum_{o_j \in \mathcal{O}_r} cost_j \leq 1 + \ln(f)$.

Let P be the subset of $\mathcal{O}$ that can be partners with p. At each iteration i of the algorithm, let $uncov_i$ be the number elements in P that do not have a partner. Let $last$ be the smallest number such that $uncov_{last} = 0$. Let $R_P = \{r_i \in R | (i \leq last) \wedge (COV_i \cap P \not\equiv \emptyset)\}$. From here on, we shall renumber each element in R_P as $r_1, \ldots, r_{|R_P|}$ by the order they are picked in the algorithm (i.e. if an element is picked that cannot partner with anything in P, we ignore it and continue numbering with the next available number, we will COV_i and the iterations of the algorithm as well, but do not re-define the set based on the new numbering).

We note that for each iteration i, the number of items in P that are partnered is equal to $uncov_{i-1} - uncov_i$. Hence,

$$\sum_{o_j \in \mathcal{O}_r} cost_j = \sum_{i=1}^{last} \frac{uncov_{i-1} - uncov_i}{|COV_i|}$$

At each iteration of the algorithm, let $PCOV_i$ be the subset of observations that are covered for the first time if region p is picked instead of region r_i. We note, that for all iterations in $1, \ldots, last$, the region p is considered by the algorithm as one of its options for greedy selection. Therefore, as p is not chosen, we know that $|COV_i| \leq$

$|PCOV_i|$. Also, by the definition of $ucov_i$, we know that $|PCOV_i| = ucov_{i-1}$. This gives us:

$$\sum_{o_j \in \mathcal{O}_r} cost_j \leq \sum_{i=1}^{last} \frac{uncov_{i-1} - uncov_i}{ucov_{i-1}}$$

Using the algebraic manipulations of [28] (page 1037), we get the following:

$$\sum_{o_j \in \mathcal{O}_r} cost_j \leq H_{|P|}$$

Where H_j is the jth harmonic number. By definition of the symbol f (maximum number of observations supported by a single partner), we obtain the statement of the claim.

(Proof of Claim 2): Combining claims 1-2, we get $|R| \leq \sum_{r_i \in R^*}(1 + \ln(f))$, which gives us the statement.

□

D.1.11 Proof of Proposition 10

I-REP-MC-Z $\leq_p$ CC

Proof. Follows directly from Theorem 25. □

D.1.12 Proof of Proposition 46

The algorithm, FIND-REGION runs $O(|\mathcal{O}|)$ time, and region r (associated with the returned set $\mathcal{O}_r$) encloses p.

Proof. PART 1: FIND-REGION consists of a single loop that iterates $|\mathcal{O}|$ times.

PART 2: Suppose, the region enclosing point p has a different label. Then, there is either a bit in *label* that is incorrectly set to 1 or 0. As only observations which are $\leq$ from β have the associated bit position set to 1, then all 1 bits are correct. As we exhaustively consider all observations, the 0 bits are correct. Hence, we have a contradiction. □

D.1.13 Proof of Proposition 11

An α-approximation algorithm for CC is an α-approximation for KREP.

Proof. Follows directly from Theorem 25. □

D.1.14 Proof of Proposition 48

REGION-GEN has a time complexity $\Theta(|\mathcal{O}| \cdot f \cdot \frac{\pi \cdot \beta^2}{g^2})$.

Proof. For any given observation, the number of points in the grid that can be in a partnered region is $\frac{\pi \cdot \beta^2 - \alpha^2}{g^2}$. Hence, the first loop of the algorithm and the size of L are both bounded by $|\mathcal{O}| \cdot \frac{\pi \cdot \beta^2}{g^2}$. We note that the lookup and insert operations for the hash table T do not affect the average-case complexity - we assume these operations take constant time based on [28], hence the statement follows. □

Appendix E

Appendix for Chapter 6

E.1 MCA where the Solution is an Explanation

In Section 6.5 we study the **MCA** problem, but do not require the solution to be an explanation. In fact, it may often not be an explanation. Consider the following example.

Example E.1.1. *Suppose that the drug-enforcement agents from Example 6.5.1 consider the set $\mathcal{C} \equiv \{p_{45}, p_{48}, p_{50}\}$. Note that p_{45} can be partnered with observations o_1, o_2, p_{48} can be partnered with observations o_3, o_5 and p_{50} can be partnered with observation o_5. Hence, there is no element in $\mathcal{C}$ that can be partnered with o_4 – which means it is not an explanation. However, let us compute the expected agent benefit. Computing the reward (wrt* ***crf****) for each explanation function from Example 6.3.3, we get the following:*

$$\boldsymbol{crf}^{(dist)}(\mathsf{ex\_fcn}_1(\mathcal{O}, 3), \{p_{45}, p_{48}, p_{50}\}) = 1$$

$$\boldsymbol{crf}^{(dist)}(\mathsf{ex\_fcn}_2(\mathcal{O}, 3), \{p_{45}, p_{48}, p_{50}\}) = 1$$

Hence, the expected agent benefit in this case must be 1 *– which is optimal (expected agent benefit must be in the range* $[0,1]$*). Therefore, we have shown that we can have an optimal solution to MCA that is not an explanation in our example.*

We can also construct an instance of the MCA problem where there is no optimal solution that is also explaining. Stepping away from our running example for a moment consider the following case of a geospatial abduction problem. Consider observations o_1, o_2*. Let* $p_1, p_2, p_3, p_4, p_5, p_6$ *be the only feasible points, the first two being only partnered with* o_1 *and the rest being only partnered with* o_2*. Consider an adversary who will pick one of the following explanations as a strategy with uniform probability:*

- $\{p_1, p_3\}$
- $\{p_1, p_4\}$
- $\{p_2, p_5\}$
- $\{p_2, p_6\}$

Let us consider the reward function ***crf*** *with* $dist = 0$ *and* $B = 2$*. Therefore, the maximal counter-adversary strategy would be the set* $\{p_1, p_2\}$ *- this would give an expected agent benefit of* 0.5*. However, this set is not an explanation - observations* o_2 *is not covered. If we require the counter-adversary strategy to be an explanation, the set* $\{p_1, p_3\}$ *would be optimal. However, the expected agent benefit would only be* 0.375 *in this case.*

Hence, we shall also consider a the special case of a maximal counter-adversary strategy that is also an explanation.

Definition 115 (Maximal Explaining Counter-Adversary Strategy)**.** *Given a set of observations, $\mathcal{O}$, reward function* $\boldsymbol{rf}$ *and explanation function distribution* exfd *(of explanation for* $\mathcal{O}$*), a* ***maximal explaining counter-adversary strategy****,* $\mathcal{C}$*, an explanation for* $\mathcal{O}$ *such that* $\mathsf{EXB}^{(\boldsymbol{rf})}(\mathcal{C}, \mathsf{exfd})$ *is maximized.*

Again, for the case in which the reward function is monotonic, we shall include an cardinality requirement B for the set $\mathcal{C}$.

We formalize the optimization problem associated with finding a maximal explaining counter-adversary strategy.

MCA-Exp

INPUT: Space $\mathcal{S}$, feasibility predicate, feas, real numbers α, β, set of observations, $\mathcal{O}$, natural numbers k, B, reward function $\boldsymbol{rf}$, and explanation function distribution exfd.

OUTPUT: The maximal explaining counter-adversary strategy, $\mathcal{C}$.

The below corollary shows us that **MCA-Exp** is NP-hard.

Corollary 18. ***MCA-Exp*** *is NP-hard.*

We note that the proof of the above corollary follows directly from the result of Theorem 33. The associated problem is in the complexity class NP – this follows trivially from the membership results for the problem of finding an explanation and the **MCA** problem.

An Exact Algorithm For MCA-Exp. A naive, exact, and straightforward approach to the **MCA-Exp** problem would simply consider all subsets of L pf cardinality $\leq k_{\mathcal{C}}$ and pick the one which maximizes the expected agent benefit and is an explanation. This is the same as the naive approach we presented for **MCA**. Obviously, this approach has a complexity $O(\binom{|L|}{k_{\mathcal{C}}})$ - and is not practical. This is unsurprising as we showed this to be an NP-complete problem.

The following theorem shows that this problem reduces to the maximization of a submodular function over a uniform matroid - which can imply a practical algorithm to address this problem.

Theorem 60. ***MCA-Exp*** *reduces in polynomial time to the maximization of a submodular function wrt a uniform matroid.*

Proof Sketch. *Given an instance of* ***MCA-Exp*** *as follows:*

Space $\mathcal{S}$, feasibility predicate, feas*, real numbers α, β, set of observations, $\mathcal{O}$, natural numbers $k, k_{\mathcal{C}}$, reward function* ***rf****, and explanation function distribution* ***exfd****, we construct an instance of the maximization of a submodular function as follows (L is the set of all possible partners).*

1. *Let M be a uniform matroid consisting of all subsets of L of cardinality $\leq k_{\mathcal{C}}$*

2. *Let function $f_{submod} : 2^L \to \Re$ be defined as follows:*

   $$f_{submod}(\mathcal{C}) = \mathit{EXB}^{(\mathbf{rf})}(\mathcal{C}, \mathit{exfd}) + 2 \cdot |\{o \in \mathcal{O} | \exists p \in \mathcal{C} \ s.t. \ (d(o,p) \in [\alpha, \beta]) \wedge (\mathsf{feas}(p))\}|$$

In the remainder of the proof proceeds as follows. First, we show that $f_{submod}(\mathcal{C})$ is submodular. Then, we prove that if there is a solution to ***MCA-Exp*** *then the submodular maximization problem returns a value greater than or equal to $2 \cdot |\mathcal{O}|$. Then we show that if the submodular maximization problem returns a value greater than or equal to $2 \cdot |\mathcal{O}|$ then there is a solution to* ***MCA-Exp****. Finally, we complete the proof by showing that if* ***MCA-Exp*** *returns a value b, then the submodular maximization problem returns a value $b + 2 \cdot |\mathcal{O}|$ and that if the maximization of f_{submod} returns value b, then* ***MCA-Exp*** *returns a value $b - 2 \cdot |\mathcal{O}|$.* ■

Although, due to the construction of Theorem 60 an $\frac{1}{\alpha}$ approximation of f_{submod} does not necessarily yield an $\frac{1}{\alpha}$ approximation of **MCA-Exp**, we still can apply the local search or greedy algorithms as a heuristic by simply replacing calls to the function $\mathsf{EXB}^{(\boldsymbol{rf})}$ with calls to f_{submod}.

E.2 Proofs

E.2.1 Proof of Lemma 19

#GCD is #P-complete and there is no FPRAS for #GCD unless NP == RP.

Proof. **CLAIM 1:** #GCD is in #P.

Clearly, as the total number of "yes" answers is bounded by 2^K, this problem is in the complexity class #P.

CLAIM 2: #GCD is #P-hard.

We have to show a parsimonious or weakly parsimonious reduction from a known $\#P$-complete problem. In [71], the authors show that the counting version of the dominating set problem (#DOMSET) is #P-complete based on a weakly parsimonious reduciton from the counting version of vertex cover. It is important to note that the consruction used in this proof uses a graph with a maximum degree of three. This shows that the counting version of the dominating set problem on a graph with a maximum degree of three is also #P-hard as well. In [123], the authors show a parsimonious reduction from the dominating set problem (with maximum degree of three) to GCD. Hence, as the reduction is parsimonuous, and the associated counting probelm is $\#P$-hard, then the statement of the claim follows.

CLAIM 3: There is no FPRAS for #GCD unless NP == RP.

By Leamm 19 and [71], conisder the following chain of polynomial-time parsimonious or weakly parsimonious reductions:

$$\#SAT \rightarrow \#3CNFSAT \rightarrow \#Pl3CNFSAT$$

$$\#Pl3CNFSAT \rightarrow \#Pl1Ex3SAT \rightarrow \#Pl1Ex3MonoSAT$$

$$\#Pl1Ex3MonoSAT \rightarrow \#PlVC \rightarrow \#Pl3DS \rightarrow \#\mathsf{GCD}$$

Hence, as all of the reductions are PTIME, parsimonious or weakly parsimonious, and planarity preserving (for planar problems), by the results of [38], the statement follows. □

E.2.2 Proof of Theorem 27

The counting version of k-**SEP** is #P-Complete and has no FPRAS unless NP=RP.

Proof. Follows directly from the fact that the number of solutions is bounded by 2^k (memebrship) and hardness follows directly from the parsimonious reduction shown in [158] and Lemma 19. $\square$

E.2.3 Proof of Proposition 49

If a reward function meets axioms 1 and 2, then then the incremental increase obtained by adding a new element decreases on a superset. Formally:
If $\mathcal{C} \subseteq \mathcal{C}'$, and point $p \in \mathcal{S}$ s.t. $p \notin \mathcal{C}$ and $p \notin \mathcal{C}'$, then $\boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C} \cup \{p\}) - \boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C}) \geq \boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C}' \cup \{p\}) - \boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C}')$.

Proof. Suppose, BWOC, for $\mathcal{C} \subseteq \mathcal{C}'$, and point $p \in \mathcal{S}$ s.t. $p \notin \mathcal{C}$ and $p \notin \mathcal{C}'$, then

$$\boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C} \cup \{p\}) - \boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C}) < \boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C}' \cup \{p\}) - \boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C}')$$

We know that $\mathcal{C}' \cup \{p\} \equiv \mathcal{C}' \cup (\mathcal{C} \cup \{p\})$. Hence:

$$\boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C} \cup \{p\}) - \boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C}) < \boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C}' \cup (\mathcal{C} \cup \{p\})) - \boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C}')$$

Also, we know that $\mathcal{C} \equiv (\mathcal{C} \cup \{p\}) \cap \mathcal{C}'$, so we get:

$$\boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C} \cup \{p\}) - \boldsymbol{rf}(\mathcal{E}_{gt}, (\mathcal{C} \cup \{p\}) \cap \mathcal{C}') < \boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C}' \cup (\mathcal{C} \cup \{p\})) - \boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C}')$$

Which leads to:

$$\boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C}') + \boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C} \cup \{p\}) - \boldsymbol{rf}(\mathcal{E}_{gt}, (\mathcal{C} \cup \{p\}) \cap \mathcal{C}') < \boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C}' \cup (\mathcal{C} \cup \{p\}))$$

Which is a clear violation of Axiom 2, hence we have a contradiction. ☐

E.2.4 Proof of Proposition 50

prf is a valid reward function.

Proof. In this proof, we define $pt1(\mathcal{E}_{gt}, \mathcal{C}), pt2(\mathcal{E}_{gt}, \mathcal{C})$ as follows:

$$\begin{aligned} pt1(\mathcal{E}_{gt}, \mathcal{C}) &= \frac{|\{p \in \mathcal{E}_{gt} | \exists p' \in \mathcal{C} \;\; s.t. \;\; d(p, p') \leq dist\}|}{2 \cdot |\mathcal{E}_{gt}|} \\ pt2(\mathcal{E}_{gt}, \mathcal{C}) &= \frac{|\{p \in \mathcal{C} | \;\nexists p' \in \mathcal{E}_{gt} \;\; s.t. \;\; d(p, p') \leq dist\}|}{2 \cdot |\mathcal{S}|} \end{aligned}$$

Hence, $\boldsymbol{prf}^{(dist)}(\mathcal{E}_{gt}, \mathcal{C}) = 0.5 + pt1(\mathcal{E}_{gt}, \mathcal{C}) - pt2(\mathcal{E}_{gt}, \mathcal{C})$. As we know the maximum value of both $pt1(\mathcal{E}_{gt}, \mathcal{C}), pt2(\mathcal{E}_{gt}, \mathcal{C})$ is 0.5, we know that ***prf*** is in $[0, 1]$. As $pt1(\mathcal{E}_{gt}, \mathcal{E}_{gt}) = 0.5$ and $pt2(\mathcal{E}_{gt}, \mathcal{E}_{gt}) = 0$, then Axiom 1 is also satisfied. Consider ***crf*** (Definition 62). Later, in Proposition 51, we show that this function is submodular, meeting Axiom 2. By Definitions 62, we can easily show that $pt1(\mathcal{E}_{gt}, \mathcal{C}) = 0.5 \cdot \boldsymbol{crf}^{(dist)}(\mathcal{E}_{gt}, \mathcal{C})$. As $pt1(\mathcal{E}_{gt}, \mathcal{C})$ is a positive linear combination of submodular functions, it is also submodular. Now consider $pt2(\mathcal{E}_{gt}, \mathcal{C})$. Any element added to any set $\mathcal{C}$ has the same effect – it either lowers the value by $\frac{1}{2 \cdot |\mathcal{S}|}$ or does not affect it – hence it is trivially submodular. Therefore, it follows that ***prf*** is submodular as it is a positive-linear combination of submodular functions. ☐

E.2.5 Proof of Proposition 51

crf is a valid, monotonic reward function.

Proof. CLAIM 1: $\boldsymbol{crf}$ satisfies reward Axiom 1.

Clearly, if $\mathcal{C} \equiv \mathcal{E}_{gt}$, then the numerator is $|\mathcal{E}_{gt}|$, which equals the denominator.

CLAIM 2: $\boldsymbol{crf}$ satisfies reward function Axiom 2.

Suppose, BWOC, there exists explanations $\mathcal{C}, \mathcal{C}'$ s.t. $\mathcal{C} \cup \mathcal{C}'$ is an explanation and $\boldsymbol{crf}^{(dist)}(\mathcal{E}_{gt}, \mathcal{C} \cup \mathcal{C}') > \boldsymbol{crf}^{(dist)}(\mathcal{E}_{gt}, \mathcal{C}) + \boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C}') - \boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C} \cap \mathcal{C}')$. Therefore, $card(\{p \in \mathcal{E}_{gt} | \exists p' \in \mathcal{C} \cup \mathcal{C}' \ s.t. \ d(p, p') \leq dist\})$ is greater than $card(\{p \in \mathcal{E}_{gt} | \exists p' \in \mathcal{C} \ s.t. \ d(p, p') \leq dist\}) + card(\{p \in \mathcal{E}_{gt} | \exists p' \in \mathcal{C}' \ s.t. \ d(p, p') \leq dist\}) - card(\{p \in \mathcal{E}_{gt} | \exists p' \in \mathcal{C} \cap \mathcal{C}' \ s.t. \ d(p, p') \leq dist\})$. We have a contradiction; indeed, by basic set theory we see that both sides of this strict inequality are actually equal.

CLAIM 3: $\boldsymbol{crf}$ is zero-starting.

Clearly, if $\mathcal{C} \equiv \emptyset$, the numerator must be 0, the statement follows.

CLAIM 4: $\boldsymbol{crf}$ is monotonic.

Suppose, BWOC, there exists $\mathcal{C} \subseteq \mathcal{C}'$ s.t. $\boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C}) > \boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C}')$. Then $card(\{p \in \mathcal{E}_{gt} | \exists p' \in \mathcal{C} \ s.t. \ d(p, p') \leq dist\}) > card(\{p \in \mathcal{E}_{gt} | \exists p' \in \mathcal{C}' \ s.t. \ d(p, p') \leq dist\})$. Clearly, this is not possible as $\mathcal{C} \subseteq \mathcal{C}'$ and we have a contradiction. $\square$

E.2.6 Proof of Proposition 52

$\boldsymbol{frf}$ is a valid, monotonic reward function.

Proof. CLAIM 1: $\boldsymbol{frf}$ satisfies all reward function axioms (*i.e.*, is valid).

Bounds We must show $\boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C}) \in [0,1]$. For each point $p \in \mathcal{E}_{gt}$, let $l_p^{\mathcal{C}} = \min_{p' \in \mathcal{C}} d(p,p')^2$. By the definition of the distance function d, we know $0 \leq l_p^{\mathcal{C}} < \infty$. Now let function $f(l_p^{\mathcal{C}}) = \frac{1}{|\mathcal{E}_{gt}| + \min_{p' \in \mathcal{C}} d(p,p')^2} = \frac{1}{|\mathcal{E}_{gt}| + l_p^{\mathcal{C}}}$. Since $0 \leq l_p^{\mathcal{C}} < \infty$, we see $0 < f(l_p^{\mathcal{C}}) \leq \frac{1}{|\mathcal{E}_{gt}|}$. Clearly, the summation over $|\mathcal{E}_{gt}|$ points $p \in \mathcal{E}_{gt}$ yields an answer in $\left(0, |\mathcal{E}_{gt}| \cdot \frac{1}{|\mathcal{E}_{gt}|}\right] = (0,1] \subset [0,1]$.

Axiom 1 If $\mathcal{C} \equiv \mathcal{E}_{gt}$, for each $p \in \mathcal{E}_{gt}$, there exists $p' \in \mathcal{C}$ s.t. $d(p,p') = 0$. Hence, by the definition of $\boldsymbol{frf}$, $\boldsymbol{frf}(\mathcal{E}_{gt}, \mathcal{C}) = 1$ in this case.

Axiom 2 We must show that our version of the triangle inequality holds, that is $\boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C} \cup \mathcal{C}') \leq \boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C}) + \boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C}') - \boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C} \cap \mathcal{C}')$. From above, $\boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C} \cup \mathcal{C}') = \sum_{p \in \mathcal{E}_{gt}} f(l_p^{\mathcal{C} \cup \mathcal{C}'})$. For each point $p \in \mathcal{E}_{gt}$, let $p_* = argmin_{p' \in \mathcal{C} \cup \mathcal{C}'} d(p,p')^2$. Without loss of generality, assume $p_* \in \mathcal{C}$, then $l_p^{\mathcal{C}} = l_p^{\mathcal{C} \cup \mathcal{C}'}$ thus $f(l_p^{\mathcal{C}}) = f(l_p^{\mathcal{C} \cup \mathcal{C}'})$. Since $p_* \in \mathcal{C}$, we have $p_* \in \mathcal{C} \cap \mathcal{C}'$ or $p_* \in \mathcal{C} \cap \bar{\mathcal{C}}'$.

If $p_* \in \mathcal{C} \cap \mathcal{C}'$: Then $f(l_p^{\mathcal{C} \cap \mathcal{C}'}) = f(l_p^{\mathcal{C}})$. However, since $p_* \in \mathcal{C}'$ we have, as above, $f(l_p^{\mathcal{C}'}) = f(l_p^{\mathcal{C}}) = f(l_p^{\mathcal{C} \cup \mathcal{C}'})$. Thus

$$\sum_{p \in \mathcal{E}_{gt}} \left[f(l_p^{\mathcal{C}}) + f(l_p^{\mathcal{C}'}) - f(l_p^{\mathcal{C} \cap \mathcal{C}'}) \right] \tag{E.1}$$

$$= \sum_{p \in \mathcal{E}_{gt}} \left[f(l_p^{\mathcal{C} \cup \mathcal{C}'}) + f(l_p^{\mathcal{C} \cup \mathcal{C}'}) - f(l_p^{\mathcal{C} \cup \mathcal{C}'}) \right] \tag{E.2}$$

$$= \sum_{p \in \mathcal{E}_{gt}} f(l_p^{\mathcal{C} \cup \mathcal{C}'}) \tag{E.3}$$

So $\boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C} \cup \mathcal{C}') = \boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C}) + \boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C}') - \boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C} \cap \mathcal{C}')$ for this case.

