ABSTRACT

Title of dissertation: PROBLEMS ORIGINATING FROM THE PLANNING OF AIR TRAFFIC MANAGEMENT INITIATIVES

Alexander Stewart Estes
Doctor of Philosophy, 2018

Dissertation directed by: Michael Ball
R. H. Smith School of Business
and
Institute of Systems Research

When weather affects the ability of an airport to accommodate flights, a ground delay program is used to control the rate at which flights arrive at the airport. This prevents excessive congestion at the airport. In this thesis, we discuss several problems arising from the planning of these programs. Each of these problems provides insight that can be applied in a broader setting, and in each case we develop generalizations of these results in a wider context. We show that a certain type of greedy policy is optimal for planning a ground delay program when no air delays are allowed. More generally, we characterize the conditions under which policies are optimal for a dynamic stochastic transportation problem. We also provide results that ensure that certain assignments are optimal, and we apply these results to the problem of matching drivers to riders in an on-demand ride service.

When flights are allowed to take air delays, then a greedy policy is no longer optimal, but flight assignments can be produced by solving an integer program. We

establish the strength of an existing formulation of this problem, and we provide a new, more scalable formulation that has the same strength properties. We show that both of these methods satisfy a type of equity property. These formulations are a special case of a dynamic stochastic network flow problem, which can be modeled as a deterministic flow problem on a hypergraph. We provide strong formulations for this general class of hypergraph flow problems.

Finally, we provide a method for summarizing a dataset of ground delay programs. This summarization consists of a small subset of the original data set, whose elements are referred to as "representative" ground delay programs. More generally, we define a new class of data exploration methods, called "representative region selection" methods. We provide a framework for evaluating the quality of these methods, and we demonstrate statistical properties of these methods.

PROBLEMS ORIGINATING FROM THE PLANNING OF AIR TRAFFIC MANAGEMENT INITIATIVES

by

Alexander Stewart Estes

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
2018

Advisory Committee:
Professor Michael Ball, Chair/Advisor
David Lovell
Ilya Ryzhov
Bruce Golden
Paul Smith

Dedication

Dedicated to my wife, Ying Han.

Acknowledgments

I would like to thank my dissertation advisor, Michael Ball, for his guidance, insight, and support through the years of research that went into this thesis. I would also like to thank David Lovell, who also provided guidance and had a hand in a good part of this work. Thank you to the rest of my committee members, Bruce Golden, Paul Smith, and Ilya Ryzhov for their insightful comments and questions. Thank you also to my other collaborators for working with me: Mark Hansen, Yulin Liu, Sreeta Gorripaty, Yi Liu, Alexei Pozdnoukhov from University of California-Berkeley and Kennis Chan, Corey Warner, and John Schade from ATAC. Thank you to NASA, and our sponsor John Schade, for funding for this project and for giving me the opportunity to pursue this research. Thank you to Bo Deng and Al Peterson at University of Nebraska-Lincoln for providing me with early research experiences that helped to grow my interest in mathematics.

Thank you to Konstantina Trivisa, Alverda McCoy, Jessica Sadler, and for all of the students in AMSC for maintaining a welcoming, friendly department. I am grateful to the friends and colleagues that I've made along the way who have shared their experience, advice and companionship.

Thank you to my parents, John and Kathy Estes, for encouraging my interest in mathematics and science and for supporting me over the years. Thank you also to my sister, Rachel Estes, who is a pretty good sister, I guess. Thank you to the rest of the family for their support and collective wisdom. Thank you to my wife, Ying Han, for being with me the whole way.

Thank you to everyone else who has helped me. Finally, thank you, for reading this thesis. I hope that you can find some knowledge or inspiration here.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

ARTCC	Air Route Traffic Control Center
ABF-Graph	Acyclic Branching-on-compaths F-Graph
ANSP	Air Navigation Service Provider
ARQM	Axiomatic Representative Quality Measure
CTD	Controlled Time of Departure
CQM	Clustering Quality Measure
DAG	Directed Acyclic Graph
D-RBD	Dynamic Ration-By-Distance
DET	Density Estimation Tree
DSTP	Dynamic Stochastic Transportation Problem
ETA	Estimated Time of Arrival
FAA	Federal Aviation Administration
FSFS	First-Scheduled First-Served
FSFS-D	First-Scheduled First-Served Within Duration
GDP	Ground Delay Program
GS	Ground Stop
IP	Integer Program(ming)
KC-UPR	K-Center Unsupervised Prototype Reduction
LP	Linear Program(ming)
MDS	Minimum Dominating Set
NTML	National Traffic Management Logs
ODRS	On-Demand Ride Service
RBD	Ration-By-Distance
RQM	Representative Quality Measure
RRS	Representative Region Selection
SAGHP	Single Airport Ground-Holding Problem
TMI	Traffic Management Initiative
TU	Totally Unimodular
UPR	Unsupervised Prototype Reduction

Chapter 1: Introduction

The demand for a resource in the national airspace system can exceed the availability of that resource. When this occurs, traffic management initiatives (TMIs) are used to restrict access to that resource, so that the quantity of flights attempting to use some resource does not exceed its availability. For example, the ability of an airport to accommodate arriving flights can decrease when there is adverse weather, and that airport may no longer be able to accommodate the scheduled quantity of arrivals. If these flights were allowed to proceed as scheduled, they would have to wait in the airspace near the airport until they have an opportunity to land. This wastes fuel, and could lead to an unsafe situation if the number of waiting flights becomes unmanageable. A type of TMI called a Ground Delay Program (GDP) can be used to prevent such a situation. In a GDP, flights destined for an impacted airport are held on the ground before they depart. These flights then take delays on the ground rather than in the air, leading to less fuel waste and avoiding a potentially dangerous situation.

This dissertation examines three topics related to the planning of GDPs at an airport. In each case, study of the target problem led to broader insights and to results for a more general type of problem.

The first topic concerns greedy algorithms in ground delay program planning. In [1], it was shown that under certain idealized conditions, the optimal manner of assigning delays in a GDP is given by a greedy algorithm called ration-by-distance (RBD). A connection between this result and an existing result for a broader class of problems was shown in [2]. More specifically, RBD is a special case of a greedy algorithm for a transportation problem, and necessary and sufficient conditions for optimality of such algorithms were given in [3]. The existing RBD algorithm is only known to be optimal in a two-stage setting. We define a similar greedy policy for a multi-stage setting. Since the existing results are connected to the transportation problem, it seems natural to suspect that the new result is connected to a multi-stage version of the transportation problem. We show that this is the case. While there is some existing work on multi-stage transportation problems [4], the existing work is mostly concerned with heuristics or approximate solution methods. As far as we are aware, none of the existing work provides a result in the multi-stage setting that is comparable to the single-stage result proven in [3]. We produce such a result, providing necessary and sufficient conditions for the optimality of greedy policies in the multi-stage setting. These results are used to show that the multi-stage greedy policy for the GDP planning is optimal. While this result is useful for determining whether or not a policy is optimal, it provides little guidance when the conditions are not satisfied. For this reason, we also develop conditions under which we can guarantee that certain assignments are optimal. This does not prescribe an entire policy, but could be used to simplify a problem or to inform heuristics. We apply this result to the problem of matching drivers with passengers in an on-demand ride

service.

The second topic concerns integer programming models for GDP planning. Empirical results indicate that an existing model has a strong formulation [5], but there are no existing theoretical results that explain this. We provide such results, and we produce a more scalable formulation that has the same strength properties. We also show that both the existing model and the new model satisfy a type of equity property. This problem can be thought of as a special case of a stochastic network flow problem. The deterministic equivalent of such problems can be represented as a flow problem on a hypergraph. While the polyhedral properties of flow problems on graphs are well known, less is known about flow problems on hypergraphs. We define a class of hypergraph flow problems connected to stochastic dynamic network flow problems. We show that the problem of finding an integral solution to one of these problems is NP-Hard, but we show that the related integer programming formulations are strong.

The third and final topic involves data support for humans planning GDP decisions. While optimization models may eventually replace human decision-making, GDPs are currently planned by humans, albeit with a set of analytics-based decision support tools. These decision makers are guided largely by their own experience and intuition. There exists a wealth of data on these programs, but for the most part there are no tools to make this data accessible to the decision makers. We propose a method that could be used as part of such a tool. This method takes a data set of ground delay programs and selects a subset of these programs as "representative" programs. This would provide guidance to decision-makers by informing them of

the types of actions that were used in the past. We define a distance measure for use with this method that ensures the results are interpretable. We compare this method against other methods designed for similar purposes. More generally, we study the problem of summarizing a data set, and we define a new class of data exploration methods that we call "representative region selection" (RRS) methods. These methods describe a data set by selecting a small set of regions of the space in which data lie. In a similar fashion to other unsupervised learning methods, there are potentially many valid criteria that can be used to measure the quality of a method. Existing literature on clustering methods has provided "axioms" that should be satisfied by any technique that measures the quality of clustering [6]. We define analogous axioms for measuring quality of representative regions, and we provide a quality measure that satisfies these axioms. Using this quality measure, we construct two RRS methods. We provide conditions under which the RRS methods satisfy statistical consistency properties, and we demonstrate that these conditions are satisfied by our two proposed methods.

This dissertation has seven chapters. The remaining six chapters are organized in three parts, corresponding to the three topics mentioned above. Each part consists of two chapters. The first chapter describes the results in the more general context, while the second chapter applies the results to the more specific problem. For the most part, each chapter can be read independently of the others. However, proofs of results in the more specific problem context usually rely upon the results provided in the broader setting. More detailed background and related literature for each problem is provided within its corresponding chapter.

Part I

Optimal Greedy Algorithms and Dynamic Ration-By-Distance

Chapter 2: Monge Properties in Dynamic, Stochastic Transportation Problems

The material in this chapter appears in the paper "Monge Properties, Optimal Greedy Policies, and Policy Improvement for the Dynamic Stochastic Transportation," which is available on SSRN, abstract number 3067310.

In this chapter, we consider a dynamic, stochastic extension to the transportation problem. For the deterministic problem, there are known necessary and sufficient conditions under which a greedy algorithm achieves the optimal solution. We define a distribution-free type of optimality and provide analogous necessary and sufficient conditions under which a greedy policy achieves this type of optimality in the dynamic, stochastic setting. We also provide weaker conditions under which it is possible to strengthen an existing policy.

2.1 Introduction.

The transportation problem is one of the most fundamental and well-known problems in combinatorial optimization. This problem involves moving resources from a set of sources to a set of destinations in a manner that minimizes costs. In its original formulation, the problem is a deterministic, single-stage problem where

all information is known and all decisions are made at once. However, in many realistic problems, decisions can take place in multiple stages, and new information may be gained as the problem progresses.

In this chapter, we consider a version of the transportation problem that is dynamic in the sense that the problem occurs over several time periods, and is stochastic in the sense that new supplies and demands arrive according to a random process. We call this problem the dynamic, stochastic, transportation problem (DSTP). This problem was originally formulated in [4], where it was applied to the problem of matching trucks to loads. It is potentially applicable in a wide range of settings. The DSTP could be used to match workers to jobs when the workers are independent contractors and the jobs are produced by customer requests, where it is uncertain both which contractors will be available and what requests will be made. One case where these conditions occur is in on-demand ride services. In these services, a customer provides her or his current location and a desired destination. The customer generally does not provide any advance notice and wishes to be taken to the destination as quickly as possible. Contractors who are willing to drive customers provide their location. The company providing the service then must match each customer with an available driver, preferably in an efficient manner. This problem is discussed in more detail in Section 3.3. Another application is that of air traffic management initiative planning, discussed in Section 3.2. There are numerous other potential applications such as matching organ donors with recipients, matching customers in online dating services, assigning customer orders to supply depots, providing ads to customers, and college admissions procedures.

A well-known result for the classic transportation problem provides conditions under which a greedy algorithm would achieve an optimal solution [3]. This result was subsequently extended to more general conditions by [7] and [8]. We provide a more complete description of this result in Section 2.3. The properties that allow for an optimal greedy solution for the transportation problem are known as Monge properties, as these types of properties were first observed by the mathematician Gaspard Monge [9]. The main result of this chapter uses Monge properties in order to provide conditions under which a type of greedy policy can achieve an optimal solution for the DSTP. A second result uses similar properties to provide conditions under which a policy can be improved. We show how these results may be applied to an air traffic management problem and an on-demand ride sharing problem respectively.

2.1.1 Layout.

A discussion of relevant work and the contributions of this chapter is given in Section 2.2. The classic formulation of the transportation problem is described in Section 2.3, and a description of the existing results for that problem is given. The DSTP is defined in Section 2.4. In Section 2.5, we define a distribution-free type of optimality for dynamic, stochastic optimization problems. The main theoretical results are presented in Section 2.6 and Section 2.7. In the former section, a class of greedy policies is defined, and necessary and sufficient conditions are provided for the optimality of these policies. Conclusions and possibilities for further research

are discussed in Section 2.8.

2.2 Relevant Work and Contributions.

Our work builds on the existing work on the DSTP and on the results concerning the classic transportation problem. Our work is also related to other fields of research.

Dynamic Stochastic Network Optimization Problems. The DSTP problem that we discuss in this chapter is essentially the same as the problem formulated in [4], although we treat the problem in slightly more generality. The main concern in [4] was the development of a tractable approximate solution method for the problem, and much of the work on dynamic, stochastic optimization problems with similar structure has also focused on developing approximate methods, e.g. [10, 11, 12, 13, 14, 15]. These works also tend to assume perfect knowledge of the distribution of the stochastic process present in the problem. We instead find conditions under which there exist simple optimal policies, and we do not use any information concerning the stochastic process. As far as the author is aware, there exists only one other work with a similar purpose. In [16] a simpler version of the problem is studied, and Monge properties are applied to this context. However, this existing work focuses on single arcs. Conditions are provided under which some specific arc should receive priority over another specific arc, and under which it is guaranteed to be optimal to assign resources to some specific arc. We instead consider entire policies, and we provide conditions under which an optimal greedy policy exists and conditions

under which a policy can be improved.

Monge Properties and Optimal Greedy Algorithms in Optimization. Many problems admit simple optimal policies when Monge properties are present. For an overview, see [17] or [18]. Almost all of the existing research applies these properties to single-stage problems. As far as the author is aware, the aforementioned work [16] is the only existing work that attempts to study these properties in a multi-period stochastic context. Other structures that allow greedy optimal policies, such as matroids and greedoids, have also been studied; see for example [19, 20, 21, 22].

Online Matching Algorithms. A special case of the DSTP is the online bipartite matching problem, first studied in [23]. In the online bipartite matching problem, there is a bipartite graph with independent sets X and Y. At the beginning of the problem, the nodes of X are visible, while the nodes of Y and all edges are hidden. In each iteration, a node y from Y and its adjacent edges are revealed by an omniscient adversary. The algorithm decides which node of X to match y with, or decides not to match y with any node. The goal is to maximize the *competitive ratio*, which is the ratio between the size of the achieved matching as compared to the largest matching that could have been produced if the nodes of Y had been known visible from the beginning. There is a substantial body of work on algorithms for this problem and variants of this problem; see for example [24, 25, 26]. This work tends to focus on bounding or characterizing the achievable competitive ratio and on producing algorithms that achieve good competitive ratios.

We instead provide conditions under which an optimal solution is achievable for a more general problem. Equivalently, this means that we provide conditions under which a competitive ratio of 1 is achievable.

Two-Sided and Many-to-Many Matching. The two-sided matching problem in economics is superficially similar to the transportation problem. In this problem, there are two disjoint groups of agents, and the goal is to match agents from one group with agents from another group. This was originally studied by [27], and there is now a extensive body of literature on two-siding matchings. For an overview, see [28]. This is a special case of the many-to-many matching problem; see for example [29, 30, 31, 32]. Two-sided matchings have also been studied in dynamic settings, e.g. [33, 34]. A significant difference between this problem and the transportation problem is that in a two-sided matching problem the agents are assumed to be rational actors with preferences. In consequence, the study of these problems largely focuses on properties such as stability and incentive-compatibility that are not relevant to the transportation problem.

2.2.1 Contributions.

The main contributions in this chapter are as follows:

- We define a distribution-free type of optimality called *oracle optimality*.
- A class of policies called *subset greedy policies* are defined for the dynamic stochastic transportation problem.

- We provide necessary and sufficient conditions under which subset greedy policies are oracle optimal.

- We provide conditions which allow for policies to be improved.

2.3 The Transportation Problem and Relevant Results.

In this section, we discuss the formulation of the transportation problem, and we present some existing results about the transportation problem that we make use of later in this chapter.

2.3.1 Formulation of Transportation Problem.

In the traditional formulation of the transportation problem, there is a set of supply nodes $\mathscr{A}$ and a set of demand nodes $\mathscr{B}$. Each supply node a has a supply s_a of some resource and each demand node b has a demand d_b for that resource. The goal is to decide how much of the resource to send from each supply node to each demand node. A pair (a, b) consisting of a supply node a and a demand node b is often referred to as an *arc*. It is assumed that resources can be moved along any arc (a, b), and that this movement would incur a cost of c_{ab} per unit. In a more general form of the problem, it may be infeasible to transport resources along some arcs. In this case, let $\mathscr{F}$ be the set of feasible arcs, a subset of $\mathscr{A} \times \mathscr{B}$. This problem has a well-known linear programming (LP) formulation given by:

$$\min \sum_{(a,b)\in\mathscr{F}} c_{ab}x_{ab}$$

subject to:

$$\sum_{b:(a,b)\in\mathscr{F}} x_{ab} = s_a \qquad \text{for all } a \in \mathscr{A},$$

$$\sum_{a:(a,b)\in\mathscr{F}} x_{ab} = d_b \qquad \text{for all } b \in \mathscr{B},$$

$$x_{ab} \geq 0 \qquad \text{for all } (a,b) \in \mathscr{F}.$$

It is well-known that the corresponding constraint matrix is totally unimodular, and thus if all of the supplies and demands are integral, then all extreme points of the LP will also be integral.

2.3.2 Conditions For Optimality of Greedy Algorithms.

Any ordering $\prec$ on the set of feasible arcs provides a type of greedy algorithm for the transportation problem. The algorithm simply considers each arc in order and moves as many of the remaining resources along the arc as possible. Necessary and sufficient conditions under which this greedy solution will be optimal are already known. Consider the sequence of feasible arcs formed by arranging the arcs of $\mathscr{F}$ according to the ordering $\prec$. This permutation is called a Monge sequence if for any a, a' in $\mathscr{A}$ and any b, b' in $\mathscr{B}$ such that

- (a,b), (a,b') and (a',b) are feasible,
- $(a,b) \prec (a,b')$ and $(a,b) \prec (a',b)$,

then the arc (a',b') is feasible and the costs satisfy

$$c_{ab} + c_{a'b'} \leq c_{ab'} + c_{a'b}.$$

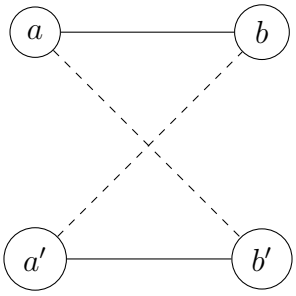


Figure 2.1: Demonstration of Monge Conditions. If $(a, b) \prec (a, b')$ and $(a, b) \prec (a', b)$ then the greedy algorithm would produce a solution using the solid arcs rather than the dashed arcs. The Monge conditions ensure that the former is feasible and no more expensive than the latter assignment.

If the supply and demand admit a feasible solution and the ordering is a Monge sequence, then the greedy procedure will produce the optimal solution. Conversely, if some greedy algorithm produces the optimal solution for every feasible choice of supply and demand, then that greedy algorithm must use an ordering that is a Monge sequence.

Theorem 1 (Due to [3, 7, 8])**.** *Let there be a transportation with feasible arcs $\mathscr{F}$ and costs $\boldsymbol{c}$. Let $\prec$ be an ordering on $\mathscr{F}$. If $\prec$ is a Monge sequence, then the greedy algorithm with ordering $\prec$ produces an optimal solution for any vectors $\boldsymbol{s}$ and $\boldsymbol{d}$ of supplies and demands that admit a feasible solution. Conversely, if the greedy algorithm with ordering $\prec$ produces an optimal solution for all vectors $\boldsymbol{s}$ and $\boldsymbol{d}$ of supplies and demands that admit a feasible solution, then $\prec$ is a Monge sequence.*

Theorem 1 was shown for unconstrained transportation problems in [3], and was later extended in [7] and [8].

The intuition behind this is as follows. Consider a simple problem where there

are two supply nodes a and a' and two demand nodes b and b'. Suppose that (a, b), (a, b') and (a', b) are all feasible and there is some ordering $\prec$ with $(a, b) \prec (a, b')$ and $(a, b) \prec (a', b)$. If there is a unit of supply at a and a' and a unit of demand at b and b', then the greedy algorithm will assign a unit to the arc (a, b) rather than arc (a, b') or (a', b). This would leave no remaining supply or demand at a or b, and one unit at a' and b'. In order for the greedy algorithm to achieve a feasible solution, the arc (a', b') must then be feasible. Furthermore, in order for the greedy solution to be optimal, then the solution that uses arcs (a, b) and (a', b') must be less expensive then the one that uses (a', b) and (a, b'). This motivates the cost requirement. A diagram of this situation is shown in Figure 2.1. While this example is relatively simplistic, this same intuition extends into more complicated situations, and is essentially the same argument that provides the proof of optimality.

2.4 The Dynamic Stochastic Transportation Problem.

We present a formal model of the dynamic stochastic transportation problem (DSTP) here. This model closely resembles the one proposed in [4], although it is slightly more general. The existing formulation is based upon the assignment problem, so that the supply or demand available at any node is always zero or one. We allow the supply or demand to be any non-negative real number. The existing formulation also assumes that any supply node can be matched with any demand node, while we allow for infeasible arcs. We also make some notational changes in order to make some results easier to state.

As in the standard transportation problem, there is a set of supply nodes and demand nodes, and there is a set of arcs between these nodes. We allow the costs of arcs and the feasibility of arcs to vary over time, but we assume that this follows a deterministic process. That is, at the beginning of the problem, it is known which arcs will be feasible in each time period and what the cost of each arc will be. The uncertainty in the problem comes from the arrival of new supplies and demands. At the beginning of the problem, there is some supply or demand present at each node, which is known. In each time period, a new, random quantity of supply or demand will arrive at each node.

The DSTP takes place in T discrete time periods. Let $\mathscr{A}$ and $\mathscr{B}$ be the supply and demand nodes respectively. Let $\mathscr{F}_t$ be the set of feasible arcs in time period t. In each time period, some amount of supply or demand may arrive at each possible node. The amount of arriving supply and demand is determined by some stochastic process Ω, which is characterized by a set of possible instances ω. Let $s^{\text{new}}(t, a; \omega)$ be the amount of supply that arrives at node a in time t under instance ω, and let $d^{\text{new}}(t, b; \omega)$ be defined similarly for demand at node b. A policy must be chosen that decides how much resource should be transported along each arc in each time period. For a policy π, let $x(t, a, b; \pi; \omega)$ be the amount of resource that is transported along arc (a, b) in time period t under instance ω. This amount is removed from both nodes a and b. Similarly to the standard problem, there is an associated per-unit cost. We allow this cost to vary over time and we let $c(t, a, b)$ be the cost of moving one unit of resource along arc (a, b) in time t.

The state of the problem at time t can be characterized by the amount of

remaining supplies and demands at the nodes. We let $s(t, a; \pi; \omega)$ be the amount of supply remaining at the beginning of time period t under policy π in instance ω, and we let $d(t, b; \pi; \omega)$ be defined similarly for demand at destination b. The transitions of the problem are given as follows. Let

$$s^x(t, a; \pi; \omega) = s(t, a; \pi; \omega) - \sum_{b:(a,b)\in\mathscr{F}} x(t, a, b; \pi; \omega);$$

$$d^x(t, b; \pi; \omega) = d(t, b; \pi; \omega) - \sum_{a:(a,b)\in\mathscr{F}} x(t, a, b; \pi; \omega).$$

The variables $s^x(t, a; \pi; \omega)$ and $d^x(t, b; \pi; \omega)$ give the amount of respective supply and demand remaining at nodes a and b after the policy π has made its allocations in time period t. The post-decision state is comprised of these variables. After the policy's allocations are made then the new supplies and demands are observed. The state at the beginning of the next time period is given by

$$s(t + 1, a; \pi; \omega) = s^x(t, a; \pi; \omega) + s^{\text{new}}(t + 1, a; \omega);$$

$$d(t + 1, b; \pi; \omega) = d^x(t, b; \pi; \omega) + d^{\text{new}}(t + 1, b; \omega).$$

We let $C(t; \pi; \omega)$ be the cost incurred in time period t by policy π in instance ω, and let $\mathbf{C}(\pi; \omega)$ be the total cost incurred by policy π in instance ω. That is,

$$C(t; \pi; \omega) = \sum_{(a,b)\in\mathscr{F}_t} x(t, a, b; \pi; \omega)c(t, a, b)$$

and

$$\mathbf{C}(\pi; \omega) = \sum_{t=0}^{T} C(t; \pi; \omega).$$

There are natural restrictions on the policy π. The amount of resource allocated along an arc cannot be more than the supply or demand present at the end nodes.

Thus, for a policy π to be valid it must be true that

$$\sum_{b:(a,b)\in\mathscr{F}_t} x(t,a,b;\pi;\omega) \leq s(t,a;\pi;\omega) \qquad \forall t \in \{0,\ldots,T\}, a \in \mathscr{A}, \omega \in \Omega;$$

$$\sum_{a:(a,b)\in\mathscr{F}_t} x(t,a,b;\pi;\omega) \leq d(t,b;\pi;\omega) \qquad \forall t \in \{0,\ldots,T\}, b \in \mathscr{B}, \omega \in \Omega.$$

A policy should not have knowledge of future events that have not happened yet, so $x(t,a,b;\pi;\omega)$ should be a function of the observed values up to time t. We require that at the end of the final time period, all of the supplies and demands have been matched, and we will say that a policy is feasible if this is true. That is, π is feasible in instance ω if

$$\sum_{t=0}^{T} \sum_{b:(a,b)\in\mathscr{F}_t} x(t,a,b;\pi;\omega) = \sum_{t=0}^{T} s^{\text{new}}(t,a;\omega) \qquad \text{for all } a \in \mathscr{A};$$

$$\sum_{t=0}^{T} \sum_{a:(a,b)\in\mathscr{F}_t} x(t,a,b;\pi;\omega) = \sum_{t=0}^{T} d^{\text{new}}(t,b;\omega) \qquad \text{for all } b \in \mathscr{B}.$$

The components that characterize a transportation problem are therefore the set of supply nodes $\mathscr{A}$, set of demand nodes $\mathscr{B}$, initial supplies $\mathbf{s}_0$, initial demands $\mathbf{d}_0$, time horizon T, cost function c, feasible arcs $(\mathscr{F}_0, ..., \mathscr{F}_T)$, and arrival process Ω. We make minimal assumptions concerning the arrival process Ω. For the problem to be well-defined, the quantities of supply and demand that arrive in each time period must be finite. In order for our results to hold, the quantities of arriving supplies and demands should depend only on some exogenous process, and not on any decisions that are made.

2.4.1 The Time-Partitioned Dynamic Transportation Problem

A special case of the DSTP is one where each node has a unique time period in which new supply or demand can arrive, and in all other time periods the node will not receive any new supply or demand. This is formalized as follows.

Definition 1 (Time-Partitioned)**.** *A DSTP is* ***time-partitioned*** *if there exists a partition* $\{\mathscr{A}_0, ..., \mathscr{A}_T\}$ *of* $\mathscr{A}$ *and a partition* $\{\mathscr{B}_0, ..., \mathscr{B}_T\}$ *of* $\mathscr{B}$ *that satisfies the following:*

- *For any time period* t *and for any* a *in* $\mathscr{A}_t$*, then* $s^{new}(\tau, a; \omega) = 0$ *for any instance* ω *and for any time period* $\tau \neq t$*.*
- *For any time period* t *and for any* b *in* $\mathscr{B}_t$*, then* $d^{new}(\tau, b; \omega) = 0$ *for any instance* ω *and for any time period* $\tau \neq t$*.*

We note that in a time-partitioned DSTP, the amount of supply remaining at any node of $\mathscr{A}_t$ will be zero at the beginning of time periods 0 to $t-1$, and will be decreasing from time period t onwards.

Lemma 1. *Let there be a time-partitioned DSTP. Let there be some* t*, and let there be some* $a \in \mathscr{A}_t$*. Let* π *be any policy, and let* ω *be any instance of any arrival process* Ω*. Then*

- *For any* $\tau \leq t$

$$s(\tau, a; \pi; \omega) = 0,$$

- *for any* $\tau > t$

$$s(\tau, a; \pi; \omega) \leq s(\tau - 1, a; \pi; \omega),$$

- *and when* $\tau = t$

$$s(\tau, a) = s^{new}(t, a; \pi; \omega).$$

This is easily observed from the definition of the problem transition and the time-partitioning property. A similar statement applies to the demand nodes. Without loss of generality, we can assume that any dynamic transportation problem is time-partitioned. If a dynamic transportation problem is not time-partitioned, the problem can be modified by creating a duplicates of each node for each time period. Then, supply that would originally arrive at some node a in some time period t will now arrive at the duplicate of a corresponding to time t, and similar modifications are made for demand. Such a problem is equivalent to the original and is time-partitioned. For time-partitioned problems, we can also assume without loss of generality that

$$\mathscr{F}_t \subseteq \left(\bigcup_{\tau=0}^{t} \mathscr{A}_\tau \right) \times \left(\bigcup_{\tau=0}^{t} \mathscr{B}_\tau \right).$$

2.4.2 The Deterministic Problem.

Let us suppose some information is obtained that guarantees that the random arrival process will follow instance ω. If the DSTP is time-partitioned, then we can formulate the problem as a single-stage transportation problem with supply nodes $\mathscr{A}$ and demand nodes $\mathscr{B}$ (this is not true in general). We refer to this problem as the deterministic problem corresponding to instance ω. Suppose some supply arrives at node a in time t and that some demand arrives at node b in time t'. This supply can be matched with this demand if and only if there is a time τ later than

t and t' such that (a, b) is feasible at time τ. For a supply node a, let $t(a)$ be the time period such that $a \in \mathscr{A}_{t(a)}$. Define $t(b)$ similarly for a demand node b. Then the set of feasible arcs for the deterministic problem, denoted $\mathscr{F}^*$, is given by

$$\mathscr{F}^* := \{(a, b) : (a, b) \in \mathscr{F}_\tau \text{ for some } \tau \geq \max\{t(a), t(b)\}\}.$$

For each feasible arc (a, b), there is a corresponding decision variable x_{ab} that determines how much supply is sent across the arc (a, b). These variables do not describe the time at which resources are sent along the arc. We make the assumption that the resource would be sent at the lowest cost time among all possible times. The corresponding cost of sending one unit of resource from a to b would then be given by

$$c^*_{ab} = \min\{c(\tau, a, b) : (a, b) \in \mathscr{F}_\tau, \tau \geq \max\{t(a), t(b)\}\}.$$

Thus, the resulting transportation problem is given by:

$$\min \sum_{(a,b)\in\mathscr{F}^*} c^*_{ab} x_{ab}$$

such that:

$$\sum_{b:(a,b)\in\mathscr{F}^*} x_{ab} = s^{\text{new}}(t, a; \omega) \qquad \forall t, a \in \mathscr{A}_t,$$

$$\sum_{a:(a,b)\in\mathscr{F}^*} x_{ab} = d^{\text{new}}(t, b; \omega) \qquad \forall t, b \in \mathscr{B}_t.$$

We will use the notation that $\mathbf{C}^*(\omega)$ is the optimal cost of the deterministic problem for instance ω. Note that the definition of a Monge sequence depends only on the feasible arcs $\mathscr{F}^*$ and costs $\mathbf{c}^*$, neither of which depend on the instance ω. Thus, we can say that a sequence is (or is not) a Monge sequence for the deterministic problem without specifying the instance ω.

Remark 1. *Let ω be an instance of some time-disjoint arrival process. A permutation of $\mathscr{F}^*$ is a Monge sequence for the deterministic problem corresponding to ω if and only if it is a Monge sequence for the deterministic problem corresponding to ω' for any instance ω' of any time-disjoint arrival process.*

This is analogous to the fact that the definition of a Monge sequence in a standard transportation problem only depends on the cost and feasibility of the arcs, and does not depend on the quantities of supply and demand.

2.5 Oracle Optimality.

For problems similar to the DSTP, the goal is typically to identify a policy that minimizes the total expected costs incurred over the time horizon. The dynamic assignment problem specified in [4] uses this type of objective. Detailed information concerning the distribution of the arrival process would generally be needed to solve this problem. We do not assume that such information is available and will therefore not attempt to find a policy that is optimal in expectation. We provide three different manners by which optimality may be defined independent of an arrival process.

The results of a policy in some instance ω. may be compared against the deterministic solution. The policy may then be defined to be optimal if it always achieves the value of the deterministic solution. We call this *oracle optimality*.

Definition 2. *A policy π is oracle optimal if*

$$\mathbf{C}(\pi;\omega) = \mathbf{C}^*(\omega)$$

for any instance ω of any possible arrival process Ω.

A policy could also be compared against other policies. We will say that a policy is *state-by-state dominant* if it always performs at least as well as any other policy.

Definition 3. *A policy π is state-by-state dominant if*

$$\mathbf{C}(\pi;\omega) \leq \mathbf{C}(\phi;\omega)$$

for any policy ϕ and any instance ω of any possible arrival process Ω.

Alternatively, it is possible to take a robust approach and consider the worst-case performance of the policy. We will say that a policy is *robustly dominant* if its worst-case performance is better than any other policy whenever such a concept is well-defined.

Definition 4. *A policy π is robustly dominant if for any policy ϕ and any arrival process Ω such that $\sup_{\omega\in\Omega} \mathbf{C}(\phi;\omega)$ exists and is finite then $\sup_{\omega\in\Omega} \mathbf{C}(\pi;\omega)$ exists and*

$$\sup_{\omega\in\Omega} \mathbf{C}(\pi;\omega) \leq \sup_{\omega\in\Omega} \mathbf{C}(\phi;\omega).$$

As it turns out, all three of these definitions are equivalent.

Proposition 1. *The following are equivalent:*

1. *The policy π is oracle optimal,*
2. *The policy π is state-by-state dominant,*
3. *The policy π is robustly dominant.*

A proof of this proposition is given in Appendix A.1. This type of optimality is much stronger than typical optimality in expectation, and not all instances have policies that attain this type of optimality. However, we can provide conditions under which it is possible to attain this type of optimality.

2.6 Subset-Greedy Policies.

In a greedy algorithm for a single-stage transportation problem, there is a complete ordering on the arcs, and resources are transferred along these arcs in order until all resources have been moved. In the setting of the dynamic transportation problem, a greedy algorithm could be applied to the remaining resources in each time period. However, it may be advantageous to omit some arcs from consideration in some time period. For example, consider a situation in which moving resources from supply node a to demand node b will be less expensive in time period $t+1$ than it would be in time period t. Then, a necessary (but not sufficient) condition for the optimality of some policy is that resources are never allocated along arc (a, b) in time period t.

Following this logic, we define a class of policies called *subset-greedy* policies. For each time period, a subset-greedy policy has an associated subset U_t of the feasible arcs $\mathscr{F}_t$, along with an ordering $\prec_t$ of U_t. The policy decides its allocations in time t by applying a greedy policy to the subset U_t using the ordering $\prec_t$. That is, in the time period t, the allocation made by a subset greedy policy π is determined by following the procedure defined in Algorithm 1. In general, it is not a trivial task

to determine which subset of arcs U_t should be used in each time period, nor is it trivial to select the ordering $\prec_t$ that should be used. Analogously, it is not a trivial task to arrange the arcs of the deterministic transportation problem into a Monge sequence if one exists. In the latter case, efficient algorithms have been developed e.g. to procure such a sequence, e.g. [35, 36]. We leave the development of such an algorithm for the general DSTP as an avenue of future research.

Algorithm 1: Subset Greedy Policy π in Time Period t.

Inputs: supply and demand nodes $\mathscr{A}$ and $\mathscr{B}$, remaining supply $s(t, a; \pi; \omega)$ for each a in $\mathscr{A}$, remaining demand $d(t, b; \pi; \omega)$ for each b in $\mathscr{B}$, subset U_t, and ordering $\prec_t$ on U_t.
Let $q_a = s(t, a; \pi; \omega)$ for each a in $\mathscr{A}$. Let $r_b = d(t, b; \pi; \omega)$ for each b in $\mathscr{B}$. Let $F = U_t$.
while F is not empty. **do**
  Let (a, b) be the minimal arc in F according to $\prec_t$. Remove (a, b) from F.
  Let $m = \min\{q_a, r_b\}$.
  Set $x(t, a, b; \pi; \omega) = m$.
  Update $q_a = q_a - m$ and $r_b = r_b - m$.
end while
Return **x**

If a subset greedy policy is optimal, then for each arc (a, b) in the subset U_t it must be true that this arc achieves its minimum cost in time t; if there were a time t' where the cost were lower, then it would be better to make assignments on (a, b) in the time period t'. Similarly, every feasible arc must appear in at least one of the subsets $U_0, \ldots, U_T$, or it is possible to construct feasible arrival instances in which the subset greedy policy is not feasible. Under these assumptions, it is possible to show that the assignments made by a subset greedy policy are equivalent to those made by a greedy algorithm for the deterministic problem. In order to simplify notation, if there is an ordering $\prec$ on some set S and there are subsets $V_1, V_2 \subseteq S$

then we will say that $V_1 \prec V_2$ if $v_1 \prec v_2$ for any $v_1 \in V_1$ and $v_2 \in V_2$. We will let $\mathscr{M}_t$ be the set of arcs in time t whose minimum feasible cost of movement is achieved in time period t. That is,

$$\mathscr{M}_t = \{(a,b) \in \mathscr{F}_t : c(t,a,b) = c^*_{ab}\}.$$

First, we can show that for any subset greedy policy, there is an equivalent subset-greedy policy where all of the subsets are disjoint.

Lemma 2. *Let there be a time-partitioned dynamic transportation problem. Let there be a subset-greedy policy π with subsets $U_0, U_1, ..., U_T$ and corresponding orderings $\prec_0, \prec_1, ..., \prec_T$. Then there exists an equivalent subset-greedy policy π' with subsets $U'_0, U'_1, ..., U'_T$ and corresponding orderings $\prec'_0, \prec'_1, ..., \prec'_T$, where the subsets $U'_0, U'_1, ..., U'_T$ are disjoint. In particular, such a policy can be achieved by choosing*

$$U'_t = U_t \setminus \left(\bigcup_{t'=0}^{t-1} U_{t'} \right)$$

and by choosing $\prec'_t$ to be the restriction of $\prec_t$ to the set U'_t.

The proof of this lemma is given in Appendix A.2. We can then show that a subset greedy policy produces an assignment that is equivalent to that of a greedy algorithm for the deterministic problem.

Lemma 3. *Let there be a time-partitioned dynamic transportation problem. Let there be a subset-greedy policy π with subsets $U_1, ..., U_T$ with corresponding orderings $\prec_1, ..., \prec_T$. If the subsets $U_1, ..., U_T$ form a partition of $\mathscr{F}^*$, then there exists an ordering $\prec^*$ on $\mathscr{F}^*$ such that for all arcs $(a,b) \in \mathscr{F}^*$ and for any instance ω of any*

arrival process Ω *then*

$$\sum_{t=0}^{T} x(t, a, b; \pi; \omega) = x^*_{a,b}(\omega),$$

where $\mathbf{x}^*(\omega)$ *is the solution provided by the greedy algorithm with ordering* $\prec^*$ *for the deterministic problem corresponding to* ω*. In particular,* $\prec^*$ *is defined by:*

1. $U_\tau \prec^* U_{\tau'}$ *for any times* τ, τ' *with* $\tau < \tau'$*,*

2. *for* $u_1, u_2 \in U_t$ *then* $u_1 \prec^* u_2$ *if and only if* $u_1 \prec_t u_2$*.*

The proof of this lemma is straightforward but tedious, and is omitted for this reason. The essence of the proof is that when the steps of the greedy algorithm are compared against those taken by the subset-greedy policy, it is observed that the same steps are performed in the same order. The only difference is that the actions of the subset-greedy policy are associated with time-periods while those of the greedy algorithm are not. Now, we can use known properties of Monge sequences to produce sufficient and necessary conditions for the oracle-optimality of a subset-greedy process.

Theorem 2. *Let there be a time-partitioned dynamic transportation problem. Let* π *be a subset-greedy policy with subsets* $U_0, U_1, ..., U_T$ *that form a partition of* $\mathscr{F}^*$*, and let* $\prec_0, \prec_1, ..., \prec_T$ *be the corresponding orderings. Let* $\prec^*$ *be the corresponding ordering on* $\mathscr{F}^*$ *as described in Lemma 3. Then* π *is oracle-optimal if any only if* $U_t \subseteq \mathscr{M}_t$ *for all* t *and* $\prec^*$ *produces a Monge sequence for the deterministic problem.*

Our conditions state that a subset-greedy policy is oracle optimal if and only if it always makes allocations at the minimum cost time and it corresponds to a Monge sequence of the deterministic problem. A proof is given in Appendix A.3.

Corollary 1. *Let there be a time-partitioned dynamic transportation problem with cost function c and feasible arcs $\mathscr{F}_0, \mathscr{F}_1, ..., \mathscr{F}_T$. This problem admits an oracle-optimal subset-greedy policy π if and only if there exists a partition $U_0, ..., U_T$ of $\mathscr{F}^*$ and an ordering $\prec^*$ of $\mathscr{F}^*$ such that:*

- *$U_t \subseteq \mathscr{M}_t$ for all t,*
- *$\prec^*$ produces a Monge sequence for the deterministic problem,*
- *$U_\tau \prec^* U_{\tau'}$ for any $\tau < \tau'$.*

This corollary states that a problem admits an oracle optimal subset-greedy policy if and only if there exists a Monge sequence for the deterministic problem such that the sequence is compatible with the times at which the arcs achieve their minimum cost. The corollary follows immediately from Theorem 2 and Lemma 2.

2.7 Monge Stars and Policy Improvements.

When a complete Monge sequence is not present, an oracle optimal subset-greedy policy is not possible, but there may be a weaker type of Monge property that can still be exploited. In this section, we provide conditions where such structure can be used to improve the quality of an existing policy. To simplify notation, this section assumes that the time period under consideration is time period zero. These results may naturally be applied to allocations in any time period, since it is always possible to treat the current time period as if it were time period zero.

We define a new type of Monge structure called a *Monge star*. This is a

property of a supply node $\aleph$ and a set B of demand nodes. The definition is as follows.

Definition 5 (Monge star)**.** *Consider a dynamic transportation problem. Let* $\aleph$ *be a supply node and let* B *be a set of demand nodes. The pair* $(\aleph, B)$ *is a* ***Monge star*** *if the following properties are satisfied for each* $\beta \in B$*:*

1. $(\aleph, \beta) \in \mathscr{M}_0$,
2. *for any* $a \in \mathscr{A}$, $b \in \mathscr{B}$, $\tau > 0$ *and* $\tau' > 0$ *such that* $(a, \beta) \in \mathscr{F}_\tau$ *and* $(\aleph, b) \in \mathscr{F}_{\tau'}$, *then there exists some* τ^* *such that* $\tau^* \geq t$, $\tau^* \geq \tau'$, $(a, b) \in \mathscr{F}_{\tau^*}$ *and* $c(0, \aleph, \beta) + c(\tau^*, a, b) \leq c(\tau, a, \beta) + c(\tau', \aleph, b)$.

Similarly to the Monge sequence property, the Monge star property involves feasibility of arcs and the cost function. In both cases, each arc from some set of arcs is compared against its adjacent arcs. In a Monge sequence, every feasible arc is compared against its adjacent arcs if those arcs occur later in the sequence. In a Monge star, the arcs of the subset are compared against the adjacent arcs that occur in later time periods. This condition ensures that matching the supplies and demands within this subset in the current time period will produce a better result than saving these resources for future periods. The presented definition involves a single supply node and a set of demand nodes. However, analogous results could be produced for a single demand node a set of supply nodes.

Theorem 3. *Let there be a time-partitioned dynamic transportation problem with a Monge star* $(\aleph, B)$*. Let* π *be any policy. Let* $\boldsymbol{r}$ *be any vector of assignments on the*

arcs of $\mathscr{F}_0$ *such that*

$$\begin{aligned}
&\sum_{b \in B} r_{\aleph b} \leq s^x(0, \aleph; \pi), \\
&\quad r_{\aleph b} \leq d^x(0, \beta; \pi) && \forall \beta \in B, \\
&\quad r_{ab} = 0 && \forall (a, b) \textit{ in } \mathscr{F}_0 \textit{ such that } a \neq \aleph \textit{ or } \beta \notin B, \\
&\quad r_{ab} \geq 0.
\end{aligned}$$

Then there exists a policy ϕ *such that*

$$x(0, a, b; \phi) = r_{ab} + x(0, a, b; \pi) \qquad \textit{for all } (a, b) \in \mathscr{F}_0$$

and such that

$$\mathbf{C}(\phi; \omega) \leq \mathbf{C}(\pi; \omega)$$

for any instance ω *of any arrival process* Ω.

This theorem essentially states the following. Suppose that after π makes its assignments in the initial time period, there is some supply left at $\aleph$ and some demand left at nodes of B. If $(\aleph, B)$ is a Monge star, then it is possible to create a new policy that improves upon the policy π by using these residual supplies and demands. This is useful because it can be difficult to decide whether to use some resources in the current time period or to save these resources for a future time period. The results in Theorem 3 can be used to identify situations where it is certainly better to use the resources in the current time period. The proof of this theorem is given in Appendix A.4.

An immediate corollary to this theorem is that if $(\aleph, B)$ is a Monge star, then there exists an optimal policy π' such that after π' makes its initial assignment, it is not possible to match any more supplies from $\aleph$ with demands from B. Let $R(\aleph, B; \pi)$ be the minimum of the remaining supply at $\aleph$ and the remaining demand at B after π has made its assignment,

$$R(\aleph, B; \pi) = \min \left\{ s^x(0, \aleph; \pi), \sum_{\beta \in B} d^x(0, \beta; \pi) \right\}. \tag{2.1}$$

Corollary 2. *Let there be a time-partitioned dynamic transportation problem with a Monge star $(\aleph, B)$. If there exists a policy π that is optimal in expectation, then there exists policy π' that is optimal in expectation such that $R(\aleph, B; \pi) = 0$.*

Note that this does not necessarily imply that the optimal policy matches the supply from $\aleph$ with demand from B. Indeed, it could be that any optimal policy π' matches all of the supply from $\aleph$ with demand nodes that are not in B. The theorem simply guarantees that there is an optimal policy that either uses all of the supply at $\aleph$ in the initial time period or uses all of the demand at B in the initial time period. The corollary stated here concerns optimality in expectation. However, this can be replaced with nearly any other notion of optimality (for example, oracle optimality) and the statement will still be true.

2.8 Conclusions and Further Work.

In this chapter, we formulate the dynamic, stochastic transportation problem (DSTP), which is an extension of the classic single-stage transportation problem. Our formulation is based on that in [4]. We provide three distribution-free ways

of defining optimality in a dynamic, stochastic transportation problem. These definitions are shown to be equivalent, and we use the term *oracle optimality* to refer to this type of optimality. We define a class of greedy policies for the DSTP, and we provide necessary and sufficient conditions for these policies to be oracle optimal. Weaker conditions are also provided under which it is possible to improve the performance of policies for the DSTP.

Our results provide conditions under which subset greedy policies are oracle optimal. It may be possible to provide broader conditions under which a weaker type of optimality, such as optimality in expectation or second-order stochastic dominance, is achieved. Other classes of policies could also be considered. For example, it may be possible to provide a characterization of when myopic policies are optimal. Classes of DSTP instances could be considered instead of classes of policies. The DSTP does not always admit an oracle optimal policy, which raises the question of which instances can be solved to oracle optimality.

There is also further work to be done regarding computational properties of the results that are presented in this chapter. We provide a characterization for when a greedy subset policy is optimal and for when a DSTP instance admits an optimal greedy subset policy. However, these results do not immediately provide an efficient algorithm for testing whether a greedy subset policy is optimal, nor do they provide an efficient algorithm for constructing an optimal greedy subset policy if one exists. Given a policy π, we provide conditions under which a superior policy ϕ exists. In the proof of this result, we construct a policy ϕ, but we did not concern ourselves with the computational efficiency of this construction, so it is likely that

there are more efficient ways to implement such a policy.

Another avenue of future research would be to apply these Monge properties to other types of dynamic problems. Monge properties have been used to provide efficient algorithms for other deterministic problems, including network flows problems and the traveling salesman problem. It would also be possible to produce dynamic, stochastic versions of these problems, and it may be possible to produce similar types of results for these problems.

Chapter 3: Monge Properties in Air Traffic Management and On-Demand Ride Services

The material in this chapter appears in the paper "Monge Properties, Optimal Greedy Policies, and Policy Improvement for the Dynamic Stochastic Transportation," which is available on SSRN, abstract number 3067310.

In this chapter, we apply the results from Chapter 2 to two problems in air traffic management and in on-demand ride services. We use these results to show that a greedy algorithm is optimal when planning a type of air traffic management initiative. We also specify conditions under which a passenger and driver should not be left unassigned in an on-demand ride services problem.

3.1 Layout

Section 3.2 describes the application in air traffic management, while section 3.3 describes the application to on-demand ride services. Each of these sections begins by providing some background for the application and some relevant literature. Conclusions are provided in Section 3.4.

3.2 Dynamic Ration By Distance

The ability of an airport to accommodate incoming flights can be negatively affected by factors such as weather that are both random and beyond human control. This can cause the amount of scheduled traffic to exceed that which the airport can handle. If these flights were allowed to continue under this schedule, the flights would be forced to form a queue in the airspace near the airport. Flights must consume fuel to stay in this queue, and it may become difficult for air traffic controllers to safely manage the flights if the queue becomes too long. In order to prevent these difficulties, an air navigation service provider (ANSP) can delay the departure of some flights destined for the impacted airport. In this way, delays are shifted from the air to the ground where they can be taken more safely and efficiently. Several integer programming formulations have been studied for this problem, for example [5, 37, 38, 39]. In [1], a greedy algorithm was shown to be optimal for a two-stage version of the problem in which flights are not allowed to take any air delays. It was later shown that this result can be related to the existing results concerning Monge properties in the single-stage transportation problems (the Monge-related results are discussed in Section 2.3). The ground holding problem was reformulated as a transportation problem and the greedy algorithm was shown to produce a Monge sequence [2]. We produce a multi-period dynamic version of the problem, and show that a greedy policy is optimal in this setting as well. As far as we know, this is the first optimal dynamic policy for the single airport ground holding problem that does not require the distribution of the stochastic process.

3.2.1 A Dynamic Model for the Single Airport Ground Holding Problem

We provide a formulation for a dynamic version of the single airport ground holding problem, which we refer to as the *multi-period single airport ground holding problem*. There is a set of flights $\mathscr{A}$. Each flight a has a corresponding flight duration $\tau^{\text{fly}}(a)$ and an original scheduled time of arrival $\tau^{\text{OTA}}(a)$. We assume there is no variance or uncertainty in flight times. Most flights are present in the schedule before the planning period of the problem. However, some flights may announce their intent to fly to the destination airport with little advance warning. Let $\mathscr{A}_0$ be the set of scheduled flights at the beginning of the planning period, and let $\mathscr{A}_t$ be the set of potential "pop-up" flights that may announce in time period t their intent to fly to the destination. We may assume without loss of generality that each potential pop-up flight has a unique time $\tau^{\text{new}}(a)$ at which this announcement may occur, and if the announcement does not occur at this time then the flight will not take place. Let $\tau^{\text{new}}(a)$ be defined to be zero for those flights a that are present in the schedule at the initiation of the planning horizon. Then according to some random process, in each time period t, some subset of the potential pop-up flights $\mathscr{A}_t$ will announce their intent to fly, while the remaining potential flights of $\mathscr{A}_t$ will not occur.

There is also a set of slots. A slot b has a corresponding time $\tau^{\text{slot}}(b)$, and it represents the resources required to accommodate the arrival of a single flight at the airport in the corresponding time period. Multiple slots may have the same

corresponding time period if the airport can accommodate multiple arrivals in the same time period. Since exogenous, random elements such as weather can affect the ability of the airport to handle arrivals, then the availability of slots is stochastic. We may assume that for each slot b, there is a unique time period $\tau^{\text{new}}(b)$ in which it is revealed whether or not that slot will exist. Let $\mathscr{B}_0$ be the set of slots that are certain to exist at the beginning of the planning horizon, and let $\mathscr{B}_t$ be the set of slots whose existence (or non-existence) is revealed in time period t. If a flight were assigned to a slot whose availability is not certain, then there is a risk that the flight would not be able to land when it reaches the airport. This flight would have to wait in the air until another slot is available. For the purposes of this problem, we do not allow such a situation. Thus, at any given time, we only allow assignments involving slots that are certain to exist.

As an example, suppose that in good conditions a particular airport can accommodate two flights per minute, but when there is fog the airport can accommodate only one flight per minute. Suppose that our planning horizon begins at 6:00 a.m. and fog is present at the beginning of this planning horizon, and suppose that we divide the planning horizon into one minute time periods. It is uncertain when the fog will clear. Then, in the initial stage of the problem, there will be a single slot corresponding to each of the times 6:00 a.m., 6:01 a.m., 6:02 a.m., and so on. No new slots will become present until the fog clears, at which point a new slot will be available in every minute greater than the current time. Consider some possible time at which the fog may clear, for example 8:00 a.m. Then the set $\mathscr{B}_{8:00}$ consists of those slots whose existence (or non-existence) is revealed at 8:00 a.m. Thus, for

any slot $b \in \mathscr{B}_{8:00}$, the time $\tau^{\text{new}}(b) = 8{:}00$ a.m. The slot times in $\mathscr{B}_{8:00}$ vary. For each of the slot times 8:00 a.m., 8:01 a.m., and so on there is a single corresponding slot of $\mathscr{B}_{8:00}$ with that given slot time. If the fog clears at 8:00, all of these slots will be revealed, while if the fog clears at any other time a different set of slots will be revealed.

At the beginning of the planning period each scheduled flight may tentatively be assigned a time slot at the airport. When such an assignment is made, the flight is issued a controlled time of departure, which is the time that the flight would need to depart in order to make the slot time. That is, if a flight a were assigned to a slot b, then the resulting controlled time of departure would be:

$$\tau^{\text{CTD}}(a, b) = \tau^{\text{slot}}(b) - \tau^{\text{fly}}(a).$$

If the controlled time of departure is reached and no changes have been made, the flight departs at that time. Once a flight has departed, the assigned slot cannot be altered. However, if new information is obtained before the flight departs, the flight may be reassigned to a new slot. For example, if fog clears at an airport and new slots become available, then perhaps a flight could be given an earlier slot. In each time period, a policy produces a new assignment for all remaining flights. The cost of a policy is measured by the delay experienced by each flight. Let b be the final slot that a flight a is assigned to when it departs. Then the delay is given by

$$\tau^{\text{slot}}(b) - \tau^{\text{OTA}}(a).$$

There are constraints on the assignments that can be made. We assume that a flight

cannot arrive earlier that its scheduled arrival time. Thus, it must be true that

$$\tau^{\text{OTA}}(a) \leq \tau^{\text{slot}}(b)$$

for any flight a assigned to a slot b. A flight also cannot be assigned to a slot in a time period t if the flight could not possibly reach the destination before the slot time occurs. That is, it should be true that

$$t + \tau^{\text{fly}}(a) \leq \tau^{\text{slot}}(b)$$

for any flight a assigned to a slot b in time period t. Finally, each flight can be assigned to at most one slot, and no slot can be assigned to more than one flight. A solution to this problem is feasible if every flight is assigned to a slot by the end of the problem. However, the number of slots can exceed the number of flights, and it is not necessary for every slot to be assigned to a flight. The goal is to minimize the total incurred ground delay.

3.2.2 Optimality of Dynamic Ration-by-Distance

The ration-by-distance (RBD) procedure [1] is known to be optimal for the first stage of a two-stage version of the problem. This algorithm simply considers slots in order of slot time and assigns each slot to the flight with the longest flight time that can use the slot. The second stage can then be solved as a simple special case of an assignment problem. We extend the ration-by-distance (RBD) algorithm into a policy for the dynamic ground holding problem, and we refer to this new policy as the dynamic ration-by-distance (D-RBD) policy. In the initial stage, the

D-RBD policy uses the RBD algorithm to produce a tentative assignment of flights to slots. In each subsequent stage, the RBD algorithm is rerun to produce a new set of tentative assignments. In each time period t, if a flight must depart in time period t in order meet its tentatively assigned slot time, then that flight departs and the assignment is realized. Thus, in time period t, if a flight a is tentatively assigned to a slot b such that

$$t + \tau^{\text{fly}}(a) = \tau^{\text{slot}}(b)$$

then the flight will depart immediately and that assignment is in fact a permanent assignment. This policy is described in detail in Algorithm 2.