If $p_* \in \mathcal{C} \cap \bar{\mathcal{C}}'$: Then, from above, we are still guaranteed that $f(l_p^{\mathcal{C}}) = f(l_p^{\mathcal{C} \cup \mathcal{C}'})$, thus $\boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C} \cup \mathcal{C}') = \boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C})$. This reduces our problem to showing

$\boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C}') - \boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C} \cap \mathcal{C}') \geq 0$. However, $\boldsymbol{rf}$ is monotonic (shown below); since $\mathcal{C} \cap \mathcal{C}' \subseteq \mathcal{C}'$, then $\boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C} \cap \mathcal{C}') \leq \boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C}')$ and our claim holds.

A similar proof holds for the case $p_* \in \mathcal{C}'$.

CLAIM 2: $\boldsymbol{frf}$ is monotonic and zero-starting. The property of zero-starting follows directly from the definition of $\boldsymbol{frf}$.

By way of contradiction, assume there is some $\mathcal{C} \subset \mathcal{C}'$ s.t. $\boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C}) > \boldsymbol{rf}(\mathcal{E}_{gt}, \mathcal{C}')$. Then, as above, $\sum_{p \in \mathcal{E}_{gt}} f(l_p^{\mathcal{C}}) > \sum_{p \in \mathcal{E}_{gt}} f(l_p^{\mathcal{C}'})$. However, since $\mathcal{C} \subset \mathcal{C}'$, we have $l_p^{\mathcal{C}} \geq l_p^{\mathcal{C}'}$ for each $p \in \mathcal{E}_{gt}$. Similarly, $f(l_p^{\mathcal{C}}) \leq f(l_p^{\mathcal{C}'})$ and thus $\sum_{p \in \mathcal{E}_{gt}} f(l_p^{\mathcal{C}}) \leq \sum_{p \in \mathcal{E}_{gt}} f(l_p^{\mathcal{C}'})$, which is our contradiction. □

E.2.7 Proof of Proposition 53

$\boldsymbol{wrf}$ is a valid, monotonic reward function.

Proof. CLAIM 1: $\boldsymbol{wrf}$ satisfies all reward function axioms (*i.e.*, is valid).

Domain We must show $\boldsymbol{wrf}^{(W,dist)}(\mathcal{E}_{gt}, \mathcal{C}) \in [0, 1]$. As $(\mathcal{C} \cap \mathcal{E}_{gt}) \subseteq \mathcal{E}_{gt}$ and W only returns positive values, this function can only return values in $[0, 1]$.

Axiom 1 If $\mathcal{C} \equiv \mathcal{E}_{gt}$, then for each $p \in \mathcal{E}_{gt}$, there exists $p' \in \mathcal{C}$ s.t. $d(p, p') = 0$. This causes the numerator to equal $\sum_{p \in \mathcal{C}} W(p)$. As $\mathcal{C} \equiv \mathcal{E}_{gt}$, the is equivalent to the denominator, so $\boldsymbol{wrf}(\mathcal{E}_{gt}, \mathcal{C}) = 1$ in this case.

Axiom 2 We must show the inequality $\boldsymbol{wrf}^{(W,dist)}(\mathcal{E}_{gt}, \mathcal{C} \cup \mathcal{C}') \leq \boldsymbol{wrf}^{(W,dist)}(\mathcal{E}_{gt}, \mathcal{C}) + \boldsymbol{wrf}^{(W,dist)}(\mathcal{E}_{gt}, \mathcal{C}') - \boldsymbol{wrf}^{(W,dist)}(\mathcal{E}_{gt}, \mathcal{C} \cap \mathcal{C}')$. This proof is similar to the proof of Axiom 2 in Proposition 51.

CLAIM 2: $\boldsymbol{wrf}$ is monotonic and zero-starting.

The property of zero-starting if shown by when $\mathcal{C} \equiv \emptyset$, the numerator must be 0, hence, $\boldsymbol{wrf}(\mathcal{E}_{gt}, \emptyset) = 0$. By way of contradiction, assume there is some $\mathcal{C} \subset \mathcal{C}'$ s.t. $\boldsymbol{wrf}^{(W,dist)}(\mathcal{E}_{gt}, \mathcal{C}) > \boldsymbol{wrf}^{(W,dist)}(\mathcal{E}_{gt}, \mathcal{C}')$. Then

$$\frac{\sum_{\{p \in \mathcal{E}_{gt} | \exists p' \in \mathcal{C} \ s.t. \ d(p,p') \leq dist\}} W(p)}{\sum_{p' \in \mathcal{E}_{gt}} W(p')} > \frac{\sum_{\{p \in \mathcal{E}_{gt} | \exists p' \in \mathcal{C}' \ s.t. \ d(p,p') \leq dist\}} W(p)}{\sum_{p' \in \mathcal{E}_{gt}} W(p')}$$

Since $\mathcal{C} \subset \mathcal{C}'$, we have

$$\frac{\sum_{\{p \in \mathcal{E}_{gt} | \exists p' \in \mathcal{C} \ s.t. \ d(p,p') \leq dist\}} W(p)}{\sum_{p' \in \mathcal{E}_{gt}} W(p')} >$$

$$\frac{\sum_{\{p \in \mathcal{E}_{gt} | \exists p' \in \mathcal{C} \ s.t. \ d(p,p') \leq dist\}} W(p)}{\sum_{p' \in \mathcal{E}_{gt}} W(p')} + \frac{\sum_{\{p \in \mathcal{E}'_{gt} | \exists p' \in (\mathcal{C}' \cap \mathcal{C}) \ s.t. \ d(p,p') \leq dist\}} W(p)}{\sum_{p' \in \mathcal{E}_{gt}} W(p')}$$

Where $\mathcal{E}'_{gt} \equiv \{p \in \mathcal{E}_{gt} | \ \nexists p' \in \mathcal{C} \ s.t. \ d(p,p') \leq dist\}$. Hence,

$$0 > \boldsymbol{wrf}^{(W,dist)}(\mathcal{E}'_{gt}, \mathcal{C}' \cap \mathcal{C})$$

Which violates the first axiom, which was shown to apply to $\boldsymbol{wrf}^{(W,dist)}$ by Claim 1—a contradiction. □

E.2.8 Proof of Theorem 28

OAS is NP-hard.

Proof. CONSTRUCTION: Given an input $\langle P, b, K \rangle$ of GCD, we create an instance of **OAS** in PTIME as follows:

- Set $\mathcal{S}$ to be a grid large enough that all points in P are also points in $\mathcal{S}$.

- $\mathsf{feas}(p) = \mathsf{TRUE}$ iff $p \in P$

- $\alpha = 0$, $\beta = b$, $\mathcal{O} \equiv P$, $k = |P|$

- Let $\boldsymbol{rf}(\mathcal{E}_1, \mathcal{E}_2) = 1$ if $\mathcal{E}_1 \subseteq \mathcal{E}_2$, and $\frac{|\mathcal{E}_1|}{|\mathcal{S}|}$ otherwise.

  This satisfies reward axiom 1 as $\mathcal{E}_1 \subseteq \mathcal{S}$, axiom 2 by definition, and the satisfiaction of axiom 3, along with montoniticity (wrt the second argument) can easily be shown by the fact that explanations that are not supersets of $\mathcal{E}_1$ (lets callthem $\mathcal{E}_2, \mathcal{E}_3$) that $\boldsymbol{rf}(\mathcal{E}_1, \mathcal{E}_2) = \boldsymbol{rf}(\mathcal{E}_1, \mathcal{E}_3)$.

- Let $\mathsf{ex\_fcn}(O, num)$ that returns set O when $num = |O|$ and is otherwise undefined. Let $\mathsf{exfd}(\mathsf{ex\_fcn}) = 1$ and 0 otherwise.

CLAIM 1: If $\mathcal{E}_{gt}$ as returned by **OAS** has a cardinality of $\leq K$, then the answer to GCD is "yes".

Suppose, BWOC, that $card(\mathcal{E}_{gt}) \leq K$ and GCD answers "no." This is an obvious contradiction as $\mathcal{E}_{gt}$ is a subset of P (by how feasibility was defined) where all elements of P are within a radius of b and $\mathcal{E}_{gt}$ also meets the cardinlality requirement of GCD.

CLAIM 2: If the answer to GCD is "yes" then $\mathcal{E}_{gt}$ as returned by **OAS** has a cardinality of less than or equal to K.

Suppose, BWOC, GCD returns "yes" but $\mathcal{E}_{gt}$ returned by **OAS** has a cardinaity greater than K. By the result of GCD, there exists a set P' of cardinality K s.t. each point in P (hence $\mathcal{O}$) is of a distance $\leq \beta$ from a point in P'. This, along with the definitoin of feasibility, make P' a valid K-explanation for $\mathcal{O}$. We note that $\mathsf{ex\_fcn}(P, |P|) = P$ and that exfd assigns this reward function a probability of one.

Hence, the expected adversarial detriment for any explanation $\mathcal{E}'_{gt}$ is $\boldsymbol{rf}(\mathcal{E}'_{gt}, P)$. As P' is an explanation of cardinality less than $\mathcal{E}_{gt}$, it follows that $\boldsymbol{rf}(P', P) < \boldsymbol{rf}(\mathcal{E}_{gt}, P)$ – which is a contradiction. □

E.2.9 Proof of Theorem 29

If the reward function is computable in PTIME, then **OAS-DEC** is NP-complete.

Proof. NP-harndness follows from Theorem 28. To show NP-completeness, a witness simply consists of $\mathcal{E}_{gt}$. We note that, as the reward function is computable in PTIME, finding the expected adversarial detriment for $\mathcal{E}_{gt}$ and comparing it to R can also be accomplished in PTIME. □

E.2.10 Proof of Theorem 30

Finding the set of all adversarial optimal strategies that provide a "yes" answer to **OAS-DEC** is #P-hard.

Proof. Let us assume that we know one optimal adversarial strategy and can compute the expected adversarial detriment from such a set – let us call this value D. Given an instance of GCD, we can create an instance of **OAS-DEC** as in Theorem 28, where we set $R = D$. Suppose we have an algorithm that produces all adversarial strategies. If we iterate through all strategies in this set, and count all strategies with a cardinality $\leq K$ (the K from the instance of GCD), we have counted all solutions to GCD – thereby solving the counting version of GCD, a #P-hard problem that is difficult to approximate by Lemma 19. $\square$

E.2.11 Proof of Proposition 55

Setting up the ***wrf*/*frf*** Constraints can be accomplished in $O(|\mathbf{EF}| \cdot k \cdot |\mathcal{O}| \cdot \Delta)$ time (provided the weight function W can be computed in constant time).

Proof. First, we must run $\mathsf{POSS\text{-}PART}$ - which reuqires $O(|\mathcal{O}| \cdot \Delta)$ operations. This results in a list of size $O(|\mathcal{O}| \cdot \Delta)$. For each explanation function, $\mathsf{ex\_fcn}$, we must compare every element in L with each element of $\mathsf{ex\_fcn}(\mathcal{O})$ - which would require $O(k \cdot |\mathcal{O}| \cdot \Delta)$ time. As there are $|\mathbf{EF}|$ explanation functions, the statement follows. $\square$

E.2.12 Proof of Proposition 56

The ***wrf***, ***frf*** Constraints have $O(|\mathcal{O}| \cdot \Delta)$ variables and $1 + |\mathcal{O}|$ constraints.

Proof. As list L is of size $O(|\mathcal{O}| \cdot \Delta)$, and there is one variable for every element of L - there are $O(|\mathcal{O}| \cdot \Delta)$ variables. As there is a constraint for each observation, plus a constraint to ensure the cardinality requirement (k) is met, there are $1 + |\mathcal{O}|$ constraints. $\square$

E.2.13 Proof of Proposition 54

Given ***wrf*** or ***frf*** Constraints:

1. Given set $\mathcal{E}_{gt} \equiv \{p_1, \ldots, p_n\}$ as a solution to **OAS** with ***wrf***(***frf***), if variables $X_1, \ldots, X_n$ - corresponding with elements in $\mathcal{E}_{gt}$ are set to 1 - and the rest

of the variables are set to 0, the objective function of the constraints will be minimized.

2. Given the solution to the constraints, if for every $X_i = 1$, we add point p_i to set $\mathcal{E}_{gt}$, then $\mathcal{E}_{gt}$ is a solution to **OAS** with ***wrf***(***frf***).

Proof. PART 1: Suppose BWOC, that there is a set of variables $X'_1, \ldots, X'_m$ that is a solution to the constraints s.t. the value of the objective function is less than if variables $X_1, \ldots, X_n$ were used. Then, there are points $p'_1, \ldots, p'_m$ in set L that correspond with the X_i's s.t. they cover all observations and that the expected adversarial detriment is minimized. Clearly, this is a contradiction.

PART 2: Suppose BWOC, that there is a set of points $\mathcal{E}'_{gt}$ s.t. the expected adversarial detriment is less than $\mathcal{E}_{gt}$. Clearly, $\mathcal{E}_{gt}$ is a valid explanation that minimizes the expected adversarial detriment by the definiton of the constraints - hence a contradiction. □

E.2.14 Proof of Proposition 57

The ***wrf***/***frf*** constraints can be transformed into a purely linear-integer form in $O(|\mathcal{O}|^2 \cdot \Delta)$ time.

Proof. Obviously, in both sets of constraints, the denominator of the objective function is strictly positive and non-zero. Hence, we can directly apply the Charnes-Cooper transformation [22] to obtain a purely integer-linear form. This transformation requires $O(\textit{number of variables} \times \textit{number of constraints})$. Hence, the $O(|\mathcal{O}|^2 \cdot \Delta)$ time complexity of the operation follows immediately from Proposition 56. □

E.2.15 Proof of Proposition 58

Given the constraints of Definition 70 or Definition 71, if we consider the linear program formed by setting all X_i variables to be in $[0,1]$, then the value returned by the objective function will be a lower bound on the value returned by the objective function for the mixed integer-linear constraints, and this value can be obtained in $O(|\mathcal{O}|^{3.5} \cdot \Delta^{3.5})$ time.

Proof. CLAIM 1: The linear relaxation of Definition 70 or Definition 71 provides a lower bound on the objective function value for the full integer-linear constraints. As an optimal value returned by the integer-linear constraints would also be a solution, optimal wrt minimality, for the linear relaxation, the statement follows.
CLAIM 2: The lower bound can be obtained in $O(|L|^{3.5})$ time.
As there is a variable for each element of L, the size of L is $O(|\mathcal{O}| \cdot \Delta)$, and the claim follows immediately from the result of [79]. $\square$

E.2.16 Proof of Porposition 59

Solving Definition 70 or Definition 71, where for some subset $L' \subset L$, every variable X_i associated with some $p_i \in L'$ is set to 0, the resulting solution will be an upper bound on the objective function for the constraints solved on the full set of variables.

Proof. Suppose, BWOC, that the solution for the objective function on the reduced MILP would be less than the actual MILP. Let $X_1, \ldots, X_n$ be the variables set to 1 for the reduced MILP in this scenario. We note, that setting the same variables to the full MILP would also be a solution, and could not possibly be less than a minimal solution – hence a contradiction. □

E.2.17 Proof of Theorem 31

If $\mathcal{E}_{gt}$ is an optimal adversarial strategy, there exists a core explanation $\mathcal{E}_{core} \subseteq \mathcal{E}_{gt}$.

Proof. CLAIM 1: For any explanation $\mathcal{E}$, there is an explanation $\mathcal{E}' \subseteq \mathcal{E}$ s.t. there are no two elements $p, p' \in \mathcal{E}'$ such that $\forall o \in \mathcal{O}$ s.t. o, p are partners, then o, p' are also partners.

Consider $\mathcal{E}$. If it does not already have the quality of claim 1, then by a simple induction, we can remove elements until the resulting set does.

CLAIM 2: If $\mathcal{E}_{gt}$ is an optimal adversarial stratgey, there is a no $p_j \in L - \mathcal{E}_{gt}$ s.t. there exists $p_i \in \mathcal{E}_{gt}$ where $const_j < const_i$ and $\forall o \in \mathcal{O}$ s.t. o, p_i are partners, then o, p_j are also partners.

Suppose, BWOC, there is a $p_j \in L - \mathcal{E}_{gt}$ s.t. there exists $p_i \in \mathcal{E}_{gt}$ where $const_j < const_i$ and $\forall o \in \mathcal{O}$ s.t. o, p_i are partners, then o, p_j are also partners.. Consider the set $(\mathcal{E}_{gt} - \{p_i\} \cup p_j$. This set is still an explanation and $\mathsf{EXR}^{(rf)}(\mathsf{exfd}, (\mathcal{E}_{gt} - \{p_i\} \cup p_j) < \mathsf{EXR}^{(rf)}(\mathsf{exfd}, \mathcal{E}_{gt})$ – which would be a contradiction as $\mathcal{E}_{gt}$ is an optimal adversarial stratgey.

CLAIM 3: There is an explanation $\mathcal{E} \subseteq \mathcal{E}_{gt}$ s.t. condition 2 of Definition 72 holds.

Consider the set $\mathcal{E} \equiv \{p_i \in \mathcal{E}_{gt} | \nexists p_j \in \mathcal{E}_{gt}\ s.t.\ (const_j < const_i) \wedge$ $(\forall o \in \mathcal{O}\ s.t.\ o, p_i\ are\ partners,\ then\ o, p_j\ are\ also\ partners)\}$. Note that any observation coverd by a point in $\mathcal{E}_{gt} - \mathcal{E}$ is covered by a point in $\mathcal{E}$ – hence $\mathcal{E}$ is an explanation. Further, by the definitoin of $\mathcal{E}$ and claim 2, this set meets condition 2 of Definition 72.

CLAIM 4: Set $\mathcal{E}$ from claim 3 is a core explanation.

By claim 3, $\mathcal{E}$ is a valid explanation and meets condition 2 of Definition 72. As it is a valid explanation, by claim 1, it also meets condition 1 of Definition 72. $\square$

E.2.18 Proof of Theorem 32

If an oracle that for a given k, $\mathcal{O}$, and exfd returns a core eplanation $\mathcal{E}_{core}$ that is guaranteed to be a subset of the optimal adversarial strategy associated with k, $\mathcal{O}$, and exfd, then we can find an optimal adversarial strategy in $O(\Delta \cdot |\mathcal{O}| \cdot \log(\Delta \cdot |\mathcal{O}|) + (k - |\mathcal{E}_{core}|)^2)$ time.

Proof. CLAIM 1: For explanation $\mathcal{E}$ and point $p_i \in L - \mathcal{E}$, $\mathsf{EXR}^{(rf)}(\mathsf{exfd}, \mathcal{E}) > \mathsf{EXR}^{(rf)}(\mathsf{exfd}, \mathcal{E} \cup \{p_i\})$ iff $const_i < \mathsf{EXR}^{(rf)}(\mathsf{exfd}, \mathcal{E})$.

If: Suppose $const_i < \mathsf{EXR}^{(rf)}(\mathsf{exfd}, \mathcal{E})$. Let $\mathsf{EXR}^{(rf)}(\mathsf{exfd}, \mathcal{E}) = \frac{a}{b}$. Hence, $\mathsf{EXR}^{(rf)}(\mathsf{exfd}, \mathcal{E} \cup \{p_i\}) = \frac{a+const_i}{b+1}$. Suppose, BWOC, $\mathsf{EXR}^{(rf)}(\mathsf{exfd}, \mathcal{E}) \leq \mathsf{EXR}^{(rf)}(\mathsf{exfd}, \mathcal{E} \cup \{p_i\})$. Then, $\frac{a}{b} \leq \frac{a+const_i}{b+1}$. This give us $a \cdot b + a \leq a \cdot b + const_i \cdot b$, which give us $\mathsf{EXR}^{(rf)}(\mathsf{exfd}, \mathcal{E}) \leq const_i$ – a contradiction.

Only-if: Suppose $\mathsf{EXR}^{(rf)}(\mathsf{exfd}, \mathcal{E}) > \mathsf{EXR}^{(rf)}(\mathsf{exfd}, \mathcal{E} \cup \{p_i\})$. Let $\mathsf{EXR}^{(rf)}(\mathsf{exfd}, \mathcal{E}) = \frac{a}{b}$. Hence, $\frac{a}{b} > \frac{a+const_i}{b+1}$ - which proves the claim.

CLAIM 2: For explanation $\mathcal{E}$ and points $p_i, p_j \in L - \mathcal{E}$ where $const_i < const_j$., then $\mathsf{EXR}^{(rf)}(\mathsf{exfd}, \mathcal{E} \cup \{p_i\}) > \mathsf{EXR}^{(rf)}(\mathsf{exfd}, \mathcal{E} \cup \{p_j\})$.

Straightforward algebera similiar to claim 1.

CLAIM 3: Algorithm BUILD-STRAT returns an optimal adversarial strategy.

We know that $\mathcal{E}_{core}$ must be in the optimal adversarial strategy. Hence, we suppose BWOC, that for the remaining elements, that ther eis a better set of elements - cardinality between 0 and $k - |\mathcal{E}_{core}|$ s.t. the expected adversarial detriment is lower. However, this contradicts claims 1-2.

CLAIM 4: Algorithm BUILD-STRAT runs in time $O(\Delta \cdot |\mathcal{O}| \cdot \log(\Delta \cdot |\mathcal{O}|) + (k - |\mathcal{E}_{core}|)^2)$.

Sorting the set $L - \mathcal{E}_{core}$ can be accomplished in $O(\Delta \cdot |\mathcal{O}| \cdot \log(\Delta \cdot |\mathcal{O}|))$ time. The remainder can be accomplished in $O((k - |\mathcal{E}_{core}|)^2)$ time. □

E.2.19 Proof of Lemma 20

Given an optimal adversarial strategy, $\mathcal{E}_{gt}$, if core explanation $\mathcal{E}_{core}$, of size δ, is a subset of $\mathcal{E}_{gt}$, then $\mathcal{E}_{core}$ is δ-core optimal.

Proof. Suppose BWOC, that $\mathcal{E}_{core}$ was not δ-core optimal. Then, given a δ-core optimal explanation $\mathcal{E}'_{core}$, we could conclude that $\mathsf{EXR}^{(rf)}(\mathsf{exfd}, (\mathcal{E}_{gt} - \mathcal{E}_{core}) \cup \mathcal{E}'_{core}) < \mathsf{EXR}^{(rf)}(\mathsf{exfd}, (\mathcal{E}_{gt})$ - which cannot be true as $\mathcal{E}_{gt}$ is an optimal adversarial strategy – hence a contradiction. □

E.2.20 Proof of Lemma 21

1. If explanation $\mathcal{E}$ is δ-core optimal, then $\mathcal{E} \subseteq L^{**}$.

2. If for some natural number δ, there exists an explation of size δ, then there exists a δ-core optimal explanation $\mathcal{E}$ s.t. $\mathcal{E} \subseteq L^*$.