Algorithm 2: Dynamic Ration-by-Distance (D-RBD) algorithm for time period t

Inputs:
- set $\mathcal{A}$ of flights that have not departed by time t.
- set $\mathcal{B}$ of slots that have not received a permanent assignment by time t.

Let $A = \mathcal{A}$. This stores the flights that have not yet been handled in this time period.
for Each slot b in $\mathcal{B}$, ordered by $\tau^{\text{slot}}(b)$ ascending: **do**
  Let A' be the flights a in A such that $\tau^{\text{OTA}}(a) \leq \tau^{\text{slot}}(b)$ and $t + \tau^{\text{fly}}(a) \leq \tau^{\text{slot}}(b)$.
  if A' is not empty **then**
    Let a be a flight of A' with maximum flight duration $\tau^{\text{fly}}(a)$.
    if $t + \tau^{\text{fly}}(b) = \tau^{\text{slot}}(a)$ **then**
      Permanently assign flight a to slot b, i.e., flight a departs for slot b.
    else
      Tentatively assign flight a to slot b.
    end if
    Remove a from A.
  end if
end for

It can be shown that the D-RBD policy is oracle optimal for this multi-period ground holding problem.

Theorem 4. *The D-RBD policy is oracle optimal for the dynamic single airport ground holding problem.*

A proof of this theorem is given in Appendix B.1. The intuition is as follows. Suppose that a flight a_1 could make use of a slot b_1 if it were to depart immediately. The benefit of not taking this action is that the slot remains available for some later flight a_2. Then, flight a_1 must be assigned to some later slot b_2. It is possible to show that flight a_2 can be assigned to b_2. It is also possible to show that when flights a_1 and a_2 are assigned to slots b_1 and b_2 respectively then the total delay is no more than when these assignments are reversed. In fact, this is essentially observing that the problem exhibits Monge structure. This allows us to make use of the results developed in Section 2.6. First, we show that the problem can be reformulated as a DSTP. Next, we show that the D-RBD policy is equivalent to a subset greedy policy. Then, we show that this subset greedy policy satisfies the conditions of Theorem 2. This proves the optimality. As far as we are aware, this is the first greedy, dynamic policy proven optimal in this context.

3.3 Policy Improvement for On-demand Car Services

Consider an on-demand ride service provider that connects each prospective passenger with a driver who is willing to pick up the passenger and transport him or her to the desired destination. We consider the setting of an on-demand ride service (ODRS) based on ride-hailing apps. Using a mobile device, a passenger may at any time state his or her location and desired destination. The ODRS has visibility on

the location of available drivers and potential passengers. However, both passenger requests and drivers may appear with no advanced warning. The role of the ODRS is to match drivers and passengers taking into account three performance criteria: passenger waiting time, driver waiting time and excess (non-revenue) driver travel time. Several models and optimization approaches have been proposed for this problem. See for example [40, 41, 42, 43] or [44]. We consider a relatively simple model that is a special case of the dynamic transportation problem.

3.3.1 A Model for the On-Demand Ride Service Problem

We use a fixed time horizon with T discrete time periods. Suppose there is a space M of possible locations where both drivers and passengers may appear. Assume that there is a distance measure δ on M, so that $\delta(l_1, l_2)$ gives the distance between the locations l_1 and l_2 of M. This distance measure is assumed to be a metric, so that:

- $\delta(l, l') \geq 0$ for any $l, l' \in M$,
- $\delta(l, l') = 0$ if and only if $l = l'$ for any $l, l' \in M$,
- $\delta(l, l') = \delta(l', l)$ for any $l, l' \in M$,
- $\delta(l_1, l_2) + \delta(l_2, l_3) \geq \delta(l_1, l_3)$ for any $l_1, l_2, l_3 \in M$.

In each time period, some new drivers may become available at any of the locations in M. Each driver is characterized solely by her or his location and the time at which he or she became available. Similarly, new passengers may make a request

at any of the locations of M, and each passenger is entirely characterized by the location and time at the request is made. The passenger would in practice also have an associated destination. We assume that the cost of transporting the passenger between the pickup location and the destination does not depend on the driver that is assigned to the passenger. We also assume that the distribution of future driver locations is not dependent on the destinations of the passengers that drivers are assigned to. Thus, we can omit any information concerning destinations. In reality, this last assumption is not true. If the driver is willing to pick up a new passenger soon after dropping off a passenger, then the driver's location will be close to the previous passenger's destination. A practical matching mechanism would likely need to take this under consideration. While the assumption may not be entirely realistic, this simplification allows us to develop some criteria under which the assignment of some passenger to some driver is likely to be a good assignment. Further work would be required to combine this guideline with other realistic considerations in order to develop an effective mechanism.

We will refer to the time at which a passenger makes a request or the time at which a driver becomes available as the arrival time of that driver or passenger. For a passenger or driver a, we will notate this as $t^{\text{arr}}(a)$. At any time, any available driver may be assigned to any passenger. This incurs a cost associated with sending the car to the pickup location that depends on the distance from the driver's location to the passenger's location. There is also a cost associated with how long the driver had to wait and how long the passenger had to wait. We assume that the cost of waiting is the same for all passengers and is the same for all drivers, but the cost for

drivers can be different than that of passengers. When a driver has been assigned to a passenger, both are removed from consideration in the problem.

This problem can be posed as a dynamic transportation problem in the following manner. The set of supply nodes $\mathscr{A}_t$ that can receive supply in time period t consists of the set of passengers that could potentially request a ride in that time period. Similarly, the set of demand nodes $\mathscr{B}_t$ consists of those drivers that could potentially become available in time period t. The stochastic arrival process then determines which potential passengers and drivers arrive in each time period. For each new passenger or driver, one unit of supply or demand arrives at the corresponding node. The cost of assigning passenger a to driver b in time period t is

$$c(t, a, b) = \delta(a, b) + w^p\left(t - t^{\text{arr}}(a)\right) + w^d\left(t - t^{\text{arr}}(b)\right).$$

Here, δ is the aforementioned distance function between the location of passenger a and the driver b. The function w^p is the passenger cost function, and is a function of the amount of time that the passenger waits to be matched. The function w^d is the driver cost function, and is a function of the amount of time that the driver waits to be matched. We do not forbid any assignments of drivers to passengers, so all arcs are feasible. If an assignment is prohibitively expensive, this should be reflected in the costs.

We make some assumptions about the cost of waiting. The cost of waiting should not decrease as the waiting time increases, so that

$$w^p(t') \geq w^p(t) \text{ for any } t' > t,$$

and similarly for w^d. We also assume that a passenger will become increasingly

impatient the longer she or he waits. Therefore, the cost of waiting an additional time period increases as the passenger continues to wait,

$$w^p(t+2) - w^p(t+1) \geq w^p(t+1) - w^p(t) \text{ for any } t.$$

We make a similar assumption for the driver waiting cost w^d. We refer to functions with this property as *marginally non-decreasing* functions.

3.3.2 Monge Stars for On-demand Car Services.

In general, a myopic policy for the on-demand car service is not necessarily optimal. For example, suppose there is currently one passenger request and one driver available. The myopic solution would be to assign the driver to the passenger. If the driver is very close to the passenger, this is likely the best solution. However, if the driver is very far from the passenger, then it is possible that a new driver will become available in a location closer to the existing passenger and that a new passenger will make a request closer to the existing driver. In this case, it may be better not to make the myopic assignment. This would incur greater waiting costs, but allows the possibility of reducing the costs of sending the drivers to the locations of passengers. This raises the question of when the myopic assignment should be made. Our results on Monge subsets (discussed in Section 2.7) can be used to provide conditions under which such an assignment should be made.

Theorem 5. *Let π be a policy. Let $\aleph$ be a passenger and let B be a set of drivers, who are all unassigned after π makes its assignment in period t, and suppose that the following are satisfied:*

1. *There are no unassigned passengers with an earlier arrival time than* $\aleph$*;*

2. *For any driver* β *in* B*, there are no unassigned drivers with an earlier arrival time than* β*;*

3. $2\delta(\aleph, \beta) \leq w^p\left((t+1) - t^{arr}(\aleph)\right) - w^p\left(t - t^{arr}(\aleph)\right)$*;*

4. *For any* $\beta \in B$*,* $2\delta(\aleph, \beta) \leq w^d\left((t+1) - t^{arr}(\beta)\right) - w^d\left(t - t^{arr}(\beta)\right)$*;*

5. *The waiting cost functions* w^p *and* w^d *are non-decreasing and marginally non-decreasing.*

For any driver β *in* B*, there is a policy* ϕ *that matches* $\aleph$ *with* β *in time period* t *and that performs at least as well as* π *under any possible arrival process.*

To summarize, this theorem concerns the unassigned passenger and a subset of the drivers that have been waiting the longest. If the marginal costs of waiting for both the drivers and the passenger are greater than twice the distance cost, and if the waiting cost functions satisfy the assumptions that we have made, then it is better to assign the passenger to any of the drivers rather than to leave the passenger unassigned. In order to prove this theorem, we show that $(\aleph, B)$ is a Monge star and then we apply the results from Theorem 3. The proof is given in Appendix B.2.

3.4 Conclusion

Two applications of the DSTP are presented. The first application is the problem of issuing ground delays to flights destined for a particular airport. We provide a greedy, dynamic policy for this problem. Our results for the DSTP are used

to show that this policy is oracle optimal. The second application is the problem of matching passengers with drivers for an on-demand ride service. We provide conditions under which we can guarantee that it is better to assign a passenger with a driver rather than to leave these agents unassigned. Further work could develop heuristics or implementation strategies based on these observations.

Part II

Hypergraph Flows and Strong, Equitable Integer Program Formulations

Chapter 4: Facets in F-Graphs

In this chapter, we study a type of network flow problem that we call the minimum-cost F-graph flow problem. This problem generalizes the typical minimum-cost network flow problem by allowing the underlying network to be a directed hypergraph rather than a directed graph. This new problem is pertinent because it can be used to model network flow problems that occur in a dynamic, stochastic, environment. We formulate this problem as an integer program, and we study specifically the case where every node has an edge with no capacity constraint. We show that even with this restriction, the problem of finding an integral solution is NP-Hard. However, we can show that all of the inequality constraints of our formulation are either facet-defining or redundant.

4.1 Concepts from Integer Programming

We begin by providing some basic definitions and results that are important for integer programming, and that are necessary for understanding this chapter. A more detailed explanation of these concepts can be found in any standard textbook

on integer programming, such as [45]. An integer program (IP) takes the form

$$\min \mathbf{c}^T \mathbf{x}$$

subject to:

$$A\mathbf{x} \leq \mathbf{b}$$

$$\boldsymbol{x} \in \mathbb{Z}^n$$

where A is a $m \times n$ matrix and $\mathbf{b}$ is a vector of $\mathbb{R}^m$. That is, the goal is to minimize a linear function over the set of integral points that fall within the polytope defined by $\{\boldsymbol{x} \in \mathbb{R}^n : A\boldsymbol{x} \leq b\}$. Each row A_i of the matrix A in combination with the corresponding element b_i of $\boldsymbol{b}$ can be referred to as a constraint. More generally, we will use the term ***constraint*** to refer to any pair $(\boldsymbol{a}, b)$ consisting of a vector $\boldsymbol{a}$ in $\mathbb{R}^n$ and a real number b. Such a pair is taken to represent the inequality

$$\boldsymbol{a}^T \boldsymbol{x} \leq b.$$

A constraint $(\boldsymbol{a}, b)$ is ***valid*** for a set of points S if

$$\boldsymbol{a}^T \boldsymbol{x} \leq b$$

for all $\boldsymbol{x} \in S$.

For a given set of integral points S in $\mathbb{Z}^n$, a polytope $\mathscr{P}$ in $\mathbb{R}^n$ is said to be a ***valid formulation*** for S if the integral points in $\mathscr{P}$ are exactly the points of the set S. The feasible region of an IP may have many valid formulations. Existing methods for solving IPs tend to perform better when the polytope representing the feasible region is as small as possible. If two polytopes $\mathscr{P}_1$ and $\mathscr{P}_2$ are both valid

for the same set of points, then we say that $\mathscr{P}_1$ is stronger than $\mathscr{P}_2$ if $\mathscr{P}_1 \subset \mathscr{P}_2$. For a given set S of integral points in $\mathbb{Z}^n$ that has a valid formulation, it can be shown that the strongest valid polytope is exactly the convex hull of S, defined by

$$\text{conv}(S) = \left\{ \sum_{i=1}^{k} \alpha_i \boldsymbol{s}_i : \boldsymbol{s}_1, ..., \boldsymbol{s}_k \in S, \alpha_1, ..., \alpha_k \in \mathbb{R}_0^+, \sum_{i=1}^{k} \alpha_i = 1 \right\}.$$

Here, by $\mathbb{R}_0^+$, we mean the set of non-negative real numbers.

A set of points $\boldsymbol{x}_1, ..., \boldsymbol{x}_k$ in $\mathbb{R}^k$ is said to be ***affinely independent*** if there are no real numbers $\alpha_1, ..., \alpha_k$ such that:

- $\sum_{i=1}^{k} \alpha_i = 0$,
- $\sum_{i=1}^{k} \alpha_i \boldsymbol{x}_i = 0$,
- there is some i such that $\alpha_i \neq 0$.

The ***dimension*** of a polytope of $\mathscr{P}$ (or, more broadly, any set in $\mathbb{R}^n$) is d if the size of a maximal set of affinely-independent points in $\mathscr{P}$ is $d + 1$. A polytope in $\mathbb{R}^n$ is ***full-dimensional*** if its dimension is n. Given a polytope $\mathscr{P}$ of dimension d, a constraint $(\boldsymbol{a}, b)$ is said to be ***facet-defining*** for $\mathscr{P}$ if the constraint is valid and the dimension of

$$\{\boldsymbol{x} \in \mathscr{P} : \boldsymbol{a}^T \boldsymbol{x} = b\}$$

is equal to $d - 1$.

Strong formulations for integer programs can be obtained by identifying the facet-defining constraints of the convex hull of integer solutions. Suppose that there is some full-dimensional polytope $\mathscr{P} = \{x \in \mathbb{R}^n : A\boldsymbol{x} \leq \boldsymbol{b}\}$ that is valid for some

set S of integral points in $\mathbb{Z}^n$, and suppose that one of the constraints (A_i, b_i) of this polytope is facet-defining for the convex hull of S. Further suppose that some polytope $\mathscr{P}' = \{x \in \mathbb{R}^n : A'\boldsymbol{x} \leq \boldsymbol{b}'\}$ is also valid for S and is stronger than $\mathscr{P}$. Then, it can be shown that $\mathscr{P}'$ has a constraint (A'_j, b'_j) such that

$$(A'_j, b'_j) = (\lambda A_i, \lambda b_i)$$

for some positive real number λ. That is, if a full-dimensional polytope has a constraint that is facet-defining for the convex hull of integral solutions, then any stronger polytope also has this constraint (possibly multiplied by a constant).

4.2 Background, Contributions and Layout

The minimum-cost network flow problem is a well-studied combinatorial optimization problem with many applications. In this problem, some nodes in a directed graph supply some quantity of resource while others demand quantities. The goal is to route the resource through the graph from the supply nodes to the demand nodes while minimizing costs. This problem has a well-known formulation as a linear program (LP) where the constraint matrix of this LP is totally unimodular (TU), ensuring that any extreme point of the feasible region is integral. Thus, the simplex method produces an integral optimal solution (see for example [46]).

Many real problems are dynamic, and decisions happen in multiple stages. When there is no uncertainty in the problem parameters, a dynamic network flow problem can be reduced to a static network flow problem on a *time-extended* graph. Each node of the time-extended network corresponds to a node of the physical

network at a certain point in time. See for example [47], [48], [49], or [50] for surveys on these types of problems.

Practical applications of this problem often have stochastic elements. For example, the time required to ship some resource along a road may be uncertain, as may be the demand for a resource in a future time period. Several approximate solution methods for stochastic, dynamic network flow problems have been studied [e.g. 10, 11, 12, 13, 14, 15, 51], but there is little existing research on the polyhedral structure of these problems. Unlike the deterministic variant, stochastic network flow problems do not exhibit the same structure as a static network flow problem on a directed graph. Instead, these stochastic problems can be expressed as a deterministic problem on a directed hypergraph, specifically a class of hypergraphs called F-graphs. We discuss this connection in Section 4.3, and existing works have also noted that network problems with uncertainty can be expressed as problems on directed hypergraphs [e.g. 52].

As far as we are aware, the only existing work that studies integrality properties of polyhedra resulting from stochastic network flow problems is [12]. This existing work provides a characterization of the extreme points of a stochastic dynamic network in which there is exactly one supply node and one demand node. This characterization is then used in a decomposition algorithm. Polyhedral properties of some similar problems have been studied. A special class of stochastic network flow problems was shown in [53] to have TU constraints. The extreme points of a class of linear programming problems called Leontief substitution systems was studied in [54]. In [55], these problems were expressed as flow problems on hypergraphs, some

properties of extreme point solutions were proved, and conditions were provided under which solutions to these problems are integral. These results can be applied to stochastic network flow problems when the edges are have no capacity constraints, but cannot be applied to networks with capacitated edges. A later work [56] examined simplex operations for hypergraph flow problems with edge capacities, but did not provide any results related to the integral feasible points of these problems.

In this chapter, we provide a more detailed study of the convex hull of integral solutions arising from the minimum-cost flow problem on a class of hypergraph flow problems that arise in stochastic network flow problems. We show that the problem of finding an integral optimal solution to such a problem is NP-hard, and we provide necessary and sufficient conditions under which each constraint defines a facet of the convex hull of integral feasible solutions.

The layout of the chapter is as follows. In Section 4.3 we define a dynamic stochastic network flow problem that motivates our interest in the F-graph flow problem. Section 4.4 provides a definition of an F-graph and the minimum-cost F-graph flow problem. Section 4.5 defines some important structure occurring in F-graphs, and defines a special class of F-graphs that we refer to as ABF-graphs. This class of graphs is a generalization of directed acyclic graphs (DAGs). Several lemmas that assist in the construction of feasible solutions for F-graph flow problems are provided in Section 4.6. The problem of finding an optimal integral solution to a F-graph flow problem in which every node has at least one uncapacitated edge is shown to be NP-Hard in Section 4.7. Section 4.8 shows that each inequality constraint of the F-graph flow problem is either facet-defining or redundant. A

full-dimensional formulation whose constraints are equally strong is also provided. Conclusions are provided in Section 4.9.

4.3 Motivation: Stochastic Network Flow Problem

The problem that motivates this work is a general type of dynamic stochastic network flow problem. The problem takes place in a discretized time horizon $t = 1, ..., T$. In each time period, the decision maker is faced with a set of nodes. At each node, there is a quantity of resources and some outgoing edges whose costs and capacities are known. The decision maker must choose how much of each resource to assign to each edge. All of the resources present at each node must be allocated, and the capacities of the edges must be respected. At the time of allocation, it may be uncertain what will happen to the resources allocated to each edge, as these resources will reach their destination in some later time period. The resources may be "delivered", which is to say the they are removed from the system and no longer have any effect on the problem. Otherwise, the resources allocated along the edge will eventually arrive at some destination node. However, both the destination node and the time at which the resources will arrive can be uncertain.

After the decision has been made in some time period t, the transition to the next time period $t + 1$ can be described as follows. First, if any resources allocated in previous time periods reach their destination at time $t+1$, then this destination is revealed. If these resources are "delivered", then they are removed from the problem. Otherwise, these arriving resources contribute to the quantity of resources available

at their respective destination nodes. Second, some (possibly zero) new quantity of resource is produced at each node. Finally, the outgoing edges that are available in time $t+1$ are revealed, along with their corresponding costs and capacities. Any of these transitions can be uncertain, and would be determined by an underlying stochastic process.

We use the convention in the following discussion that edges in each time period are considered to distinct from those in other time periods; the edge that departs v in time period t and arrives at w in time period $t+1$ is considered a different edge than one that departs v in time period $t+1$ and arrives at w in time period $t+2$. Since each edge has a corresponding time period, we omit time periods from variables associated with edges.

4.3.1 Markov Decision Process Formulation

This is rigorously formulated as a Markov decision process as follows. Let there be a set of nodes V. In the initial time period, for each node v there is a non-negative quantity q_v^0 and a set of outgoing edges $E_0^+(v)$. Each edge e in $E_0^+(v)$ is either uncapacitated or has a corresponding capacity $a^0(e)$ and cost $c^0(e)$. Let E_t^+ be the set of all outgoing edges present at time t, and let $\mathscr{C}_t$ be the set of capacitated edges in time t. After the initial time period, the state of the problem in the time period t is described by:

- the quantity $q_{t,v}$ of resource at each node v in V present at time t,
- the outgoing edges E_t^+ available at time t,

- the cost c_e of each edge e in E_t^+,
- the capacity a_e of each edge e in $\mathscr{C}_t(v)$,
- the set of edges R_t that were assigned flow in some previous time period that have yet to reach their destination,
- the quantity of flow f_e that was assigned to each edge e of R_t.

In each time period, the decision to be made is the amount of flow f_e to be assigned to each outgoing edge of each node, with the requirement that the entire quantity of resource present at each node is assigned to outgoing edges of the node,

$$\sum_{e \in E_t^+(v)} f_e = q_{t,v} \qquad \forall v \in V,$$

and that the flow does not exceed capacity,

$$f_e \leq a_e \qquad \forall e \in E_t^+.$$

This incurs a cost of

$$C^t(\boldsymbol{f}^t) = \sum_{e \in E_t^+(v)} c_e f_e. \tag{4.1}$$

The post-decision state is formed simply by adding the new allocations to the collection of allocated resources that have yet to reach their destination. We will use the notation R_t^f to refer to the post-decision set of edges that have received flow,

$$R_t^f := R_t + E_t^+.$$

Then, the post-decision state consists of the edges e of R_t^f and the associated flows f_e.

Let Ω be the set of possible realizations of the uncertain parameters. We refer to the elements of Ω by the conventional term ***scenarios***. Let $E_t^-(v;\omega)$ be the set of edges e that arrive at v in time period t under scenario ω, and let $E_t^-(\omega)$ be the set of all edges that are delivered or reach their destination in time period t under scenario t. Let $\kappa_{t,v}(\omega)$ be the quantity of new demand that is produced in time period t at node v under scenario ω. For time period t, let $E_t^+(\omega)$ by the set of available outgoing edges under scenario ω, and let $\mathscr{C}_t(\omega)$ be the set of capacitated edges under scenario ω. There is possible notational confusion between the set of outgoing edges $E_t^+(v)$ from a node v and the set of all outgoing edges $E_t^+(\omega)$ in a scenario ω, but this should be clear from context. For each edge e in E_t, let $c_e(\omega)$ be the corresponding cost and for each edge e in $\mathscr{C}_t$ let $a_e(\omega)$ be the corresponding capacity in scenario Ω. Then, the transitions are given by:

$$
\begin{aligned}
q_{t+1,v} &= \kappa_{t+1,v}(\omega) + \sum_{e:E_{t+1}^-(v;\omega)} f_e && \forall v \in V, \\
E_{t+1}^+ &= E_{t+1}^+(\omega), \\
c_e &= c_e(\omega) && \forall e \in E_{t+1}^+, \\
a_e &= a_e(\omega) && \forall e \in \mathscr{C}_{t+1}, \\
R_{t+1} &= R_t^f \setminus E_{t+1}^-(\omega).
\end{aligned}
$$

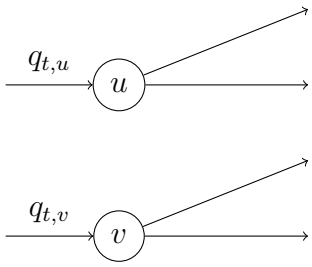
Naturally, the quantity of flow assigned in previous time periods does not change over time, so we have omitted this from the description of the transition. The state may also include the information state. For example, in time period t some information may be revealed regarding which arcs will be available in time $t+1$. The form of the

information state is dependent on the particular problem. The framework provided here is applicable for a broad range of stochastic processes and information states. However, we do require that all of the stochastic elements (the available outgoing edges, the destinations of edges, the quantities of resources produced, the capacities, and the costs) are independent of any decision that is made and are determined entirely by an exogenous random process. The goal is to identify a policy that minimizes the total expected costs. The problem is summarized in Figure 4.1.

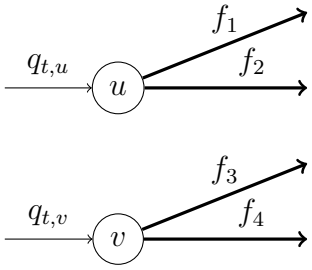
4.3.2 Example: Production Planning

Here, we give a more concrete example of a stochastic network flow problem. Suppose that a company has some factories where a product is manufactured, and some stores where a product is sold. Each factory has a maximum quantity of resources that it can produce in each time period. Any resources produced in a time period must either be shipped to a store or stored at the location of production. It takes a certain amount of time to ship product to a store. In each time period, the store sells some quantity of product and generates some revenue. The quantity sold cannot exceed the demand for this time period, which is not known in advance. Any unsold product must be stored at the store, which incurs a storage cost.

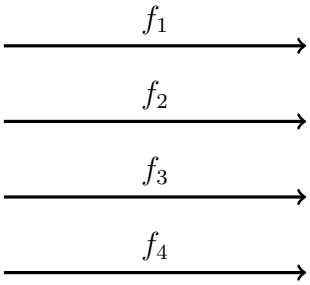
Let us define the parameters of the problems as follows:



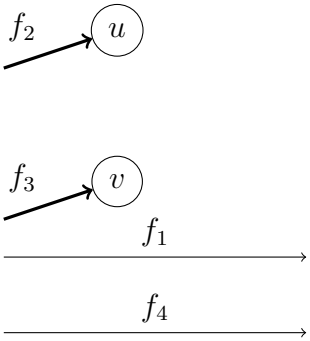
(a) Initial State: quantity of resources at each node, available edges.



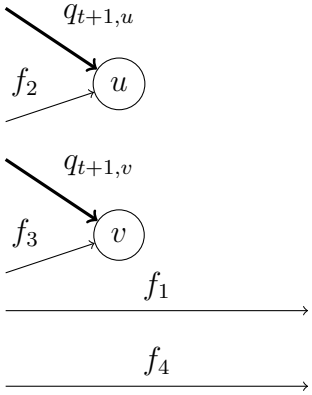
(b) Decision: quantity of resource allocated to each edge.



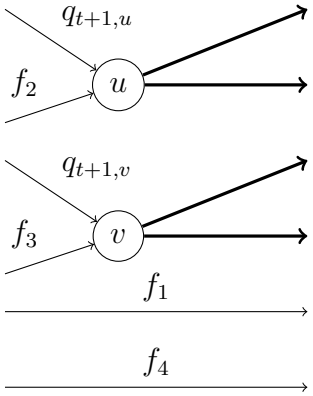
(c) Post-decision state: quantity of resources that have yet to reach their destination.



(d) Transition I: some previously-allocated resources reach their destination.



(e) Transition II: new resources are produced at nodes.



(f) Transition III: new edges become available.

Figure 4.1: Stochastic Network Flow Problem

T - number of time periods.

X - set of factory locations.

Y - set of store locations.

$m_{t,x}$ - maximum quantity of resource that can be produced in time period t at factory x.

$c_{t,x}^{\text{prod}}$ - per-unit cost to produce the resource at factory x in time period t.

$c_{t,z}^{\text{store}}$ - per-unit cost to store resource at factory or store z.

$c_{t,x,y}^{\text{ship}}$ - per-unit cost to ship the resource from factory x in time period t to arrive at store y in some later time period.

$\tau(x, y)$ - time required to ship the resource from factory x to store y.

$r_{t,y}$ - per-unit revenue from selling the resource at store y in time period t.

$a_{t,d}$ - the demand at store d in time period t.

For the moment, assume that the demand $a_{t,d}$ is uncertain, while all other parameters are known in advance. This can be formulated as a stochastic network flow problem in the following fashion. The vertices are given by $V = X \times Y$. That is, each vertex either corresponds to a factory or to a store. In each time period, $m_{t,x}$ resources are produced at each factory x, while no resources are produced at any store.

Each vertex corresponding to a factory x has the following available outgoing edges in time period t:

- a single edge $e_{t,x}^{\text{store}}$ that represents storage at x. The cost of this edge is $c_{t,x}^{\text{store}}$.