Proof. Proof of Part 1:

Suppose, BWOC, exists explanation $\mathcal{E}$ s.t. for some δ, $\mathcal{E}$ is δ-core optimal and $\mathcal{E} \not\subseteq L^{**}$. Then, there exists some $p_i \in \mathcal{E} \cap (L - L^{**})$. By the definition of L^{**}, there exists a $p_j \in L^{**}$ s.t. $const_j < const_i$ and $\forall o \in \mathcal{O}\ s.t.\ o, p_i$ are partners, then o, p_j are also partners. Hence, the set $(\mathcal{E} - \{p_i\}) \cup \{p_j\}$ is also an explanation of size δ and has a lower expected detriment. From the definition of δ-core optimal, this is a contradiction.

Proof of Part 2:

Suppose, BWOC, for some δ s.t. there is an explanation of this size, there does not exist a δ-core optimal explanation $\mathcal{E}$ s.t. $\mathcal{E} \subseteq L^*$. By the proof of part 1, we know that an δ-core optimal explanation must be within L^{**}. Further, by the definition of L^*, for any point $p_i \in L^{**} - L^*$, there exists point $p_j \in L^*$ s.t. $const_j = const_i$ and $\forall o \in \mathcal{O}\ s.t.\ o, p_i$ are partners, o, p_j are also partners. Hence, for some δ-core explanation that is not a subset of L^*, any $p_i \in \mathcal{E} \cap (L^{**} - L^*)$ can be replace with some $p_j \in L^*$ - and the resulting set is still an explanation, optimal, and of cardinality δ - a contradiction. $\square$

E.2.21 Proof of Proposition 60

The δ-core cosntraints require $O(\Delta \cdot |\mathcal{O}|)$ variables and $1 + |\mathcal{O}|$ constraints.

Proof. Mirrors propositon 54. □

E.2.22 Proof of Proposition 61

Given δ-core cosntraints:

1. Given set δ-core optimal explanation $\mathcal{E}_{core} \equiv \{p_1, \ldots, p_n\}$, if variables $X_1, \ldots, X_n$ - corresponding with elements in $\mathcal{E}_{gt}$ are set to 1 - and the rest of the variables are set to 0, the objective function of the constraints will be minimized.

2. Given the solution to the constraints, if for every $X_i = 1$, we add point p_i to set $\mathcal{E}_{core}$, then $\mathcal{E}_{core}$ is a δ-core optimal soluton.

Proof. From Lemma 21, we know that for any δ s.t. there exists and explanation of that size, there is a δ-core explanation $\mathcal{E}$ that is a subset of L^*. Hence, the rest of the proof mirrors the proof of Proposition 54 □

E.2.23 Proof of Theorem 33

MCA is NP-hard.

Proof. Consider an instance of GCD consisting of set of points P, integer b, and integer K. We construct an instance of MCA as follows:

CONSTRUCTION:

- Set $\mathcal{S}$ to be a grid large enough that all points in P are also points in $\mathcal{S}$. We will use M, N to denote the length and width of $\mathcal{S}$.

- $\mathsf{feas}(p) = \mathsf{TRUE}$ iff $p \in P$

- $\alpha = 0$, and $\beta = \sqrt{M^2 + N^2}$, $\mathcal{O} \equiv P$, $k = K$, and $B = K$
- Let $\boldsymbol{rf}(\mathcal{E}_1, \mathcal{E}_2)$ be $\boldsymbol{crf}$ where $dist = b$.
- Let functions $\mathsf{ex\_fcn}_1, \ldots, \mathsf{ex\_fcn}_{|P|}$ be explanation functions - each $\mathsf{ex\_fcn}_i$ corresponding to a unique $p_i \in P$. Let $\mathsf{ex\_fcn}_i(\mathcal{O}, num) = \{p_i\}$ for all $num > 0$. Note that each p_i is an explanation for the set P as it is of cardinality $\leq k$, is feasible, and is guarantted to be with $[\alpha, \beta]$ from all other points in P as $[\alpha, \beta] = [0, \sqrt{M^2 + N^2}]$
- Let $\mathsf{exfd}(\mathsf{ex\_fcn}_i) = \frac{1}{|P|}$ for all i.

CLAIM 1: $\boldsymbol{crf}^{(dist)}(\{p_i\}, \mathcal{C}) = 1$ iff there exists $p' \in \mathcal{C}$ s.t. a disc of radius b (note $b = dist$) centered on p' covers p_i. $\boldsymbol{crf}^{(dist)}(\{p_i\}, \mathcal{C}) = 0$ iff there does not exist $p' \in \mathcal{C}$ s.t. a disc of radius b centered on p' covers p_i.

Follows directly from the definition of $\boldsymbol{crf}$.

CLAIM 2: If the expected agent benefit is 1, then for all i, $\boldsymbol{crf}^{(dist)}(\{p_i\}, \mathcal{C}) = 1$.

Suppose, BWOC, that the expected agent benefit is 1 and there exists some p_i s.t. $\boldsymbol{crf}^{(dist)}(\{p_i\}, \mathcal{C}) \neq 1$. Then, for a singleton set, $\boldsymbol{crf}^{(dist)}(\{p_i\}, \mathcal{C}) = 0$. Hence, for the $\mathsf{ex\_fcn}_i$ associated with p_i, $\boldsymbol{crf}^{(dist)}(\mathsf{ex\_fcn}_i(\mathcal{O}), \mathcal{C}) = 0$. So, by the definition of expected agent benefit, it is not possible for the expected agent benefit to be 1 – a contradiction.

CLAIM 3: If MCA returns an optimal counter-adversary strategy with an expected expected agent benefit of 1, then GCD must also return "yes."

Suppose, BWOC, MCA returns a stratgey with an expected agent benefit of 1 and the corresponding of GCD returns "no." Then there does not exist a K-sized cover for the points in P. However, the set $\mathcal{C}$ is of cardinlaity K and by claims 1-2 covers all points in P. Hence, a contradiction.

CLAIM 4: If GCD return "yes" then MCA must return an optima counter-adversary strategy with an expected agent benefit of 1.

Suppose, BWOC GCD returns "yes" and MCA reutrns returnsa an optimal strategy with an expected agent benefit < 1. However, by the answer to GCD, there must exist $P' \subseteq P$ of cardinality k that is within distance b of all points in P. Hence, for all i, $\boldsymbol{crf}^{(dist)}(\{p_i\}, \mathcal{C}) = 1$ (as $b = dist$). So, the expected agent benefit must also be 1. Hence, a contradiction.

Proof of theorem: Follows directly from claims 3-4. □

E.2.24 Alternate Proof of Theorem 33

MCA is NP-hard (shown in the case where the reward function is not monotonic and the agent has no budget).

Proof. Consider an instance of GCD consisting of set of points P, integer b, and integer K. We construct an instance of MCA as follows:

CONSTRUCTION: The construction is the same for the first proof of Theorem 33 in Section E.2.23 (the encoding of GCD) except the reward function is $\mathbf{krf}_k^{dist}(\mathcal{E}_{gt}, \mathcal{C})$

defined as follows

$$\frac{1}{2}+\frac{|\{p \in \mathcal{E}_{gt} | \exists p' \in \mathcal{C} \;\; s.t. \;\; d(p,p') \leq b\}|}{2 \cdot |\mathcal{E}_{gt}|} \qquad if\ |\mathcal{C}| \leq k$$

$$\frac{1}{2}+\frac{|\{p \in \mathcal{E}_{gt} | \exists p' \in \mathcal{C} \;\; s.t. \;\; d(p,p') \leq b\}|}{2 \cdot |\mathcal{E}_{gt}|} - \frac{|\mathcal{C}|-k}{2 \cdot |\mathcal{S}|} \qquad otherwise$$

CLAIM 1: Given some $k \geq |\mathcal{A}|$, the function **krf** is a valid reward function.

Clearly, $\mathbf{krf}_k^b(\mathcal{A},\mathcal{A}) = 1$. To show submodularity (the second axiom), we must show the following for $\mathcal{C} \subseteq \mathcal{C}'$ and $p \notin \mathcal{C}'$:

$$\mathbf{krf}_k^b(\mathcal{A}, \mathcal{C} \cup \{p\}) - \mathbf{krf}_k^b(\mathcal{A}, \mathcal{C}) \geq \mathbf{krf}_k^b(\mathcal{A}, \mathcal{C}' \cup \{p\}) - \mathbf{krf}_k^b(\mathcal{A}, \mathcal{C}') \tag{E.4}$$

There are six possible cases:

1. $|\mathcal{C}' \cup \{p\}| \leq k$: submodularity follows from the submodularity of ***crf***
2. $|\mathcal{C}' \cup \{p\}| > k, |\mathcal{C}'| \leq k, |\mathcal{C} \cup \{p\}| \leq k$: in this case, the left-hand side of inequality E.4 is positive and the right-hand side is negative, submodularity immediately follows
3. $|\mathcal{C}' \cup \{p\}| > k, |\mathcal{C}'| > k, |\mathcal{C} \cup \{p\}| \leq k$: in this case, the left-hand side of inequality E.4 is positive and the right-hand side is negative, submodularity immediately follows
4. $|\mathcal{C}' \cup \{p\}| > k, |\mathcal{C}'| \leq k, |\mathcal{C} \cup \{p\}| > k, |\mathcal{C}| \leq k$: this is the case where $\mathcal{C} \equiv \mathcal{C}'$, both sides of inequality E.4 are equal

5. $|\mathcal{C}' \cup \{p\}| > k, |\mathcal{C}'| > k, |\mathcal{C} \cup \{p\}| > k, |\mathcal{C}| \leq k$: the right-hand side of inequality E.4 either increases or decreases by, at most, the amount the left side decreases by - the left hand side always decreases

6. $|\mathcal{C}' \cup \{p\}| > k, |\mathcal{C}'| \leq k, |\mathcal{C} \cup \{p\}| > k, |\mathcal{C}| > k$: the right-hand side of inequality E.4 either increases or decreases by, at most, the amount the left side decreases by - the left hand side always decreases

PROOF OF THEOREM: Mirrors the proof in Section E.2.23, as this reward function is maximized (returns a value of 1) for the mixed adversarial strategy in the construction iff each point is within distance b of some point in the agent's strategy, *and* the agents strategy is of cardinality $\leq k$ (anything of a greater cardinality would give a reward less than 1). Therefore, we can follow the remainder of that proof and obtain the same result. □

E.2.25 Proof of Theorem 34

MCA-DEC is NP-complete, provided the reward function can be evaluated in PTIME.

Proof. CLAIM 1: Membership in NP.

Given an explanation, $\mathcal{C}$, we can evaluate it reward and if it is an explanation in PTIME.

CLAIM 2: MCA-DEC is NP-hard.

Follows directly from Theorem 33 □

E.2.26 Proof of Theorem 35

Counting the number of strategies that provide a "yes" answer to **MCA-DEC** is #P-complete and has no FPRAS unless NP==RP.

Proof. Theorem 33 shows a parsimonious reduction from GCD to **MCA**. Hence, we can simply apply Lemma 19 and the statement follows. $\square$

E.2.27 Proof of Theoerm 36

For a fixed $\mathcal{O}, k, \mathsf{exfd}$, the expected agent benefit, $\mathsf{EXB}^{(\boldsymbol{rf})}(\mathcal{C}, \mathsf{exfd})$ has the following properties:

1. $\mathsf{EXB}^{(\boldsymbol{rf})}(\mathcal{C}, \mathsf{exfd}) \in [0, 1]$

2. For $\mathcal{C} \subseteq \mathcal{C}'$ and some point $p \in \mathcal{S}$ where $p \notin \mathcal{C} \cup \mathcal{C}'$, the following is true:

   $\mathsf{EXB}^{(\boldsymbol{rf})}(\mathcal{C} \cup \{p\}, \mathsf{exfd}) - \mathsf{EXB}^{(\boldsymbol{rf})}(\mathcal{C}, \mathsf{exfd}) \geq \mathsf{EXB}^{(\boldsymbol{rf})}(\mathcal{C}' \cup \{p\}, \mathsf{exfd}) - \mathsf{EXB}^{(\boldsymbol{rf})}(\mathcal{C}', \mathsf{exfd})$

   (i.e. expected agent benefit is sub-modular for MCA)

Proof. Part 1 follow directly from the definition of a reward function and expected agent benefit.

For part 2, for some set $\mathcal{C}$ and fixed exfd, we have:

$$\mathsf{EXB}^{(\boldsymbol{rf})}(\mathcal{C}, \mathsf{exfd}) = \sum_{\mathsf{ex\_fcn} \in \mathbf{EF}} \boldsymbol{rf}(\mathcal{C}, \mathsf{ex\_fcn}(\mathcal{O}, k)) \cdot \mathsf{exfd}(\mathsf{ex\_fcn})$$

Which is a positive, linear combination of submodular functions – hence $\mathsf{EXB}^{(\boldsymbol{rf})}$ must also be submodualr. $\square$

E.2.28 Proof of Proposition 62

$\mathsf{MCA\text{-}LS}$ has time complexity of $O(\frac{1}{\epsilon} \cdot |L|^3 \cdot F(\mathsf{exfd}) \cdot \lg(|L|)$ where $F(\mathsf{exfd})$ is the time complexity to compute $\mathsf{EXB}^{(\mathit{rf})}(\mathcal{C}, \mathsf{exfd})$ for some set $\mathcal{C} \subseteq L$.

Proof. We note that one iteration of the algorithm requires $O(|L| \cdot F(\mathsf{exfd}) + |L| \cdot \lg(|L|))$ time. We shall assume that $O(|L| \cdot F(\mathsf{exfd})$ dominates $O(|L| \cdot \lg(|L|))$. By Theorem 3.4 of [47], the number of iterations of the algorithm is bounded by $O(\frac{1}{\epsilon} \cdot |L|^2 \cdot \lg(|L|)$ where $F(\mathsf{exfd})$, hence the statement follows. □

E.2.29 Proof of Proposition 63

$\mathsf{MCA\text{-}LS}$ is an $(\frac{1}{3} - \frac{\epsilon}{|L|})$-approximation algorithm for **MCA**.

Proof. By Theorem 36, we can be assured that when the “if” statement at line 4c is TRUE, then there are no further elements in $\mathcal{C}^*$ that will afford an incremental increase of $> (1 + \frac{\epsilon}{|L|^2}) \cdot \mathsf{EXB}^{(\mathit{rf})}(\mathcal{C}, \mathsf{exfd})$, even if the last element is not yet reached. Hence, we can apply Theorem 3.4 of [47] and the statement follows. □

E.2.30 Proof of Corollary 12

For a fixed $\mathcal{O}, k, \mathsf{exfd}$, if the reward function is montonic, then the expected agent benefit, $\mathsf{EXB}^{(\mathit{rf})}(\mathcal{C}, \mathsf{exfd})$ is also montonic and zero-starting.

Proof. The zero-starting aspect of expected agent benefit follows directly from the definitions of zero-starting and expected agent benefit.

Consider the definition of $\mathsf{EXB}^{(\boldsymbol{rf})}$:

$$\mathsf{EXB}^{(\boldsymbol{rf})}(\mathcal{C} \cup \{p\}, \mathsf{exfd}) - \mathsf{EXB}^{(\boldsymbol{rf})}(\mathcal{C}, \mathsf{exfd}) \geq \mathsf{EXB}^{(\boldsymbol{rf})}(\mathcal{C}' \cup \{p\}, \mathsf{exfd}) - \mathsf{EXB}^{(\boldsymbol{rf})}(\mathcal{C}', \mathsf{exfd})$$

As $\boldsymbol{rf}$ is montonic by the statement, and exfd is fixed, $\mathsf{EXB}^{(\boldsymbol{rf})}$ is a positive linear combination of montonic functions, so the statement follows. □

E.2.31 Proof of Proposition 64

The complexity of MCA-GREEDY-MONO is $O(B \cdot |L| \cdot F(\mathsf{exfd}))$ where $F(\mathsf{exfd})$ is the time complexity to compute $\mathsf{EXB}^{(\boldsymbol{rf})}(\mathcal{C}, \mathsf{exfd})$ for some set $\mathcal{C} \subseteq L$ of size B.

Proof. The outer loop at line 4 iterates B times, the inner loop at line 4b iterates $O(|L|)$ times, and at each inner loop, at line 4(b)ii, the function $\mathsf{EXB}^{(\boldsymbol{rf})}(\mathcal{C}, \mathsf{exfd})$ is computed with costs $F(\mathsf{exfd})$. There is an additional $O(|L| \cdot \lg(|L|))$ sorting operation after the inner loop which, under most non-trivial cases, is dominated by the $O(|L| \cdot F(\mathsf{exfd}))$ cost of the loop. The statement follows. □

E.2.32 Proof of Corollary 13

MCA-GREEDY-MONO is an $(\frac{e}{e-1})$-approximation algorithm for MCA (when the reward function is montonic).

First, we define incremental increase:

Definition 116. *For a given $p_i \in L$ at some iteration j of the outer loop of GREEDY-MONO (the loop starting at line 4), the incremental increase, $inc_i^{(j)}$, is defined as follows:*

$$inc_i^{(j)} = \mathit{EXB}^{(\boldsymbol{rf})}(\mathcal{C}^{(j-1)} \cup \{p_i\}, \mathcal{E}_{gt}) - \mathit{EXB}^{(\boldsymbol{rf})}(\mathcal{C}^{(j-1)}, \mathcal{E}_{gt})$$

Where $\mathcal{C}^{(j-1)}$ is the set of points in L selected by the algorithm after iteration $j-1$.

Proof. CLAIM 1: For any given iteration j of $\mathsf{GREEDY\text{-}MONO}$ and any $p_i \in L$, $inc_i^{(j)} \geq inc_i^{(j+1)}$

By Definition 116, the statement of the proposition is equivalent to the following:

$$\mathsf{EXB}^{(rf)}(\mathcal{C}^{(j-1)} \cup \{p_i\}, \mathcal{E}_{gt}) - \mathsf{EXB}^{(rf)}(\mathcal{C}^{(j-1)}, \mathcal{E}_{gt}) \geq \mathsf{EXB}^{(rf)}(\mathcal{C}^{(j)} \cup \{p_i\}, \mathcal{E}_{gt}) - \mathsf{EXB}^{(rf)}(\mathcal{C}^{(j)}, \mathcal{E}_{gt})$$

Obviously, as $\mathcal{C}^{(j-1)} \subseteq \mathcal{C}^{(j)}$, this has to be true by the submodularity of $\mathsf{EXB}^{(rf)}$, as proved in Theorem 36.

(Proof of Proposition): By claim 1, we can be assured that any point not considered by the inner loop will not have a greater incremental increase than some point already considered in that loop. Hence, our algorithm provides the same result as the greedy algorithm of [127]. We know that the results of [127] state that a greedy algorithm for a non-decreasing, submodularity function F s.t. $F(\emptyset) = 0$ is a $\frac{e}{e-1}$ approximation algorithm for the associated maximization problem. Theroem 36 and Corollary 12 show that these properties hold for finding a maximal counter-adversary startegy when the reward function is montonic. Hence, by [127], the statement follows. $\square$

E.2.33 Proof of Theoerem 37

$\mathsf{MCA\text{-}GREEDY\text{-}MONO}$ provides the best approximation ratio for MCA (when the reward function is monotonic) unless $P == NP$.

Proof. The MAX-K-COVER [46] is defined as follows.

INPUT: Set of elements, S and a family of subsets of S, $\mathcal{H} \equiv \{H_1, \ldots, H_{max}\}$, and positive integer K.

OUTPUT: $\leq K$ subsets from $\mathcal{H}$ s.t. the union of the subsets covers a maximal number of elements in S.

In [46], the author proves that for any $\alpha < \frac{e}{e-1}$, there is no α-approximation algorithm for MAX-K-COVER unless $P == NP$. We show that an instance of MAX-K-COVER can be embedded into an instance of MCA where the reward function is monotonic and zero-starting in PTIME. By showing this, we can leverage the result of [46] and Corollary 13 to prove the statement. We shall define the reward function $\boldsymbol{srf}(\mathcal{E}_{gt}, \mathcal{C}) = 1$ iff $|\mathcal{E}_{gt} \cap \mathcal{C}| \geq 1$ and $\boldsymbol{srf}(\mathcal{E}_{gt}, \mathcal{C}) = 0$ otherwise. Clearly, this reward function meets all the axioms, is zero-starting, and monotonic. We create a space $\mathcal{S}$ s.t. the number of points in $\mathcal{S}$ is greater than or equal to $|\mathcal{H}|$. For each subset in $\mathcal{H}$, we create an observation at some point in the space. We shall call this set $\mathcal{O}_{\mathcal{H}}$ and say that o_H is the element of $\mathcal{O}_{\mathcal{H}}$ that corresponds with set $H \in \mathcal{H}$. We set $\mathsf{feas}(p) = \mathsf{true}$ *iff* $p \in \mathcal{O}_{\mathcal{H}}$. We set $\alpha = 0$, β to be equal to the diagonal of the space, and $k = |\mathcal{O}_{\mathcal{H}}|$. Hence, any non-empty subset of $\mathcal{O}_{\mathcal{H}}$ is a valid explanation for $\mathcal{O}$. For each $x \in S$, we define explanation function $\mathsf{ex\_fcn}_x$ s.t. $\mathsf{ex\_fcn}_x(\mathcal{O}_{\mathcal{H}}, k) = \{o_H \in \mathcal{O}_{\mathcal{H}} | x \in H\}$. We define the explanation function distribution exfd to be a uniform distribution over all $\mathsf{ex\_fcn}_x$ explanation functions. We set the budget $B = K$. Clearly, this construction can be accomplished in PTIME. We note that any solution to this instance of MCA must be subset of $\mathcal{O}_{\mathcal{H}}$, for if it is not, we can get rid of the extra elements and have no change to the expected agent benefit. Hence, each $p \in \mathcal{C}$ will correspond to an element of $\mathcal{H}$, so we shall use the notation p_H to denote a point in the solution that corresponds with some $H \in \mathcal{H}$ (as each $o \in \mathcal{O}_{\mathcal{H}}$ corresponds with some $H \in \mathcal{H}$).

CLAIM 1: Given a solution $\mathcal{C}$ to MCA, the set $\{H \in \mathcal{H} | p_H \in \mathcal{C}\}$ is a solution to MAX-K-COVER.

Clearly, this solution meets the cardinality constraint, as there is exactly one element in $\mathcal{O}_{\mathcal{H}}$ for each element of $\mathcal{H}$ and $\mathcal{C}$ is a subset of $\mathcal{O}_{\mathcal{H}}$. Suppose, BWOC, there is some other subset of $\mathcal{H}$ that covers more elements in S. Let $\mathcal{H}'$ be this solution to MAX-K-COVER and $\mathcal{C}'$ be the subset of $\mathcal{O}_{\mathcal{H}}$ that corresponds with it. We note that for some $x \in S$ in $\mathcal{C}'$, $\boldsymbol{srf}(\mathsf{ex\_fcn}_x(\mathcal{O}_{\mathcal{H}}, k), \mathcal{C}') = 1$ iff there is some $H \in \mathcal{H}'$ s.t. $x \in H$ and $\boldsymbol{srf}(\mathsf{ex\_fcn}_x(\mathcal{O}_{\mathcal{H}}, k), \mathcal{C}') = 0$ otherwise. Hence, the expected agent benefit is the fraction of elements in S covered by $\mathcal{H}'$. If $\mathcal{H}'$ is the optimal solution to MAX-K-COVER, then $\mathcal{C}'$ must provide a greater expected agent benefit than $\mathcal{C}$, which is clearly a contradiction.