- for each store location y, an edge $e^{\text{store}}_{t,y}$ that represents shipping resources from x to y. The cost of this edge is $c^{\text{ship}}_{t,x,y}$.

- a single edge $e^{\text{unused}}_{t,x}$ that represents unused production capacity, with cost $-c^{\text{prod}}_{t,x}$. That is, producing k units less than the maximum production capacity would save $kc^{\text{prod}}_{t,x}$ units of cost.

Each vertex corresponding to a factory y in time period t has the following available outgoing edges:

- a single edge $e^{\text{store}}_{t,y}$ that represents storage at y. The cost of this edge is $c^{\text{store}}_{t,y}$.

- a single edge $e^{\text{sell}}_{t,y}$ that represents selling resources from x to y. The cost of this edge is $c^{\text{sell}}_{t,y}$, and the capacity of this edge under scenario ω is $a_{t,y}(\omega)$.

In each time period t, a single incoming edge will reach each factory x, specifically the storage arc $e^{\text{store}}_{t-1,x}$. Similarly, the storage arc $e^{\text{store}}_{t-1,y}$ will reach each store location y. In addition, for each factory x, the shipping arc $e^{\text{ship}}_{t-\tau(x,y),y}$ will reach the store location y. In this case, the network structure does not depend on the stochastic process, and we can represent the problem by an extended time space network with some uncertain parameters. Figure 4.2 shows the extended network for a problem instance with a single factory and a single store and where the shipping time is a single time period.

Even when the network structure is not affected by the random process, this problem is generally not equivalent to a standard network flow problem due to the stochastic elements. Decisions in later periods can incorporate more information

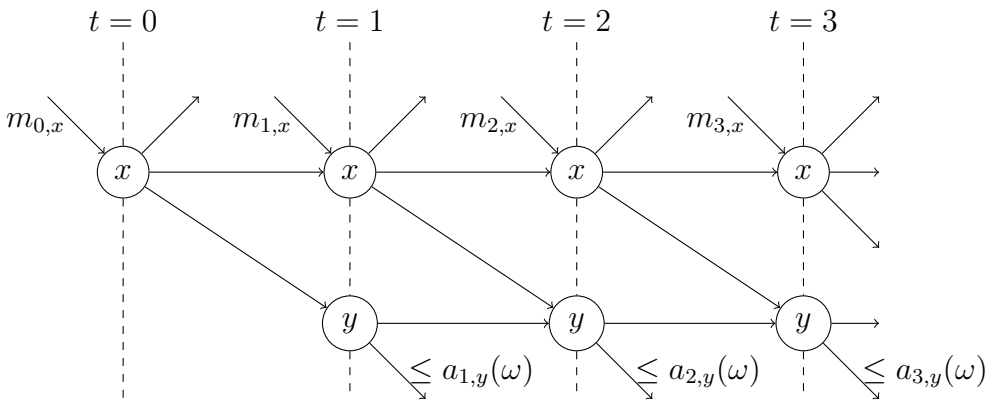


Figure 4.2: Production planning as a stochastic network

than those in earlier time periods. For example, suppose that the demands follow two scenarios ω_1 and ω_2 that are the same in time periods 0 and 1, and then become different in time period 2. Then, decisions made in time periods 0 and 1 are made in absence of any information, while those made in time period 2 are made in response to the revealed demand information. While the problem cannot be represented by a standard deterministic network, it can be represented as a deterministic hypergraph. This is shown in Figure 4.3. The hyperedges are colored in order to make the figure clearer.

This framework also allows uncertainty that would affect the structure of the network. For example, we could allow travel times to be uncertain. In this case, let $\tau(t, x, y; \omega)$ be the time it takes for resources shipped from factory x at time period t to arrive at time period y. The incoming edges that arrive at store y in time period t would now be defined as follows: for each factory x and for each time period $s < t$, the edge $e^{\text{ship}}_{s,x,y}$ reaches the store y in time period t if $s + \tau(t, x, y; \omega) = t$.

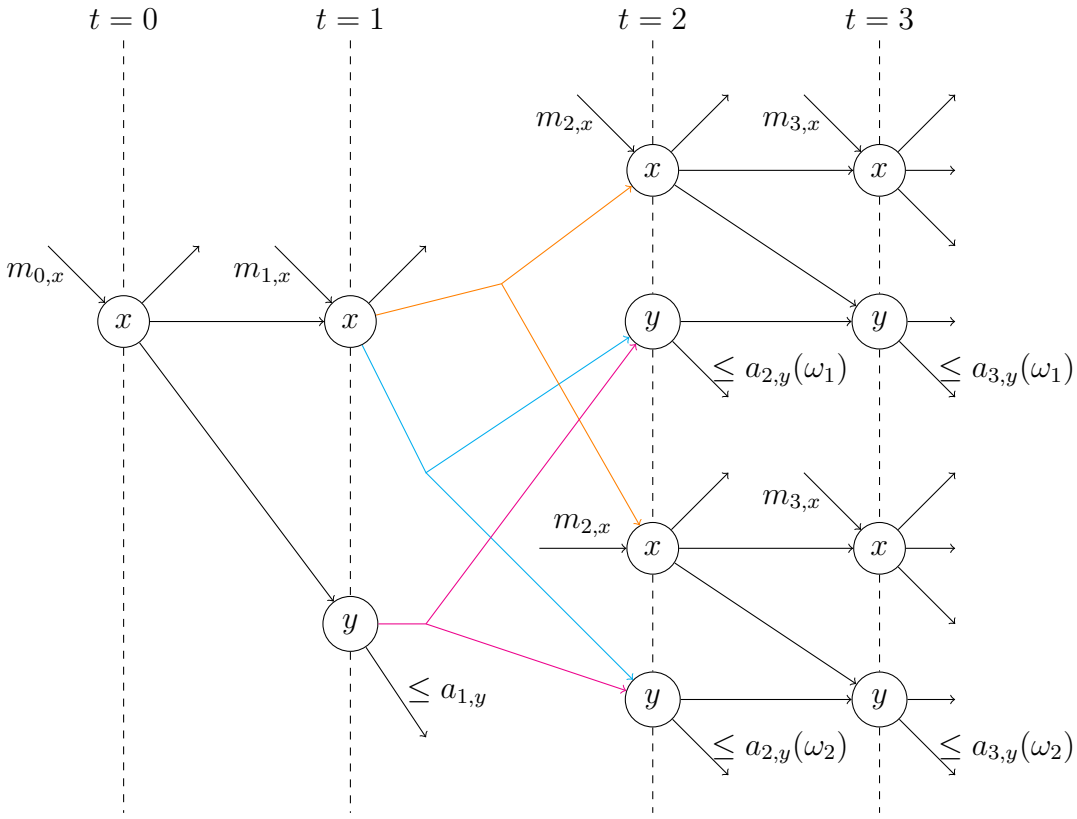


Figure 4.3: Production planning problem as a deterministic hypergraph

Another example of a stochastic network flow problem can be found in Chapter 5, where it is shown that an air traffic management problem can be modeled as a stochastic network flow problem.

4.3.3 Integer Programming Formulation

An optimal policy to a stochastic network flow problem is a solution to the stochastic program,

$$\min_{\boldsymbol{f}} \sum_{e \in E_0^+} c_e f_e + \mathbb{E}\left[Q_1(\mathbf{f}; \omega)\right]$$

subject to

$$\begin{aligned} \sum_{e \in E_0^+(v)} f_e &= q_{0,v} && \forall v \in V, \\ f_e &\leq a_e && \forall e \in \mathscr{C}_0, \\ f_e &\geq 0; \end{aligned}$$

where

$$Q_t(\mathbf{f}; \omega) := \min \sum_{e \in E_t^+(\omega)} c_e(\omega) f_e + \mathbb{E}\left[Q_{t+1}(\mathbf{f}; \omega)\right]$$

subject to

$$\begin{aligned} \sum_{e \in E_t^+(v;\omega)} f_e &= \kappa_{t,v}(\omega) + \sum_{e : E_t^-(v;\omega)} f_e && \forall v \in V, \\ f_e &\leq a_e(\omega) && \forall e \in \mathscr{C}_t(\omega), \\ f_e &\geq 0; \end{aligned}$$

and $Q_{T+1} \equiv 0$.

When there is a finite set of scenarios, any stochastic program can be rewritten as a single problem that selects the decisions that are made in all stages under all scenarios. This is called the ***deterministic equivalent***. The deterministic equivalent of this problem can be formulated as a flow problem on a hypergraph. We make the typical assumption that the set of scenarios Ω is arranged in a ***scenario tree***. A scenario is a rooted tree in which each node corresponds to a subset of the scenarios Ω, and the t^{th} level of the tree is a partition of Ω that corresponds to the time period t and refines the $(t-1)^{\text{th}}$ level. The interpretation is that if ω_1 and ω_2 fall in some node in the t^{th} level of the scenario tree, it is impossible to differentiate between these two scenarios using the information available at time period t. Thus, in order for a solution to be implementable, any decision made at time t must be the same under all scenarios that fall in the same node in the t^{th} level of the tree; we will refer to such nodes as "nodes at time t". In order to distinguish nodes of the scenario tree Ω from the nodes of the underlying flow problem, we will always refer to the former as ***scenario nodes***. We will use the notation $\mathscr{N}_t$ to refer to the set of scenario nodes at time t in scenario tree, and we will let $p(n)$ be the sum of the probabilities of the scenarios in the scenario node n. For more on scenario trees and deterministic equivalents, the reader may consult any standard reference on stochastic programming, for example [57].

We make the natural assumption that all quantities that are revealed at time t are the same across any scenarios that share the same scenario node at time t, since these scenarios should be indistinguishable at time t. Then, random quantities that

are revealed at time t can be described as a function of scenario nodes rather than scenarios. For example, we can use the notation $\kappa_{t,v}(n)$ to describe the quantity of resource produced at the node v at time t under the scenario node n, where n is a scenario node at time t. Similar notation will apply to other random quantities.

The deterministic equivalent is then given by:

$$\min_{\boldsymbol{f}} \sum_{t=0}^{T} \sum_{n \in \mathscr{N}_t} \sum_{e \in E_t^+} p(n) c_e(n) f_e(n) \tag{4.2}$$

subject to

$$\sum_{e \in E_t^+(v;n)} f_e(n) = \kappa_{t,v}(n) + \sum_{e : E_t^-(v;n)} f_e(n), \quad \forall v \in V, t \in \{0, \ldots, T\}, n \in \mathscr{N}_t, \tag{4.3}$$

$$f_e(n) \leq a_e(n) \qquad \forall t \in \{0, \ldots, T\}, e \in \mathscr{C}_t(n), n \in \mathscr{N}_t, \tag{4.4}$$

$$f_e(n) \geq 0 \qquad \forall t \in \{0, \ldots, T\}, e \in \mathscr{C}_t(n), n \in \mathscr{N}_t. \tag{4.5}$$

The constraints of this problem certainly resemble a network flow problem. Constraints (4.3) look like conservation of flow constraints, while constraints (4.4) and (4.5) look like capacity and non-negativity constraints, respectively. As we noted in Section 4.3.2, due to the presence of stochastic elements, the structure underlying this flow problem is not a standard graph. Consider an edge e that exits some node v under some scenario node n at time t. For simplicity, suppose that the scenario node n consists of two scenarios ω_1 and ω_2 that become distinguishable at time $t+1$. Suppose that e arrives at some node w in time period $t+1$ under scenario ω_1, but arrives at some node x in time period $t+1$ under scenario ω_2. Then the edge e will be treated as an incoming edge in two separate conservation of flow constraints, one corresponding to the node w under scenario ω_1 in time period $t+1$,

and one corresponding to the node x under scenario ω_2 in the time period $t+1$. This is instead a network flow problem on a class of directed hypergraphs, called F-graphs, for which edges are allowed to have multiple destination nodes. In fact, the underlying networks for instances of this problem exhibit additional structure, so we can restrict our attention to a special type of F-graph called an ABF-graph. This is defined in the subsequent sections, and the connection between ABF-graphs and this problem is described in more detail in Section 4.5.1.

4.4 Definition of F-Graph and the F-Graph Flow Problem

The minimum-cost network flow problem is defined on a directed graph. The extension that we consider is defined on a type of directed hypergraph, which is called an *F-graph*. See, for example, [52] for a survey of directed hypergraphs and their applications.

An ***F-graph*** H consists of a set of nodes $V(H)$ and a set of *F-edges* $E(H)$. An ***F-edge*** is an ordered pair (v, W) where $v \in V(H)$ and where W is a non-empty subset of $V(H)$. When there is no confusion, we will often refer to F-edges simply as edges. The node v is referred to as the *origin node* of e while the nodes W are referred to as the *destination nodes* of e. In this work, it is convenient to allow destination nodes that are not nodes of the hypergraph. Thus, an F-edge is more accurately an ordered pair (v, W) where $v \in V(H)$ and $W \subseteq \mathscr{W}$ for some set $\mathscr{W}$. We will often use the term edge or arc instead of F-edge. If none of the destination nodes of some edge e are in $V(H)$, then we may refer to the edge e as a ***delivery***

edge. The interpretation is that these edges represent delivery of resource to some customer, which removes the resource from the network. The ***forward star*** of a node v, denoted $E^+(v;H)$ is the set of edges $\{e \in E(H) : e = (v, W) \text{ for some } W\}$, and the ***backward star*** of a node v, denoted $E^-(v;H)$ is the set of edges $\{e \in E(H) : e = (w, W) \text{ such that } v \in W\}$. If there is a single incoming or outgoing edge of a node v, we will sometimes denote this edge by $e^-(v;H)$ or $e^+(v;H)$ respectively.

We define a minimum-cost flow problem on F-graphs that extends the minimum-cost network flow problem on directed graphs. The parameters are similar to the network flow problem. For each edge e, there is a corresponding per-unit cost c_e. Each node v has some corresponding quantity of resource q_v that is produced at the node. Edges in a set $\mathscr{C}$ have a specified capacity, and the flow across each edge e in $\mathscr{C}$ is not allowed to be greater than some value a_e. Edges not in $\mathscr{C}$ are referred to as uncapacitated edges. The ***minimum-cost F-graph flow*** problem, or ***F-graph flow*** problem for short, is defined by

$$\min \sum_{e \in E(H)} c_e f_e$$

subject to

$$\sum_{e \in E^+(v;H)} f_e = q_v + \sum_{e \in E^-(v;H)} f_e \qquad \forall v \in V(H), \tag{4.6}$$

$$f_e \leq a_e \qquad \forall e \in \mathscr{C}, \tag{4.7}$$

$$f_e \geq 0 \qquad \forall e \in E(H). \tag{4.8}$$

The only difference between the standard network flow problem and this problem is that edges are F-edges instead of standard directed edges. The constraints

(4.6),(4.7), and (4.8) will be referred to as conservation, capacity and non-negativity constraints respectively. For a given F-graph H, a vector of quantities $\mathbf{q}$, a set of capacitated edges $\mathscr{C}$, and a vector of edge capacities $\mathbf{a}$, we denote the corresponding polytope defined by the constraints of problem by $\mathscr{P}_1(H, \mathbf{q}, \mathscr{C}, \mathbf{a})$. A flow problem has all of these parameters with the addition of a vector of costs $\mathbf{c}$. We will denote such a flow problem by $\mathscr{F}(H, \mathbf{q}, \mathscr{C}, \mathbf{a}, \mathbf{c})$.

In the typical setup of the minimum-cost network flow problem there are no delivery edges, the sum of q_v across the nodes is zero, and the goal is to route resources from nodes with positive q_v to nodes with negative q_v. We will instead require that q_v is non-negative for each node and the goal is to route resources from the nodes with positive q_v to delivery arcs, which allow these resources to exit the network. These alterations do not cause any loss of generality. Consider a formulation $\mathscr{P}$ of the problem where q_v sums to zero across the nodes v and the network has no delivery edges. This can be translated into a formulation $\mathscr{P}'$ for the new setting as follows. For each node v, define a new quantity by $q'_v = q_v$ if q_v is positive and $q'_v = 0$ otherwise. For each node v such that q_v is negative, add a single outgoing delivery edge $d(v)$ from v with capacity $|q_v|$ and with cost $c_{d(v)} = 0$. Note that in any feasible solution, the amount of flow on this delivery edge must be q_v. Then, any feasible solution $\boldsymbol{\lambda}$ for $\mathscr{P}$ has a corresponding feasible solution $\boldsymbol{\lambda}'$ for $\mathscr{P}'$ that is formed by setting $f_{d(v)} = q_v$ on the aforementioned delivery edges and leaving all other variables unchanged. Conversely, any feasible solution $\boldsymbol{\lambda}'$ for $\mathscr{P}'$ has a corresponding feasible solution $\boldsymbol{\lambda}$ for $\mathscr{P}$ that is formed by omitting the variables $f_{d(v)}$ on these delivery edges. Thus, these formulations are equivalent.

For some graph H, a ***subgraph*** S of H is a graph such that $V(S) \subseteq V(H)$ and such that $E(S) \subseteq E(H)$. In a manner consistent with the definitions given above, it must be true that the origin node of every edge in $E(S)$ is contained in $V(S)$ but the destination nodes need not be contained in $V(S)$. If S is a subgraph of H and $E(S) \neq E(H)$, then we will say that S is a ***proper subgraph***. We use the notation $S \subseteq H$ to indicate that S is a subgraph of H, while $S \subset H$ denotes that S is a proper subgraph of H. For a set of edges $\mathcal{E} \in E(H)$, we say the induced subgraph $H[\mathcal{E}]$ is the graph such that

$$V(H[\mathcal{E}]) = \{v \in V(H) : v \text{ is an origin node for some edge } e \in \mathcal{E}\}$$

and

$$E(H[\mathcal{E}]) = \mathcal{E}.$$

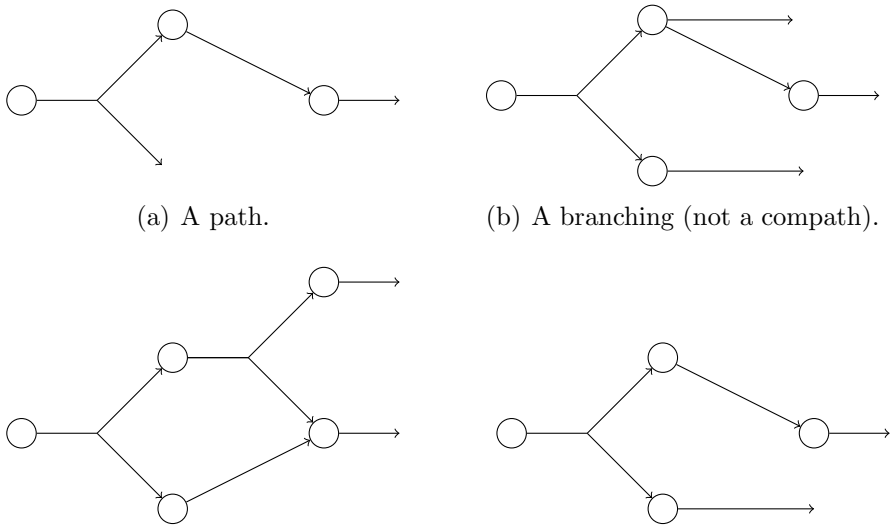
In some cases, we will discuss a flow problem on a subgraph S of some graph H in which all relevant parameters take the same value as a flow problem $\mathscr{F}(H, \mathbf{q}, \mathscr{C}, \mathbf{a}, \mathbf{c})$ on the entire graph. In this case, we denote the flow problem on the subgraph as $\mathscr{F}(S, \mathbf{q}, \mathscr{C}, \mathbf{a}, \mathbf{c})$ with the understanding that only the elements of $\mathbf{q}, \mathscr{C}, \mathbf{a}$, and $\mathbf{c}$ relevant to S are used. Similar notation applies to the polytope $\mathscr{P}$.

4.5 Paths, Compaths, Branchings and ABF-graphs

Two fundamental structures in F-graphs are *paths* and *compaths*. These extend the notion of a path in a network. A ***path*** P in a graph H is the induced subgraph for a set of edges $e_0 = (v_0, W_0), e_1 = (v_1, W_1)..., e_k = (v_k, W_k)$ such that each edge

e_j is in $E(H)$ and such that $v_{j+1} \in W_j$ for each j from 1 to k. A path ***starts at*** v if v is the origin node of the first edge of P and ***ends at*** w if w is a destination node of the last edge in P. For two nodes v and w, we say that the path P is a ***path from*** v ***to*** w if the path starts at v and ends at w. This definition is in accordance with existing sources, for example [52]. For a node v, a ***compath*** P ***starting at*** v is a subgraph of H such that v is the unique node of $V(P)$ with no incoming edge in $E(P)$ and every node of $V(P)$ has exactly one outgoing edge in $E(P)$. Note that every path starting at v is a compath starting at v. A compath (or path) P starting at v is ***maximal*** if there is no compath (or path) P' starting at v such that $P \subset P'$. The term compath was introduced in [12] for the stochastic network flow setting, where it was shown that the extreme points of a special type of stochastic network are compaths. The definition that we give here generalizes that notion. An ***edge selection*** is a set of edges $\mathcal{E}$ such that every vertex of $V(H)$ has exactly one outgoing edge in $\mathcal{E}$. For a node v, a ***branching rooted at*** v is a subgraph B such that every node of $V(B)$ except for v has exactly one incoming edge, while v has no incoming edge. This extends the notion of a branching (also called an arborescence or out-tree) in a directed graph. Existing works have studied the polyhedra of branchings in graphs and the problem of generating an optimal spanning branching; see for example [58] or [59]. Examples of paths, compaths, and branchings are given in Figure 4.4.

Every path starting at v is both a compath and a branching starting at v. A compath is not necessarily a branching, and vice-versa. An F-graph is ***branching on compaths*** if every compath is a branching. Consider a network with a single



(a) A path.

(b) A branching (not a compath).

(c) A compath (not a branching).

(d) A branching that is also a compath, but is not a path.

Figure 4.4: Paths, Compaths and Branchings

supply node and a single delivery edge, and suppose that there is exactly one unit of resource available at the supply node. A solution to an F-graph flow problem would route this unit of resource from the supply node through the network to the delivery edge. If the F-graph has a compath that is not a branching, then there may be solutions in which more than a single unit of resource is delivered. This is demonstrated in Figure 4.5. Within the context of stochastic network flow problems as described in Section 4.3, this behavior seems unusual. We would not expect that resources could be created simply by moving a single unit of resources through the network. Indeed, the F-graphs that arise from the aforementioned stochastic network flow problem are branching on compaths. In this case, hyperedges represent points in time at which two scenarios become distinct. These scenarios should not subsequently interact with each other.

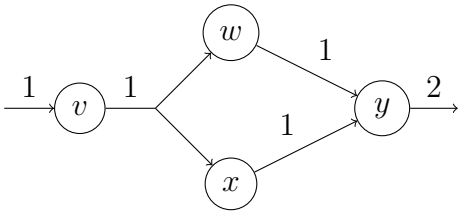


Figure 4.5: F-graphs that have non-branching compaths allow resources to be created.

We will say that an F-graph H is ***acyclic*** if for any nodes v and w such that there is a path from v to w then there is no path from w to v. This extends the notion of acyclicity in directed graphs. Stochastic network flow problems are generally acylic because nodes represent both a time and a location, and edges move forward in time. We will refer to acyclic branching-on-compath F-graphs as ABF-graphs. Note that in a DAG, a compath is simply a path, and every acyclic path is a branching. Thus, ABF-graphs are a generalization of DAGs. Similarly to DAGs, acyclic F-graphs have a ***natural partial ordering*** on their nodes and edges. This ordering is defined as follows:

- For v and w in $V(H)$, $v \prec^H w$ if there is a path from v to w.
- For v in $V(H)$ and $e = (v', W)$ in $E(H)$, $v \prec^H e$ if $v = v'$ or there is a path from v to v'; $e \prec^H v$ if there is a path from v' to v.
- For $e = (v_1, W_1)$ and $\eta = (v_2, W_2)$ in $E(H)$, $e \prec^H \eta$ if there exists some node $w \in W_1$ such that $w \preceq^H v_2$.

The definitions of a path and a compath can be expressed in terms of this partial ordering. A path P is a subgraph of H such that the nodes and edges of P are

totally ordered under the partial ordering $\prec^P$. A compath P starting at v is the induced subgraph for a set of edges $\mathscr{E}$ such that no two edges share the same origin node and such that v is the unique minimal node under the partial ordering $\prec^P$. A compath (or path) P is maximal if any maximal edge e under the partial ordering $\prec^P$ is also a maximal edge under the partial ordering $\prec^H$. It will be convenient to have compact notation for the subgraph induced by the elements preceding some node v in some F-graph H according to the natural partial ordering. We will let $L(v; H)$ be the subgraph of H given by:

$$V(L(v; H)) = \{w \in E(H) : w \preceq^H v\},$$

$$E(L(v; H)) = \{e \in E(H) : e \prec^H v\}.$$

In a similar fashion, we define $U(v; H)$ to be the subgraph where

$$V(U(v; H)) = \{w \in E(H) : w \succeq^H v\},$$

$$E(U(v; H)) = \{e \in E(H) : e \prec^H v\}.$$

For an edge e of H, the subgraphs $L(e; H)$ and $U(e; H)$ are defined similarly.

When an F-graph is acyclic, there is an alternative characterization of the branching-on-compaths property that is useful.

Lemma 4. *Let H be an acyclic F-graph. Then H is branching on compaths if and only if $\{U(w; H) : w \in W\}$ is a collection of disjoint sets for every edge $e = (v, W)$ in $E(H)$.*

Proof. First we prove the forward direction, that if H is branching on compaths then $\{U(w; H) : w \in W\}$ for every edge $e = (v, W)$ in $E(H)$. In fact, we prove the

contrapositive. Assume that there is some edge $e = (v, W)$ where $\{U(w, H) : w \in W\}$ is not a collection of disjoint sets. Then we can produce a compath that is not a branching. By the assumption, there exists some $\rho \in V(H)$ and some distinct $w_1, w_2 \in W$ such that $\rho \succeq^H w_1$ and $\rho \succeq^H w_2$. In particular, let ρ be a minimal node such that $\rho \succeq^H w_1$ and $\rho \succeq^H w_2$. Let P_1 be the path from w_1 to ρ and let P_2 be the path from w_2 to ρ. We claim that $P^* = e + P_1 + P_2$ is a compath. It is clear that v is the unique minimal node of P^*. Next, we need to show that each node has at most one outgoing edge. Suppose that some node $u \in P^*$ has two outgoing edges e_1 and e_2. Since the paths P_1 and P_2 start at nodes strictly greater than v, it is clear that $u \neq v$. Consequently, $e \neq e_1$ and $e \neq e_2$, so these edges must have come from P_1 and P_2. In any path each node has only one outgoing edge, so one of the two edges is in P_1 while the other edge is in P_2. This implies that $u \succeq^H w_1$ and $u \succeq^H w_2$, since those nodes are the first nodes in the respective paths. Since ρ is the final node of the path P_1, then $u \preceq^P \rho$. We defined ρ to be a minimal node such that $\rho \succeq^H w_1$ and $\rho \succeq^H w_2$, so in fact it must be true that $u = \rho$. However, ρ has no outgoing edges in the path P_1 or P_2. This is a contradiction. Thus, there is no node $u \in P^*$ that has two outgoing edges e_1 and e_2. Thus, P is a compath. Next, we show that ρ has two incoming edges in the compath P. Let η_1 be the incoming edge of ρ in P_1 and let η_2 be the incoming edge of ρ in P_2. Suppose that $\eta_1 = \eta_2$, and let y be the origin node of this edge. Then, it would be true that $y \succeq^H w_1$ and $y \succeq^H w_2$ since w_1 and w_2 are the first nodes in their respective paths. It would also be true that $y \prec^H \rho$ since there is a path from y to ρ consisting of the single edge η_1. However, this is a contradiction since by definition, ρ is a minimal node

such that $\rho \succeq^H w_1$ and $\rho \succeq^H w_2$. Thus, the node v has two incoming edges in the compath P. We have now constructed a compath P of H that is not a branching, which completes the proof of the forward direction.