CLAIM 2: Given a solution $\mathcal{H}'$ to MAX-K-COVER, the set $\{o_H \in \mathcal{O}_{\mathcal{H}} | H \in \mathcal{H}'\}$ is a solution to MCA.

Again, that the solution meets the cardinality requirement is trivial (mirrors that part of claim 1). Suppose, BWOC, there is some set $\mathcal{C}$ that provides a greater maximum benefit than $\{o_H \in \mathcal{O}_{\mathcal{H}} | H \in \mathcal{H}'\}$. Let $\mathcal{H}'' \equiv \{H \in \mathcal{H} | p_H \in \mathcal{C}\}$. As with claim 1, the expected agent benefit for $\mathcal{C}$ is equal to the fraction of elements in S covered by $\mathcal{H}''$, which is a contradiction as $\mathcal{H}'$ is an optimal solution to MAX-K-COVER. $\square$

E.2.34 Proof of Corollary 18

MCA-Exp is NP-hard.

Proof. Consider the construction in Theorem 33. As any non-empty subset of P - which are all the feasible points in the space - is an explanation - then solution to MCA is also a solution to MCA-Exp. $\square$

E.2.35 Proof of Theorem 60

MCA-Exp reduces in polynomial time to the maximization of a submodular function wrt a uniform matroid.

Proof. Given an instance of **MCA-Exp** as follows:

Space $\mathcal{S}$, feasibility predicate, feas, real numbers α, β, set of observations, $\mathcal{O}$, natural numbers k, B, reward function $\boldsymbol{rf}$, and explanation function distribution exfd.

Let L be the set of all possible partners. Consider the following construction.

1. Let M be a uniform matroid consisting of all subsets of L of cardilnality $\leq B$
2. Let function $f_{submod} : 2^L \to \Re$ be defined as follows:

$$f_{submod}(\mathcal{C}) = \mathsf{EXB}^{(\boldsymbol{rf})}(\mathcal{C}, \mathsf{exfd}) + 2 \cdot |\{o \in \mathcal{O} | \exists p \in \mathcal{C} \ s.t. \ (d(o,p) \in [\alpha, \beta]) \wedge (\mathsf{feas}(p))\}|$$

CLAIM 1: $f_{submod}(\mathcal{C})$ is submodular.

As $\mathsf{EXB}^{(\boldsymbol{rf})}(\mathcal{C}, \mathsf{exfd})$, all we will show that $2 \cdot |\{o \in \mathcal{O} | \exists p \in \mathcal{C} \ s.t. \ (d(o,p) \in [\alpha, \beta]) \wedge (\mathsf{feas}(p))\}|$ is submodular, as a positive linear combination of submoudlar functions is also submodular. Suppose, BWOC that it is not submodular, hence, for some

$\mathcal{C} \subset \mathcal{C}'$ and $p'' \notin \mathcal{C}'$, we have the following:

$$2 \cdot |\{o \in \mathcal{O} | \exists p \in \mathcal{C} \cup \{p''\} \ s.t.\ (d(o,p) \in [\alpha, \beta]) \wedge (\mathsf{feas}(p))\}| -$$

$$2 \cdot |\{o \in \mathcal{O} | \exists p \in \mathcal{C} \ s.t.\ (d(o,p) \in [\alpha, \beta]) \wedge (\mathsf{feas}(p))\}| <$$

$$2 \cdot |\{o \in \mathcal{O} | \exists p \in \mathcal{C}' \cup \{p'\} \ s.t.\ (d(o,p) \in [\alpha, \beta]) \wedge (\mathsf{feas}(p))\}| -$$

$$2 \cdot |\{o \in \mathcal{O} | \exists p \in \mathcal{C}' \ s.t.\ (d(o,p) \in [\alpha, \beta]) \wedge (\mathsf{feas}(p))\}|$$

We can re-write this as follows:

$$2 \cdot |\{o \in \mathcal{O} | o \textit{ and } p'' \textit{ are partners and } \nexists p''' \in \mathcal{C} \textit{ that can also be a partner for } o\}| <$$

$$2 \cdot |\{o \in \mathcal{O} | o \textit{ and } p'' \textit{ are partners and } \nexists p''' \in \mathcal{C}' \textit{ that can also be a partner for } o\}|$$

Clearly, as $\mathcal{C} \subseteq \mathcal{C}'$, this cannot hold - hence we have a contradiction.

CLAIM 2: If there is a solution to **MCA-Exp** then the submodular maximization problem returns a value greater than or equal to $2 \cdot |\mathcal{O}|$.

Suppose, BWOC, there is a solution to **MCA-Exp**, and the submodular maximization problem returns a value less than $2 \cdot |\mathcal{O}|$. However, any solution to $\mathcal{C}$ to **MCA-Exp**, we know the following:

$$2 \cdot |\{o \in \mathcal{O} | \exists p \in \mathcal{C} \ s.t.\ (d(o,p) \in [\alpha, \beta]) \wedge (\mathsf{feas}(p))\}| = 2 \cdot |\mathcal{O}|$$

hence, a contradiction.

CLAIM 3: If the submodular maximization problem returns a value greater than or equal to $2 \cdot |\mathcal{O}|$ then there is a solution to **MCA-Exp**.

Suppose, BWOC, claim 3 is false. However, we know that

$$\mathsf{EXB}^{(\boldsymbol{rf})}(\mathcal{C}, \mathsf{exfd}) \leq 1$$

Hence, the only way for the submodular maximization problem returns a value greater than or equal to $2 \cdot |\mathcal{O}|$ is if the vertices chosen to produce such a value is an explanation – hence a contradiction.

CLAIM 4: If **MCA-Exp** returns a value b, then the submodular maximization problem returns a value $b + 2 \cdot |\mathcal{O}|$.

By claim 2, we know for solution $\mathcal{C}$ to **MCA-Exp**, for some $\mathcal{C}'$ set of elements that maximizes f_{submod} that:

$$2 \cdot |\{o \in \mathcal{O} | \exists p \in \mathcal{C}' \; s.t. \; (d(o,p) \in [\alpha, \beta]) \wedge (\mathsf{feas}(p))\}| = 2 \cdot |\mathcal{O}|$$

Hence, any set that maximizes f_{submod} is an explantion that maximizes the quantity $\mathsf{EXB}^{(rf)}(\mathcal{C}, \mathsf{exfd})$ - which, by definition, is also a set that can be a solution to **MCA-Exp**.

CLAIM 5: If the maximization of f_{submod} returns value b, then **MCA-Exp** returns a value $b - 2 \cdot |\mathcal{O}|$.

Consider set $\mathcal{C}'$ that maximizes f_{submod}. By claim 3, this is an explantion that maximizes $\mathsf{EXB}^{(rf)}(\mathcal{C}, \mathsf{exfd})$. Hence, by the definition of **MCA-Exp**, it will also give a solution to **MCA-Exp** and by the definition of f_{submod}, returns a value $b - 2 \cdot |\mathcal{O}|$.

Proof of theorem: follows directly from claims 2-5. $\square$

Appendix F

Appendix for Chapter 7

F.1 Proofs

F.1.1 Proof of Theorem 38

Given GBGOP $(\mathcal{M}, s_0, \mathcal{A}, \mathsf{C}, IC, \mathbf{c}, \Theta_{in}, \Theta_{out})$, finding an optimal solution $SOL \subseteq \mathcal{A} \times \mathcal{M}$ is NP-hard. This result holds even if for each $a \in \mathcal{A}, p \in \mathcal{M}$, it is the case that $\forall g'(p') \in a(p)$, $p' = p$ - i.e. each action only affects the point is is applied to).

Proof. The known NP-hard problem of SET-COVER [46] is defined as follows.

INPUT: Set of n elements, S and a family of m subsets of S, $\mathcal{H} \equiv \{H_1, \ldots, H_m\}$.

OUTPUT: $\mathcal{H}' \subseteq \mathcal{H}$ of minimal cardinality s.t. $\bigcup_{H \in \mathcal{H}'} H \equiv S$.

Given an instance of SET-COVER, we embed it in a GBGOP as follows:

- $\mathcal{G} = \{g_1, \ldots, g_n\}$ - each predicate in $\mathcal{G}$ corresponds to an element in S

- $\mathcal{M}$ consists of a single point, p

- $\mathcal{A} = \{a_1, \ldots, a_m\}$ - each action in $\mathcal{A}$ corresponds to an element in $\mathcal{H}$. Each a_i is defined as follows: $a_i(p) = \bigcup_{x_j \in H_i}\{g_j(p)\}$

- C returns 1 for all aciton-point pairs.
- $s_0 = \emptyset$, $IC = \emptyset$, $\mathbf{c} = m$
- $\Theta_{in} = \bigcup_{g_i \in \mathcal{G}}\{g_i(p)\}$
- $\Theta_{out} = \emptyset$

Clearly, this construction can be performed in PTIME. The corecetness of the embeding follows directly from claims 1-2 below.

CLAIM 1: If $\mathcal{H}' \subseteq \mathcal{H}$ is an optimal solution to SET-COVER, then $\bigcup_{H_i \in \mathcal{H}'}\{(a_i, p)\}$ is an optimal solution to the GBGOP.

First, we show that $\bigcup_{H_i \in \mathcal{H}'}\{(a_i, p)\}$ is a solution to the GBGOP. As $\mathbf{c}$ is the cardinality of $\mathcal{A} \times \mathcal{M}$ and $IC = \emptyset$, the first two requirement is trivially met. To meet the third requirement, we observe that $appl(\emptyset, \bigcup_{H_i \in \mathcal{H}'}\{(a_i, p)\} = \bigcup_{H_i \in \mathcal{H}'} a_i(p)$ and by the construction $\bigcup_{H_i \in \mathcal{H}'} a_i(p) = \bigcup_{g_i \in \mathcal{G}}\{g_i(p)\} = \Theta_{in}$. Now we show that the solution is optimal. Let $SOL = \bigcup_{H_i \in \mathcal{H}'}\{(a_i, p)\}$. Suppose, BWOC, there is some solution $SOL' \subseteq \mathcal{A} \times \mathcal{M}$ s.t. $|SOL'| < |SOL|$. Hence, there is some $\mathcal{H}''$ where $SOL' = \bigcup_{H_i \in \mathcal{H}''}\{(a_i, p)\}$. Hence, $|\mathcal{H}''| < |\mathcal{H}'|$. By the construction, $\mathcal{H}''$ also covers all elements of S - which implies $\mathcal{H}''$ is more optimal than $\mathcal{H}'$ - a contradiction.

CLAIM 2: If SOL is an optimal solution to the BMGOP, then $\{H_i | (a_i, p) \in SOL\}$ is an optimal solution to SET-COVER.

Clearly, by the construction, $\{H_i | (a_i, p) \in SOL\}$ covers all elements in S. Let $\mathcal{H}'$

be this set. Suppose, BWOC, that there is some set $\mathcal{H}'' \subseteq \mathcal{H}$ where $|\mathcal{H}''| < |\mathcal{H}'$ and $\mathcal{H}''$ covers all the elements in S. Then, we can construct $SOL' = \bigcup_{H_i \in \mathcal{H}''} \{(a_i, p)\}$. By the first claim, this must also be a solution to GBGOP. Further, it must have a smaller cardinality than SOL - a contradiction. □

F.1.2 Proof of Theorem 39

Given BMGOP $(\mathcal{M}, s_0, \mathsf{B}, \mathcal{A}, \mathsf{C}, IC, k, \mathbf{c})$, finding an optimal solution $SOL \subseteq \mathcal{A}$ is NP-hard.

Proof. The known NP-hard problem of MAX-K-COVER [46] is defined as follows.

INPUT: Set of n elements, S and a family of m subsets of S, $\mathcal{H} \equiv \{H_1, \ldots, H_m\}$, and positive integer K.

OUTPUT: $\leq K$ subsets from $\mathcal{H}$ s.t. the union of the subsets covers a maximal number of elements in S.

Given an instance of MAX-K-COVER, we embed it in a BMGOP as follows:

- $\mathcal{G} = \{g_1, \ldots, g_n\}$ - each predicate in $\mathcal{G}$ corresponds to an element in S
- $\mathcal{M}$ consists of a single point, p
- B is a $|B_{\mathcal{L}}|$-sized vector of 1's
- $\mathcal{A} = \{a_1, \ldots, a_m\}$ - each action in $\mathcal{A}$ corresponds to an element in $\mathcal{H}$. Each a_i is defined as follows: $a_i(p) = \bigcup_{x_j \in H_i} \{g_j(p)\}$
- C is a $|\mathcal{A} \times \mathcal{M}|$-sized vector of 1's
- $s_0 = \emptyset$, $IC = \emptyset$, $k = K$, $\mathbf{c} = K$

Clearly, this construction can be performed in PTIME. The corecetness of the embeding follows directly from claims 1-2 below.

CLAIM 1: If $\mathcal{H}' \subseteq \mathcal{H}$ is an optimal solution to MAX-K-COVER, then $\bigcup_{H_i \in \mathcal{H}'} \{(a_i, p)\}$ is an optimal solution to the BMGOP.

Suppose, BWOC, there is some solution $SOL' \subseteq \mathcal{A}$ s.t. $\sum_{A_i \in appl(\bigcup_{H_i \in \mathcal{H}'}\{(a_i,p)\}, s_0)} b_i < \sum_{A_i \in appl(SOL', s_0)} b_i$. As there is only one point in $\mathcal{M}$ (point p), then each action-point pair in SOL' must have unique action - let $\mathcal{A}'$ be these actions. By the construction, each action in $\mathcal{A}'$ is associated with a subset in $\mathcal{H}$ - let $\mathcal{H}^*$ be this set of subsets. Clearly, by the cost vector in the construction, $|\mathcal{H}^*| \leq k$, or else SOL' is not a solution to the BMGOP. We also know that $appl(SOL', s_0) = (\emptyset \cup \{a(p) | (a, p) \in SOL'\})$ - again, as there is only one point in $\mathcal{M}$, $appl(SOL', s_0)$ can be associated with a subset of $\mathcal{G}$ - let us call this $\mathcal{G}'$. Further, by the construction, $\mathcal{G}'$ is associated with a subset of S, which we shall denote S'. By the construction, S' is the union of all subsets in $\mathcal{H}^*$. And, by how we defined B, $|S'|$ is greater than the number of elements covered by the optimal solution - which is a contradiction.

CLAIM 2: If SOL is an optimal solution to the BMGOP, then $\{H_i | (a_i, p) \in SOL\}$ is an optimal solution to MAX-K-COVER.

Suppose, BWOC, that there set $\mathcal{H}' \subseteq \mathcal{H}$ s.t. $|\mathcal{H}'| \leq k$ and $|\bigcup_{H_j \in \mathcal{H}'} H_j| > |\bigcup_{H_i | (a_i, p) \in SOL} H_i|$. Let $\mathcal{A}'$ be the subset of actions associated with $\mathcal{H}'$ and $SOL' = \{(a_i, p) | a_i \in \mathcal{A}'\}$. By the cost vector, we know that $|SOL'| \leq k$ which means SOL' is a valid solution. By the construciton, and the fact there is only one point in $\mathcal{M}$, we know that $appl(SOL', s_0) = |\bigcup_{H_i \in \mathcal{H}'} H_i|$, which must be larger than $appl(SOL, s_0)$ - hence a contradiction. $\square$

F.1.3 Proof of Theorem 40

If for some $\epsilon > 0$, there is a PTIME algorithm to approximate GBGOP within $(1-\epsilon) \cdot \ln(|\mathcal{A} \times \mathcal{M}|)$, then $NP \subset TIME(|\mathcal{A} \times \mathcal{M}|^{O(\lg \lg |\mathcal{A} \times \mathcal{M}|)})$ (NP has a slightly super-polynomial algorithm).

Proof. Suppose, by way of contradiction, that for some $\epsilon > 0$, there is a PTIME algorithm to approximate GBGOP within $(1-\epsilon) \cdot \ln(|\mathcal{A} \times \mathcal{M}|)$ and $NP \not\subset TIME(|\mathcal{A} \times \mathcal{M}|^{O(\lg \lg |\mathcal{A} \times \mathcal{M}|)})$. By Theorem 38, the same algorithm could approximate SET-COVER withihn $(1-\epsilon) \cdot \ln(|\mathcal{H}|)$ and we would have $NP \not\subset TIME(|\mathcal{H}|^{O(\lg \lg |\mathcal{H}|)})$. Howevewr, if we could obtain such an approximation factor for SET-COVER, Theorem 4.4 of [46] tells us that $NP \subset TIME(|\mathcal{H}|^{O(\lg \lg |\mathcal{H}|)})$ - a contradiction. □

F.1.4 Proof of Theorem 41

Given BMGOP $(\mathcal{M}, s_0, \mathsf{B}, \mathcal{A}, \mathsf{C}, IC, k, \mathbf{c})$, finding an optimal solution SOL of action-point pairs cannot be approximated in PTIME within a ratio of $\frac{e-1}{e} + \epsilon$ for some $\epsilon > 0$ (where e is the inverse of the natural log) unless **P=NP**, even when $IC = \emptyset$. (There is no polynomial-time algorithm that can approximate an optimal solution within a factor of about 0.63 unless P=NP.)

Proof. Suppose, by way of contradiction, that an algorithm existed for finding a solution to a BMGOP within $1 - 1/e + \epsilon$ of optimal for some $\epsilon > 0$. Then we could use the construction of Theorem 39 to obtain an approximate solution to **MAX-K-COVER** within a factor of $1 - 1/e + \epsilon$ for some $\epsilon > 0$. By Theorem 5.3 of [46], this would imply **P==NP**, which contradicts the statement of the theorem. □

F.1.5 Proof of Theorem 42

Given GBGOP $(\mathcal{M}, s_0, \mathcal{A}, \mathsf{C}, IC, \mathbf{c}, \Theta_{in}, \Theta_{out})$, if the cost function and all actions $a \in \mathcal{A}$ can be polynomially computed, then determining if there is a solution SOL for the instance of the GBGOP s.t. for some real number k, $|SOL| \leq k$ is in-NP.

Proof. As all calculations of actions, and cost can be performed in PTIME, and checking if a given solution satisfies the integrity constraints can also be performed in PTIME, the verification of a solution is also achievable in PTIME, which gives us membership in the complexity class NP. □

F.1.6 Proof of Theorem 43

Given BMGOP $(\mathcal{M}, s_0, \mathsf{B}, \mathcal{A}, \mathsf{C}, IC, k, \mathbf{c})$, if the cost function, benefit function, and all actions $a \in \mathcal{A}$ can be polynomially computed, then determining if there is a solution SOL for the instance of the BMGOP s.t. for some real number val, $\sum_{A_i \in appl(SOL, s_0)} b_i \geq val$ is in-NP.

Proof. As all calculations of actions, cost, and benefit can be performed in PTIME, and checking if a given solution satisfies the integrity constraints can also be performed in PTIME, the verification of a solution is also achievable in PTIME, which gives us membership in the complexity class NP. □

F.1.7 Proof of Theorem 44

Counting the number of solutions to a GBGOP (under the assumptions of Theorem 42) is #P-complete.

Proof. CLAIM 1: There is a 1-1 encoding of MONSAT into a GBGOP.

The MONSAT problem is defined as per [145] below. INPUT: Set of m clauses C, each with K disjuncted literals, no literals are negations, L is the set of atoms, $|L| = n$.

OUTPUT: "Yes" iff there is a subset of L such that if the atoms in the subset are true, all of the clauses in C are satisfied.

We use the following encoding.

- $\mathcal{G} = \{g_1, \ldots, g_m\}$ - each predicate in $\mathcal{G}$ corresponds to an clause in C (predicate g_j corresponds with clause ϕ_j)
- $\mathcal{M}$ consists of a single point, p
- $\mathcal{A} = \{a_1, \ldots, a_n\}$ - each action in $\mathcal{A}$ corresponds to an element in L (action a_i corresponds with lieteral ℓ_i) . Each a_i is defined as follows: $a_i(p) = \{g_j(p) | \{\ell_i\} \models \phi_j$
- C returns 1 for all aciton-point pairs.

- $s_0 = \emptyset$, $IC = \emptyset$, $\mathsf{c} = n$
- $\Theta_{in} = \bigcup_{g_i \in \mathcal{G}} \{g_i(p)\}$
- $\Theta_{out} = \emptyset$

Clearly, the above construction can be accomplished in PTIME.

CLAIM 1.1 If $SOL \subseteq \mathcal{A} \times \mathcal{M}$ is a solution to GBGOP, then there exists $L' \subseteq L$ that is a solution to MONSAT where $|SOL| = |L'|$.

Consider the set of literals $L' = \{\ell_i | (a_i, p) \in SOL\}$. Clearly, $|L'| = |SOL|$. Suppose, BWOC, there is a clause $\phi_i \in C$ s.t. there is no $\ell \in L'$ where $\{\ell\} \models \phi$. Clearly, the element $g_i(p)$ is in Θ_{in}, so there must be some action-point-pair (a_j, p) in SOL s.t. $g_i(p) \in a_j(p)$ (otherwise, SOL is not a solution). This implies there is some $\ell_j \in L'$ that corresponds with a_j and, by the construction $\{\ell_j \models \phi$ - a contradiction.

CLAIM 1.2 If $L' \subseteq L$ is a solution to MONSAT then there exists $SOL \subseteq \mathcal{A} \times \mathcal{M}$ that is a solution to GBGOP where $|SOL| = |L'|$.

Consider the set $SOL = \{(a_i, p) | \ell_i \in L'\}$. Clearly $|L'| = |SOL|$. As $\mathsf{c} = n$ and $IC = \emptyset$, the first two requirements of SOL to be a solution are trivially met. The set $appl(\emptyset, \{(a_i, p) | \ell_i \in L'\}) = \bigcup_{\ell_j \in L'} g_j(p)$. Hence, as L' has a literal that satisfies each clause, and by the construction, we know the third requirement of being a solution is met.

CLAIM 2: Counting the solutions to a GBGOP is $\#P$-hard.

Using the construction of claim 1, we can embedd the counting version of MONSAT (number of solutions) into the counting version of a GBGOP (number of solutions). As there is a 1-1 correspondance, the reduction is parsimonious. Hence, using the construction of claim 1, if we find that there are N solutions to the GBGOP, the corresponding instance of MONSAT also has exactly N solutions.

CLAIM 3: Counting the solutions to a GBGOP is in the complexity class $\#P$.

We use the two requirements for membership in-#P.

(i) Witnesses must be verifiable in PTIME (shown in Theorem 42).

(ii) The number of solutions to GBGOP x^{Δ} - where x is depends on the input and Δ is a constant. We know that the number of solutions is bounded by $\sum_{i=0}^{|\mathcal{A}\times\mathcal{M}|}\binom{|\mathcal{A}\times\mathcal{M}|}{i}$ which is less than $\gamma \cdot |\mathcal{A} \times \mathcal{M}|^{|\mathcal{A}\times\mathcal{M}|}$ for some constant γ. □

F.1.8 Proof of Theorem 45

For some $\epsilon > 0$, approximating the number of solutions to a GBGOP within a factor of $2^{|\mathcal{A}\times\mathcal{M}|^{1-e}}$ is NP-hard.

Proof. Suppose, BWOC, there is some $\epsilon > 0$ s.t. there is a PTIME algorithm to approximate the number of solutions to a GBGOP with cardinality $\leq k$ within a factor of $2^{|\mathcal{A}\times\mathcal{M}|^{1-e}}$. Hence, the same algorithm could be used to approximate counting the solution to MONSAT (using the construction of Theorem 44 within a factor of $2^{n^{1-e}}$. However, this contradicts Theorem 3.2 of [145]. □

F.1.9 Proof of Lemma 22

Given GBGOP $\Gamma = (\mathcal{M}, s_0, \mathcal{A}, \mathsf{C}, IC, \mathbf{c}, \Theta_{in}, \Theta_{out})$, for any optimnal solution $SOL \subseteq R$, there is an optimal solution $SOL' \subseteq R^*$.

Proof. We show this by proving that for any set $W = SOL \cap (R - R^*)$, there is some set $W' \subseteq R^* - (R^* \cap SOL)$ s.t. $(SOL - W) \cup W'$ is also a solution. Hence, $|(SOL - W) \cup W'| = |SOL|$. By Definition 86, for any $(a_i, p_i) \in R - R^*$, there is some $(a_j, p_j) \in R^*$ s.t. $c_j \leq c_i$, (a_j, p_j) appears in the same or fewer integrity constraints, and $a_i(p_i) - (s_0 \cap a_i(p_i)) \subseteq a_j(p_j)$. Hence, for any $(a_i, p_i) \in W$, there is a corresponding element in W' that we can use to replace (a_i, p_i) without increasing cost, violating any integrity constraints, or not covering an element of Θ_{in}. $\square$

F.1.10 Proof of Proposition 66

Suppose $\Gamma = (\mathcal{M}, s_0, \mathcal{A}, \mathsf{C}, IC, \mathbf{c}, \Theta_{in}, \Theta_{out})$ is a GBGOP and $IP(\Gamma)$ is its corresponding integer program. We can create such a program with a variable for every element of R^* (instead of R) and the statement of Proposition 65 still holds true.

Proof. Follows directly from Lemma 22 and Proposition 65. $\square$

Appendix G

Appendix for Chapter 8

G.1 Proofs for Section 8.3

G.1.1 Proof of Proposition 70

If agg is positive-linear, then it is montonic.

Proof. Follows directly from Definitions 97-98. $\square$

G.1.2 Proof of Proposition 71

If a SNOP-query is not zero-starting w.r.t. a social network $\mathcal{S}$ and a GAP $\Pi \supseteq \Pi_{\mathcal{S}}$, and the aggregate is positive-linear, it can be expressed as a zero-starting SNOP-query in linear time while still maintaining a positive-linear aggregate.

Proof. CONSTRUCTION: Let $C = value(\emptyset)$.

Create a new SNOP query with aggregate $agg'(X) = agg(X) - C$.

We shall use the notation $value'$ to refer to the value function for the above construction.

CLAIM For any set $\mathbf{V}'$, $value(\mathbf{V}') = value(\mathbf{V}') + C$.

Follows directly from the construction. $\square$

G.1.3 Proof of Lemma 23

Given SNOP query $Q = (agg, VC, k, g(V))$ (w.r.t. SN $\mathcal{S}$ and GAP $\Pi \supseteq \Pi_{\mathcal{S}}$), if agg is monotonic (Definition 97), then $value$ (defined as per Q and Π) is montonic.

Proof. By the definition of $\mathbf{T}$, the annotation of any vertex atom montonically increases as we add more facts of the form $g(V) \leftarrow$ to the logic program. Hence, by the monotonicity of agg, the statement follows. $\square$

G.1.4 Proof of Lemma 24

Given SNOP query $Q = (agg, VC, k, g(V))$ (w.r.t. SN $\mathcal{S}$ and GAP $\Pi \supseteq \Pi_{\mathcal{S}}$), if VC is applied a-priori (as per Definition 101), the set of pre-answers (to query Q) is a uniform matroid.

Proof. Let $\mathbf{V}_{cond}$ be the set of veritces in $\mathbf{V}$ s.t. for each $v \in \mathbf{V}_{cond}$, $g(v) : 1 \wedge \bigwedge_{pred \in \ell_{vert}(v)} pred(v) : 1 \models VC[V/v]$.

CLAIM 1: For an a-priori VC SNOP query, any subset of $\mathbf{V}_{cond}$ of cardinality $\leq k$ is a pre-answer.

Suppose, BWOC, some subset of $\mathbf{V}' \subseteq \mathbf{V}_{cond}$ of cardinality $\leq k$ is not a pre-answer.

Obvouisly, all such subsets meet the cardinality requirement. Then, there must exist some $v' \in \mathbf{V'}$ s.t. $g(v') : 1 \wedge \bigwedge_{pred \in \ell_{vert}(v')} pred(v') : 1 \not\models VC[V/v']$. By Definition 101, this is a contradiction.

CLAIM 2: There is no subset $\mathbf{V'} \subseteq \mathbf{V}$ where $\mathbf{V'} \cap (\mathbf{V} - \mathbf{V}_{cond}) \not\equiv \emptyset$ that is a pre-answer.

Clearly, this would have an element that would not satisfy the a-priori VC, and hence, not be a pre-answer.

Proof of lemma: Any subset of size $\leq k$ of $\mathbf{V}_{cond}$ is a uniform matroid by definition. Also, from claims 1-2, we know that this family of sets also corresponds exactly with the set of pre-answers. Hence, the statement of the lemma follows. $\square$

G.1.5 Proof of Theorem 47

Given SNOP query $Q = (agg, VC, k, g(V))$ (w.r.t. SN $\mathcal{S}$ and GAP $\Pi \supseteq \Pi_{\mathcal{S}}$) if the following criteria are met:

- Π is a linear GAP

- VC is applied a-priori

- agg is positive linear,

then $value$ (defined as per Q and Π) is **sub-modular**.

In other words, for $\mathbf{V}_{cond} \equiv \{v' | v' \in \mathbf{V}$ and $(g(v') : 1 \wedge \bigwedge_{pred \in \ell_{vert}(v')} pred(v') : 1 \models$

$VC[V/v'])\}$ and sets $\mathbf{V}_1 \subseteq \mathbf{V}_2 \subseteq \mathbf{V}_{cond}$ and $v \in \mathbf{V}_{cond}$, $v \notin \mathbf{V}_1 \cup \mathbf{V}_2$, the following holds:

$$value(\mathbf{V}_1 \cup \{v\}) - value(\mathbf{V}_1) \geq value(\mathbf{V}_2 \cup \{v\}) - value(\mathbf{V}_2)$$

Proof. CLAIM 1: For some $\mathbf{V}'$, if $A_i : \mu_i \in \mathbf{T}_{\{\Pi \cup \{g(v'):1\leftarrow \,|\, v'\in\mathbf{V}'\}}$ s.t. there is no $\mu_i' > \mu_i$ where $A_i : \mu_i' \in \mathbf{T}_{\{\Pi \cup \{g(v'):1\leftarrow \,|\, v'\in\mathbf{V}'\}}$ then, there exists a polynomial of the following form:

$$f_i(X_1, \ldots, X_{|\mathbf{V}|}) = \lambda_1 \cdot X_1 + \ldots + \lambda_{|\mathbf{V}|} \cdot X_{|\mathbf{V}|} + \lambda_{|\mathbf{V}|+1}$$

s.t. if each X_i where $V_i \in \mathbf{V}'$ is set to 1 and each X_i where $V_i \notin \mathbf{V}'$ is set to 0, then $f_i(X_1, \ldots, X_{|\mathbf{V}|}) = \mu_i$.

(Proof of claim 1): Consider all of the rules in $\{\Pi \cup \{g(v') : 1 \leftarrow$. If $A_i : \mu_i \in \mathbf{T}_{\{\Pi \cup \{g(v'):1\leftarrow \,|\, v'\in\mathbf{V}'\}}$, then there must exist a rule that causes the annotation of A_i to equal μ_i. As the annotation in all rules is a linear function, we can easily re-write it in the above form, based on the presence of annotated atoms in the body formed with the goal predicate.

CLAIM 2: For some $\mathbf{V}'$, if $A_i : \mu_i \in \mathbf{T}_{\{\Pi \cup \{g(v'):1\leftarrow \,|\, v'\in\mathbf{V}'\}} \uparrow j$, s.t. there is no $\mu_i' > \mu_i$ where $A_i : \mu_i' \in \mathbf{T}\{\Pi \cup \{g(v') : 1 \leftarrow \,|\, v' \in \mathbf{V}'\} \uparrow j$ then, there exists a polynomial of the following form:

$$f_i(X_1, \ldots, X_{|\mathbf{V}|}) = \lambda_1 \cdot X_1 + \ldots + \lambda_{|\mathbf{V}|} \cdot X_{|\mathbf{V}|} + \lambda_{|\mathbf{V}|+1}$$

s.t. if each X_i where $V_i \in \mathbf{V}'$ is set to 1 and each X_i where $V_i \notin \mathbf{V}'$ is set to 0, then $f_i(X_1, \ldots, X_{|\mathbf{V}|}) = \mu_i$.

(Proof of claim 2): We will show that if the statement of the claim is true for the $j-1$ application of $\mathbf{T}$, then it is true for application j. The proof of the claim relies on this subclaim along with claim 1. If the claim holds for application $j-1$, then for each annotated atom $A_i' : \mu_i'$, there is an associated polynomial as per the statement. Consider the rule that fires in the jth application of the operator that causes rule A_i to be annotated with μ_i. We can re-write this as a polynomial of the above form, simply by substituting the polynomial for each annotation associated with A_i' from the previous iteration. As all of the polynomials are being substituted into variable positions of a polynomial, the result is still a polynomial, which can easily be re-arranged to resemble that of the claim.

CLAIM 3: For some $\mathbf{V}'$, if $A_i : \mu_i \in lfp(\mathbf{T}_{\{\Pi \cup \{g(v') : 1 \leftarrow \ | \ v' \in \mathbf{V}'\}})$, s.t. there is no $\mu_i' > \mu_i$ where $A_i : \mu_i' \in lfp(\mathbf{T}\{\Pi \cup \{g(v') : 1 \leftarrow \ | \ v' \in \mathbf{V}'\})$ then, there exists a polynomial of the following form:

$$f_i(X_1, \ldots, X_{|\mathbf{V}|}) = \lambda_1 \cdot X_1 + \ldots + \lambda_{|\mathbf{V}|} \cdot X_{|\mathbf{V}|} + \lambda_{|\mathbf{V}|+1}$$

s.t. if each X_i where $V_i \in \mathbf{V}'$ is set to 1 and each X_i where $V_i \notin \mathbf{V}'$ is set to 0, then $f_i(X_1, \ldots, X_{|\mathbf{V}|}) = \mu_i$.

(Proof of claim 3): Follows directly from claims 1-2.

CLAIM 4: For some $\mathbf{V}_i \subseteq \mathbf{V}$, there exists a polynomial of the following form:

$$f_i(X_1, \ldots, X_{|\mathbf{V}|}) = \lambda_1 \cdot X_1 + \ldots + \lambda_{|\mathbf{V}|} \cdot X_{|\mathbf{V}|} + \lambda_{|\mathbf{V}|+1}$$

s.t. if each X_i where $V_i \in \mathbf{V}'$ is set to 1 and each X_i where $V_i \notin \mathbf{V}'$ is set to 0, then

$f_i(X_1, \ldots, X_{|\mathbf{V}|}) = value(\mathbf{V}_i)$.

(Proof of claim 4): Consider all atoms formed with predicate *goal* in the lfp where the annotation is maximum. By claim 3, each is associated with a polynomial. A positive linear combination of all these polynomials is a polynomial of the form in this claim, and is equivalent to *value*.

CLAIM 5: $value(\mathbf{V}_1 \cup \{v\}) - value(\mathbf{V}_1) \geq value(\mathbf{V}_2 \cup \{v\}) - value(\mathbf{V}_2)$.

(Proof of claim 5): By the definition of *value*, as VC is applied a-priori, we know that *value* is defined on all subsets of $\mathbf{V}_{cond}$.

We define the following polynomial functions, which are associated with *value* for the various subsets of $\mathbf{V}$ in claim 5 (with some re-arrangement, Greek letters resemble constants, X variables can be either 0 or 1 - signifying if the associated subscript is includes in the associated set).

1. $f_1(X_{\mathbf{V}_1}, X_{\{v\}}, X_{\mathbf{V}_2-\mathbf{V}_1}) = \alpha_1 \cdot X_{\mathbf{V}_1} + \beta_1 \cdot X_{\{v\}} + \gamma_1 \cdot X_{\mathbf{V}_2-\mathbf{V}_1} + \lambda_1$

   $value(\mathbf{V}_1 \cup \{v\}) = f_1(1,1,0) = \alpha_1 + \beta_1 + \lambda_1$

2. $f_2(X_{\mathbf{V}_1}, X_{\{v\}}, X_{\mathbf{V}_2-\mathbf{V}_1}) = \alpha_2 \cdot X_{\mathbf{V}_1} + \beta_2 \cdot X_{\{v\}} + \gamma_2 \cdot X_{\mathbf{V}_2-\mathbf{V}_1} + \lambda_2$

   $value(\mathbf{V}_1) = f_2(1,0,0) = \alpha_2 + \lambda_2$

3. $f_3(X_{\mathbf{V}_1}, X_{\{v\}}, X_{\mathbf{V}_2-\mathbf{V}_1}) = \alpha_3 \cdot X_{\mathbf{V}_1} + \beta_3 \cdot X_{\{v\}} + \gamma_3 \cdot X_{\mathbf{V}_2-\mathbf{V}_1} + \lambda_3$

   $value(\mathbf{V}_2 \cup \{v\}) = f_3(1,1,1) = \alpha_3 + \beta_3 + \gamma_3 + \lambda_3$

4. $f_4(X_{\mathbf{V}_1}, X_{\{v\}}, X_{\mathbf{V}_2-\mathbf{V}_1}) = \alpha_4 \cdot X_{\mathbf{V}_1} + \beta_4 \cdot X_{\{v\}} + \gamma_4 \cdot X_{\mathbf{V}_2-\mathbf{V}_1} + \lambda_4$

   $value(\mathbf{V}_2) = f_4(1,0,1) = \alpha_4 + \gamma_4 + \lambda_4$

CLAIM 5.1: $\alpha_4 + \gamma_4 + \lambda_4 \geq \alpha_2 + \gamma_3 + \lambda_2$

(Proof of claim 5.1): We note that the constants in the f_i's defined earlier all correspond directly with constants seen in rules. Hence, as $f_4(1,0,1)$ corresponds with the maximum possible value for $value(\mathbf{V}_2)$, there can be no constants other than $\alpha_4, \gamma_4, \lambda_4$ that sum to a value greater than $value(\mathbf{V}_2)$. The statement of claim 5.1 immediately follows.

CLAIM 5.2: $\alpha_1 + \beta_1 + \lambda_1 \geq \alpha_3 + \beta_3 + \lambda_3$

(Proof of claim 5.2): Mirrors claim 5.1, (in this case, $value(\mathbf{V}_1 \cup \{v\})$ is the maximum possible value of $f_1(1,1,0)$).

(Completion of claim 5 / theorem): Suppose, BWOC, claim 5 is not true. Then, it must be the case that

$$value(\mathbf{V}_1 \cup \{v\}) - value(\mathbf{V}_1) < value(\mathbf{V}_2 \cup \{v\}) - value(\mathbf{V}_2)$$

This would imply:

$$\alpha_1 + \beta_1 + \lambda_1 + \alpha_4 + \gamma_4 + \lambda_4 < \alpha_3 + \beta_3 + \gamma_3 + \lambda_3 + \alpha_2 + \lambda_2$$

By claim 5.2, we have the following:

$$\alpha_4 + \gamma_4 + \lambda_4 < \gamma_3 + \alpha_2 + \lambda_2$$

Which contradicts claim 5.1. The statement of the theorem follows. $\square$

G.1.6 Proof of Theorem 48

Finding an answer to SNOP query $Q = (agg, VC, k, g(V))$ (w.r.t. SN $\mathcal{S}$ and GAP $\Pi \supseteq \Pi_{\mathcal{S}}$) is NP-hard (even if Π is a linear GAP, $VC = \emptyset$, $agg = SUM$ and $value$ is zero-starting).

Proof. The known NP-hard problem of max k-cover [46] as follows.

MAX K-COVER

INPUT: Set of elements, S and a family of subsets of S, $\mathcal{H} \equiv \{H_1, \ldots, H_{max}\}$, and positive integer K.

OUTPUT: $\leq K$ subsets from $\mathcal{H}$ s.t. the union of the subsets covers a maximal number of elements in S.

We shall make the following assumptions of **MAX-K-COVER**

1. $|\mathcal{H}| > K$
2. There is no $H \in \mathcal{H}$ s.t. $H \equiv \emptyset$

CONSTRUCTION: Given **MAX K-COVER** input $S, \mathcal{H}, K$ we create a SNOP-query as follows.

1. Set up social network $\mathcal{S}$ as follows:
    (a) $\mathsf{EP} \equiv \{edge\}$
    (b) $\mathsf{VP} \equiv \{vertex\}$
    (c) For every element of $\mathcal{H}$, and every element of S, we create an element of V. We shall denote subsets of V, V_S and $V_{\mathcal{H}}$ as the vertices corresponding

with S and $\mathcal{H}$ respectively. For some $s \in S$, v_s is the corresponding vertex. For some $H \in \mathcal{H}$, v_H is the corresponding vertex. Note that set $V \equiv V_S \cup V_{\mathcal{H}}$

(d) For each $H \in \mathcal{H}$, if $s \in H$ draw add edge (v_H, v_s) to set E

(e) For each $v \in V$, $\ell_{vert}(v) = vertex$

(f) For each $(v, v') \in E$, $\ell_{edge}(v, v') = edge$

(g) For each $(v, v') \in E$, $w(v, v') = 1$

2. Set up program Π as follows:

   (a) Embed $\mathcal{S}$ into Π.

   (b) Add diffusion rule $vertex(V) : X \leftarrow vertex(V') : X \wedge edge(V', V) : 1$ to Π

3. Set up SNOP-query Q as follows:

   (a) $agg = SUM$

   (b) $VC = true$

   (c) $k = K$ (the K from SET_COVER)

   (d) $g = vertex$

Additionally, we will use the following notation:

1. V' is a pre-answer to the constructed query

2. $value(V')$ is the value of the constructed query for pre-answer V'

3. V'_{ans} is an answer to the constructed query

CLAIM 1: The construction can be performed in PTIME.

Straightforward.

CALIM 2: Program Π is a linear GAP.

Follows directly form Definition 96.

CLAIM 3: An answer V'_{ans} to the SNOP query cannot contain a vertex in $v_s \in V_S$ **and** a vertex in $v_H \in V_{\mathcal{H}}$ s.t. $s \in H$.

BWOC, an optimal solution could have an element v_s as described in the claim. By assumption 1, there are more than K elements in $V_{\mathcal{H}}$ and all of them have an edge to some element of V_S by assumption 2. It is obvious that v_s will be annotated with a 1 in the fixed point, and that no elements of $V_{\mathcal{H}} - V'_{ans}$ will be annotated with 1 in the fixed point. Hence, we can pick any element of $V_{\mathcal{H}} - V'_{ans}$ and *value* will be at least one greater than the "optimal" solution – hence a contradiction.

CLAIM 4: If an answer $V'_{ans} \cap V_S \not\equiv \emptyset$, then we can construct an alternative optimal solution such that $V'_{ans} \cap V_S \equiv \emptyset$.

As no element in $V'_{ans} \cap V_S \not\equiv \emptyset$ has an outgoing neighbor, and by assumption 1, we can be assured that $|V'_{ans} - V_{\mathcal{H}}| > |V'_{ans} \cap V_S|$, we can replace the elements of $V'_{ans} \cap V_S$ in V'_{ans} with elements from $V'_{ans} - V_{\mathcal{H}}$ and still be ensured of an optimal solution.

CLAIM 5: Given a set $\mathcal{H}' \subseteq \mathcal{H}$ that ensures an optimal solution to **MAX-K-COVER**, we can construct an optimal V'_{ans} to the SNOP query.

CASE 1 (claim 5): $|\mathcal{H}'| = K$.

Let OPT be the number of elements of S covered in the optimal solution of **MAX-K-COVER**. For each $H \in \mathcal{H}'$, we pick the corresponding element of $V_{\mathcal{H}}$. Obviously, $value(V'_{ans}) = OPT + K$. Suppose, we could pick a different element of $\mathbf{V}$ and get a solution with a higher *value*. As no element of S has an outgoing edge, replacing one of the elements from the constructed set with one of these will not ensure a greater solution. If we could pick an element from $V_{\mathcal{H}} - V'_{ans}$, then this would obviously imply a solution to **MAX-K-COVER** s.t. more than OPT elements of S are covered – clearly this is a contradiction as $\mathcal{H}'$ is an optimal cover.

CASE 2 (claim 5): $|\mathcal{H}'| < K$.

Create $\mathcal{H}''$ with all of the elements of $\mathcal{H}'$ and $K - |\mathcal{H}'|$ elements of $\mathcal{H} - \mathcal{H}'$. Clearly, this is also an optimal solution to **MAX-K-COVER** (as cardinality is not optimized, just needs to be below K). We can now apply case 1 of this claim.

CLAIM 6: Given V'_{ans}, we can constructively create a subset of $\mathcal{H}$ that, if picked, ensures an optimal solution to **MAX-K-COVER**.

CASE 1 (claim 6): $V'_{ans} \subseteq V_{\mathcal{H}}$

Simply pick each H associated with each $v_H \in V'_{ans}$. Let $OPT' = value(V'_{ans})$ note that $OPT' = K + SPREAD$ where $SPREAD$ corresponds with the number of 1-annotated elements of V_S in the fixed point. If there is a different subset of $\mathcal{H}$ that can be picked, (i.e. a more optimal solution to **MAX-K-COVER**), then we can create a solution to the SNOP query where some $SPREAD' > SPREAD$ elements of V_S become annotated with 1 in the fixed point. Clearly, this would imply a more optimal solution to the SNOP query – a contradiction.

CASE 2 (claim 6): $V_S - V'_{ans} \not\equiv \emptyset$

From this solution, we can use claim 4 to create an optimal solution s.t. case 1 applies.

The proof of the theorem follows directly from claims 5-6. $\square$

G.1.7 Proof of Theorem 49

Finding an answer to a decision problem associated with SNOP query $Q = (agg, VC, k, g(V))$ (w.r.t. SN $\mathcal{S}$ and GAP $\Pi \supseteq \Pi_{\mathcal{S}}$) where agg and the functions in $\mathcal{F}$ are polynomially computable is in-NP.

Proof. We utilize the following decision problem:

Definition 117 (SNOP-DEC)**.** *An instance of the decision problem related to a SNOP-query accepts the input for the query plus real number target. The decision problem returns "yes" iff there exists pre-answer V' s.t. $value(V') \geq target$ and* *"no" otherwise.*

CLAIM 1: SNOP-DEC is NP-hard.