In the opposite direction, suppose that H is not branching on compaths. Then there exists some compath P such that some node $u \in V(P)$ has two incoming edges $e_1 = (v_1, W_1)$ and $e_2 = (v_2, W_2)$ in P. Let v_0 be the starting node of compath P. By definition, there exists some paths P_1 and P_2 in P from v_0 to v_1 and v_2 respectively. Define $P_1' = P_1 + e_1$ and $P_2' = P_2 + e_2$, which are both paths in P from v_0 to u. Let η_1 be the minimum edge that is contained in path P_1' but not contained in path P_2'; such an edge must exist since e_2 is not in P_1'. The edge η_1 cannot be the first edge of P_1', since the node v_0 has only one outgoing edge in the compath P. Thus, the edge η_1 must have an edge $e' = (v', W')$ immediately preceding it. The path P_2' must have an edge following e', since all edges prior to and including e' are in P_1' but the edge e_2 is not in P_1'; call this edge η_2. Since no two edges in a compath share origin nodes, then η_1 and η_2 must have different origin nodes. Thus, there exist distinct nodes w_1 and w_2 of W' such that η_1 is an outgoing arc of w_1 and η_2 is an outgoing arc of w_2. Then, P_1' contains a path from w_1 to u and u is in $U(w_1; P)$. Similarly, u is in $U(w_2; P)$. Thus, $\{U(w; H) : w \in W'\}$ is not a collection of disjoint sets. $\square$

An important property of ABF-graphs is that certain subgraphs are DAGs with delivery edges. In particular, any ABF-graph with a unique maximal node is a DAG.

Lemma 5. *Let H be an ABF-graph with a unique maximal node ρ. Then for all*

edges $e = (v, W)$ *in* $E(H)$ *it is true that* $|W \cap V(H)| \leq 1$.

Proof. Suppose that there is some F-arc (v, W) in $E(H)$ where $|W \cap V(H)| \geq 2$. Let w and w' be distinct elements of $W \cap V(H)$. By definition, $w \preceq^H \rho$ and $w' \preceq^H \rho$. Thus, there exists paths P and P' from w and w' to ρ respectively. Let e' be the minimal edge of P' such that some destination node of e' is also a destination node of an edge of P. This edge must exist because ρ is a destination node for the final edge of P' and the final edge P. Let P^* be the path consisting of the edges of P' that are less than or equal to the edge e'. This construction ensures that in the subnetwork $P^* \cup P$, the node ρ' has two distinct incoming edges and there is no node that has two outgoing edges. This in turn ensures the subgraph $e + (P \cup P^*)$ is a compath, since the node v is a minimal node for this subgraph and it remains true that no node has two outgoing edges. However, the node ρ' has two incoming edges, so this compath is not a branching. This is a contradiction, so it must be true that each edge can have at most one destination node in V. □

As indicated below, this result implies that the constraint matrix of a flow problem on an ABF-graph is TU when that graph has a unique maximal node.

Lemma 6. *Let there be an ABF-graph with a unique maximal node* ρ*. The constraint matrix of the corresponding flow problem is TU when the problem is written*

in equality form,

$$\begin{aligned}
\sum_{e \in E^+(v)} f_e - \sum_{e \in E^-(v)} f_e &= q_v && \forall v \in V(H), \\
f_e + s_e &= a_e && \forall e \in \mathscr{C}, \\
f_e &\geq 0 && \forall e \in E(H), \\
s_e &\geq 0 && \forall e \in \mathscr{C}.
\end{aligned}$$

Proof. This follows almost immediately from Lemma 5 and the well-known property that the constraint matrix of the minimum-cost network flow problem on a directed graph is TU. There is a small detail that must be taken care of; ABF-graphs may have delivery edges while typical directed graphs do not. However, it is easy to see that the constraints of a flow problem on a DAG H with delivery edges are a subset of the constraints of a flow problem on a standard DAG G in which a node has been placed at the end of each delivery edge of H. Thus, the constraints will be TU. □

4.5.1 ABF-Graphs and Stochastic Network Flow

Recall the stochastic network flow problem described in Section 4.3, whose formulation is given by:

$$\min \sum_{t=0}^{T} \sum_{n \in \mathscr{N}_t} \sum_{e \in E_t^+} p(n) c_e(n) f_e(n)$$

subject to

$$\sum_{e \in E_t^+(v;n)} f_e(n) = \kappa_{t,v}(n) + \sum_{e:E_t^-(v;n)} f_e(n) \qquad \forall v \in V, t \in \{0, ..., T\}, n \in \mathscr{N}_t,$$

$$f_e(n) \leq a_e(n) \qquad \forall t \in \{0, ..., T\}, e \in \mathscr{C}_t(n), n \in \mathscr{N}_t,$$

$$f_e(n) \geq 0 \qquad \forall t \in \{0, ..., T\}, e \in \mathscr{C}_t(n), n \in \mathscr{N}_t.$$

This is a network flow problem on an ABF-graph that is defined as follows. Let

$$V(H) = \{(t, n, v) : t \in \{1, ..., T\}, n \in \mathscr{N}_t, v \in V\}.$$

The interpretation of node (t, n, v) of $V(H)$ is that it represents the node v of V at time t under the scenarios in the scenario node n. For each edge e of $E_t^+(v; \omega)$, there is a corresponding F-edge η in $E(H)$. The origin node of η is $(t, n_t(\omega), v)$ where $n_t(\omega)$ is the scenario node at time t containing ω. The destination nodes of η are the set:

$$\{(\tau, n_\tau(\omega), w) : e \in E_\tau^-(w; n_\tau(\omega))\}.$$

The F-edge η represents the uncertainty in the network edge e; the multiple destination nodes represent the possible destinations of the edge e under the various scenarios. The cost c_η of each F-edge η is simply the cost c_e of the corresponding network edge e. Similarly, η is in $\mathscr{C}$ if and only if e is in $\mathscr{C}_t$, and the capacity a_η is set equal to a_e. It is straightforward to verify that this graph is an ABF-graph.

Since graphs resulting from stochastic network flow problems are acyclic and branching on compaths, for the most part we will restrict our study to this class of graphs.

4.6 Constructing Feasible Solutions

We present several lemmas that assist in the construction and modification of feasible solutions. These lemmas will be useful in the proof of subsequent results. Let $\boldsymbol{\lambda}$ be a solution to a F-graph flow problem on some F-graph H. We will let $f_e(\boldsymbol{\lambda})$ be the value of the variable f_e in the solution $\boldsymbol{\lambda}$ for each edge e in $E(H)$. We will use the notation

$$F^+(\boldsymbol{\lambda}) := \{e \in E(H) : f_e(\boldsymbol{\lambda}) > 0\},$$

the set of edges that take positive flow in the solution $\boldsymbol{\lambda}$. Similarly, we define

$$F^0(\boldsymbol{\lambda}) := \{e \in E(H) : f_e(\boldsymbol{\lambda}) = 0\},$$

the set of edges that take zero flow in the solution $\boldsymbol{\lambda}$. For a set of edges $\mathcal{E}$ we let $\boldsymbol{\ell}(\mathcal{E})$ be the solution (not usually feasible) defined by

$$f_e(\boldsymbol{\ell}(\mathcal{E})) = \begin{cases} 1 & e \in \mathcal{E}, \\ 0 & \text{otherwise.} \end{cases}$$

For a subgraph S, we use the shorthand $\boldsymbol{\ell}(S) = \boldsymbol{\ell}(E(S))$.

Lemma 7 states that the subgraph induced by the edges greater than some node v in an edge selection X is always a maximal compath.

Lemma 7. *Let X be an edge selection of an ABF-hypergraph H. Then $U(v; X)$ is a maximal compath.*

Proof. The subnetwork $U(v; X)$ is clearly a compath since v is the unique minimal node of $U(v; X)$, and every node has exactly one outgoing edge in X. We can also

show that it is maximal. Let $e^* = (x, W)$ be a maximal edge of $E(U(v; X))$ in X and suppose that e^* is not a maximal edge in H. In this case, there exists some edge $\tilde{e} = (x', W')$ in $E(H)$ such that there is a path P from some node w in W to x'. However, since $x' \in V(H)$ and X is an edge selection, then there exists some edge e' in $E(X)$ whose origin node is w'. Then, $e' \succ^X e^*$, which is a contradiction because e^* is a maximal edge in X. Thus, $U(v; X)$ is a maximal compath. □

Lemma 8 allows the creation of larger compaths from smaller compaths.

Lemma 8. *Let $e = (v, W)$ be any edge of an ABF-graph H. For each $w \in V(H) \cap W$, let P_w be a compath beginning at w. Then the subnetwork $P^* = e + \sum_{w \in V(H) \cap W} P_w$ is a compath. Furthermore, P^* is maximal if and only if each P_w is maximal.*

Proof. It is easy to see that v is the unique minimal node of P^*. It follows from Lemma 4 that the paths P_w for $w \in V \cap W$ must be vertex-disjoint, which in turn implies that each node of P^* has at most one outgoing edge. It is also clear that an edge is a maximal edge of P^* if and only if it is a maximal edge of P_w for some $w \in V(H) \cap W$. This immediately implies that P^* is maximal if and only if each P_w is maximal. □

Lemma 9 states that there is always a compath contained in the edges that take positive flow in a feasible solution to a flow problem.

Lemma 9. *Let $\boldsymbol{\phi}$ be any feasible solution to an F-graph flow problem on some ABF-graph H, and let $e = (v, W)$ be an edge of $F^+(\boldsymbol{\phi})$. Then there exists a maximal compath P in $F^+(\boldsymbol{\phi})$ starting at v whose first edge is e.*

Proof. Induction. This is clearly true for any maximal edge e, because the edge itself is a maximal compath. Suppose that there exists some edge $e = (v, W)$ in $F^+(\boldsymbol{\phi})$ such that the property is true for all edges $e' \succ e$; we will show that the property is also true for e. If $W \cap V(H)$ is empty, then the edge is maximal and the property is again satisfied. Suppose $W \cap V(H)$ is not empty. Since there is positive flow on the edge e, then for each vertex $w \in W \cap V(H)$ there must exist some edge η_w outgoing from w such that η_w has positive flow; otherwise, the conservation constraint would not be satisfied at w. By the induction hypothesis, there is a maximal compath P_w starting at w. Then, by Lemma 8, $e + \sum_{w \in V(H) \cap W} P_w$ is a maximal compath. □

Lemma 9 states that adding a unit of flow along compath and while removing a unit of flow along a compath that starts at the same node preserves feasibility, as long as edge capacities are not violated.

Lemma 10. *Let $\boldsymbol{\phi}$ be an integral feasible solution to an F-graph flow problem on some ABF-graph H. Let P' be any maximal compath in $F^+(\boldsymbol{\phi})$ starting at a node v of $V(H)$. Let P be any maximal compath starting at the same node v such that*

$$f_e(\boldsymbol{\phi}) \leq a_e - 1$$

for any edge e in $(E(P) \cap \mathscr{C}) \setminus E(P')$. Then $\boldsymbol{\phi} + \boldsymbol{\ell}(P) - \boldsymbol{\ell}(P')$ is an integral feasible solution. Furthermore, $E(\boldsymbol{\ell}(P)) \subseteq F^+(\boldsymbol{\phi} + \boldsymbol{\ell}(P) - \boldsymbol{\ell}(P'))$.

Proof. Note that the variables in $\boldsymbol{\ell}(P)$ and $\boldsymbol{\ell}(P')$ only appear in the conservation constraints for nodes that fall in the paths P and P', the edge capacity constraints for edges in these paths, and the non-negativity constraints for edges in these paths.

Since P' is a compath within the edges of $\boldsymbol{\phi}$ that have positive flow and the solution is integral, then all variables of $\boldsymbol{\phi} - \boldsymbol{\ell}(P')$ take positive values, which in turn implies that all variables of $\boldsymbol{\phi} + \boldsymbol{\ell}(P) - \boldsymbol{\ell}(P')$ take positive values.

The node conservation constraint at v will still be satisfied because the addition of $\boldsymbol{\ell}(P)$ increases the outgoing flow by one, while the subtraction of $\boldsymbol{\ell}(P')$ decreases this flow by one, and neither of the paths alter the incoming flow. For any node w other than v, note that since P' and P are branchings and maximal, the node w has exactly one outgoing and one incoming arc, so the addition of $\boldsymbol{\ell}(P)$ and the subtraction of $\boldsymbol{\ell}(P')$ do not violate the conservation constraint.

Observe that

$$f_e(\boldsymbol{\phi} + \boldsymbol{\ell}(P) - \boldsymbol{\ell}(P')) = \begin{cases} f_e(\boldsymbol{\phi}) + 1 & e \in E(P) \setminus E(P'), \\ f_e(\boldsymbol{\phi}) & e \in E(P') \cap E(P), \\ f_e(\boldsymbol{\phi}) - 1 & e \in E(P') \setminus E(P). \end{cases}$$

From this and the assumptions of the lemma it is apparent that the edge capacity constraints are preserved. It follows from this same observation that $E(\boldsymbol{\ell}(P)) \subseteq F^+(\boldsymbol{\phi} + \boldsymbol{\ell}(P) - \boldsymbol{\ell}(P'))$. Let e be some edge of the path P. From the observation, it is immediate that e receives positive flow in $\boldsymbol{\phi} + \boldsymbol{\ell}(P) - \boldsymbol{\ell}(P')$ if e is not in P'. If e is in P', then

$$f_e(\boldsymbol{\phi} + \boldsymbol{\ell}(P) - \boldsymbol{\ell}(P')) = f_e(\boldsymbol{\phi})$$

and, by the assumptions of the lemma, $f_e(\boldsymbol{\phi}) \geq 1$. Thus, in either case, the edge e receives positive flow in $\boldsymbol{\phi} + \boldsymbol{\ell}(P) - \boldsymbol{\ell}(P')$. □

Given a path P and an edge selection X, it is possible to extend the path into a maximal compath using only edges from the edge selection.

Lemma 11. *Let P be a compath in an ABF-graph H starting at some node v in $V(H)$, and let X be an edge selection of H. There exists a maximal compath P' of H starting at v such that $E(P) \subseteq E(P') \subseteq E(P) \cup X$.*

Proof. We prove this by induction on the number of edges in P. If P consists of a single edge $e = (v, W)$, then it follows from Lemmas 7 and 8 that

$$P' = e + \sum_{w \in W} U(w; X)$$

is a maximal compath. Suppose that this theorem holds for any compath with k edges, and let P be a compath with $k+1$ edges. Let $e = (v, W)$ be the outgoing edge of v in P. Consider $U(w; P)$ for some w in $W \cap V(H)$. If w has an outgoing edge in P then $U(w; P)$ is a compath, and this compath cannot include the edge e, so it has at most k edges. By induction, there exist a maximal compath P_w starting at w such that $U(w; P) \subseteq P_w$ and $E(P_w) \subseteq E(U(w; P)) \cup X$. If w has no outgoing edge in P, then let $P_w = U(w; X)$, which is a maximal compath by Lemma 7. Then, we claim that

$$P' = e + \sum_{w \in W \cap V(H)} P_w$$

is a maximal compath that satisfies the desired properties. Lemma 8 guarantees that this is a maximal compath starting at v.

By definition of each compath P_w, $E(P_w) \subseteq E(U(w; P)) \cup X$, so it must be

true that

$$E(P') \subseteq e + \left(\bigcup_{w \in W} E(U(w; P)) \right) \cup X.$$

From this, any edge η in P' is either the edge e, is in the subgraph $U(w; P)$ for some $w \in W \cap V(H)$, or is an edge of X. Thus, any edge of P' must be an edge of $E(P) \cup X$.

Let η be an edge of P. If $\eta = e$, then it is clear that η is an edge of P'. If $\eta ! = e$, then it must be true that $\eta \succ^P w$ for some destination node w of e. This implies that w has an outgoing edge in P and that η is in $U(w; P)$. By definition, $P_w = U(w; P)$ when w has an outgoing edge in P, so η is in P_w. Thus, $E(P) \subseteq E(P')$. □

4.7 Semi-Uncapacitatedness and NP-Completeness

In subsequent results, we will examine ABF-graph flow problems in which every node has an outgoing uncapacitated edge. We will refer to such flow problems as ***semi-uncapacitated***. In this case, we can find an edge selection in which every edge is uncapacitated. This is a useful property because it ensures that feasible solutions for flow problems on subgraphs can always be extended into feasible solutions for the flow problem on the entire graph.

Lemma 12. *Let $\boldsymbol{q}, \mathscr{C}$, and $\boldsymbol{a}$ be parameters defining the polytope $\mathscr{P}_1(H, \boldsymbol{q}, \mathscr{C}, \boldsymbol{a})$ of a flow problem on H, and suppose that there exists an edge selection X that consists of edges in $E(H) \setminus \mathscr{C}$. Let $\boldsymbol{\phi}$ be an (integral) feasible solution in $P_1(S, \boldsymbol{q}, \mathscr{C}, \boldsymbol{a})$ for some subnetwork S of H. Then there exists an (integral) feasible solution $\boldsymbol{\phi}'$ in*

$\mathscr{P}_1(H, \boldsymbol{q}, \mathscr{C}, \boldsymbol{a})$ such that $f_e(\boldsymbol{\phi}) = f_e(\boldsymbol{\phi}')$ for all edges $e \in E(S) \setminus X$ and such that $f_e(\boldsymbol{\phi}') = 0$ for all edges $e \in E(H) \setminus (X \cup E(S))$.

Proof. For each node $v \in V(H)$, let

$$R_v = q_v + \sum_{e \in E^-(v,S)} f_e(\boldsymbol{\phi}) - \sum_{e \in E^+(v,S)} f_e(\boldsymbol{\phi}).$$

For any node $v \in V(S)$, $R_v = 0$ from the conservation constraints in $P_1(S, \boldsymbol{q}, \mathscr{C}, \mathbf{a})$. Note that no node v from $V(H) \setminus V(S)$ can have outgoing arcs in S. Thus, $R_v \geq 0$ for all v. We claim that the solution

$$\boldsymbol{\phi}' := \boldsymbol{\phi} + \sum_{v \in V(H)} R_v \boldsymbol{\ell}(U(v; X))$$

has the desired properties. Since $R_v = 0$ for any node v in $V(S)$ then $f_e(\boldsymbol{\phi}) = f_e(\boldsymbol{\phi}')$ for all edges $e \in E(S) \setminus X$. It is also easy to see that any edge capacity constraints will be satisfied by $\boldsymbol{\phi}'$ since it takes the same values as $\boldsymbol{\phi}$ in all capacitated edges.

Note that for any edge $e \in X$,

$$f_e(\boldsymbol{\phi}') := f_e(\boldsymbol{\phi}) + \sum_{v \in V(L(e;X))} R_v.$$

Consider some node $v \in V(H)$;

$$\begin{aligned} q_v + \sum_{e \in E^-(v,H)} f_e(\boldsymbol{\phi}') - \sum_{e \in E^+(v,H)} f_e(\boldsymbol{\phi}') & \\ = q_v + \sum_{e \in E^-(v,S)} f_e(\boldsymbol{\phi}) - \sum_{e \in E^+(v,S)} f_e(\boldsymbol{\phi}) & \\ + \sum_{e \in E^-(v,X)} \sum_{w \in V(L(e;X))} R_w - \sum_{\substack{w \in V(H): \\ w \prec^X e^+(v;X)}} R_w. & \end{aligned}$$

We claim that the collection $\{V(L(e; X)); e \in E^-(v; X)\}$ is a partition of $V(L(v; X)) \setminus \{v\}$. It is straightforward to verify that the union of this collection is equal to

$V(L(v;X)) \setminus \{v\}$. Suppose that these sets are not disjoint, so that there is some node w in $V(L(\eta_1;X))$ and in $V(L(\eta_2;X))$ for distinct edges η_1 and η_2 in $E^-(v;X)$. By Lemma 7, $U(w;X)$ is a compath. Since H is an ABF-graph, this compath is a branching, which implies that v has at most one incoming edge in $U(w;X)$. However, both η_1 and η_2 are in $U(w;X)$, and both are incoming edges of v. This is a contradiction. This shows that $\{V(L(e;X)); e \in E^-(v;X)\}$ is a partition of $V(L(v;X)) \setminus \{v\}$. Using this fact, the right-hand-side can be rewritten as

$$\begin{aligned}
\text{RHS} &= q_v + \sum_{e \in E^-(v,S)} f_e(\boldsymbol{\phi}) - \sum_{e \in E^+(v,S)} f_e(\boldsymbol{\phi}) + \sum_{\substack{w \in V(H): \\ w \prec^X v}} R_w - \sum_{\substack{w \in V(H): \\ w \preceq^X v}} R_w \\
&= q_v + \sum_{e \in E^-(v,S)} f_e(\boldsymbol{\phi}) - \sum_{e \in E^+(v,S)} f_e(\boldsymbol{\phi}) - R_v \\
&= 0.
\end{aligned}$$

This demonstrates that $\boldsymbol{\phi}'$ is feasible. Further note that if $\boldsymbol{\phi}$ is integral, then each R_v is an integer. Thus, $\boldsymbol{\phi}'$ will also be integral. This completes the proof. □

When all edges are uncapacitated, then F-graph flow problems fall in a class of problems called Leontief substition problems, which are known to have integral optimal solutions that can be obtained by polynomial-time algorithms [55]. Semi-uncapacitated problems are unlikely to have such efficient solutions. In fact, the decision problem corresponding to a semi-uncapacitated ABF-hypergraph flow problem is NP-complete.

Theorem 6. *The problem of whether there exists an integral solution to the semi-uncapacitated ABF-graph flow problem with objective k is NP-complete.*

Proof. The semi-uncapacitated ABF-graph flow problem is in NP since it is a special case of an integer programming problem, which is known to be in this class.

We prove that this problem is NP-complete by reduction from set packing. Let there be a finite universe $\mathscr{U}$ of items and let $\mathscr{S}$ be a collection of subsets of $\mathscr{U}$. Define a corresponding F-graph problem $\Gamma(\mathscr{U}, \mathscr{S}) = \mathscr{F}(H, \mathbf{q}, \mathscr{C}, \mathbf{a}, \mathbf{c})$ as follows. The graph has one node for each element of $\mathscr{S}$ and one node for each item of $\mathscr{U}$. The edges are defined so that every node has two edges, one of which is an uncapacitated delivery edge with cost 0. For each node corresponding to some element S of $\mathscr{S}$, the other outgoing edge has destination nodes $\{u \in \mathscr{U} : u \in S\}$. In other words, this edge connects the node corresponding to the set S with the nodes corresponding to the elements in S. The cost of this edge is $|S| - 1$ and its capacity is 1. For nodes corresponding to items in $\mathscr{U}$, the second outgoing edge is a delivery edge with capacity 1 and cost -1. Every node corresponding to an element of $\mathscr{S}$ has a demand quantity of 1, while those nodes corresponding to elements in $\mathscr{U}$ have a demand quantity of 0. It can be easily seen that the size of $\Gamma(\mathscr{U}, \mathscr{S})$ is bounded by a polynomial of the size of $(\mathscr{U}, \mathscr{S})$.

An example of this construction is provided in Figure 4.6. This figure shows the corresponding F-graph flow problem for a set packing problem that is defined as follows. The universe of items $\mathscr{U}$ is the set $\{a, b, c, d\}$. The collection of subsets $\mathscr{S}$ has three elements: $S_1 = \{a, b, c\}$, $S_2 = \{a, b\}$ and $S_3 = \{c, d\}$. The value shown in parenthesis for each node is the corresponding quantity of resource present at that node. The value on each edge outside of the parentheses is the cost of the edge, while the value within the parentheses is the edge capacity if such a value is present.

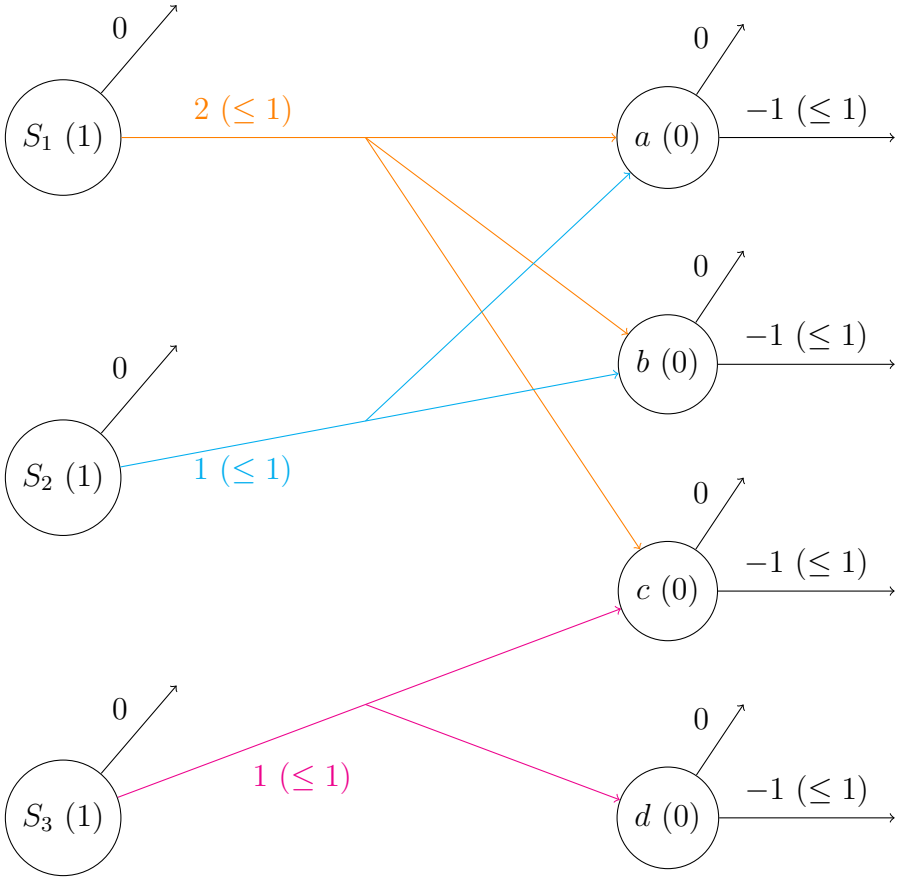


Figure 4.6: Semi-uncapacitated F-graph flow problem constructed from set packing problem, $\mathscr{U} = \{a, b, c, d\}$, $\mathscr{S} = \{\{a, b, c\}, \{a, b\}, \{c, d\}\}$.

.

We claim that a set $(\mathscr{U}, \mathscr{S})$ has a set packing of size at least k if and only if the flow problem $\Gamma(\mathscr{U}, \mathscr{S})$ has an integral feasible solution with objective value of at most $-k$. In the forward direction, let A be a collection of k disjoint elements in $\mathscr{S}$. Then define a corresponding solution to the flow problem as follows. Let H be the underlying hypergraph of the flow problem $\Gamma(\mathscr{U}, \mathscr{S})$. Consider a node S corresponding to an element in $\mathscr{S}$. If S is in the set packing A, then assign a unit of flow to the capacitated outgoing edge and no flow to the uncapacitated delivery edge. Conversely, if S is not in the set packing then assign a unit of flow on the uncapacitated delivery edge and none to the outgoing capacitated edge. Next, consider a node u corresponding to an item in $\mathscr{U}$. If u is covered by the set packing A (i.e. there exists some $B \in A$ such that $u \in B$), then assign a unit on flow on the capacitated outgoing delivery edge. Otherwise, assign no flow on either of these edges. It is straightforward to verify that this solution is feasible and that the objective value of this solution is $-k$. This completes the proof of the forwards direction.

In the backwards direction, let $\boldsymbol{\lambda}$ be an integral solution to $\Gamma(\mathscr{U}, \mathscr{S})$ with objective value of $-k$. We claim that there exists a solution $\boldsymbol{\lambda}'$ that achieves an objective value no smaller than $-k$ and such that no two edges of $F^+(\boldsymbol{\lambda}')$ share a destination node. Suppose that $\boldsymbol{\lambda}$ has two edges $e = (S, W)$ and e' with positive flow such that e and e' share some destination node u. Then there are at least two units of flow entering the node u, which implies that at least one unit of flow must be assigned to the uncapacitated arc e^* that is outgoing from u. Each of the other nodes of W must also have at least one outgoing delivery edge with positive flow in

λ, since these nodes receive a unit of flow along the edge e. Thus, we can choose a set of edges X that contains exactly one edge with positive flow for each node of $W - u$. Note that $e + X$ is a maximal compath starting at S. Let η be the uncapacitated delivery edge that is outgoing from S, which by itself is a maximal compath starting at v. Then, we can define a new solution $\boldsymbol{\lambda} - \boldsymbol{\ell}(e + e^* + X) + \boldsymbol{\ell}(\eta)$ that is feasible by Lemma 10. Removing the unit of flow on the edge e decreases the objective value by $|S| - 1$; increasing the flow on η has no affect on the objective value; decreasing the flow on e^* also has no affect on the objective value; there are $|S| - 1$ edges in X and the cost of each of these edges is at least -1, so removing a unit of flow on each of these edges cannot increase the objective value by more than $|S| - 1$. Thus, the objective value of $\boldsymbol{\lambda} - \boldsymbol{\ell}(e + e^* + X) + \boldsymbol{\ell}(\eta)$ cannot be higher than the objective value of $\boldsymbol{\lambda}$. In the new solution, the number of pairs of edges that share a destination node has decreased. Thus, by repeating this procedure, we can procure a solution $\boldsymbol{\lambda}'$ in which no pairs of edges share a destination node and with objective value no smaller than $-k$.