We do this by a reduction from SET_COVER.

CONSTRUCTION: Given instance $S, \mathcal{H}, K$ of SET_COVER, we create K instances of SNOP-DEC, each identified with index $i \in [1, K]$, that each use the same construction used to show the NP-hardness of a SNOP query with the following two exceptions:

- Set k in SNOP-DEC to i
- Set $target$ in SNOP-DEC to $i + |S|$

CLAIM 1.1: The construction can be performed in PTIME.

Straightforward. CLAIM 1.2: If there is a solution to the set cover problem, at least one of the constructed instances of SNOP-DEC will return "yes."

Suppose, that there is a solution to the set-cover problem, that causes the selection of m elements of $\mathcal{H}$ (where $m \leq K$). By the construction, there exists an instance of SNOP-DEC such that $target = m + |S|$ and $k = m$. We simply pick the k vertices in $V_{\mathcal{H}}$ corresponding with the covers, and by the construction, after running Π, all of the vertices in V_S will have an annotation to the vertex atoms formed by *marked* of 1. Hence, the aggregate will be $m + |S|$ - which is greater than $target$, so that instance of SNOP-DEC returns "yes."

CLAIM 1.3: If there is no solution to the set cover problem, all of the instances of SNOP-DEC will return "no."

Suppose there is no solution to $\mathsf{SET\_COVER}$ and one of the constructed instances of SNOP-DEC returns "yes." Then, for some $i \in [1, K]$, there are i vertices that can be picked to change the annotation of the *vertex* vertex atoms to ensure that the aggregate is greater than or equal to $i + |S|$. As, at most, only i vertex atoms can be picked, and only atoms in V_S can change annotation due to Π, all i vertices associated with the vertex atoms must be in $V_{\mathcal{H}}$ to ensure that we have the most possible vertex atoms formed with *vertex* that have a non-zero annotation. However, in order for all of the vertices in V_S to have the annotations of the associated *vertex* vertex atom increase to 1, there must be at least one incoming edge to each element of V_S from one of the i atoms from $V_{\mathcal{H}}$. By how $\mathcal{S}$ is constructed, this would imply a set-cover of size i, which would be a contradiction.

PROOF OF CLAIM 1: Follows directly from claims 1.1-1.3.

CLAIM 2: SNOP-DEC is in-NP (with the conditions in the statement).

Suppose, we are given a set V'. We can easily verify this solution in PTIME as follows: (i) verify V' is a valid pre-answer can easily be done in PTIME by checking that $|V'| \leq k$ and that $\forall v' \in V'$, $VC(v')$ is true. (ii) by the assumptions about agg and the functions in $\mathcal{F}$, we can compute $value(V')$ in PTIME as well. the statement follows. $\Box$

G.1.8 Proof of Theorem 50

Answering a SNOP query $Q = (agg, VC, k, g(V))$ (w.r.t. SN $\mathcal{S}$ and GAP $\Pi \supseteq \Pi_{\mathcal{S}}$), cannot be approximated in PTIME within a ratio of $\frac{e-1}{e} + \epsilon$ for some $\epsilon > 0$ (where e is the inverse of the natural log) unless **P==NP** – even if Π is a linear GAP, $VC = \emptyset$, $agg = SUM$ and *value* is zero-starting.

(That is, there is no polynomial-time algorithm that can approximate *value* within a factor of about 0.63 under standard assumptions.)

Proof. Suppose, BWOC, there is an α-approximation algorithm for an SNOP query. Hence, we can approximate *value* returned by SNOP within a factor of $1 - 1/e + \epsilon$ for some $\epsilon > 0$. Using the **MAX-K-COVER** reduction in Theorem 48, for SNOP answer V'_{ans}, the cardinality of the covered elements of S in **MAX-K-COVER** is $value(V'_{ans}) - K$. Hence, this approximation algorithm would provide a solution to **MAX-K-COVER** within a factor of $1 - 1/e + \epsilon$ for some $\epsilon > 0$. By Theorem 5.3 of [46], this would imply **P==NP**, which contradicts the statement of the theorem. $\square$

G.1.9 Proof of Theorem 51

Counting the number of answers to SNOP query $Q = (agg, VC, k, g(V))$ (w.r.t. SN $\mathcal{S}$ and GAP $\Pi \supseteq \Pi_{\mathcal{S}}$) is #P-complete.

Follows directly from Lemmas 32 and 33.

Lemma 32. *The counting version of the SNOP query answering problem (we shall call it #SNOP) is #P-hard.*

Proof. We now define the known #P-Complete problem, MONSAT [145] and a variant of it used in this proof:

Counting K-Monotone CNF Sat. (#MONSAT)

INPUT: Set of clauses C, each with K disjuncted literals, no literals are negations, L is the set of atoms.

OUTPUT: Number of subsets of L such that if the atoms in the subset are true, all of the clauses in C are satisfied.

Counting K-Monotone CNF Sat. - Exact (#MONSAT-EQ)

INPUT: Set of clauses C, each with K disjuncted literals, no literals are negations, L is the set of atoms and natural number m.

OUTPUT: Number of subsets of L - each with cardinality of exactly m - such that if the atoms in the subset are true, all of the clauses in C are satisfied.

We now define the following problem used in the proof:

#SNOP-EQ

INPUT: Same as SNOP-DEC.

OUTPUT: Number of pre-answers V' that would causes a "yes" answer to SNOP-DEC *and* $|V'| = k$.

CLAIM 1: #MONSAT$\leq_p$#MONSAT-EQ and #MONSAT-EQ is #P-hard Consider the following construction (CONSTRUCTION 1):

Let L be the set of atoms associated with #MONSAT. Create $|L|$ instances of #MONSAT-EQ - each with a cardinality constraint (m) in $[1, |L|]$, and the remainder of the input the same as #MONSAT.

(Proof of claim 1): The sum of the solution to the $|L|$ instances of #MONSAT-EQ is equal to the solution to #MONSAT.

Every possible satisfying assignment counted as a solution to #MONSAT has a unique cardinality associated with it, which is in $[1, |L|]$. The claim follows trivially from this fact and construction 1 (which can be performed in PTIME).

CLAIM 2: #MONSAT-EQ$\leq_p$#SNOP-EQ and #SNOP-EQ is #P-hard

Consider the following construction (CONSTRUCTION 2):

Given #MONSAT-EQ input (C, K, L, m), we create an instance of #SNOP-EQ as follows.

1. Set up social network $\mathcal{S}$ as follows:

    (a) $\mathsf{EP} \equiv \{edge\}$

    (b) $\mathsf{VP} \equiv \{vertex\}$

    (c) For every element of C, and every element of L, we create an element of V. We shall denote subsets of V, V_C and V_L as the vertices corresponding with C and L respectively. For some $a \in C$, v_a is the corresponding vertex. For some $b \in L$, v_b is the corresponding vertex.

(d) For each $a \in C$, if b is in clause C, add edge (v_b, v_a) to set E

(e) For each $v \in V$, $\ell_{vert}(v) = vertex$

(f) For each $(v, v') \in E$, $\ell_{edge}(v, v') = edge$

(g) For each $(v, v') \in E$, $w(v, v') = 1$

2. Set up program Π as follows:

   (a) Embed $\mathcal{S}$ into Π

   (b) For each $v \in V$, add fact $vertex(v) : 0$ to Π

   (c) Add diffusion rule $vertex(v) : 1 \leftarrow vertex(v') : 1 \wedge edge(v', v) : 1$ to Π

3. Set up SNOP-query Q as follows:

   (a) $agg = SUM$

   (b) $VC = \emptyset$

   (c) $k = m$ (the m from #MONSAT-EQ)

   (d) $g = vertex$

   (e) $target = |C| + k$

CLAIM 2.1: Construction 2 can be performed in PTIME.

Straightforward.

CLAIM 2.2: If there is a solution to given an instance of MONSAT-EQ, then given construction 2 as input, SNOP-EQ will return "yes". For each $a \in L$ in the solution to MONSAT-EQ, change the annotation of $vertex(v_a)$ to 1 in Π_{facts}. There are

$m = k$ such vertices. By the construction, this will cause the $|C|$ vertices of V_C to increase their annotation - resulting in an aggregate of $|C| + k$, causing SNOP-EQ to return "yes".

CLAIM 2.3: If, given construction 2 as input, SNOP-EQ returns "yes", then a solution to given an instance of MONSAT-EQ such that k is the cardinality of the solution.

We note that selecting any vertex in V' not in V_L will result in an $value(V') < |C|+k$, as fewer than $|C|$ nodes will have their annotation increase after running Π. The only way to achieve an $value(V') = |C| + k$ is if there exists a set of k vertices in V_L such that there is an outgoing edge from at least one of the picked vertices to each node in V_C. This is only possible if there exists a solution to the MONSAT-EQ problem.

CLAIM 2.4: There is a 1-1 correspondence between solution to MONSAT-EQ and SNOP-EQ using construction 2.

As each literal in a MONSAT-EQ solution corresponds to exactly one vertex in a SNOP-EQ, and by claims 2.2-2.3, the claim follows.

PROOF OF CLAIM 2: Follows directly from claims 2.1-2.4.

CLAIM 3: #SNOP-EQ$\leq_p$#SNOP, #SNOP is #P-hard

Consider the following construction (CONSTRUCTION 3):

Let k be the cardinality constraint associated with #SNOP-EQ. Create two instances of #SNOP, one with a cardinality constraint of k and one with the constraint of $k-1$, and the remainder of the input is the same as #SNOP-EQ.

PROOF OF CLAIM 3: First, note that construction 3 can be performed in PTIME. We show that the solution to #SNOP with cardinality constraint $k-1$ subtracted from the solution to #SNOP with cardinality constraint k is the solution to #SNOP-EQ. As the solution to #SNOP with cardinality constraint $k-1$ is the number of all V''s that are a solution with cardinality of $k-1$ or less, and the solution to #SNOP with cardinality constraint k is the number of all V''s that are a solution with cardinality of k or less, the difference is the number of all V''s with a cardinality of exactly k.

PROOF OF LEMMA: Follows directly from claims 3. $\square$

Lemma 33. *If the aggregate function agg is polynomially computable and functions in $\mathcal{F}$ are polynomially computable, then #SNOP is in-#P.*

Proof. We use the two requirements for membership in-#P as presented in [87].

(i) Witnesses must be verifiable in PTIME (shown in the NP-Completness of a SNOP-query).

(ii) The number of solutions to #SNOP is bounded by $x'^{k'}$ - where k' is a constant. We know that the number of solutions is bounded by $\sum_{i\leq k}\binom{|V|}{i}$ which is less than $c\cdot|V|^k$ for some constant c. $\square$

G.1.10 Proof of Theorem 52

Given SNOP query $Q = (agg, VC, k, g(V))$ (w.r.t. SN $\mathcal{S}$ and GAP $\Pi \supseteq \Pi_{\mathcal{S}}$), finding $\bigcup_{V'_{ans} \in \mathsf{ans}(Q)} V'_{ans}$ is NP-hard.

Proof. We shall refer to the problem of finding $\bigcup_{V'_{ans} \in \mathsf{ans}(Q)} V'_{ans}$ as SNOP-ALL. We show that SNOP-ALL is $\leq_p$ solving a SNOP-query.

Given set an instance of SNOP-ALL and vertex set V^*, $|V^*| \leq k$ let SNOP-ALL(V^*) be the modification of of the instance of SNOP-ALL where the value k is reduced by $|V^*|$ and for each $v_j^* \in V^*$, the fact $g(v_i) : 1$ is added to Π.

Consider the following informal algorithm (FIND-SET) that takes an instance of SNOP-ALL (Q) and some vertex set V^*, $|V^*| \leq k$.

1. If $|V^*| = k$, return V^*
2. Else, solve SNOP-ALL(V^*), returning set V''.
   (a) If $V'' - V^* \equiv \emptyset$, return V^*
   (b) Else, pick $v \in V'' - V^*$ and return FIND-SET$(Q, V^* \cup v)$

Note, that the above algorithm can only iterate k times.

CLAIM 1: The V^* returned by FIND-SET is a valid solution to the SNOP-query (with the same input for Q).

First, we number the elements in V^* as $v_1, \ldots, v_{size}$ - where v_1 is picked as the first element in the solution and vertex v_i is added at the ith recursive call of FIND-SET. We know that $size \leq k$

BASE CASE: There is a set of vertices of size $\leq$ $size$ that is a solution to the

SNOP-query s.t. vertex v_1 is in that set - follows directly from the definition of SNOP-ALL.

INDUCTIVE HYPOTHESIS: For some $k' \leq size$, we assume that for vertices $v_1, \ldots, v_{k'-1}$ there is some set of vertices of size $\leq k$ that is a solution to the SNOP-query s.t. vertices $v_1, \ldots, v_{k'-1}$ are in that set.

INDUCTIVE STEP: For some $k' \leq size$, consider vertices $v_1, \ldots, v_{k'}$. By the inductive hypothesis, vertices $v_1, \ldots, v_{k'-1}$ are in a $\leq k$-sized solution. By the construction, and the definition of SNOP-ALL, we know that vertex $v_{k'}$ must also be in that set as well.

CLAIM 2: Given some V' as a solution to the SNOP-query, the algorithm FIND-SET can be run in such a way to return that set.

Number each vertex in V' as $v_1, \ldots, v_{size}$. By the definition of SNOP-ALL, upon the i'th call to FIND-SET, we are guaranteed that the vertices $v_i, \ldots, v_{size}$ will be in set V''. Simply pick vertex v_i follow the algorithm to the next recursive call, the claim immediately follows.

PROOF OF PROPOSITION: Note the construction can be accomplished in PTIME. The proposition follows directly from claims 1-2. $\square$

G.1.11 Proof of Theorem 53

Given SNOP query $Q = (agg, VC, k, g(V))$ (w.r.t. SN $\mathcal{S}$ and GAP $\Pi \supseteq \Pi_{\mathcal{S}}$), finding $\bigcup_{V'_{ans} \in \mathsf{ans}(Q)} V'_{ans}$ reduces to $|V| + 1$ SNOP-queries.

Proof. We set up $|V|$ SNOP-queries as follows:

- Let k_{all} be the k value for the SNOP-ALL query and and for each SNOP-query i, let k_i be the k for that query. For each query i, set $k_i = k_{all} - 1$.
- Number each element of $v_i \in V$ such that $g(v_i)$ and $VC(v_i)$ are true. For the ith SNOP-query, let v_i be the corresponding element of V
- Let Π_i refer to the GAP associated with the ith SNOP-query and Π_{all} be the program for SNOP-ALL. For each program Π_i, add fact $g(v_i) : 1$
- For each SNOP-query i, the remainder of the input is the same as for SNOP-ALL.

After the construction, do the following:

1. We shall refer to a SNOP-query that has the same input as SNOP-ALL as the "primary query." Let ${V'_{ans}}^{(pri)}$ be an answer to this query and $value({V'_{ans}}^{(pri)})$ be the associated value.
2. For each SNOP-query i, let ${V'_{ans}}^{(i)}$ be an answer and $value({V'_{ans}}^{(i)})$ be the associated value.
3. Let V'', the solution to SNOP-ALL be initialized as $\emptyset$.
4. For each SNOP-query i, if $value({V'_{ans}}^{(i)}) = value({V'_{ans}}^{(pri)})$, then add vertex v_i to V''.

CLAIM 1: If for the ith SNOP-query, if $value({V'_{ans}}^{(i)}) = value({V'_{ans}}^{(pri)})$, then v_i must be in the solution to SNOP-ALL.

Suppose, by way of contradiction, that for the ith query, $value({V'_{ans}}^{(i)}) = value({V'_{ans}}^{(pri)})$, but v_i is not in the solution to SNOP-ALL. Then, there is no V' of size $\leq k$ s.t. $v_i \in V'$ and V' is an answer to a the primary SNOP-query. However, this is a contradiction, as given v_i and the vertices returned by the ith query, we are guaranteed this to be a valid answer to the primary query.

CLAIM 2: For each v_i in a solution to SNOP-ALL, the ith SNOP query returns a value s.t. $value({V'_{ans}}^{(i)}) = value({V'_{ans}}^{(pri)})$.

Suppose, by way of contradiction, that there is some v_i in the solution to SNOP-ALL s.t. the ith query returns a value that is not equal to the value returned by the primary. However, by the definition of SNOP-ALL, this is not possible, hence a contradiction.

PROOF OF PROPOSITION: Note the construction can be accomplished in PTIME. The proposition follows directly from claims 1-2. □

G.2 Proofs for Section 8.5

G.2.1 Proof of Proposition 72

Suppose Π is any GAP. Then:

1. $\mathbf{S}_\Pi$ is monotonic.

2. $\mathbf{S}_\Pi$ has a least fixpoint $lfp(\mathbf{S}_\Pi)$ and $lfp(\mathbf{T}_\Pi) = grd(lfp(\mathbf{S}_\Pi))$.

   That is, $lfp(\mathbf{S}_\Pi)$ is a non-ground representation of the (ground) least fixpoint operator $\mathbf{T}_\Pi$.

Proof. Part 1 follows directly from the definition – for a given atom A and interpretation I, $\mathbf{S}(I)(A) \geq I(A)$.

Part 2 follows directly from the definitions of $\mathbf{S}$ and $\mathbf{T}$. $\square$

G.2.2 Proof of Theorem 54

Given SNOP query $Q = (agg, VC, k, g(V))$ (w.r.t. SN $\mathcal{S}$ and GAP $\Pi \supseteq \Pi_{\mathcal{S}}$), if agg is monotonic then:

- There is an answer to the SNOP-query Q w.r.t. the GAP Π iff SNOP-Mon$(\Pi, agg, VC, k, g(V))$ does not return NIL.

- If SNOP-Mon$(\Pi, agg, VC, k, g(V))$ returns any result other than NIL, then that result is an answer to the SNOP-query Q w.r.t. the GAP Π.

Proof. Part 1 ($\Leftarrow$): Suppose there is an answer to the query and SNOP-Mon returns NIL. Then there is some set of vertices, sol of cardinality $\leq k$, s.t. $\Pi \cup \bigcup_{v \in sol} g(v) : 1 \models VC$. However, such a set would obviously have been added as a tuple into $Todo$ at step 2 or step 4(c)iB. Hence, a contradiction.

Part 1 ($\Rightarrow$): Suppose there is no answer to the query and SNOP-Mon returns NIL. Then, there is no set of vertices, sol of cardinality $\leq k$, s.t. $\Pi \cup \bigcup_{v \in sol} g(v) : 1 \models VC$. SNOP-Mon performs such a check at line 4b. Hence, a contradiction.

Part 2: Suppose, BWOC, there exists a set of vertices that is a solution, sol, of cardinality $\leq k$, s.t. $\bigcup_{v \in sol} g(v) : 1$ is not what is returned by SNOP-Mon and $value(\Pi \cup \bigcup_{v \in sol} g(v) : 1$ is greater than $bestVal$. We note that SNOP-Mon considers most sets of vertices of cardinality $\leq k$. Further, the monotonicity of agg and line 4(c)i tell us that the only solutions not considered are ones guaranteed to have a $value$ less than $bestVal$ – hence, a contradiction. $\square$

G.2.3 Proof of Proposition 73

Given SNOP query $Q = (agg, VC, k, g(V))$ (w.r.t. SN $\mathcal{S}$ and GAP $\Pi \supseteq \Pi_{\mathcal{S}}$), the complexity of GREEDY-SNOP is $O(k \cdot |\mathbf{V}| \cdot F(|\mathbf{V}|))$ where $F(|\mathbf{V}|)$ is the time complexity to compute $value(V')$ for some set $V' \subseteq \mathbf{V}$ of size k.

Proof. The outer loop at line 2 iterates k times, the inner loop at line 2b iterates $O(|\mathbf{V}|)$ times, and at each inner loop, at line 2(b)i, the function $value$ is computed with costs $F(|\mathbf{V}|)$. The statement follows. $\square$

G.2.4 Proof of Theorem 55

If SNOP query $Q = (agg, VC, k, g(V))$ (w.r.t. SN $\mathcal{S}$ and GAP $\Pi \supseteq \Pi_{\mathcal{S}}$) meets the following criteria:

- Π is a linear GAP

- VC is applied a-priori

- agg is positive linear

- $value$ is zero-starting.

Then GREEDY-SNOP is an $(\frac{e}{e-1})$-approximation algorithm for the query.

Proof. The results of [127] state that a greedy algorithm for a non-decreasing, submodularity function F s.t. $F(\emptyset) = 0$ is a $\frac{e}{e-1}$ approximation algorithm for the associated maximization problem. In Section 8.3, we show that a query meeting the criteria of the statement satisfies the requirements. The statement follows. $\square$

G.2.5 Proof of Proposition 74

For all ground atoms A and vertices v, $INC_{i-1}(v)(A) \geq INC_i(v)(A)$.

Proof. Consider the following values: $I_{i-1}(v)(A), I_{i-2}^{(alg)}(A), I_i(v)(A), I_{i-1}^{(alg)}(A)$. These correspond with the following sets of vertices, respectively: $SOL_{i-2} \cup \{v\}, SOL_{i-2}, SOL_{i-2} \cup \{v\} \cup (SOL_{i-1} - SOL_{i-2}), SOL_{i-2} \cup (SOL_{i-1} - SOL_{i-2})$. Hence, by claim 3 of Theorem 47, we can associate the values $I_{i-1}(v)(A), I_{i-2}^{(alg)}(A), I_i(v)(A), I_{i-1}^{(alg)}(A)$ with linear functions with three variables corresponding to the sets $SOL_{i-2}, \{v\}, (SOL_{i-1} - SOL_{i-2})$. If the variables corresponding to the set of vertices are set to 1 and the rest zero, then the function corresponds to the value assigned to A by that interpretation. Consider the following four functions:

$$\begin{aligned}
f_1(X_1, X_2, X_3) &= a_1 \cdot X_1 + b_1 \cdot X_2 + c_1 \cdot X_3 + d_1 \\
f_2(X_1, X_2, X_3) &= a_2 \cdot X_1 + b_2 \cdot X_2 + c_2 \cdot X_3 + d_2 \\
f_3(X_1, X_2, X_3) &= a_3 \cdot X_1 + b_3 \cdot X_2 + c_3 \cdot X_3 + d_3 \\
f_4(X_1, X_2, X_3) &= a_4 \cdot X_1 + b_4 \cdot X_2 + c_4 \cdot X_3 + d_4
\end{aligned}$$

Where X_1, X_2, X_3 correspond to $SOL_{i-2}, \{v\}, (SOL_{i-1} - SOL_{i-2})$ respectively.

$$\begin{aligned}
f_1(1,1,0) &= I_{i-1}(v)(A) &&= a_1 + b_1 + d_1 \\
f_2(1,0,0) &= I^{(alg)}_{i-2}(A) &&= a_2 + d_2 \\
f_3(1,1,1) &= I_i(v)(A) &&= a_3 + b_3 + c_3 + d_3 \\
f_4(1,0,1) &= I^{(alg)}_{i-1}(A) &&= a_4 + c_4 + d_4
\end{aligned}$$

Note 1: We note, using the same techniques as claims as 5.1-5.2 of Theorem 47, that there is no i_1, i_2, i_3 not equal to 1 where $a_{i_1} + b_{i_2} + d_{i_3} > a_1 + b_1 + d_1$ and no i_1, i_2, i_3 not equal to 4 where $a_{i_1} + c_{i_2} + d_{i_3} > a_4 + c_4 + d_4$.

So, suppose, BWOC, the statement of the proposition does not hold. Then, we have:

$$\begin{aligned}
a_1 + b_1 + d_1 - a_2 - d_2 &< a_3 + b_3 + c_3 + d_3 - a_4 - c_4 - d_4 \\
a_1 + b_1 + d_1 + a_4 + c_4 + d_4 &< a_3 + b_3 + c_3 + d_3 + a_2 + d_2
\end{aligned}$$

And by **Note 1**,

$$a_1 + b_1 + d_1 > b_3 + a_2 + d_2$$

Which, subtracting from both sides, gives us:

$$a_4 + c_4 + d_4 < a_3 + c_3 + d_3$$

Which contradicts **Note 1**. The statement of the proposition follows. □

G.2.6 Proof of Lemma 25

For all programs Π and any atom A,

$$lfp(S_{PROG(\Pi)})(A) = lfp(S_{\Pi})(A)$$

Proof. Follows directly from Definition 105. □

G.2.7 Proof of Lemma 26

If $\Pi_3 \equiv \Pi_1 \cup \Pi_2$, then for any atom A,

$$lfp(\mathbf{S}_{\Pi_3})(A) = lfp(\mathbf{S}_{PROG(\Pi_1) \cup PROG(\Pi_2)})(A)$$

Proof. CLAIM 1: $lfp(\mathbf{S}_{\Pi_3})(A) \geq lfp(\mathbf{S}_{PROG(\Pi_1) \cup PROG(\Pi_2)})(A)$

By the monotonicity of $\mathbf{S}$, we know that for all A, $lfp(S_{\Pi_3})(A) \geq lfp(S_{\Pi_1})(A)$ and $lfp(S_{\Pi_3})(A) \geq lfp(S_{\Pi_2})(A)$. Further, as $\Pi_1, \Pi_2 \subseteq \Pi_3$, it follows that $PROG(\Pi_1), PROG(\Pi_2) \subseteq PROG(\Pi_3)$, meaning that $PROG(\Pi_1) \cup PROG(\Pi_2) \subseteq PROG(\Pi_3)$.

By the monotonicity of $\mathbf{S}$, it follows that for any atom A, $lfp(\mathbf{S}_{\Pi_3})(A) \geq lfp(\mathbf{S}_{PROG(\Pi_1) \cup PROG(\Pi_2)})(A$

CLAIM 2: $lfp(\mathbf{S}_{\Pi_3})(A) \leq lfp(\mathbf{S}_{PROG(\Pi_1) \cup PROG(\Pi_2)})(A)$

Going the other direction, by Definition 105, $\Pi_1 \subseteq PROG(\Pi_1)$ and $\Pi_2 \subseteq PROG(\Pi_2)$. Therefore, for all A, $lfp(\mathbf{S}_{\Pi_1 \cup \Pi_2})(A) \leq lfp(\mathbf{S}_{PROG(\Pi_1) \cup PROG(\Pi_2)})(A)$, which means that $lfp(\mathbf{S}_{\Pi_3})(A) \leq lfp(\mathbf{S}_{PROG(\Pi_1) \cup PROG(\Pi_2)})(A)$.

The statement of the lemma follows directly from claims 1-2. □

G.2.8 Proof of Proposition 75

If $\Pi_3 \equiv \Pi_1 \cup \Pi_2$, then for any atom A,

$$lfp(\mathbf{S}_{\Pi_3})(A) = lfp(\mathbf{S}_{PROG(PROG(\Pi_1) \cup PROG(\Pi_2))})(A)$$

Proof. By Lemma 25,

$$lfp(\mathbf{S}_{PROG(PROG(\Pi_1) \cup PROG(\Pi_2))})(A) = lfp(\mathbf{S}_{PROG(\Pi_1) \cup PROG(\Pi_2)})(A)$$

By Lemma 26, the statement of the proposition follows. □

G.2.9 Proof of Proposition 76

$inc_i^{(opt)} \leq inc_{i-1}^{(opt)}$.

Proof. This proposition is equivalent to the statement for all $\mathbf{V}' \subseteq \mathbf{V}$ and all $v, v' \notin \mathbf{V}'$, then

$$value(\mathbf{V}' \cup \{v, v'\}) - value(\mathbf{V}' \cup \{v'\}) \leq value(\mathbf{V}' \cup \{v\}) - value(\mathbf{V}')$$

Where v is the vertex added by the greedy algorithm at iteration $i-1$ and v' is the vertex added by the greedy algorithm at iteration i. Obviously, as $\mathbf{V}' \cup \{v\} \supseteq \mathbf{V}'$, this is a special case of submoduarity, which is proved for this special case of queries in Theorem 47. □

G.2.10 Proof of Corollary 14

$inc_i(v) \leq inc_{i-1}(v)$.

Proof. This proposition is equivalent to the statement for all $\mathbf{V}' \subseteq \mathbf{V}$ and all $v, v' \notin \mathbf{V}'$, then

$$value(\mathbf{V}' \cup \{v, v'\}) - value(\mathbf{V}' \cup \{v'\}) \leq value(\mathbf{V}' \cup \{v\}) - value(\mathbf{V}')$$

Where v' is the vertex added at iteration $i-1$. The statement of the corollary holds as a result of Proposition 76. □

G.2.11 Proof of Proposition 77

For $j \leq i$,
$inc_i(v) \leq agg\left(\{\min\left(1, INC_j(v)(g(v')) + I^{(alg)}_{i-1}(g(v'))\right) - I^{(alg)}_{i-1}((g(v')))|v' \in \mathbf{V}\}\right)$

Proof. Follows directly from Observation 1 and Proposition 74. □

G.2.12 Proof of Theorem 57

If the nodes in $GS^{(\epsilon)}_i(cand^{(\epsilon)}{}_i)$ corresponding with elements of $cand^{(\epsilon)}{}_i'$ are an independent set of $GS^{(\epsilon)}_i(cand^{(\epsilon)}{}_i)$, then the greedy algorithm can select all vertices in $cand^{(\epsilon)}{}_i'$ and still obtain a solution within $\frac{e^\epsilon - 1}{e^\epsilon}$ of optimal.

Proof. By the definition of an independent set and vertex spread, for any $v, v' \in cand^{(\epsilon)}{}_i'$, we know that $spread^{(\epsilon)}_i(v) \cap spread^{(\epsilon)}_i(v') \equiv \emptyset$ as there is no edge between them in the spread-graph. Hence, if v is selected on iteration i, we know we can select v' on iteration $i+1$ as $inc^{(\epsilon)}_{i+1}(v') \geq inc^{(\epsilon)}_i(v')$ as $spread^{(\epsilon)}_i(v) \cap spread^{(\epsilon)}_i(v') \equiv \emptyset$. We also know, on iteration $i+1$, the set $cand^{(\epsilon)}{}_i' - \{v, v'\}$ is an independent set of the spread graph of $cand^{(\epsilon)}{}_{i+1}$, so we can select every other element of $cand^{(\epsilon)}{}_i'$ as well. □

G.2.13 Proof of Proposition 78

The complexity of GREEDY-SNOP2 is $O(k \cdot |\mathbf{V}| \cdot F(|\mathbf{V}|))$ where $F(|\mathbf{V}|)$ is the time complexity to compute $value(V')$ for some set $V' \subseteq \mathbf{V}$ of size k.

Proof. There are two main operations that incur a cost additional to GREEDY-SNOP (see Proposition 73, however they are both dominated by other operations.

1. At each iteration of the loop at line 4, $inc^{(up)}(v)$ is computed for each vertex, giving an upper bound of the incremental increase for the iteration. There are $O(|\mathbf{V}|)$ of these operations and each operations costs less than the computation of the fixed point, so they are dominated by line 4e.

2. After the completion of the inner loop at line 4e, the algorithm may create a spread graph and find an independent set. Under the assumption that GREEDY-INDEP-SET or a similar algorithm is used, this operation is also dominated by the inner loop at line 4e.

□

G.2.14 Proof of Proposition 79

Given a SNOP-query meeting the following criteria:

- Π is a linear GAP
- VC is applied a-priori
- agg is positive linear

- *value* is zero-starting

Then GREEDY-SNOP2 is an $\frac{e^{\epsilon}}{e^{\epsilon}-1}$-approximation algorithm for the query.

Proof. Below we note the main differences between GEEDY-SNOP and GREEDY-SNOP2 as show how they still allow the approximation guarantee of the statement:

1. The least fixed point is computed using saved logic programs that capture previously computed annotations. By Proposition 75, this has no effect on the approximation ratio.

2. Ignoring vertices whose associated upper bound on the incremental increase is below this quantity for vertices already considered does not affect approximation ratio by Corollary 14. Further, this upper bound on the incremental increase associated with a vertex is correct as per Proposition 77.

3. Picking a vertex whose associated incremental increase is within ϵ of optimal give the approximation ratio of the statement by Theorem 56 and the upper bound used to specify this is correct by Observation 2

4. Selecting multiple vertices that comprise an independent set of the spread graph of all vertices whose incremental increase is within ϵ of optimal allows for the approximation guarantee of the statement by Theorem 57.

□

```
RETURN-SET($G = (V, E), V' \subseteq V, v \in V$) returns $V'' \subset V$

1. $V'' = V' \cup \{v\}$
2. For all $v' \in V$ s.t. $(v, v') \in E$:
    (a) If $v' \notin V'$, do the following:
        i. Set $V^* =$ RETURN-SET($G, v', V''$)
        ii. $V'' = V'' \cup V^*$
3. Return $V''$
```

```
FIND-ALL-DNS-SETS($G = (V, E)$) returns $V_1, \ldots, V_n \subseteq V$

1. $n = 0$, $V_{rem} = V$
2. While $V_{rem} \not\equiv \emptyset$
    (a) $n++$, $V_n = \emptyset$
    (b) Pick a vertex $v \in V_{rem}$.
    (c) $V_n =$ RETURN-SET($G, \emptyset, v$)
    (d) $V_{rem} = V_{rem} - V_n$
3. Return $V_1, \ldots, V_n$
```

G.2.15 Algorithm for Finding Disjoint Node Sets

G.2.16 Proof of Proposition 80

Given a SNOP-query meeting the following criteria:

- Π is a linear GAP
- VC is applied a-priori
- agg is positive linear
- $value$ is zero-starting

Then GREEDY-SNOP-DIV is an $\frac{e^\epsilon}{e^\epsilon - 1}$-approximation algorithm for the query.

Proof. We prove the statement by showing that for any instance of GREEDY-SNOP-DIV, the solution returned is the same as that returned by an instance of GREEDY-SNOP2 – thus assuring the approximation guarantee. In this proof we shall use GREEDY-SNOP2$_i$ to refer to an instance of GREEDY-SNOP2 that considers vertices only in some DNS_i, as called by GREEDY-SNOP-DIV. We shall use GREEDY-SNOP2$_{all}$ to refer to an instance of GREEDY-SNOP2 on the same input as GREEDY-SNOP-DIV.

CLAIM 1: If the first vertex (vertex v) picked by GREEDY-SNOP2$_i$ is also picked by GREEDY-SNOP2$_{all}$, then the incremental increase for that vertex is the same for both algorithms.

We note that vertex v is independent from any vertex $v' \notin DNS_i$, so by the proof of Theorem 57, the statement of the claim follows.

CLAIM 2: If vertex v_j is picked by $\mathsf{GREEDY\text{-}SNOP2}_i$ at some iteration j, then it is picked by $\mathsf{GREEDY\text{-}SNOP2}_{all}$ only if $\mathsf{GREEDY\text{-}SNOP2}_{all}$ picks all other vertices selected by $\mathsf{GREEDY\text{-}SNOP2}_i$ before iteration j.

We show this by induction on j.

BASE CASE: $j = 2$

Consider v_1, v_2. Note that on the first iteration of $\mathsf{GREEDY\text{-}SNOP2}_i$, the algorithm found that the incremental increase of v_1 is more "optimal" than v_2. Hence, by claim 1, this vertex would also be picked by $\mathsf{GREEDY\text{-}SNOP2}_{all}$.

INDUCTIVE HYPOTHESIS:

If $\mathsf{GREEDY\text{-}SNOP2}_{all}$ picks v_j, it also selects vertices $v_1, \ldots, v_{j-2}$ picked by $\mathsf{GREEDY\text{-}SNOP2}_i$ on iterations $1, \ldots, j-2$.

INDUCTIVE STEP:

If $\mathsf{GREEDY\text{-}SNOP2}_{all}$, then by the inductive hypothesis, it selects $v_1, \ldots, v_{j-2}$. Suppose, BWOC, it picks vertex v_j before v_{j-1}. However, as $\mathsf{GREEDY\text{-}SNOP2}_i$ picks vertex v_{j-1} first, we know it is "more optimal" than v_j on $\mathsf{GREEDY\text{-}SNOP2}_i$. As the only vertices that picked $\mathsf{GREEDY\text{-}SNOP2}_{all}$ which are not independent were the same ones picked by $\mathsf{GREEDY\text{-}SNOP2}_i$, we know that $\mathsf{GREEDY\text{-}SNOP2}_{all}$ will also find v_{j-1} "more optimal" than v_j – hence a contradiction.

CLAIM 3: Any vertex picked by $\mathsf{GREEDY\text{-}SNOP2}_{all}$ contributes the same incremental increase as if it were picked by $\mathsf{GREEDY\text{-}SNOP2}_i$.

Follows from the fact that vertices in each instance are independent from each other as well as claims 1-2.

Proof of Proposition: Follows directly from claims 1-3. $\square$

Bibliography

[1] N. Agmon, S. Kraus, and G.A. Kaminka. Multi-robot perimeter patrol in adversarial settings. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA-2008)*, pages 2339–2345, 2008.

[2] N. Agmon, S. Kraus, G.A. Kaminka, and V. Sadov. Adversarial uncertainty in multi-robot patrol. In *Proc. 21st Int. Joint Conf. on Artificial Intelligence (IJCAI-2009)*, pages 1811–1817, 2009.

[3] James F. Allen and George Ferguson. Actions and events in interval temporal logic. *J. of Logic and Computation*, 4:531–579, 1994.

[4] Ethem Alpaydin. *Introduction to Machine Learning.* MIT Press, 2 edition, 2010.

[5] Roy M. Anderson and Robert M. May. Population biology of infectious diseases: Part i. *Nature*, 280(5721):361, 1979.

[6] T. Antal, S. Redner, and V. Sood. Evolutionary dynamics on degree-heterogeneous graphs. *Physical Review Letters*, 96(18):188104, 2006.

[7] Shyamanta M. Hazarika Anthony G. Cohn. Qualitative spatial representation and reasoning: An overview. volume 46, pages 1–29, 2001.

[8] K. Apt. *Principles of constraint programming.* Cambridge University Press, 2003.

[9] Sinan Aral, Lev Muchnik, and Arun Sundararajan. Distinguishing influence-based contagion from homophily-driven diffusion in dynamic networks. *Proceedings of the National Academy of Sciences*, 106(51):21544–21549, December 2009.

[10] V Asal, J Carter, and J Wilkenfeld. Ethnopolitical violence and terrorism in the middle east. In J Hewitt, J Wilkenfeld, and T Gurr, editors, *Peace and Conflict 2008.* Paradigm, 2008.

[11] Yossi Azar and Iftah Gamzu. Efficient submodular function maximization under linear packing constraints. *(submitted, preprint avaialbe from http://www.cs.tau.ac.il/ iftgam/papers/SubmodularPacking.pdf)*, 2010.

[12] Adnan Aziz, Vigyan Singhal, Felice Balarin, Robert K. Brayton, and Alberto L. Sangiovanni-vincentelli. It usually works: The temporal logic of stochastic systems. pages 155–165. Springer, 1995.

[13] C. Baral, N. Tran, and L. Tuan. Reasoning about actions in a probabilistic setting. In *Proc. AAAI 2002*, pages 507–512, 2002.

[14] Sugato Basu, Ian Davidson, and Kiri Wagstaff. *Constrained Clustering: Advances in Algorithms, Theory, and Applications*. Chapman & Hall/CRC, 2008.

[15] Paul Brantingham and Patricia Brantingham. Crime Pattern Theory. In Richard Wortley and Lorraine Mazerolle, editors, *Enviromental Criminology and Crime Analysis*, pages 78–93. 2008.

[16] Herv Brnnimann and Michael T. Goodrich. Almost optimal set covers in finite vc-dimension. *Discrete Comput. Geom*, 14:293–302, 1995.

[17] Matthias Broecheler, Paulo Shakarian, and V.S. Subrahmanian. A scalable framework for modeling competitive diffusion in social networks. *Social Computing / IEEE International Conference on Privacy, Security, Risk and Trust*, 0:295–302, 2010.

[18] Matthias Broecheler, Gerardo I. Simari, and V. S. Subrahmanian. Using histograms to better answer queries to probabilistic logic programs. In *ICLP '09: Proceedings of the 25th International Conference on Logic Programming*, pages 40–54, Berlin, Heidelberg, 2009. Springer-Verlag.

[19] Tom Bylander, Dean Allemang, Michael C. Tanner, and John R. Josephson. The Computational Complexity of Abduction, 1991.

[20] Meeyoung Cha, Alan Mislove, Ben Adams, and Krishna P. Gummadi. Characterizing social cascades in flickr. In *WOSP '08: Proceedings of the first workshop on Online social networks*, pages 13–18, New York, NY, USA, 2008. ACM.

[21] Meeyoung Cha, Alan Mislove, and Krishna P. Gummadi. A Measurement-driven Analysis of Information Propagation in the Flickr Social Network. In *In Proceedings of the 18th International World Wide Web Conference (WWW'09)*, Madrid, Spain, April 2009.

[22] A. Charnes and W. Cooper. Programming with linear fractional functionals. *Naval Research Logistics Quarterly*, 9(3):163–297, 1962.

[23] Wei Chen, Chi Wang, and Yajun Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '10, pages 1029–1038, New York, NY, USA, 2010. ACM.

[24] Vaek Chvatal. *Linear Programming*. W.H.Freeman, New York, 1983.

[25] R. Cleaveland, P. Iyer, and M. Narasimha. Probabilistic Temporal Logics via the Modal Mu-Calculus. *Theoretical Computer Science*, 342(2-3):316–350, 2005.

[26] Luca Console, Luigi Portinale, and Daniele Theseider Dupré. Focussing Abductive Diagnosis. *AI Commun.*, 4(2), 1991.

[27] Luca Console, Maria Luisa Sapino, and Daniele Theseider Dupré. The Role of Abduction in Database View Updating. *Journal of Intelligent Information Systems*, 4(3):261, 1995.

[28] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, second edition, 2001.

[29] Robin Cowan and Nicolas Jonard. Network structure and the diffusion of knowledge. *Journal of Economic Dynamics and Control*, 28(8):1557 – 1575, 2004.

[30] J. Lang D. Dubois and H. Prade. Timed possibilistic logic. *Fundamenta Informaticae*, XV:211–234, 1991.

[31] C. Damasio, L. Pereira, and T. Swift. Coherent well-founded annotated logic programs. In *Proc. Intl. Conf. on Logic Programming and Non-Monotonic Reasoning*, pages 262–276. Springer Lecture Notes in Computer Science Vol. 1730, 1999.

[32] Ian Davidson and S. S. Ravi. Clustering with constraints: Feasibility issues and the k-means algorithm. In *SDM*, 2005.

[33] Munmun De Choudhury, Hari Sundaram, Ajita John, and Dorée Duncan Seligmann. Can blog communication dynamics be correlated with stock market activity? In *HT '08: Proceedings of the nineteenth ACM conference on Hypertext and hypermedia*, pages 55–60, New York, NY, USA, 2008. ACM.

[34] Alex Dekhtyar, Michael I. Dekhtyar, and V. S. Subrahmanian. Temporal probabilistic logic programs. In *ICLP 1999*, pages 109–123, Cambridge, MA, USA, 1999. The MIT Press.

[35] J.P. Dickerson, G.I. Simari, V.S. Subrahmanian, and Sarit Kraus. A Graph-Theoretic Approach to Protect Static and Moving Targets from Adversaries. In *Proc. 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS-2010)*, pages 299–306, 2010.

[36] Jürgen Dix, Sarit Kraus, and V. S. Subrahmanian. Heterogeneous temporal probabilistic agents. *ACM TOCL*, 7(1):151–198, 2006.

[37] Silvio do Lago Pereira and Leliane Nunes de Barros. Planning with abduction: A logical framework to explore extensions to classical planning. In *Lecture Notes in Computer Science Advances in Artificial Intelligence SBIA*, 2004.

[38] Martin Dyer, Leslie A Goldberg, Catherine Greenhill, and Mark Jerrum. On the relative complexity of approximate counting problems. Technical report, Coventry, UK, UK, 2000.

[39] T. Eiter, J. Lu, and V.S. Subrahmanian. Computing non-ground representations of stable models. In *Proc. Intl. Conf. on Logic Programming and Non-Monotonic Reasoning*, pages 198–217. Springer Lecture Notes in Computer Science Vol. 1265, 1997.

[40] T Eiter, V.S. Subrahmanian, and G. Pick. Heterogeneous Active Agents, I: Semantics. *Artificial Intelligence Journal*, 108(1-2):179–255, 1999.

[41] Thomas Eiter and Georg Gottlob. The complexity of logic-based abduction. *J. ACM*, 42(1):3–42, 1995.

[42] E. A Emerson and Joseph Y. Halpern. "sometimes" and "not never" revisited: on branching versus linear time. Technical report, Austin, TX, USA, 1984.

[43] Ronald Fagin and Joseph Y. Halpern. Reasoning about knowledge and probability. *Journal of the ACM*, 41:340–367, 1994.

[44] Ronald Fagin, Joseph Y. Halpern, and Nimrod Megiddo. A logic for reasoning about probabilities. *Information and Computation*, 87:78–128, 1990.

[45] H. Cruz F.C. Coelho, C. Codeco. Epigrass: A tool to study disease spread in complex networks. *Source Code for Biology and Medicin*, 3(3), 2008.

[46] Uriel Feige. A threshold of ln n for approximating set cover. *J. ACM*, 45(4):634–652, 1998.

[47] Uriel Feige, Vahab S. Mirrokni, and Jan Vondrak. Maximizing non-monotone submodular functions. In *FOCS '07: Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science*, pages 461–471, Washington, DC, USA, 2007. IEEE Computer Society.

[48] Massimo Franceschetti, Matthew Cook, and Jehoshua Bruck. A geometric theorem for network design. *IEEE Transactions on Computers*, 53(4):483–489, 2004.

[49] Michael L. Fredman and Robert E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM*, 34(3):596–615, July 1987.

[50] David Freedman, Roger Purves, and Robert Pisani. *Statistics*. W.W. Norton and Co., 4 edition, 2007.

[51] Thom Frühwirth. Annotated constraint logic programming applied to temporal reasoning. In *PLILP: Programming Language Implementation and Logic Programming*, Madrid, 1994. Springer.