Next, we claim that in any integral solution $\boldsymbol{\lambda}$ such that there are k nodes corresponding to elements of $\mathscr{S}$ whose outgoing capacitated edge is assigned a unit of flow, then the objective value is no smaller than $-k$. We can prove this by induction on k. The case when $k = 0$ is trivial. Suppose that this is true whenever $k < l$. For some $l \geq 1$, suppose there were an integral solution $\boldsymbol{\lambda}$ such that l of the nodes corresponding to elements of $\mathscr{S}$ have an outgoing capacitated edge that is assigned a unit of flow in $\boldsymbol{\lambda}$. Let $e = (S, W)$ be one such edge. Similarly to previous arguments, there must exist a set of edges X in $F^+(\boldsymbol{\lambda})$ such that each node of W

has exactly one outgoing edge in X. The edges $e + X$ induce a maximal compath, and if we let η be the delivery edge out of S, then we can define a new feasible solution

$$\boldsymbol{\theta} := \boldsymbol{\lambda} - \boldsymbol{\ell}(e + X) + \boldsymbol{\ell}(\eta).$$

Feasibility of $\boldsymbol{\theta}$ is guaranteed by Lemma 10. Removing the unit of flow on the edge e decreases the objective value by $|S| - 1$; increasing the flow on η has no affect on the objective value; there are $|S|$ edges in X and the cost of each of these edges is at least -1, so removing a unit of flow on each of these edges cannot increase the objective value by more than $|S|$. Thus, the objective value of $\boldsymbol{\theta}$ is at most 1 greater than the objective value of $\boldsymbol{\lambda}$. Furthermore, $F^+(\boldsymbol{\theta})$ has only $l - 1$ of the capacitated edges whose origin nodes correspond to elements of $\mathscr{S}$. Applying the induction hypothesis, the objective value of $\boldsymbol{\theta}$ is no less than $l - 1$. This in turn implies that the objective value of $\boldsymbol{\lambda}$ is no less than l.

Given the above claims, we are ready to prove that if $\Gamma(\mathscr{U}, \mathscr{S})$ has an integral solution $\boldsymbol{\lambda}$ with objective value of at most $-k$ then there is a set packing A of size at least k. Let $\boldsymbol{\lambda}$ be such a solution. By an earlier claim, there exists a solution $\boldsymbol{\lambda'}$ that also achieves an objective value of at most $-k$ such that no two edges of $F^+(\boldsymbol{\lambda}')$ share a destination node. By another earlier claim, in the solution $\boldsymbol{\lambda'}$, a unit of flow must be assigned to at least k of the capacitated edges whose origin nodes correspond to elements of $\mathscr{S}$. Then, the elements of $\mathscr{S}$ corresponding to these k origin nodes form a set packing A. These elements must be disjoint since the corresponding outgoing capacitated edges do not share a destination node. This

completes the proof of the backwards direction. □

4.8 Full-Dimensional Formulation with Facet-Defining Inequalities

In this section, we first provide a full-dimensional projection of the polytope for the semi-uncapacitated F-graph flow problem. Then, we show that all of the constraints of the F-graph flow polytope are facet-defining and that each constraint has a corresponding constraint in the full-dimensional projection that is also facet-defining. We do make some assumptions to avoid degenerate cases. Specifically we assume that:

1. The capacity a_e for each capacitated edge e in $\mathscr{C}$ is strictly positive.

2. For every edge $e = (w, V)$ such that $q_w = 0$, there is a path from some node v to w such that the supply q_v is strictly positive.

Assumption 1 can be made because if $a_e = 0$ for some edge e then no flow can be assigned to that edge, and there is an equivalent flow problem in which the edge is removed. Assumption 2 can also be made because if there is no path from any node with positive demand to an edge e, then there can be no flow along the incoming edges of e. Thus, if there is no supply available at the origin node, there can be no flow assigned to the edge e and there is an equivalent flow problem in which the edge is removed. We will refer to these assumptions as the ***non-degeneracy*** assumptions.

4.8.1 Full-Dimensional Formulation

Consider an ABF-graph H, a set of capacitated edges $\mathscr{C}$, and an edge selection X of H such that none of the edges of X are capacitated. Then, given vectors of supplies $\mathbf{q}$ and capacities $\mathbf{a}$, we can define a polytope:

$\mathscr{P}_2(H, \mathbf{q}, \mathscr{C}, \mathbf{a}, X)$:

$$f_{e^+(v,X)} = \sum_{w:w\preceq^X v} \left(q_w + \sum_{e\in E^-(w;H)\setminus X} f_e - \sum_{e\in E^+(w;H)\setminus X} f_e \right) \qquad \forall v \in V(H), \tag{4.9}$$

$$f_e \leq a_e \qquad \forall e \in \mathscr{C}, \tag{4.10}$$

$$f_e \geq 0 \qquad \forall e \in E(H). \tag{4.11}$$

The polytope $\mathscr{P}_2$ was derived by manipulating the equality constraints to isolate the flow variables corresponding to edges in X. For this reason, this polytope is exactly equal to the polytope $\mathscr{P}_1(H, \mathbf{q}, \mathscr{C}, \mathbf{a})$. Similarly to $\mathscr{P}_1$, we refer to the constraints 4.9, 4.10, and 4.11 as conservation, capacity and non-negativity constraints respectively.

Lemma 13. *Let there be an ABF-graph H, a set of edges $\mathscr{C}$ in $E(H)$, and an edge selection X of H such that X is disjoint from $\mathscr{C}$. For any vector of supplies $\boldsymbol{q}$ and capacities $\boldsymbol{a}$, $\mathscr{P}_1(H, \boldsymbol{q}, \mathscr{C}, \boldsymbol{a}) = \mathscr{P}_2(H, \boldsymbol{q}, \mathscr{C}, \boldsymbol{a}, X)$.*

Proof. Define the *pseudo-rank* $r(v; X)$ of a node v to be the number of edges in the shortest path from any minimal node to v under the partial ordering $\prec^X$, with the convention that $r(v; X) = 0$ if v is a minimal node. Let $V^n(H; X)$ be the nodes that

have pseudo-rank of at most n in the poset on $V(H)$ with ordering $\prec^X$. We define $\mathscr{P}_1^n(H, \mathbf{q}, \mathscr{C}, \mathbf{a})$ to be the polytope in $\mathbb{R}^{|E|}$ formed by the conservation constraints for the nodes of $V^n(H; X)$ along with the capacity and non-negativity constraints on the outgoing edges on these nodes. That is,

$\mathscr{P}_1^n(H, \mathbf{q}, \mathscr{C}, \mathbf{a}) := \boldsymbol{f} \in \mathbb{R}^{|E|}$ such that

$$\begin{aligned}
\sum_{e \in E^+(v;H)} f_e = q_v + \sum_{e \in E^-(v;H)} f_e \quad & \forall v \in V^n(H; X), \\
f_e \leq a_e \quad & \forall e \in \mathscr{C} \text{ s.t. the origin of } e \text{ is in } V^n(H; X), \\
f_e \geq 0 \quad & \forall e \in E(H) \text{ s.t. the origin of } e \text{ is in } V^n(H; X).
\end{aligned}$$

We define $\mathscr{P}_2^n(H, \mathbf{q}, \mathscr{C}, \mathbf{a}, X)$ similarly. Since the parameters of these polytopes will remain constant throughout this proof, we will refer to these polytopes simply as $\mathscr{P}_1^n$ and $\mathscr{P}_2^n$.

We prove this property using induction on n. The nodes with pseudo-rank 0 are the minimal nodes of the poset. Note that the capacity and non-negative constraints are always the same in $\mathscr{P}_1$ and $\mathscr{P}_2$ and that for each minimal node v the conservation constraint is the same in $\mathscr{P}_1$ as it is in $\mathscr{P}_2$. Thus, $\mathscr{P}_1^0 = \mathscr{P}_2^0$. Suppose that $\mathscr{P}_1^{n-1} = \mathscr{P}_2^{n-1}$ for some $n \geq 1$ and consider some feasible solution $\boldsymbol{\phi}$ of $\mathscr{P}_1^n$. For brevity, we will use the notation:

$$R(w; H, X) = q_w + \sum_{e \in E^-(w;H) \setminus X} \boldsymbol{\phi}_e - \sum_{e \in E^+(w;H) \setminus X} \boldsymbol{\phi}_e.$$

By definition $\mathscr{P}_1^n \subseteq \mathscr{P}_1^{n-1}$ and by the induction assumption $\mathscr{P}_1^{n-1} = \mathscr{P}_2^{n-1}$. Thus, $\boldsymbol{\phi}$ is a feasible solution of $\mathscr{P}_2^{n-1}$. Consider some v of depth n. Since $\boldsymbol{\phi}$ is feasible for

$\mathscr{P}_1^n$ and there is exactly one edge $e^+(v;X)$ in X whose origin is v, then

$$\begin{aligned}\boldsymbol{\phi}_{e^+(v;X)} &= q_v + \sum_{e\in E^-(v;H)} \boldsymbol{\phi}_e - \sum_{e\in E^+(v;H)\setminus X} \boldsymbol{\phi}_e \\ &= q_v + \sum_{e\in E^-(v;X)} \boldsymbol{\phi}_e + \sum_{e\in E^-(v;H)\setminus X} \boldsymbol{\phi}_e - \sum_{e\in E^+(v;H)\setminus X} \boldsymbol{\phi}_e \\ &= R(v;H,X) + \sum_{e\in E^-(v;X)} \boldsymbol{\phi}_e.\end{aligned}$$

Each node w in $E^-(v,X)$ must have pseudo-rank strictly less than v, so by induction:

$$\sum_{e\in E^-(v;X)} \boldsymbol{\phi}_e = \sum_{e\in E^-(v,X)} \sum_{u\in V(H):u\prec^X e} R(u;H,X).$$

We claim that the collection of sets

$$\mathcal{A} := \{V(L(e;X)) : e \in E^-(v;X)\}$$

is a partition of $V(L(v;X)) - v$. It is straightforward to verify that the union of the sets of $\mathcal{A}$ is equal to $V(L(v;X)) - v$. Suppose there were edges $e_1 = (w_1, W_1)$ and $e_2 = (w_2, W_2)$ in $E^-(v;X)$ such that $V(L(e_1;X))$ and $V(L(e_2;x))$ have a common node y. By Lemma 7, $U(y;X)$ is a compath. Since H is branching on compaths, every node of $U(y;X)$ must have at most one incoming edge. However, e_1 and e_2 are distinct incoming edges of v, both of which are in $U(y;X)$. This is a contradiction, so sets of $\mathcal{A}$ must be disjoint. Using this fact, we can rewrite the summation

$$\sum_{e\in E^-(v;X)} \boldsymbol{\phi}_e = \sum_{u:u\prec^X v} R(u;H,X).$$

Thus,

$$\begin{aligned}\boldsymbol{\phi}_{e^+(v,X)} &= R(v;H,X) + \sum_{u:u\prec^X v} R(u;H,X) \\ &= \sum_{u:u\preceq^X v} R(v;H,X).\end{aligned}$$

Thus, $\boldsymbol{\phi}$ satisfies the conservation constraints of $\mathscr{P}_2^n$. As in the base case, the capacity and non-negativity constraints are the same in $\mathscr{P}_1^n$ as in $\mathscr{P}_2^n$. Thus, $\mathscr{P}_1^n \subseteq \mathscr{P}_2^n$. This sequence of steps can be reversed to show that $\mathscr{P}_2^n \subseteq \mathscr{P}_1^n$. □

While the polytopes $\mathscr{P}_1$ and $\mathscr{P}_2$ are equivalent, rearranging the constraints in this way allows us to more easily identify a full-dimensional projection of $\mathscr{P}_1$ and $\mathscr{P}_2$. We define a new polytope on the space $\mathbb{R}^{|E(H)\setminus X|}$ by

$P_3(H, \mathbf{q}, \mathscr{C}, \mathbf{a}, X)$:

$$0 \leq \sum_{w : w \preceq^X v} \left(q_w + \sum_{e \in E^-(w;H)\setminus X} f_e - \sum_{e \in E^+(w;H)\setminus X} f_e \right) \qquad \forall v \in V(H), \tag{4.12}$$

$$f_e \leq a_e \qquad \forall e \in \mathscr{C}, \tag{4.13}$$

$$f_e \geq 0 \qquad \forall e \in E(H) \setminus X. \tag{4.14}$$

The variables in this polyhedron are the variables f_e for edges $e \in E(H)\setminus X$. It is straightforward to verify that this polyhedron is the projection of $\mathscr{P}_2$ that omits the flow variables along the edges X. Under the non-degeneracy assumptions, we can show the polyhedron $\mathscr{P}_3$ has full dimension.

Theorem 7. *Let there be an ABF-graph H, a set of edges $\mathscr{C}$ in $E(H)$, and an edge selection X of H such that X is disjoint from $\mathscr{C}$. Let $\boldsymbol{q}$ and $\boldsymbol{a}$ be integral vectors of supplies and capacities respectively. If the non-degeneracy assumptions (stated at the beginning of Section 4.8) hold then the convex hull of integral solutions to $\mathscr{P}_3(H, \boldsymbol{q}, \mathscr{C}, \boldsymbol{a}, X)$ has dimension $|E \setminus X|$.*

Proof. In this proof, we identify $|E \setminus X| + 1$ integral affinely independent feasible solutions in $\mathscr{P}_1$, and we take care that the projections of these solutions are still

affinely independent in $\mathscr{P}_3$. This in turn provides the same number of solutions in $\mathscr{P}_3$. This set of solutions contains $|E \setminus X|$ solutions that correspond to the edges of $E \setminus X$, as well as one additional solution. We first define a solution $\boldsymbol{O}$ in which the flow assigned the edge $e = (v, W)$ is given by:

$$f_e(\boldsymbol{O}) := \begin{cases} \sum_{w : w \preceq^X v} q_w & \text{if } e \text{ is in } X, \\ 0 & \text{otherwise.} \end{cases}$$

It is straightforward to verify that this solution satisfies the constraints of $\mathscr{P}_2$, which implies that it is a feasible solution of $\mathscr{P}_1$.

We now construct a solution $\boldsymbol{\lambda}^e$ corresponding to each edge e in $E \setminus X$. Order the edges of $E \setminus X$ in a total ordering $\prec^*$ that is consistent with the natural partial ordering $\prec^H$. We will then construct the solutions corresponding to each arc one at a time, following the ordering $\prec^*$ and starting with the minimal arc. The solution $\boldsymbol{\lambda}^e$ for each arc e is constructed in such a way that e is $f_e(\boldsymbol{\lambda}^e) > 0$ and such that $e' \prec^* e$ for any edge e' of $F^+(\boldsymbol{\lambda}^e) \setminus X$. This implies that the solutions are affinely independent and that the projections of these solutions onto $\mathscr{P}_3$ are affinely independent. This fact is also useful in the construction of solutions in subsequent iterations. The details of the construction of $\boldsymbol{\lambda}^e$ for some edge $e = (v, W)$ are as follows. Lemmas 7 and 8 imply that

$$P_e := e + \sum_{w \in W} U(w; X)$$

is a maximal compath. There are two cases:

1. There is some node $\rho \preceq^X v$ such that $q_\rho > 0$.

2. $q_\rho = 0$ for all $\rho \preceq^X v$.

Consider case 1. We define the corresponding solution $\boldsymbol{\lambda}^e = \mathbf{O} + \boldsymbol{\ell}(P_e) - \boldsymbol{\ell}(U(v;X))$. Note that $\mathbf{O}$ has no capacitated edges with positive flow, so it is trivially true that $E(P_e)$ shares no capacitated edges in $F^+(\mathbf{O})$. Note also that because there is some node $\rho \preceq^X v$ such that $q_\rho > 0$, then

$$\sum_{w : w \preceq^X v} q_v > 0.$$

From the construction of $\mathbf{O}$, we can then see that every edge in the compath $U(v;X)$ is assigned a positive flow in $\mathbf{O}$. Thus, by Lemma 10, $\boldsymbol{\lambda}^e$ is a feasible solution. Note that e is the only edge of $E \setminus X$ that is assigned a positive flow, so it is trivially true that all of the edges $E \setminus X$ that receive positive flow are less than or equal to e in the natural ordering on H.

For case 2, let ω be a minimal node of $L(v;X)$. From the non-degeneracy assumptions we know there is some node y such that $y \preceq^H \omega$ and $q_y > 0$, and we know that $q_\omega = 0$ because $q_\rho = 0$ for all $\rho \prec^X e$. Thus, there must be an edge η into the node ω. Since ω is a minimal node in the ordering $\prec^X$, η is an edge of $E \setminus X$. The solution $\boldsymbol{\lambda}^\eta$ was constructed in a previous iteration of this process because $\eta \prec^H e$. By construction, edge η is in $F^+(\boldsymbol{\lambda}^\eta)$, while $f_\xi(\boldsymbol{\lambda}^\eta) = 0$ for any edge ξ of $E \setminus X$ such that $\xi \succ^H \eta$. Thus, the edge $e^+(\omega, X)$ must receive positive flow in $\boldsymbol{\lambda}^\eta$ because there is a positive amount of flow on an incoming edge of ω, and there is zero flow on all other outgoing edges of ω. Thus, by Lemma 9 there is a maximal compath $P_{e^+(\omega,X)}$ formed from edges of $F^+(\boldsymbol{\lambda}^\eta)$ starting at the edge

$e^+(\omega, X)$. Then, by Lemmas 7 and 10, the solution

$$\boldsymbol{\theta} := \boldsymbol{\lambda}^\eta + \boldsymbol{\ell}\left(U(\omega; X)\right) - \boldsymbol{\ell}\left(P_{e^+(\omega,X)}\right)$$

is a feasible solution in which all the edges of $U(\omega; X)$ are assigned a positive quantity of flow.

Note that it is still true in $\boldsymbol{\theta}$ that any edge of $F^+(\boldsymbol{\theta}) \setminus X$ is less than or equal to η in the natural ordering on H. Further note that in the solution $\boldsymbol{\theta}$, positive flow is assigned to all edges of $U(v; X)$ because $U(v; X)$ is a subgraph of $U(\omega; X)$. Thus, again applying Lemmas 7 and 10, the solution

$$\boldsymbol{\lambda}^e := \boldsymbol{\lambda}^\eta + \boldsymbol{\ell}\left(U(\omega; X)\right) - \boldsymbol{\ell}\left(P_{e^+(\omega,X)}\right) + \boldsymbol{\ell}\left(P_e\right) - \boldsymbol{\ell}\left(U(v; X)\right)$$

is feasible, and has positive flow assigned to the edge e. Since $\eta \prec^H e$ and e is the only edge of $E \setminus X$ that is given more flow in $\boldsymbol{\lambda}^e$ than $\boldsymbol{\theta}$, it is true that $e' \prec^H e$ for any edge e' in $F^+(\boldsymbol{\lambda}^e) \setminus X$. This completes the construction.

□

4.8.2 Facet-Defining Equalities

As it turns out, both the formulation $\mathscr{P}_1$ and its projection $\mathscr{P}_3$ are quite strong. In fact, every constraint is either a facet of the convex hull of integral solutions or is weakly dominated by other constraints in the formulation. This remains true in the full-dimensional formulation. This is described in detail in the Theorems 8, 9, and 10.

Theorem 8. *Let there be an ABF-graph H, a vector of supply quantities $\boldsymbol{q}$, a set of capacitated edges $\mathscr{C}$, a vector of demand quantities $\boldsymbol{a}$, and an edge selection X*

of $E(H)$ that contains no capacitated edges. If some edge $\eta = (v, W)$ in $E(H) \setminus X$ is the only incoming edge of some node ρ in $V(H)$ such that $q_\rho = 0$ then the non-negativity constraint for the edge η in the polytope $\mathscr{P}_1(H, \boldsymbol{q}, \mathscr{C}, \boldsymbol{a})$ is dominated by the conservation constraint of the node ρ and the non-negativity constraints on the outgoing edges of w. Otherwise, this constraint is facet-defining for the convex hull of integral solutions. This is theorem also holds if $\mathscr{P}_1(H, \boldsymbol{q}, \mathscr{C}, \boldsymbol{a})$ is replaced by $\mathscr{P}_2(H, \boldsymbol{q}, \mathscr{C}, \boldsymbol{a}, X)$ or $\mathscr{P}_3(H, \boldsymbol{q}, \mathscr{C}, \boldsymbol{a}, X)$.

Proof. In the forward direction, let there be some (possibly infeasible or fractional) solution $\boldsymbol{\phi}$ that satisfies the conservation constraint for ρ and the non-negativity constraints for the edges in $E^+(\rho, H)$ as defined for the polytope $\mathscr{P}_1$. Then,

$$\sum_{e \in E^-(\rho;H)} f_e(\boldsymbol{\phi}) - \sum_{e \in E^+(\rho;H)} f_e(\boldsymbol{\phi}) = q_\rho.$$

Since η is the only incoming edge of ρ, this can be rewritten as

$$f_\eta(\boldsymbol{\phi}) = q_\rho + \sum_{e \in E^+(\rho;H)} f_e(\boldsymbol{\phi}).$$

Since $\boldsymbol{\phi}$ satisfies the non-negativity constraints on the edge $E^+(\rho; H)$ this implies that $f_\eta(\boldsymbol{\phi}) \geq 0$.

Similarly, let there be some (possibly infeasible or fractional) solution $\boldsymbol{\phi}$ that satisfies the conservation constraint for ρ and the non-negativity constraints for the edges in $E^+(\rho, H)$ as defined for the polytope $\mathscr{P}_2$. Then,the conservation constraint for ρ states that

$$f_{e^+(\rho;W)}(\boldsymbol{\phi}) = \sum_{w : w \preceq^X v} \left(q_w + \sum_{e \in E^-(w;H) \setminus X} f_e(\boldsymbol{\phi}) - \sum_{e \in E^+(w;H) \setminus X} f_e(\boldsymbol{\phi}) \right).$$

Since η is the only incoming edge of ρ and η is not in X, ρ has no preceding nodes in X. Therefore, this can be rewritten as

$$f_{e^+(\rho;W)}(\boldsymbol{\phi}) = f_\eta(\boldsymbol{\phi}) - \sum_{e \in E^+(\rho;H)\setminus X} f_e(\boldsymbol{\phi}),$$

$$\sum_{e \in E^+(\rho;H)} f_e(\boldsymbol{\phi}) = f_\eta(\boldsymbol{\phi}).$$

Again, the non-negativity constraints on the outgoing edges of ν imply that $f_\eta \geq 0$.

The argument in the forward direction for $\mathscr{P}_3$ is nearly identical. Let there be some (possibly infeasible or fractional) solution $\boldsymbol{\phi}$ that satisfies the conservation constraint for ρ and the non-negativity constraints for the edges in $E^+(\rho, H) \setminus X$ as defined for the polytope $\mathscr{P}_3$. Then,the conservation constraint for ρ states that

$$0 \leq \sum_{w : w \preceq^X \rho} \left(q_w + \sum_{e \in E^-(w;H)\setminus X} f_e(\boldsymbol{\phi}) - \sum_{e \in E^+(w;H)\setminus X} f_e(\boldsymbol{\phi}) \right).$$

As before, η is the only incoming edge of ρ and η is not in X, so ρ has no preceding nodes in X and we can rewrite this as:

$$0 \leq f_\eta(\boldsymbol{\phi}) - \sum_{e \in E^+(\rho;H)\setminus X} f_e(\boldsymbol{\phi}),$$

$$\sum_{e \in E^+(\rho;H)\setminus X} f_e(\boldsymbol{\phi}) \leq f_\eta(\boldsymbol{\phi}),$$

and the non-negativity constraints on the outgoing edges of ν imply that $f_\eta \geq 0$.

In the opposite direction, let η be an edge of $E(H) \setminus X$ and suppose that there is no node ρ such that η is the only incoming arc. We can construct a set of affinely-independent solutions of size $|E(H) \setminus X|$ in which $f_\eta = 0$. In fact, we claim that with very slight modifications, the procedure given in the proof of Theorem 3 can provide these solutions. Start with $\mathbf{O}$, as defined in the previous proof. Order

the arcs of $E \setminus (X + \eta)$ in an ordering that is consistent with the natural partial ordering on H, and consider each arc $\xi = (v', W')$ starting with the minimal arc. The same two cases apply. If there exists some node $y \preceq^X \eta$ such that $q_y \geq 0$, then construct the solution $\boldsymbol{\lambda}^\xi$ as before. If such a node does not exist, then we construct the solution almost the same as before, but more care is taken as to which incoming edge we select. Again, let ω be a minimal node of $L(v'; X)$. As before, ω must have an incoming edge ζ in $E(H) \setminus X$. If $\zeta \neq \eta$, then we construct the solution $\boldsymbol{\lambda}^\eta$ by modifying the solution $\boldsymbol{\lambda}^\zeta$ as before. If $\zeta = \eta$, then by assumption ω must have another incoming edge κ in $E(H) \setminus X$, and we construct the solution $\boldsymbol{\lambda}^\eta$ by modifying the solution $\boldsymbol{\lambda}^\kappa$ as before.

By the same arguments as before, the projection of these solutions on to $\mathscr{P}_3$ is also affinely independent. Further note that in the construction of the solution $\boldsymbol{\lambda}^\xi$, positive flow is only assigned to edges that were considered in previous iterations of this procedure. Since no iteration is run for the edge η, none of these solutions assign positive flow on the edge η. □

In some cases, the capacity constraint on an edge can be dominated by the constraints corresponding the preceding subgraph. If sufficient flow cannot be routed to the origin node of an edge, then it could be that no feasible solution meets the capacity constraint. In this case, those preceding constraints dominate the capacity constraint. On the other hand, the capacity constraint is facet-defining whenever it is possible to route enough flow to the origin node.

Theorem 9. *Let there be an ABF-graph H, a vector of supply quantities $\boldsymbol{q}$, a set*

of capacitated edges $\mathscr{C}$, a vector of demand quantities $\boldsymbol{a}$, and an edge selection X of $E(H)$ that contains no capacitated edges. Let η be an edge in $\mathscr{C}$. Let f^* be the value to the optimization problem $\max\{f_\eta(\boldsymbol{\phi}) : \boldsymbol{\phi} \in \mathscr{P}_1(L(\eta; H), \boldsymbol{q}, \mathscr{C} - \eta, \boldsymbol{a})\}$. *If $f^* \leq a_\eta$, then the subset of constraints that correspond to the subgraph polytope $P_1(L(\eta; H), \boldsymbol{q}, \mathscr{C} - \eta, \boldsymbol{a})$ dominate the capacity constraint for η. Otherwise, the constraint is facet-defining for the convex hull of integral solutions. This theorem is also true if $\mathscr{P}_1$ is replaced by $\mathscr{P}_2$ or $\mathscr{P}_3$.*

Proof. The forward direction is trivial.

In the backwards direction, suppose that $f^* > a_\eta$. Then we can construct $|E(H) \setminus X|$ integral solutions that satisfy this constraint at equality. Let $\mathbf{a} + \boldsymbol{\ell}(\eta)$ be a vector $\mathbf{a}'$ of capacities where $a'_e = a_e$ for all $e \neq \eta$ and $a'_\eta = a_\eta + 1$. Lemmas 6 and 12 imply that there exists an integral feasible solution $\boldsymbol{\phi}$ to $\mathscr{P}_1(H, \mathbf{q}, \mathscr{C}, \mathbf{a} + \boldsymbol{\ell}(\eta))$ such that:

- $f_\eta(\boldsymbol{\phi}) = a_\eta + 1$,
- $f_e(\boldsymbol{\phi}) = 0$ for all e in $E \setminus X$ where $e \not\preceq^H \eta$.

We use a similar strategy as in previous proofs, and construct feasible solutions one at a time in an iterative procedure. In this proof, we divide this procedure into two stages. In each iteration the first stage, we define a solution $\boldsymbol{\lambda}^\xi$ corresponding to some edge ξ of $F^+(\boldsymbol{\phi}) \setminus (X + \eta)$. In each iteration of the second stage, we define a solution $\boldsymbol{\lambda}^\xi$ corresponding to an edge ξ of $F^0(\boldsymbol{\phi}) \setminus X$. We take care that across both stages, no solutions differ from $\boldsymbol{\phi}$ on the amount of flow assigned to some edge ξ until the iteration corresponding to this edge is reached. This guarantees that

these solutions are affinely independent (and that their projections onto $\mathscr{P}_3$ are also affinely independent).