[52] Bin Fu, Zhixiang Chen, and Mahdi Abdelguerfi. An almost linear time 2.8334-approximation algorithm for the disc covering problem. In *AAIM '07: Proceedings of the 3rd international conference on Algorithmic Aspects in Information and Management*, pages 317–326, Berlin, Heidelberg, 2007. Springer-Verlag.

[53] I. Fujiwara, Y. Hirose, and M. Shintani. *Can News be a Major Source of Fluctuation: A Bayesian DGSE Approach*, volume Discussion Paper Nr. 2008-E-16. Institute for Monetary and Economic Studies, Bank of Japan, 2008.

[54] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., New York, NY, USA, 1979.

[55] Els Gijsbrechts, Katia Campo, and Tom Goossens. The impact of store flyers on store traffic and store sales: a geo-marketing approach. *Journal of Retailing*, 79(1):1 – 16, 2003.

[56] Rob J. Van Glabbeek, Scott A. Smolka, and Bernhard Steffen. Reactive, generative, and stratified models of probabilistic processes. *Information and Computation*, 121:130–141, 1995.

[57] Jack A. Goldstone, Robert Bates, Ted Robert Gurr, Michael Lustik, Monty G. Marshall, Jay Ulfelder, and Mark Woodward. A global forecasting model of political instability. In *Proc. Annual Meeting of the American Political Science Association*, 2005.

[58] Teofilo F. Gonzalez. Covering a set of points in multidimensional space. *Inf. Process. Lett.*, 40(4):181–188, 1991.

[59] Georg Gottlob, Sherry Marcus, Anil Nerode, Gernot Salzer, and V. S. Subrahmanian. A non-ground realization of the stable and well-founded semantics. *Theor. Comput. Sci.*, 166(1-2):221–262, 1996.

[60] P.R. Goundan and A.S. Schultz. Revisiting the greedy approach to submodular set function maximization. Technical report, Massachusetts Institute of Technology, 2007.

[61] Mark Granovetter. Threshold models of collective behavior. *The American Journal of Sociology*, 83(6):1420–1443, 1978.

[62] P. Haddawy. Representing plans under uncertainty: A logic of time, chance and action. *PhD Thesis, Univ. of Illinois*, 1991.

[63] J. Halpern and M. Tuttle. Knowledge, probability, and adversaries. In *IBM Thomas J. Watson Research Center Tech Report*, 1992.

[64] H. Hansson and B. Jonsson. A logic for reasoning about time and probability. *Formal Aspects of Computing*, 6:512–535, 1994.

[65] Hans Hansson and Bengt Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6:102–111, 1994.

[66] S. Hart and M. Sharir. Probabilistic propositional temporal logic. *Information and Control*, 70:97–155, 1986.

[67] Herbert W. Hethcote. Qualitative analyses of communicable disease models. *Mathematical Biosciences*, 28(3-4):335 – 356, 1976.

[68] Dorit S. Hochbaum. Approximation Algorithms for the Set Covering and Vertex Cover Problems. *SIAM Journal on Computing*, 11(3):555–556, 1982.

[69] Dorit S. Hochbaum. *Approximation Algorithms for NP-Complete Problems*. PWS Publishing Co., 1997.

[70] Dorit S. Hochbaum and Wolfgang Maass. Approximation schemes for covering and packing problems in image processing and vlsi. *J. ACM*, 32:130–136, 1985.

[71] Harry B. Hunt, III, Madhav V. Marathe, Venkatesh Radhakrishnan, and Richard E. Stearns. The complexity of planar counting problems. *SIAM J. Comput.*, 27(4):1142–1167, 1998.

[72] ISW. Map of Special Groups Activity in Iraq, Institute for the Study of War. 2008.

[73] M. Jackson and L. Yariv. Diffusion on social networks. In *Economie Publique*, volume 16, pages 69–82, 2005.

[74] Robert Jeansoulin, Odile Papini, Henri Prade, and Steven Schockaert. In Robert Jeansoulin, Odile Papini, Henri Prade, and Steven Schockaert, editors, *Methods for Handling Imperfect Spatial Information*, volume 256 of *Studies in Fuzziness and Soft Computing*. Springer Berlin / Heidelberg, 2010.

[75] Lujun Jia, Rajmohan Rajaraman, and Torsten Suel. An efficient distributed algorithm for constructing small dominating sets. *Distrib. Comput.*, 15(4):193–205, 2002.

[76] D.S. Johnson. The np-completeness column: An ongoing guide. *Journal of Algorithms*, 3(2):182–195, 1982.

[77] A. C. Kakas and P. Mancarella. Database updates through abduction. In *VLDB90*, 1990.

[78] K. Kanazawa. A logic and time nets for probabilistic inference. In *Proc. AAAI 1991*, 1991.

[79] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395, 1984.

[80] Richard Karp. Reducibility Among Combinatorial Problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, page 85103. 1972.

[81] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146, New York, NY, USA, 2003. ACM.

[82] Gabriele Kern-Isberner and Thomas Lukasiewicz. Combining probabilistic logic programming with the power of maximum entropy. *Artif. Intell.*, 157(1-2):139–202, 2004.

[83] Samir Khuller, Maria Vanina Martinez, Dana Nau, Gerardo I. Simari, Amy Sliva, and Venkatramanan Siva Subrahmanian. Action probabilistic logic programs. *Annals of Mathematics and Artificial Intelligence*, 51(2–4):295–331, 2007.

[84] W. Kiessling, H. Thone, and U. Guntzer. Database support for problematic knowledge. In *Proceedings of EDBT 1992, Springer LNCS Volume 580*, pages 421–436, 1992.

[85] Michael Kifer and Eliezer L. Lozinskii. A logic for reasoning with inconsistency. *Journal of Automated Reasoning*, 9(2):179–215, 1992.

[86] Michael Kifer and V.S. Subrahmanian. Theory of generalized annotated logic programming and its applications. *J. Log. Program.*, 12(3&4):335–367, 1992.

[87] Dexter Kozen. *The Design and Analysis of Algorithms*. Springer-Verlag, New York, 1991.

[88] Stanislav Krajci, Rastislav Lencses, and Peter Vojts. A comparison of fuzzy and annotated logic programming. *Fuzzy Sets and Systems*, 144(1):173 – 192, 2004.

[89] Fabian Kuhn and Roger Wattenhofer. Constant-time distributed dominating set approximation. In *In Proc. of the 22 nd ACM Symposium on the Principles of Distributed Computing (PODC*, pages 25–32, 2003.

[90] Benjamin Kuipers. A hierarchy of qualitative representations for space. In *Working papers of the Tenth International Workshop on Qualitative Reasoning about Physical Systems*, 1996.

[91] M. Kwiatkowska, G. Norman, and D. Parker. Verifying randomized distributed algorithms with PRISM. In *Proc. Workshop on Advances in Verification (Wave'2000)*, July 2000.

[92] Marta Kwiatkowska, Gethin Norman, and David Parker. Prism: probabilistic model checking for performance and reliability analysis. *SIGMETRICS Perform. Eval. Rev.*, 36(4):40–45, 2009.

[93] Laks V.S. Lakshmanan and F. Sadri. Modeling uncertainty in deductive databases. In *Proceedings of DEXA 1994*, pages 724–733. Springer LNCS Vol. 856, 1994.

[94] Laks V.S. Lakshmanan and F. Sadri. Probabilistic deductive databases. In *Proceedings of the Intl. Logic Programming Symposium (ILPS)*. MIT Press, 1994.

[95] Laks V.S. Lakshmanan and Nematollaah Shiri. A parametric approach to deductive databases with uncertainty. *IEEE Transactions on Knowledge and Data Engineering*, 1997.

[96] Leslie Lamport. "sometime" is sometimes "not never": on the temporal logic of programs. In *POPL 1980*, pages 174–185, New York, NY, USA, 1980. ACM.

[97] Kim G. Larsen and Arne Skou. Bisimulation through probabilistic testing. *Inf. Comput.*, 94(1):1–28, 1991.

[98] D. Lehmann and S. Shelah. Reasoning about time and chance. *Information and Control*, 53:165–198, 1982.

[99] Nicola Leone, Francesco Scarcello, and V.S. Subrahmanian. Optimal models of disjunctive logic programs: Semantics, complexity, and computation. *IEEE Transactions on Knowledge and Data Engineering*, 16:487–503, 2004.

[100] Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. Predicting positive and negative links in online social networks. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 641–650, New York, NY, USA, 2010. ACM.

[101] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. Cost-effective outbreak detection in networks. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 420–429, New York, NY, USA, 2007. ACM.

[102] Kevin Leyton-Brown and Yoav Shoham. *Essentials of Game Theory: A Concise, Multidisciplinary Introduction.* Morgan and Claypool Publishers, 2008.

[103] S. Li and M. Ying. Region connection calculus: Its models and composition table. *Artif. Intell.*, 145:121 – 146, 2003.

[104] Chen Liao and Shiyan Hu. Polynomial time approximation schemes for minimum disk cover problems. *Journal of Combinatorial Optimization.*

[105] Erez Lieberman, Christoph Hauert, and Martin A. Nowak. Evolutionary dynamics on graphs. *Nature*, 433(7023):312–316, 2005.

[106] J. W. Lloyd. *Foundations of Logic Programming, Second Edition*. Springer-Verlag, 1987.

[107] J. Lu. Logic programs with signs and annotations. *Journal of Logic and Computation*, 6(6):755–778, 1996.

[108] James J. Lu, Anil Nerode, and V.S. Subrahmanian. Hybrid knowledge bases. *IEEE Transactions on Knowledge and Data Engineering*, 8(5):773–785, 1996.

[109] J.J. Lu, N.V. Murray, and E. Rosenthal. Signed formulas and annotated logics. In *Multiple-Valued Logic, 1993., Proceedings of The Twenty-Third International Symposium on*, pages 48–53, May 1993.

[110] Thomas Lukasiewicz. Probabilistic logic programming. In *ECAI*, pages 388–392, 1998.

[111] Thomas Lukasiewicz. Many-valued disjunctive logic programs with probabilistic semantics. In *In Proceedings of the 5th International Conference on Logic Programming and Nonmonotonic Reasoning, volume 1730 of LNAI*, pages 277–289. Springer, 1999.

[112] Thomas Lukasiewicz, Thomas Lukasiewicz, Gabriele Kern-isberner, and Gabriele Kern-isberner. Probabilistic logic programming under maximum entropy. In *In Proc. ECSQARU-99, LNCS 1638*, pages 279–292. Springer, 1999.

[113] Carsten Lund and Mihalis Yannakakis. On the hardness of approximating minimization problems. *J. ACM*, 41(5):960–981, 1994.

[114] A. Martelli M. Falaschi, G. Levi and C. Palamidessi. A new declarative semantics for logic languages. In *Proc. 5th Internat. Conf. Symp. on Logic Programming*, page 9931005, 1988.

[115] Wolfgang Maass. On the complexity of nonconvex covering. *SIAM J. Comput.*, 15(2):453–467, 1986.

[116] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

[117] Paolo Mancarella, Alessandra Raffaetà, and Franco Turini. Temporal annotated constraint logic programming with multiple theories. In *DEXA '99: Proceedings of the 10th International Workshop on Database & Expert Systems Applications*, 1999.

[118] A. Mannes, M. Michaell, A. Pate, A. Sliva, V.S. Subrahmanian, and J. Wilkenfeld. Stochastic Opponent Modelling Agents: A Case Study with Hezbollah. In *Proc. 2008 First Intl. Workshop on Social Computing, Behavioral Modeling and Prediction*. Springer Verlag, April 1-2, 2008.

[119] Aaron Mannes, Amy Sliva, V.S. Subrahmanian, and Jonathan Wilkenfeld. Stochastic Opponent Modeling Agents: A Case Study with Hamas. In *Proc. 2008 Intl. Conf. on Computational Cultural Dynamics*, pages 49–54. AAAI Press, Sep. 2008.

[120] V. Martinez, G.I. Simari, A. Sliva, and Venkatramanan Siva Subrahmanian. The SOMA Terror Organization Portal (STOP): Social Network and Analytic Tools for the Real-Time Analysis of Terror Groups. In Huan Liu and John Salerno, editors, *Proceedings of the First International Workshop on Social Computing, Behavioral Modeling and Prediction*, 2008.

[121] V. Martinez, G.I. Simari, A. Sliva, and V.S. Subrahmanian. CONVEX: Similarity-Based Algorithms for Forecasting Group Behavior. *IEEE Intelligent Systems*, 23(4):51–57, 2008.

[122] V. Martinez, G.I. Simari, A. Sliva, and V.S. Subrahmanian. CAPE: Automatically Predicting Changes in Terror Group Behavior. *to appear in Mathematical Methods in Counterterrorism (ed. N. Memon)*, 2009.

[123] S. Masuyama, T. Ibaraki, and T. Hasegawa. The computational complexity of the m-center problems on the plane. *Trans. IECE of Japan*, E84:57–64, 1981.

[124] P. Mateus, A. Pacheco, J. Pinto, A. Sernadas, and C. Sernadas. Probabilistic situation calculus. *AMAI*, 32:393–431(39), 2001.

[125] Nimrod Megiddo and Kenneth J. Supowit. On the complexity of some common geometric location problems. *SIAM Journal of Computing*, 13(1):182–196, 1984.

[126] A. Pentland N. Eagle and D. Lazer. Mobile phone data for inferring social network structure. In *Proc. 2008 Intl. Conference on Social and Behavioral Computing*, pages 79–88. Springer Verlag, 2008.

[127] G. L. Nemhauser, L. A. Wolsey, and M.L. Fisher. An analysis of approximations for maximizing submodular set functionsi. *Mathematical Programming*, 14(1):265–294, 1978.

[128] Raymond T. Ng and V. S. Subrahmanian. Probabilistic logic programming. *Information and Computation*, 101(2):150–201, 1992.

[129] Raymond T. Ng and Venkatramanan Siva Subrahmanian. A semantical framework for supporting subjective and conditional probabilities in deductive databases. In Koichi Furukawa, editor, *Proceedings of ICLP '91*, pages 565–580. The MIT Press, 1991.

[130] Raymond T. Ng and Venkatramanan Siva Subrahmanian. Probabilistic logic programming. *Information and Computation*, 101(2):150–201, 1992.

[131] Nils Nilsson. Probabilistic logic. *Artificial Intelligence*, 28:71–87, 1986.

[132] Susan Owicki and Leslie Lamport. Proving liveness properties of concurrent programs. *ACM Trans. Program. Lang. Syst.*, 4(3):455–495, 1982.

[133] Maurice Pagnucco. *The Role of Abductive Reasoning within the Process of Belief Revision*. PhD thesis, Basser Department of Computer Science, University of Sydney, Australia, 1996.

[134] Christos H. Papadimitriou. Worst-Case and Probabilistic Analysis of a Geometric Location Problem. *SIAM J. Comput.*, 10(3):542–557, 1981.

[135] P. Paruchuri, M. Tambe, F. Ordóñez, and S. Kraus. Security in multiagent systems by policy randomization. In *Proc. 5th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS-2006)*, 2006.

[136] Vangelis T. Paschos. A survey of approximately optimal solutions to some covering and packing problems. *ACM Comput. Surv.*, 29(2):171–209, 1997.

[137] Charles S. Peirce. *Philosophical writings of Peirce, selected and edited with an introd. by Justus Buchler.* Dover Publications New York,, 1955.

[138] Yun Peng and James A. Reggia. Plausibility of Diagnostic Hypotheses. In *Proceedings, 5th National Conference on AI (AAAI-86)*, pages 140–145, 1986.

[139] J. Pita, M. Jain, F. Ordóñez, M. Tambe, S. Kraus, and R. Magori-Cohen. Effective solutions for real-world stackelberg games: When agents must deal with human uncertainties. In *Proc. 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS-2009)*, pages 369–376, 2009.

[140] M. L. Puterman. *Markov Decision Processes*. Wiley, 1994.

[141] James A. Reggia and Yun Peng. *Abductive inference models for diagnostic problem-solving*. Springer-Verlag New York, Inc., New York, NY, USA, 1990.

[142] J. Renz and B. Nebel. On the complexity of qualitative spatial reasoning: A maximal tractable fragment of the region connection calculus. *Artif. Intell.*, 108:69 – 123, 1999.

[143] Thomas Hewitt Rich, Mildred Adams Fenton, and Carroll Lane Fenton. *The fossil book: a record of prehistoric life*. Dover Publications, 2 edition, 1996.

[144] D. Kim Rossmo and Sacha Rombouts. Geographic Profiling. In Richard Wortley and Lorraine Mazerolle, editors, *Enviromental Criminology and Crime Analysis*, pages 136–149. 2008.

[145] Dan Roth. On the hardness of approximate reasoning. *Artificial Intelligence*, 82:273–302, 1996.

[146] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 2nd edition edition, 2003.

[147] Jan Rychtář and Brian Stadler. Evolutionary dynamics on small-world networks. *International Journal of Computational and Mathematical Sciences*, 2(1), 2008.

[148] Eugene Santos and Joel D. Young. Probabilistic temporal networks: A unified framework for reasoning with time and uncertainty. *Inter. Journal of Approximate Reasoning*, 20, 1999.

[149] Paulo Santos and Murray Shanahan. Hypothesising object relations from image transitions. In *Proc. ECAI02*, 2002.

[150] Thomas C. Schelling. *Micromotives and Macrobehavior*. W.W. Norton and Co., 1978.

[151] P. Schrodt and D.J. Gerner. Cluster analysis as an early warning technique for the middle east. *Preventive Measures: Building Risk Assessment and Crisis Early Warning Systems (eds. John L. Davies and Ted Robert Gurr)*, 1998.

[152] P. Shakarian, M. Broecheler, and V.S. Subrahmanian. Using generalized annotated programs to solve social network optimization problems. *(submitted)*, 2011.

[153] P. Shakarian and V.S. Subrahmanian. Region-based Geospatial Abduction with Counter-IED Applications. In U. Kock Wiil, editor, *Counterterrorism and Open Source Intelligence (to appear)*. Springer, 2010.

[154] Paulo Shakarian, John Dickerson, and V.S. Subrahmanian. Adversarial geosptaial abudction. *(submitted)*, 2011.

[155] Paulo Shakarian, Austin Parker, Gerardo Simari, and V.S. Subramanian. Annotated probabilstic temporal logic. *ACM Transactions on Computational Logic*, 12(2), 2011.

[156] Paulo Shakarian, Gerardo Simari, and V.S. Subramanian. Annotated probabilistic temporal logic: Approximate fixpoint implementation. *ACM Transactions on Computational Logic (accepted)*, 2011.

[157] Paulo Shakarian, V.S. Subrahmanian, and Maria Luisa Sapino. SCARE: A Case Study with Baghdad. In *Proceedings of the Third International Conference on Computational Cultural Dynamics*. AAAI, 2009.

[158] Paulo Shakarian, V.S. Subrahmanian, and Maria Luisa Sapino. GAPs: Geospatial Abduction Problems. *ACM Transaction on Intelligent Systems and Technology (accepted)*, 2010.

[159] Paulo Shakarian, V.S. Subrahmanian, and Maria Luisa Sapino. Using generalized annotated programs to solve social network optimization problems. In Manuel Hermenegildo and Torsten Schaub, editors, *Technical Communications of the 26th International Conference on Logic Programming*, volume 7 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 182–191, Dagstuhl, Germany, 2010. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[160] M. Shanahan. Noise and the common sense informatic situation. In *Proc. AAAI96*, page 1098, 1996.

[161] David B. Shmoys, Eva Tardos, and Karen Aardal. Approximation algorithms for facility location problems (extended abstract). In *In Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 265–274, 1998.

[162] V. Sood, Tibor Antal, and S. Redner. Voter models on heterogeneous networks. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 77(4):041121, 2008.

[163] R.K. Srihari. Automatic indexing and content-based retrieval of captioned images. *Computer*, 28(9):49 –56, September 1995.

[164] John F. Stollsteimer. A Working Model for Plant Numbers and Locations. 45(3):631–645, Aug. 1963.

[165] Bogdan Stroe and V. S. Subrahmanian. First order heterogeneous agent computations. In *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 217–224, New York, NY, USA, 2003. ACM.

[166] V. S. Subrahmanian and Diego Reforgiato Recupero. AVA: Adjective-Verb-Adverb Combinations for Sentiment Analysis. *IEEE Intelligent Systems*, 23(4):43, 2008.

[167] Eric Sun, Itamar Rosenn, Cameron Marlow, and Thomas Lento. Gesundheit! modeling contagion through facebook news feed. In *Proceedings of the Third International Conference on Weblogs and Social Media*, San Jose, CA, May 2009. AAAI Press, AAAI Press.

[168] K. Thirunarayan and M. Kifer. A theory of nonmonotonic inheritance based on annotated logic. *Artificial Intelligence*, 60(1):23–50, 1993.

[169] S. Rebecca Thomas. The placa agent programming language. In *ECAI-94: Proceedings of the workshop on agent theories, architectures, and languages on Intelligent agents*, pages 355–370, New York, NY, USA, 1995. Springer-Verlag New York, Inc.

[170] US Army. *Intelligence Preparation of the Battlefiled (US Army Field Manual)*, FM 34-130 edition, 1994.

[171] US Army. *Counterinsurgency (US Army Field Manual)*, FM 3-24 edition, 2006.

[172] Pascal Van Hentenryck. Constraint logic programming. *The Knowledge Engineering Review*, 6(03):151–194, 2009.

[173] Moshe Y. Vardi. Automatic verification of probabilistic concurrent finite state programs. *Symp. on Foundations of Comp. Sci.*, 0:327–338, 1985.

[174] Vijay V. Vazirani. *Approximation Algorithms.* Springer, March 2004.

[175] J. Venneksn, S. Verbaeten, and M. Bruynooghe. Logic programs with annotated disjunctions. In *Proc. Intl. Conf. on Logic Programming*, pages 431–445. Springer Lecture Notes in Computer Science Vol. 3132, 2004.

[176] Kiri Wagstaff, Claire Cardie, Seth Rogers, and Stefan Schrödl. Constrained k-means clustering with background knowledge. In *ICML '01: Proceedings of the Eighteenth International Conference on Machine Learning*, pages 577–584, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[177] D.J. Watts and J. Peretti. Viral marketing for the real world. *Harvard Business Review*, May 2007.

[178] Duncan J. Watts. Networks, dynamics, and the small-world phenomenon. *The American Journal of Sociology*, 105(2):493–527, 1999.

[179] Y. Weiss and E.H. Adelson. A unified mixture framework for motion segmentation: incorporating spatial coherence and estimating the number of models. In *Computer Vision and Pattern Recognition, 1996. Proceedings CVPR '96, 1996 IEEE Computer Society Conference on*, pages 321 –326, June 1996.

[180] WEKA. WEKA 3 Data Mining, http://www.cs.waikato.ac.nz/ml/weka/. 2009.

[181] Jonathan Wilkenfeld, Victor Asal, Carter Johnson, Amy Pate, and Mary Michael. The use of violence by ethnopolitical organizations in the middle east. Technical report, National Consortium for the Study of Terrorism and Responses to Terrorism, February 2007.