Order the edges of $F^+(\boldsymbol{\phi}) \setminus X$ in an ordering consistent with $\prec^H$. Consider each edge $\xi = (v, W)$ one at a time, starting with the maximum edge. Let P_ξ be a compath in the edges $F^+(\boldsymbol{\phi})$ whose first edge is ξ (such a compath must exist by Lemma 9). Define a solution

$$\boldsymbol{\theta} := \boldsymbol{\phi} - \boldsymbol{\ell}(P_\xi) + \boldsymbol{\ell}(U(v; X)).$$

By Lemma 10, this solution is a feasible solution of $\mathscr{P}_1(H, \mathbf{q}, \mathscr{C}, \mathbf{a} + \boldsymbol{\ell}(\eta))$. There are two cases: either η is in P_ξ or η is not in P_ξ. If η is in P_ξ then $f_\eta(\boldsymbol{\theta}) = a_\eta$, so that $\boldsymbol{\theta}$ is a feasible solution of $\mathscr{P}_1(H, \mathbf{q}, \mathscr{C}, \mathbf{a})$. In this case, we can let $\boldsymbol{\lambda}^\xi = \boldsymbol{\theta}$. If η is not in P_ξ then $f_\eta(\boldsymbol{\theta}) = a_\eta + 1$. Then let P_η be a compath in $F^+(\boldsymbol{\theta})$ starting at η (Lemma 9 again ensures existence), let ρ be the origin node of η and let

$$\begin{aligned}\boldsymbol{\lambda}^\xi &:= \boldsymbol{\theta} - \boldsymbol{\ell}(P_\eta) + \boldsymbol{\ell}(U(\rho; X)) \\ &= \boldsymbol{\phi} - \boldsymbol{\ell}(P_\xi) + \boldsymbol{\ell}(U(v; X)) - \boldsymbol{\ell}(P_\eta) + \boldsymbol{\ell}(U(\rho; X)),\end{aligned}$$

which is a feasible solution for $\mathscr{P}_1(H, \mathbf{q}, \mathscr{C}, \mathbf{a} + \boldsymbol{\ell}(\eta))$ by Lemma 10. Since η is not an edge of X, then it must be true that $f_\eta(\boldsymbol{\lambda}^\xi) = f_\eta(\boldsymbol{\phi}) - 1$, so that $\boldsymbol{\lambda}^\xi$ is a feasible solution for $\mathscr{P}_1(H, \mathbf{q}, \mathscr{C}, \mathbf{a})$.

In either case, note that $f_\xi(\boldsymbol{\lambda}^\xi) = f_\xi(\boldsymbol{\phi}) - 1$. Suppose that $f_\xi(\boldsymbol{\lambda}^\psi) \neq f_\xi(\boldsymbol{\phi})$ for some edge ψ in $E(H) \setminus X$ where $\psi \neq \xi$ and $\psi \neq \eta$. Then, ξ must be in the compath P_ψ or in the compath P_η. The compath P_η starts at η and consists of edges from $F^+(\boldsymbol{\phi})$. By definition of $\boldsymbol{\phi}$, $f_e(\boldsymbol{\phi}) = 0$ for all e in $E \setminus X$ where $e \not\preceq^H \eta$. This

implies that η is the only edge of P_η that is not in X. Since $\xi \neq \eta$, it must be true that ξ is an edge in P_ψ. This implies that $\psi \prec^H \xi$, which in turn implies that $\boldsymbol{\lambda}^\psi$ was constructed in a later iteration than $\boldsymbol{\lambda}^\xi$. In other words, at the point in time when the solution $\boldsymbol{\lambda}^\xi$ is constructed, it is the only solution that differs from $\boldsymbol{\phi}$ in the amount of flow assigned on the edge ξ. Note also that $f_e(\boldsymbol{\phi}) = f_e(\boldsymbol{\lambda}^\xi)$ for any edge e in $F^0(\boldsymbol{\phi}) \setminus X$.

Next, consider the arcs of $F^0(\boldsymbol{\phi})$. Order these arcs in an ordering that agrees with the natural ordering $\prec^H$. Consider each arc ξ, starting with the minimum arc. We claim that there is a maximal compath P_ξ such that:

1. The edge ξ is an edge of P_ξ;
2. For any edge $e \in P_\xi$ such that $e \not\preceq^H \xi$, the edge e is in X;
3. The set of edges $E(P_\xi)$ is disjoint from $F^+(\boldsymbol{\phi}) \cap \mathscr{C}$;
4. The compath P_ξ starts at a node that has an outgoing edge ψ in $F^+(\boldsymbol{\phi})$.

If the origin node of ξ has a positive quantity of resource, then it is straightforward to verify that the path $U(e; X)$ is a compath that meets these conditions. Suppose that this origin node has no resource present. By the non-degeneracy conditions (given at the beginning of Section 4.8), there must exist some path P_0 from some node ρ to the origin node of ξ where $q_\rho > 0$. Let $e^* = (v, W)$ be the minimal edge of P_0 such that $E(U(e^*; P_0 + \xi))$ is disjoint from $F^+(\boldsymbol{\phi})$. Consider the case that $v = \rho$, i.e. e^* is the outgoing edge of ρ in P_0. Since $q_\rho > 0$, then there must be another outgoing edge ψ in $F^+(\boldsymbol{\phi})$. Consider the case that e^* is not the outgoing

edge of ρ in P_0. Then there exists some edge e' that precedes e^* in the path P_0; this edge must be in $F^+(\boldsymbol{\phi})$ from the definition of e^*. Since v has an incoming edge with positive flow in $\boldsymbol{\phi}$, it must also have an outgoing edge ψ in $F^+(\boldsymbol{\phi})$. Thus, in either case, $U(e^*; P_0)$ is a path in $F^0(\boldsymbol{\phi})$ whose origin node has an outgoing edge ψ in $F^+(\boldsymbol{\phi})$. By Lemma 11, there exists a maximal compath P_ξ starting at v such that $E(U(e^*, P_0 + \xi)) \subseteq E(P_\xi) \subseteq E(U(e^*; P_0 + \xi)) \cup X$. Since ξ is the maximal edge of $P_0 + \xi$, it is an edge of $U(e^*, P_0 + \xi)$, so requirement 1 is satisfied by P_ξ. All of the edges in P_0 are less than or equal to ξ and are not in $F^+(\boldsymbol{\phi}) \cap \mathscr{C}$, so P_ξ also satisfies requirement 2 and 3. By definition, the compath starts at a node that has an outgoing edge ξ in $F^+(\boldsymbol{\phi})$, so requirement 4 is also satisfied. Let P_ψ be a compath starting at ψ in $F^+(\boldsymbol{\phi})$ (the existence of this compath is guaranteed by Lemma 7). Then, we can define a solution

$$\boldsymbol{\theta} := \boldsymbol{\phi} + \boldsymbol{\ell}(P_\xi) - \boldsymbol{\ell}(P_\psi),$$

which is a feasible solution for $\mathscr{P}_1(H, \mathbf{q}, \mathscr{C}, \mathbf{a} + \boldsymbol{\ell}(\eta))$ by Lemma 10. As before, if η is an edge of P_ψ, then $\boldsymbol{\theta}$ is a feasible solution for $\mathscr{P}_1(H, \mathbf{q}, \mathscr{C}, \mathbf{a})$ and we let $\boldsymbol{\lambda}^\xi = \boldsymbol{\theta}$. If η is not in $E(P_\psi)$,

$$\begin{aligned}\boldsymbol{\lambda}^\xi &:= \boldsymbol{\theta} - \boldsymbol{\ell}(P_\eta) + \boldsymbol{\ell}(U(v, X)) \\ &= \boldsymbol{\phi} + \boldsymbol{\ell}(P_\xi) - \boldsymbol{\ell}(P_\psi) - \boldsymbol{\ell}(P_\eta) + \boldsymbol{\ell}(U(v; X)).\end{aligned}$$

which is a feasible solution for $\mathscr{P}_1(H, \mathbf{q}, \mathscr{C}, \mathbf{a})$ for the same reasons as in the first stage.

Similar to before, note that $f_\xi(\boldsymbol{\lambda}^\xi) = f_\xi(\boldsymbol{\phi}) + 1$. We also claim that $\boldsymbol{\lambda}^\xi$ is the first solution constructed in this process that differs from $\boldsymbol{\phi}$ in the amount of flow

assigned to ξ. Let $\boldsymbol{\lambda}^{\xi}$ be the solution corresponding to some ξ constructed in this second stage. Let $\boldsymbol{\lambda}^{\psi}$ be some other solution corresponding to an edge ψ such that $f_{\xi}(\boldsymbol{\lambda}^{\psi}) \neq f_{\xi}(\boldsymbol{\phi})$. Since ξ is assigned zero flow in the solution $\boldsymbol{\phi}$, then ξ must be assigned more flow in $\boldsymbol{\lambda}^{\psi}$ than in $\boldsymbol{\phi}$. In any solution constructed in the first stage, only edges of X receive more flow than in $\boldsymbol{\phi}$. Thus, $\boldsymbol{\lambda}^{\psi}$ is a second-stage solution. In a second-stage solution, $\boldsymbol{\lambda}^{\psi}$ only assigns more flow than $\boldsymbol{\phi}$ to edges of X and edges of a compath P_{ψ}. Thus, ξ is in the compath P_{ψ}. This compath is defined so that every edge of P_{ψ} is an edge of X or is less than or equal to ψ. Thus, ξ must be less than ψ. This would imply that the solution $\boldsymbol{\lambda}^{\xi}$ was constructed in a previous iteration than $\boldsymbol{\lambda}^{\psi}$. □

Theorems 8 and 9 describe the conditions under which the non-negativity and capacity constraints for edges of $E(H) \setminus X$ are facet-defining. In $\mathscr{P}_1$ and $\mathscr{P}_2$, the remaining inequality constraints are the non-negativity constraints corresponding to the edges of X. For $\mathscr{P}_3$, the conservation constraints are the constraints that have not yet been handled. There is a correspondence between these constraints. The non-negativity constraint of the edge $e^+(v; X)$ for some node v holds at equality in some solution $\boldsymbol{\phi}$ if and only if the conservation constraint at v holds at equality in the projection of $\boldsymbol{\phi}$ on to the polytope $\mathscr{P}_3$. For this reason, the conservation constraints in $\mathscr{P}_3$ are facet-defining when the non-negativity constraints on edges of X are facet-defining for $\mathscr{P}_1$ and $\mathscr{P}_2$. These conditions are described in Theorem 10. In this Theorem, we use the notation that $\mathbf{q} + \boldsymbol{\ell}(v)$ is the vector of quantities $\mathbf{q}'$

such that

$$q'_w = \begin{cases} q_w & \text{if } w \neq v, \\ q_w + 1 & \text{if } w = v. \end{cases}$$

Theorem 10. *Let there be an ABF-graph H, a vector of supply quantities $\boldsymbol{q}$, a set of capacitated edge $\mathscr{C}$, a vector of demand quantities $\boldsymbol{a}$, and an edge selection X of $E(H)$ that contains no capacitated edges. Let v be any node in $V(H)$, and let*

$$f^* = \max \left\{ \sum_{w : w \preceq^X v} \left(\sum_{e \in E^+(w) \setminus X} f_e(\boldsymbol{\phi}) - \sum_{e \in E^-(w) \setminus X} f_e(\boldsymbol{\phi}) \right) : \right.$$
$$\left. \boldsymbol{\phi} \in \mathscr{P}_1(L(v; H) + E^+(v; H), \boldsymbol{q} + \boldsymbol{\ell}(v), \mathscr{C}, \boldsymbol{a}) \right\}.$$

Note that the value of f^* will remain unchanged if $\mathscr{P}_1$ is replaced with $\mathscr{P}_2$ or $\mathscr{P}_3$. Consider the following cases:

1. *There exists a destination node ρ of $e^+(v, X)$ such that $q_\rho = 0$ and $e^+(v, X)$ is the only incoming edge of ρ.*

2. *The value f^* is less than or equal to $\sum_{w : w \preceq^X v} q_w$.*

3. *Neither case 1 nor case 2 holds.*

If case 1 holds then the non-negativity constraint of $e^+(v, X)$ is dominated by the non-negativity constraints of the outgoing edges of ρ and the conservation constraint on ρ. If case 2 holds then the non-negativity constraint of $e^+(v, X)$ is dominated by the constraints of $\mathscr{P}_1(L(v; H) + E^+(v; H), \boldsymbol{q}, \mathscr{C}, \boldsymbol{a})$ excluding the non-negativity constraint on $e^+(v, X)$. If case 3 holds, then the non-negativity constraint on $e^+(v, X)$

is facet-defining for the convex hull of integral solutions. This statement remains true if $\mathscr{P}_1$ is replaced by $\mathscr{P}_2$.

A similar statement holds true for $\mathscr{P}_3$: if case 1 holds then the conservation constraint at v is dominated by the conservation constraint at w. If case 2 holds then the conservation constraint at v is dominated by the constraints of $\mathscr{P}_1(L(v;H) + E^+(v;H), \boldsymbol{q}, \mathscr{C}, \boldsymbol{a})$ excluding the conservation constraint at v. If case 3 holds, then the conservation constraint at v is facet-defining for the convex hull of integral solutions.

Proof. **Case 1.** For $\mathscr{P}_1$ and $\mathscr{P}_2$, the proof of this statement when case 1 holds is nearly identical to the proof of the forward direction of Theorem 8, so we will only provide a proof for the statement for $\mathscr{P}_3$. Let there be some edge $e^+(v;X)$ in X that is the only incoming edge of a node ρ. Let $\boldsymbol{\phi}$ be a (possibly infeasible or fractional) solution to $\mathscr{P}_3(H, \mathbf{q}, \mathscr{C}, \mathbf{a}, X)$ that satisfies the conservation constraint at ρ and the non-negativity constraints on the outgoing edges of ρ. Then

$$0 \leq \sum_{w:w \preceq^X \rho} \left(q_w + \sum_{e \in E^-(w;H)\setminus X} f_e(\boldsymbol{\phi}) - \sum_{e \in E^+(w;H)\setminus X} f_e(\boldsymbol{\phi}) \right).$$

Since v is the node immediately preceding ρ in X, the right-hand-side can be rewritten:

$$\begin{aligned} 0 \leq \sum_{w:w \preceq^X v} & \left(q_w + \sum_{e \in E^-(w;H)\setminus X} f_e(\boldsymbol{\phi}) - \sum_{e \in E^+(w;H)\setminus X} f_e(\boldsymbol{\phi}) \right) \\ & + \left(q_\rho + \sum_{e \in E^-(\rho;H)\setminus X} f_e(\boldsymbol{\phi}) - \sum_{e \in E^+(\rho;H)\setminus X} f_e(\boldsymbol{\phi}) \right). \end{aligned}$$

By the assumptions in this case, $q_\rho = 0$ and the set $E^-(\rho; H) \setminus X$ is empty. Then

$$0 \leq \sum_{w:w \preceq^X v} \left(q_w + \sum_{e \in E^-(w;H) \setminus X} f_e(\boldsymbol{\phi}) - \sum_{e \in E^+(w;H) \setminus X} f_e(\boldsymbol{\phi}) \right) - \sum_{e \in E^+(\rho;H) \setminus X} f_e(\boldsymbol{\phi}).$$

Applying the non-negativity constraints on the outgoing edges of ρ,

$$0 \leq \sum_{w:w \preceq^X v} \left(q_w + \sum_{e \in E^-(w;H) \setminus X} f_e(\boldsymbol{\phi}) - \sum_{e \in E^+(w;H) \setminus X} f_e(\boldsymbol{\phi}) \right).$$

Thus, the conservation constraint on the node v is dominated by the conservation constraint of ρ and the non-negativity constraints on the edges of ρ. This completes the proof of the statement when condition 1 holds.

Case 2. The proof of the statement for $\mathscr{P}_2$ when case 2 holds is as follows. First, we claim that non-negativity constraint for the edge $e^+(v; H)$ of the polytope $\mathscr{P}_2\left(L(v; h) + E^+(v; H), \mathbf{q} + \ell(v), \mathscr{C}, \mathbf{a}\right)$ is a redundant constraint of this polytope. Let $\boldsymbol{\phi}$ be a feasible (possibly fractional) solution to this polytope. Then, by definition of f^* and the assumptions of the case,

$$\begin{aligned}
\sum_{w:w \preceq^X v} q_w &\geq \sum_{w:w \preceq^X v} \left(\sum_{e \in E^+(w) \setminus X} f_e(\boldsymbol{\phi}) - \sum_{e \in E^-(w) \setminus X} f_e(\boldsymbol{\phi}) \right) \\
0 &\leq \sum_{w:w \preceq^X v} \left(q_w + \sum_{e \in E^-(w) \setminus X} f_e(\boldsymbol{\phi}) - \sum_{e \in E^+(w) \setminus X} f_e(\boldsymbol{\phi}) \right) \\
1 &\leq \sum_{w:w \preceq^X v} \left(q'_w + \sum_{e \in E^-(w) \setminus X} f_e(\boldsymbol{\phi}) - \sum_{e \in E^+(w) \setminus X} f_e(\boldsymbol{\phi}) \right) \\
1 &\leq f_{e^+(v;X)}(\boldsymbol{\phi}).
\end{aligned}$$

Thus, there is no solution $\boldsymbol{\phi}$ in which the non-negativity constraint on the edge $e^+(v; H)$ holds at equality, and this constraint is redundant.

Let there be some (infeasible, possibly fractional) solution $\boldsymbol{\theta}$ in which all

constraints of the polytope $\mathscr{P}_2\left(L(v;h)+E^+(v;H),\mathbf{q},\mathscr{C},\mathbf{a}\right)$ other than the non-negativity constraint on $e^+(v,;X)$ are satisfied. We claim that the solution

$$\boldsymbol{\phi} := \boldsymbol{\theta} + \boldsymbol{\ell}(e^+(v;X))$$

is a feasible solution to $\mathscr{P}_2\left(L(v;h)+E^+(v;H),\mathbf{q}+\boldsymbol{\ell}(v),\mathscr{C},\mathbf{a}\right)$. Note that the only constraint of this polytope in which $e^+(v;X)$ appears the non-negativity constraint on this edge is the conservation constraint on v. Now,

$$\begin{aligned} f_{e^+(v;X)}(\boldsymbol{\phi}) &= f_{e^+(v;X)}(\boldsymbol{\theta}) + 1 \\ &= \sum_{w:w\preceq^X v}\left(q_w + \sum_{e\in E^-(w)\setminus X} f_e(\boldsymbol{\theta}) - \sum_{e\in E^+(w)\setminus X} f_e(\boldsymbol{\theta})\right) + 1 \\ &= \sum_{w:w\preceq^X v}\left(q'_w + \sum_{e\in E^-(w)\setminus X} f_e(\boldsymbol{\phi}) - \sum_{e\in E^+(w)\setminus X} f_e(\boldsymbol{\phi})\right). \end{aligned}$$

So $\boldsymbol{\phi}$ satisfies the conservation constraint on v. As we already showed, the non-negativity constraint on $e^+(v;X)$ is redundant. The solution $\boldsymbol{\phi}$ satisfies all other constraints, so it will satisfy this non-negativity constraint as well. Thus, $\boldsymbol{\phi}$ solution is a feasible solution for $\mathscr{P}_2\left(L(v;h)+E^+(v;H),\mathbf{q}+\boldsymbol{\ell}(v),\mathscr{C},\mathbf{a}\right)$ excluding the non-negativity constraint on $e^+(v;X)$. From the previous arguments,

$$f_{e^+(v;X)}(\boldsymbol{\phi}) \geq 1.$$

This implies that

$$f_{e^+(v;X)}(\boldsymbol{\theta}) \geq 0.$$

Thus, any feasible solution that satisfies the constraints of $\mathscr{P}_3(L(v;H)+E^+(v;H),\mathbf{q},\mathscr{C},\mathbf{a})$ excluding the non-negativity constraint on $e^+(v;H)$ will also satisfy the non-negativity constraint.

The proof of the statement for $\mathscr{P}_3$ when case 2 holds is similar. Since

$$f^* \leq \sum_{w:w \preceq^X v} q_w$$

then for any feasible solution $\boldsymbol{\phi}$ of $\mathscr{P}_3(L(v;H) + E^+(v;H), \mathbf{q} + \boldsymbol{\ell}(v), \mathscr{C}, \mathbf{a}, X)$,

$$\begin{aligned}
\sum_{w:w \preceq^X v} q_w &\geq \sum_{w:w \preceq^X v} \left(\sum_{e \in E^+(w) \setminus X} f_e(\boldsymbol{\phi}) - \sum_{e \in E^-(w) \setminus X} f_e(\boldsymbol{\phi}) \right), \\
0 &\leq \sum_{w:w \preceq^X v} \left(q_w + \sum_{e \in E^-(w) \setminus X} f_e(\boldsymbol{\phi}) - \sum_{e \in E^+(w) \setminus X} f_e(\boldsymbol{\phi}) \right), \\
1 &\leq \sum_{w:w \preceq^X v} \left(q'_w + \sum_{e \in E^-(w) \setminus X} f_e(\boldsymbol{\phi}) - \sum_{e \in E^+(w) \setminus X} f_e(\boldsymbol{\phi}) \right),
\end{aligned}$$

for any feasible solution $\boldsymbol{\phi}$ in the polytope. Thus, the conservation constraint on v is never met at equality in this polytope, so the constraint can be removed. Since v is a maximal node in this polytope, the parameter q'_v does not appear in any other constraints. Thus, the resulting polytope is equal to $\mathscr{P}_3(L(v;H)+E^+(v;H), \mathbf{q}, \mathscr{C}, \mathbf{a})$ with the conservation constraint on v removed. Thus, for any solution of $\boldsymbol{\phi}$ of $\mathscr{P}_3(L(v;H) + E^+(v;H), \mathbf{q}, \mathscr{C}, \mathbf{a})$, it is true that

$$\sum_{w:w \preceq^X v} q_w \geq \sum_{w:w \preceq^X v} \left(\sum_{e \in E^+(w) \setminus X} f_e(\boldsymbol{\phi}) - \sum_{e \in E^-(w) \setminus X} f_e(\boldsymbol{\phi}) \right).$$

This proves that the constraints of $\mathscr{P}_3(L(v;H) + E^+(v;H), \mathbf{q}, \mathscr{C}, \mathbf{a})$ excluding the conservation constraint on v dominate this conservation constraint.

Case 3. Suppose that case 3 holds. It is implied by Lemmas 6 and 12 that there exists an integral solution $\boldsymbol{\phi}$ for $\mathscr{P}_2(H, \mathbf{q} + \boldsymbol{\ell}(v), \mathscr{C}, \mathbf{a})$, such that $f_e(\boldsymbol{\phi}) = 0$ for any e in $E \setminus (X \cup E^+(v;H))$ where $e \not\preceq^H v$. Due to the assumptions of this case,

there exists a solution that additionally satisfies:

$$\sum_{w:w\preceq^X v}\left(\sum_{e\in E^+(w;H)\setminus X} f_e(\boldsymbol{\phi}) - \sum_{e\in E^-(w;H)\setminus X} f_e(\boldsymbol{\phi})\right) = \sum_{w:w\preceq^X v} q'_w,$$

$$\sum_{w:w\preceq^X v}\left(q'_w + \sum_{e\in E^-(w;H)\setminus X} f_e(\boldsymbol{\phi}) - \sum_{e\in E^+(w;H)\setminus X} f_e(\boldsymbol{\phi})\right) = 0,$$

$$f_{e^+(v,X)}(\boldsymbol{\phi}) = 0.$$

In a similar fashion to previous constructions, we use this solution to construct $|E(H)\setminus X|$ integral feasible solutions $\mathscr{P}_2(H,\mathbf{q},\mathscr{C},\mathbf{a},X)$ in which the non-negativity constraint on $f_{e^+(v;X)}$ is satisfied to equality. These are produced in a three-stage process. In each iteration of the first stage, a solution corresponding to an edge of $(F^+(\boldsymbol{\phi})\cup E^+(v;H))\setminus X$ is constructed. In the second stage, solutions corresponding to an edge of $F^+(\boldsymbol{\phi})\setminus(X\cup E^+(v;H))$ are constructed. In the final stage, a solution corresponding to an edge of $F^0(\boldsymbol{\phi})\setminus X$ is constructed. As in the proof of Theorem 9, the solutions are constructed such that $\boldsymbol{\lambda}^e$ is the first solution in this process which assigns an amount of flow on e that differs from that in $\boldsymbol{\phi}$. This ensures that the solutions are affinely independent and that the projections of these solutions into $\mathscr{P}_3$ are affinely independent as well. For any solution $\boldsymbol{\phi}$ for $\mathscr{P}_2$, the flow $f_e^+(v;X)(\boldsymbol{\phi})$ is equal to zero if and only the conservation constraint holds at equality in the projection of $\boldsymbol{\phi}$ onto $\mathscr{P}_3$, so this provides the proof for the statements for all three polytopes.

For each edge in $(F^+(\boldsymbol{\phi})\cup E^+(v;H))\setminus X$, let P_ξ be a maximal compath in $F^+(\boldsymbol{\phi})$ starting at ξ (such a compath must exist by Lemma 9) and let

$$\boldsymbol{\lambda}^\xi = \boldsymbol{\phi} - \boldsymbol{\ell}(P_\xi).$$

We claim that this is a feasible solution for $\mathscr{P}_2(H, \mathbf{q}, \mathscr{C}, \mathbf{a}, X)$. The non-negativity constraints and edge capacity constraints are clearly preserved. The conservation constraint for any node $\rho \neq v$ is preserved, since the maximal compath $P_{e^+(v;X)}$ has either one incoming and outgoing edge from ν or no incoming edges and no outgoing edges. The conservation constraint at v is satisfied because this solution has one less unit of outgoing flow at v that $\boldsymbol{\theta}$ and the quantity of supply at the node is decreased by one. It is clear that $f_{e^+(v;X)}(\boldsymbol{\lambda}^\xi) = 0$. By definition of $\boldsymbol{\phi}$, any edge assigned positive flow that is greater than an outgoing edge of v must either be an edge in X. Thus, we can see that solution $\boldsymbol{\lambda}^\xi$ only differs from ϕ in the flow assigned to ξ and the flow assigned to edges of X.

Order the edges of $F^+(\boldsymbol{\phi}) \setminus (E^+(v; H) \cup X)$ in some total ordering consistent with the natural partial ordering on H. Consider each edge ξ in order, starting with the maximum edge. Let P_ξ be a maximal compath in $F^+(\boldsymbol{\phi})$ starting at ξ (such a compath must exist by Lemma 9). Let y be the origin node of ξ. Define a solution

$$\boldsymbol{\theta} = \boldsymbol{\phi} - \boldsymbol{\ell}(P_\xi) + \boldsymbol{\ell}(U(y; X)).$$

By Lemmas 7 and 10, $\boldsymbol{\theta}$ is a feasible solution of $\mathscr{P}_2(H, \mathbf{q} + \boldsymbol{\ell}(v), \mathscr{C}, \mathbf{a})$. Furthermore, it is clear that

$$\begin{aligned} f_{e^+(v;X)}(\boldsymbol{\theta}) &\leq f_{e^+(v;X)}(\boldsymbol{\phi}) + 1 \\ &\leq 1. \end{aligned}$$

There are two cases: either $f_{e^+(v;X)}(\boldsymbol{\theta}) = 1$ or $f_{e^+(v;X)}(\boldsymbol{\theta}) = 0$. Consider the former case. Then, let $P_e^+(v; X)$ be a complete compath in $F^+(\boldsymbol{\theta})$ starting with the edge

$e^+(v;X)$ (such a compath must exist due to Lemma 9). By a similar justification as that given in the first stage construction,

$$\begin{aligned}\boldsymbol{\lambda}^\xi &:= \boldsymbol{\theta} - \boldsymbol{\ell}(P_{e^+(v;X)}),\\ &= \boldsymbol{\phi} - \boldsymbol{\ell}(P_\xi) + \boldsymbol{\ell}(U(y;X)) - \boldsymbol{\ell}(P_{e^+(v;X)}).\end{aligned}$$

is a feasible solution for $\mathscr{P}_2(H, \mathbf{q}, \mathscr{C}, \mathbf{a}, X)$. Note also that $\boldsymbol{\lambda}^\xi(e^+(v;X)) = 0$, so the non-negativity constraint of $e^+(v;X)$ holds to equality.

Consider the case that $f_{e^+(v,X)}(\boldsymbol{\theta}) = 0$. Then it must be true that $f_\eta(\boldsymbol{\phi}) \geq 1$ for some outgoing edge η of v not in X, because

$$\begin{aligned}\sum_{e \in E^+(v) \setminus X} f_e(\boldsymbol{\theta}) &= 1 + q_v + \sum_{e \in E^-(v)} f_e(\boldsymbol{\theta})\\ &\geq 1.\end{aligned}$$

Let P_η be a complete compath starting at η in $F^+(\boldsymbol{\theta}_\xi)$ (such a compath must exist due to Lemma 9). Then, let

$$\begin{aligned}\boldsymbol{\lambda}^\xi &:= \boldsymbol{\theta} - \boldsymbol{\ell}(P_\eta)\\ &= \boldsymbol{\phi} - \boldsymbol{\ell}(P_\xi) + \boldsymbol{\ell}(U(y;X)) - \boldsymbol{\ell}(P_\eta).\end{aligned}$$

For similar reasons as the previous case, $\boldsymbol{\lambda}_\xi$ is a feasible solution for $\mathscr{P}_2(H, \mathbf{q}, \mathscr{C}, \mathbf{a}, X)$ and it is clear that $f_{e^+(v,X)}(\boldsymbol{\theta}_\xi) = 0$.

We can show that in either case, each second-stage solution $\boldsymbol{\lambda}^\xi$ is the first solution that differs from $\boldsymbol{\phi}$ in the amount of flow assigned on ξ. Suppose that some solution $\boldsymbol{\lambda}^\psi$ constructed in another iteration corresponding to an edge $\psi = (v', W')$ differs from $\boldsymbol{\phi}$ in the amount of flow on ξ. As previously discussed, each first-stage

solution differs from $\boldsymbol{\phi}$ only on the flow assigned to the outgoing edge of v and to edges of X. Thus, $\boldsymbol{\lambda}^{\psi}$ must be a second- or third-stage solution. If $\boldsymbol{\lambda}^{\psi}$ is a third-stage solution, then ψ is constructed after ξ. If $\boldsymbol{\lambda}^{\psi}$ is a second-stage solution, it is constructed by making three modifications to $\boldsymbol{\phi}$: flow on compath P_{ψ} is decreased, flow on the compath $U(v'; X)$ is increased, and flow is decreased on some path $P_{e'}$ starting on an outgoing edge of v. The compath P_{ψ} only contains edges of $F^{+}(\boldsymbol{\phi})$ that are greater than or equal to ψ. The compath $U(v'; X)$ only contains edges of X. Finally, for similar reasons as in the first stage, every edge of $E(P_{e'})$ other than e' is an edge of X. If e' is in X, then all edges of $P_{e'}$ are edges of X, while if e' is not in X then it is one of the edges whose corresponding solution was constructed in the first stage. Thus, $\boldsymbol{\lambda}^{\psi}$ differs from $\boldsymbol{\phi}$ only in flow assigned to edges that are either greater than ψ or are in X. Thus, ξ must be greater than ψ, which implies that the solution $\boldsymbol{\lambda}^{\psi}$ would be constructed after the solution $\boldsymbol{\lambda}^{\xi}$.

Finally, order the edges of $F^{0}(\phi) \setminus X$ according to some total ordering that is consistent with the natural partial ordering on H. Take each edge ξ of this ordering, starting with the minimum edge. For the same reasons as in the previous theorem, we claim that there is a compath P_{ξ} such that:

1. The compath P_{ξ} contains the edge ξ;

2. For any arc $e \in P_{\xi}$ such that $e \not\preceq \xi$, then $e \in X$;

3. The edges $E(P_{\xi})$ are in $F^{0}(\boldsymbol{\phi}) \cup X$;

4. The compath P_{ξ} starts at an node that has an outgoing edge ψ in $F^{+}(\boldsymbol{\phi})$.

Let P_ψ be a compath starting at ψ in $F^+(\boldsymbol{\phi})$ (such a compath must exist by Lemma 9). Then, we can define a solution

$$\boldsymbol{\theta} := \boldsymbol{\phi} + \boldsymbol{\ell}(P_\xi) - \boldsymbol{\ell}(P_\psi).$$

This solution is a feasible solution for $\mathscr{P}_2(H, \mathbf{q}+\boldsymbol{\ell}(v), \mathscr{C}, \mathbf{a})$ by Lemma 10. Similarly to the construction of the second-stage solutions, we can see that

$$\begin{aligned} f_{e^+(v,X)}(\boldsymbol{\theta}) &\leq f_{e^+(v,X)}(\boldsymbol{\phi}) + 1 \\ &\leq 1, \end{aligned}$$

and just as before, if $f_{e^+(v,X)}(\boldsymbol{\theta}_\xi) = 1$ then we define:

$$\boldsymbol{\lambda}^\xi = \boldsymbol{\theta} - \boldsymbol{\ell}(P_{e^+(v,X)}),$$

while if $f_{e^+(v,X)}(\boldsymbol{\theta}_\xi) = 0$, then there exists some outgoing edge η of v such that $\boldsymbol{\theta}_\xi(\eta) \geq 1$, and we let

$$\begin{aligned} \boldsymbol{\lambda}^\xi &= \boldsymbol{\theta} - \boldsymbol{\ell}(P_\eta) \\ &= \boldsymbol{\phi} + \boldsymbol{\ell}(P_\xi) - \boldsymbol{\ell}(P_\psi) - \boldsymbol{\ell}(P_\eta). \end{aligned}$$

for some compath P_η in $F^+(\boldsymbol{\theta})$ starting with the edge η. Again we can show that in either case, each third-stage solution $\boldsymbol{\lambda}^\xi$ is the first solution that differs from $\boldsymbol{\phi}$ in the amount of flow assigned on ξ. Suppose the solution to some other solution $\boldsymbol{\lambda}^\psi$ corresponding to an edge $\psi = (v', W')$ differs from $\boldsymbol{\phi}$ in the amount of flow on ξ. Since $\boldsymbol{\phi}$ assigns zero flow to the edge ξ, then $\boldsymbol{\lambda}$ must assign higher flow on this edge than $\boldsymbol{\phi}$ does. Again, each first-stage solution differs from $\boldsymbol{\phi}$ only on the flow assigned to the outgoing edge of v and to edges of X. Each second-stage solution

only assigns higher flow than $\boldsymbol{\phi}$ on edges of X, while $\boldsymbol{\lambda}^{\xi}$ assigns higher flow on ξ than $\boldsymbol{\phi}$ does. Thus, $\boldsymbol{\lambda}^{\psi}$ must be a third-stage solution. The solution $\boldsymbol{\lambda}^{\psi}$ only assigns higher flow than $\boldsymbol{\phi}$ along the path P_{ψ} starting at ψ. By definition, every edge of P_{ψ} is either less than ψ or is in X. Thus, ψ is greater than ξ, which implies that $\lambda\psi$ is constructed in a later iteration than $\boldsymbol{\lambda}^{\xi}$. This completes the proof. □

4.9 Conclusion

In this chapter, we examined the polyhedral properties of minimum-cost flow problems on ABF-graphs in which every node has at least one uncapacitated outgoing edge. We showed that this class of problems is NP-hard. For each constraint, we provided conditions under which the constraint is a facet of the convex hull of integral solutions. When these conditions are not satisfied, we proved that the constraint is redundant and provided a set of dominating constraints. We provided a full-dimensional formulation in which these properties were also demonstrated. This work can be immediately applied to the stochastic, dynamic, network flow problems that arise in a variety of fields, ensuring that the formulations for these problems are strong.

There is still more work to be done to improve our understanding of the convex hull of integral solutions for this problem. This would involve identifying valid inequalities that separate fractional solutions. An ideal result would be to completely characterize the convex hull of integral solutions. Improving our knowledge of these polyhedra could lead to better decomposition methods or other solution techniques

for these types of problems. Similar work could also be done for other multistage stochastic problems, and in general more work could be done towards understanding how the structure of multistage stochastic problems relates to their single-stage counterparts.

Chapter 5: Facets in the Dynamic Single Airport Ground Holding Problem

In this chapter, we study integer programming models for assigning delays to flights that are destined for an airport whose capacity has been impacted by poor weather or some other exogenous factor. In the existing literature, empirical evidence seemed to suggest that a proposed integer programming model had a strong formulation, but no existing theoretical results explained the observation. We provide results that explain the strength of the existing model, and we propose a model whose size scales better with the number of flights in the problem and that preserves the strength of the existing model. Finally, we show that both the existing model and the new model satisfy an equity property.

5.1 Background, Contributions and Layout

Poor weather or other factors, such as runway construction, can negatively affect the ability of an airport to accommodate incoming flights. When this occurs, some flights destined for that airport must be held on the ground to prevent excess congestion. The problem of efficiently scheduling flights in a ground delay program was first studied in [60]. The variant of this problem in which no air delays are

allowed was discussed in Section 3.2, while in this chapter we examine the case in which air delays are allowed. The decision to be made is to determine the set of flights that are permitted to depart in each period. In each time period, the airport has a capacity, which is the maximum number of flights that can land in that time period. The capacity is affected by exogenous factors such as weather, and is therefore uncertain. If more flights arrive in a time period than can land, the excess flights are held in the air until they are able to land. The goal is to minimize a expected cost function, which is typically a weighted sum of ground delays and air delays.

There are many proposed solution approaches for the SAGHP, including approaches based on integer programming (IP) [5, 37, 38, 39, 61], simulation-based optimization [62], chance-constrained programming [63], and approximate dynamic programming [64, 65]. Several of these approaches were compared in [66]. The integer programming approaches can be divided into two categories: static [37, 38, 39] and dynamic [5, 61]. Static models assume that assignments are selected at the beginning of the planning period and are never altered, whereas dynamic models recognize that assignments can be modified when new information is acquired. The static models have been shown to have strong polyhedral properties, and in fact, under certain realistic conditions, the linear programming (LP) relaxation of either of these models will always have at least one integral optimal solution [37, 38]. For the model proposed in [39], it has also been shown that under certain conditions, solutions to this model satisfy a type of equity property, known as first-scheduled first-served (FSFS) [37].

The polyhedral properties of the dynamic models are not as well known. In [5], it was observed that for many problem instances, the solution of the LP relaxation was integral. However, this was not the case for all problem instances, and no explanation was provided for the observed strength of the model. It was also shown empirically that the objective function can be modified in order to encourage more equitable solutions, but there was not shown to be an equity property analogous to the one that has been established for the static model. Furthermore, the existing dynamic models do not scale well with the number of flights.

Our contributions in this chapter are as follows:

- We show that an existing dynamic model for the SAGHP is closely related to models whose constraints are all redundant or facet-defining.
- We provide a new dynamic integer programming model that scales better than existing models as the number of flights increases. For this new model, we also provide similar polyhedral results.
- We show that for certain cost functions, both the existing dynamic model and the proposed model satisfy an equity property similar to the one shown in a static model.

Layout. The layout of this chapter is as follows. In Section 5.2, we introduce an existing dynamic model for the SAGHP, and we present closely related models whose constraints are all facet-defining or redundant. In Section 5.3, we propose a new dynamic model for the SAGHP that scales better with the quantity of flights.

We show that this model is equivalent to the existing model, and we provide similar polyhedral results. Computational experiments comparing the performance the new model to the existing model are provided in Section 5.4. In Section 5.5, we show that the optimal solutions of the existing model and the new model satisfy a certain kind of equity property. We discuss a broader class of objective functions in Section 5.6, and we provide a modified version of the new model in that admits these objective functions. Results concerning an equity property are also provided for these objective functions. Finally, in Section 5.7, we provide conclusions and suggestions for future research.

5.2 Mukherjee-Hansen Model

In this section, we introduce the Mukherjee-Hansen (MH) model for the dynamic SAGHP (proposed in [5]), and we provide some results related to the polyhedral strength of closely related models. We have made some slight notational changes so that all variables are indexed by the time at which the corresponding decision is made, and so that this time is the first index of the variable. We have also made some notational changes to improve readability.

5.2.1 Formulation of the Mukherjee-Hansen Model

The Mukherjee-Hansen (MH) model, like many stochastic integer programming models, assumes that the uncertainty is characterized by a finite set of scenarios Θ, which are arranged into a scenario tree. In this case, each scenario q describes

the maximum number of flights M_t^q that can arrive at the airport in time period t. The scenario tree is a rooted tree. The t^{th} level of the tree corresponds to the t^{th} time period, and each node of the tree corresponds to a subset of Θ. The interpretation is that if scenarios q_1 and q_2 fall in the same node n in the t^{th} level of the tree, then it is impossible to tell the difference between these two scenarios at time t. Thus, any decision made in time period t must be the same in these scenarios. In order for the tree to be valid, the nodes in each level of the tree must form a partition of Ω, and each level of the tree must be a refinement of the higher level. In the MH model, the time period is divided into T discrete time periods. The parameters of the model are as follows:

Φ - set of flights;

Θ - set of scenarios;

$\mathscr{N}$ - set of all nodes of the scenario tree;

$\mathscr{N}_t$ - set of nodes of the t^{th} level of the scenario tree;

$W^{\max}$ - maximum number of flights allowed to take air delay simultaneously;

$p(q)$ - probability that scenario q occurs;

M_t^q - maximum number of flights that can land at the airport in time period t in scenario q;

c_a - cost of holding one flight in the air for one time period;

c_g - cost of holding one flight on the ground for one time period.

Typically, the cost c_a would be greater than the cost c_g because taking delay in the air is more expensive than taking delay on the ground. Each flight f has

a scheduled departure time $\tau(f)$ and a flight duration $\delta(f)$. The departure of the flight f cannot be rescheduled to any time prior to its scheduled departure $\tau(f)$, but can be delayed to a later time. The flight duration determines the amount of time required for the flight to reach the airport once it departs, so that if flight f departs at time t then it will reach the airport at the time $t + \delta(f)$. At this point, if the airport has available capacity, the flight will land. If there is no available capacity, then the flight will be held in the air and will land in a later time period. Naturally, the decision of whether the flight will depart in time period t must be made no later than the time period t. Thus, this decision must be the same in any scenarios that cannot be distinguished at time period t. We will also use the notation that Φ_t^{dep} is the set of flights that are able to depart by time t,

$$\Phi_t^{\text{dep}} := \{f \in \Phi : \tau(f) \leq t\},$$

and Φ_t^{arr} is the set of flights that are able to arrive by time t,

$$\Phi_t^{\text{arr}} := \{f \in \Phi : \tau(f) + \delta(f) \leq t\}.$$

The primary decision variables decide the time at which each flight will depart in each scenario:

$$y_{t,f}^q := \begin{cases} 1 & \text{if flight } f \text{ departs at time } t \text{ in scenario } q, \\ 0 & \text{otherwise.} \end{cases}$$

There are also auxiliary decision variables that determine the amount of air delay taken:

$$w_t^q := \text{the number of flights held in the air in time period } t.$$

The objective of the model is to minimize a weighted combination of expected ground delays and air delays:

$$\min \sum_{q\in\Theta} p(q) \left(\sum_{f\in\Phi} \sum_{t=\tau(f)}^{T+1} c_g(t-\tau(f)) y^q_{t,f} + \sum_{t=1}^{T} c_a w^q_t \right)$$

The constraints are as follows:

$$\sum_{t=\tau(f)}^{T+1} y^q_{t,f} = 1 \qquad \forall f \in \Phi, q \in \Theta, \tag{5.1}$$

$$w^q_{t-1} - w^q_t + \sum_{f\in\Phi^{\text{arr}}_t} y^q_{t-\delta(f),f} \leq M^q_t \qquad \forall t \in \{1,...,T\}, q \in \Theta, \tag{5.2}$$

$$w^q_t \leq W^{\max} \qquad \forall q \in \Theta, t \in \{0,...,T\}, \tag{5.3}$$

$$w^q_0 = w^q_T = 0 \tag{5.4}$$

$$y^q_{t,f} = y^r_{t,f} \qquad \forall t \in \{1,...,T+1\}, f \in \Phi, n \in N_t,$$
$$\text{pairs } (q,r) \text{ of scenarios in } n, \tag{5.5}$$

$$y^q_{t,f} \in \{0,1\},$$

$$w^q_t \in \mathbb{Z}^+_0.$$

Constraint (5.1) ensures that every flight is assigned a departure time in every scenario. Assigning a flight to the time period $T+1$ indicates that the flight will not receive a departure time within the planning horizon. Constraint (5.2) enforces the capacity of the airport, and constraint (5.3) enforces the maximum number of flights simultaneously receiving air delay. Constraint (5.4) takes care of some edge cases. The constraints 5.5, known as anti-anticipatory constraints, ensures that the decisions are based only on the available information. Throughout this chapter, we will use the notation that if $\boldsymbol{\lambda}$ is a solution to the MH model, then $v(\boldsymbol{\lambda})$ is the value

of the the variable v in $\boldsymbol{\lambda}$, e.g. $w_t^q(\boldsymbol{\lambda})$ is the value of w_t^q in $\boldsymbol{\lambda}$.

When a ground delay program is initiated, some flights destined to land at the affected airport might already have departed but not yet arrived. In current practice, these flights would be exempted from the ground delay program. Exempt flights would not be assigned delays in the ground delay program, and would instead be allowed to proceed along their current planned trajectory. Once an exempt flight reaches the terminal airspace, it is treated in the same fashion as flights that were controlled in the ground delay program are treated. If there is available capacity, the flight is allowed to land. Otherwise, the flight is held in the air. These exempt flights can be incorporated into the MH with only a minor alteration. Let E_t be the number of exempt flights expected to reach the destination airport at time period t. Then, we replace the set of constraints 5.2 with the constraints

$$E_t + w_{t-1}^q - w_t^q + \sum_{f \in \Phi_t^{\text{arr}}} y_{t-\delta(f),f}^q \leq M_t^q \qquad \forall t \in \{1, ..., T\}, q \in \Theta.$$

5.2.2 Mukherjee-Hansen with Strict Flow Conservation

Suppose that there were no limit on the number of flights taking air delay. In this case, we can provide a nearly equivalent model in which every constraint is either redundant or is a facet of the convex hull of integral solutions. One difference is that feasible solutions of the MH model do not necessarily conserve the number of airborne flights, while the variant discussed here does. More specifically, it is possible in the MH model that there are more flights held in the airspace near the airport in some time period than are present at that time. Since the objective

solution penalizes air holding, such a situation will not occur in any optimal solution. Our reformulation enforces the conservation constraint in a stricter fashion. The only other change made in our reformulation is in the implementation of the anti-anticipatory constraints. In the new model, a single variable is used in the place of each group of variables that would be constrained to take the same value by the anti-anticipatory constraint. The parameters of this model are nearly the same as in the MH model, although we introduce some additional notation. We let $p(n)$ to be the sum of probabilities of scenarios in the node n of the scenario tree, i.e.:

$$p(n) := \sum_{q \in n} p(q).$$

For any time s, any node n in $\mathscr{N}_s$ and for any $t \leq s$, we let $a(t, n)$ be the node of N_t such that $n \subseteq a(t, n)$. That is, $a(t, n)$ is the set of scenarios that cannot be distinguished from any of the scenarios in node n at time period t. We will refer to $a(t, n)$ as the ***ancestor*** of node n in time period t. For a scenario q, we define $n(t, q)$ to be the node of $\mathscr{N}_t$ that contains the scenario q. Note that for any $t' < t$ and scenario node n, $a(t', a(t, n)) = a(t', n)$. That is, an ancestor of an ancestor of n is also an ancestor of n. For a node n in $\mathscr{N}$, then let $t(n)$ be the corresponding time period, so that $t(n) = s$ if and only if n is a node of $\mathscr{N}_s$.

The decision variables of the reformulation include variables similar to those

in the MH model:

$$y^n_{t,f} := \begin{cases} 1 & \text{if flight } f \text{ departs at time } t \text{ in all scenarios of node } n, \\ 0 & \text{otherwise;} \end{cases}$$

$w^n_t :=$ number of flights taking air delay at time t in all scenarios of node n;

and we introduce the following new variables:

$$g^n_{t,f} := \begin{cases} 1 & \text{if flight } f \text{ takes ground delay in time } t \text{ in all scenarios of node } n, \\ 0 & \text{otherwise;} \end{cases}$$

$l^n_t :=$ number of flights landing at time t in all scenarios of node n.

Then, the model, which we will refer to as the MH-Flow model is given by:

$$\min \sum_{n \in \mathscr{N}} p(n) \left[c_g \left(\sum_{f \in \Phi^{\text{dep}}_{t(n)}} (t(n) - \tau(f)) y^n_{t(n),f} \right) + c_a w^n_{t(n)} \right]$$

$$y^n_{\tau(f),f} + g^n_{\tau(f),f} = 1 \qquad \forall f \in \Phi, n \in \mathscr{N}_{\tau(f)}, \tag{5.6}$$

$$y^n_{t,f} + g^n_{t,f} = g^{a(t,n)}_{t-1,f} \qquad \forall f \in \Phi, t \in \{\tau(f)+1, ..., T\}, n \in \mathscr{N}_t, \tag{5.7}$$

$$y^n_{T+1,f} = g^{a(T,n)}_{T,f} \qquad \forall f \in \Phi, n \in \mathscr{N}_{T+1}, \tag{5.8}$$

$$E_t + w^{a(t-1,n)}_{t-1} + \sum_{f \in \Phi^{\text{arr}}_t} y^{a(t-\delta(f),n)}_{t-\delta(f),f} = l^n_t + w^n_t \quad \forall t \in \{1, ..., T\}, n \in \mathscr{N}_t, \tag{5.9}$$

$$l^n_t \leq M^n_t \qquad \forall t \in \{1, ..., T\}, n \in \mathscr{N}_t, \tag{5.10}$$

$$w^n_0 = 0,$$

$$y^n_{t,f}, g^n_{t,f} \in \{0, 1\},$$

$$w^n_t, l^n_t \in \mathbb{Z}^+_0.$$

Constraints 5.6 - 5.8 of MH-Flow are essentially equivalent to constraint 5.1 of MH. To see this, note that for any solution $\boldsymbol{\lambda}$ that satisfies the constraints 5.6 - 5.8 it is true that

$$\sum_{t=\tau(f)}^{T+1} y_{\tau(f),f}^{a(t,n)}(\boldsymbol{\lambda}) = g_{T,f}^{a(T,n)}(\boldsymbol{\lambda}) + 1 - g_{\tau(f),f}^{a(\tau(f),n)}(\boldsymbol{\lambda}) + \sum_{s=\tau(f)+1}^{T-1} g_{t,f}^{a(t,n)}(\boldsymbol{\lambda}) - g_{t+1,f}^{a(t+1,n)}(\boldsymbol{\lambda}),$$

$$= 1.$$

for all flights f and scenario nodes n in $\mathscr{N}_{T+1}$. Conversely, let $\boldsymbol{\psi}$ be a solution to the MH problem that satisfies

$$\sum_{t=\tau(f)}^{T+1} y_{\tau(f),f}^{q}(\boldsymbol{\psi}) = 1.$$

for all flights f, time periods t, and scenario nodes t in $\mathscr{N}_t$. Then there is a solution $\boldsymbol{\lambda}$ to the MH-Flow model such that $\boldsymbol{\lambda}$ satisfies constraints $5.6 - 5.8$ and such that

$$y_{t,f}^{n(t,q)}(\boldsymbol{\lambda}) = y_{t,f}^{q}(\boldsymbol{\psi}).$$

for each flight f, time periods t, and scenario node n in $\mathscr{N}_t$. In particular, $\boldsymbol{\lambda}$ is defined by

$$g_{t,f}^{n}(\boldsymbol{\lambda}) = 1 - \sum_{s=\tau(f)}^{t} y_{s,f}^{q}(\boldsymbol{\psi})$$

for each flight f, time period t, and scenario node n. Constraints 5.9 and 5.10 enforce a stricter version of 5.2 that does not allow excess air holding.

It follows immediately from the results in Section 4.8 that each inequality constraint of the MH-Flow model defines a facet of the convex hull of integer solutions or is redundant, as it is straightforward to show that this is an ABF-graph flow problem in which all nodes have at least one uncapacitated edge. The MH-Flow model

does have more variables and constraints than the MH model. However, the results in Section 4.8 also provide a full-dimensional version of the model with less variables whose constraints are still either facet-defining or redundant. This formulation is provided in Section C.1 of Appendix C.

5.2.3 Mukherjee-Hansen Model with Diversions

Once maximum air delay constraints are imposed to the MH-Flow model, the results from Section 4.8 are no longer applicable. Here, we propose a second variant of the MH that includes maximum air delay constraints and such that the results from 4.8 can be applied. In practice, when there are large amounts of congestion at an airport it is possible to divert some of the incoming flights to another airport. We introduce the variables

$$d_t^n := \text{ number of flights diverted from airport in time period } t,$$

we replace constraint 5.9 of the MH-Flow model with the constraint

$$E_t + w_{t-1}^n + \sum_{f \in \Phi_t^{\text{arr}}} y_{t-\delta(f),f}^{a(t-\delta(f),n)} = l_t^n + w_t^n + d_t^n \qquad \forall t \in \{1, ..., T+1\}, n \in N_t,$$

and we reintroduce the maximum air delay constraints

$$w_t^n \leq W^{\max} \qquad \forall t \in \{1, ..., T+1\}, n \in N_t.$$

The objective of the problem is then

$$\min \sum_{n \in \mathcal{N}} p(n) \left[c_g \left(\sum_{f \in \Phi_{t(n)}^{\text{dep}}} (t(n) - \tau(f)) y_{t(n),f}^n \right) + c_a w_{t(n)}^n + c_d d_{t(n)}^n \right]$$

where c_d is the cost of making a diversion. We will refer to this model as the MH-Diversion model. Similar to before, the results in Section 4.8 ensure that every inequality constraint of this model is either a facet of the convex hull of integral solutions or is redundant. Also as before, the results of Section 4.8 provide a full-dimensional version of the MH-Diversion model, all of whose constraints are also either facet-defining or redundant.

5.3 Dynamic Hoffkin Model

It is possible to greatly reduce the number of variables in the MH model and its variants while maintaining the strength of the formulation. Suppose there are multiple flights that have the same flight duration, are on the ground in time period t, and are able to depart within that time period t. For the purpose of the IP models above, these flights are functionally identical. For this reason, it is possible to aggregate these flights. The decision variables then determine how many flights receive delays in each time periods rather than determining exactly which flights receive delays.

$x^n_{t,m}$:= the number of flights with flight duration m that take ground delay in time period t under the scenarios of node n;

$z^n_{t,m}$:= the number of flights with flight duration m that depart in time period t under the scenarios of node n.

Let $D_{t,m}$ be the set of flights whose earliest departure time is t and whose flight duration is m,

$$D_{t,m} := \{f \in \Phi : \tau(f) = t, \delta(f) = m\},$$

and let Δ be the set of all flight durations,

$$\Delta := \{\delta(f) : f \in \Phi\}.$$

Then, we define the D-Hoffkin model as follows:

$$\min \sum_{n \in \mathcal{N}} p(n) \left[c_g \left(\sum_{m \in \Delta} x^n_{t(n),m} \right) + c_a w^n_{t(n)} \right]$$

subject to:

$$x^n_{1,m} + z^n_{1,m} = |D_{1,m}| \quad \forall m \in \Delta, n \in N_1, \tag{5.11}$$

$$z^n_{t,m} + x^n_{t,m} - x^{a(t-1,n)}_{t-1,m} = |D_{t,m}| \quad \forall m \in \Delta, t \in \{2, ..., T\}, n \in N_t, \tag{5.12}$$

$$E_t + w^{a(t-1,n)}_{t-1} - w^n_t + \sum_{\substack{m \in \Delta: \\ m < t}} z^{a(t-m,m)}_{t-m,m} \leq M^q_t \quad \forall t \in \{1, ..., T\}, n \in \mathcal{N}_t, \tag{5.13}$$

$$w^n_t \leq W^{\max} \quad \forall t \in \{1, ..., T\}, n \in N_t, \tag{5.14}$$

$$w^n_0 = 0,$$

$$w^n_t, x^n_{t,m}, z^n_{t,m} \in \mathbb{Z}^+_0.$$

Constraints 5.11 and 5.12 of the D-Hoffkin model serve the same purpose that the constraint 5.1 serves in the MH model and that 5.6 - 5.8 serve in the MH-Flow model. These constraints ensure that all flights on the ground in each time period either receive delay or depart. Constraint 5.13 enforces the capacity of the airport in a similar way that constraint 5.2 enforces the capacity of the airport in the MH model. The term

$$\sum_{\substack{m \in \Delta: \\ m < t}} z^{a(t-m,n)}_{t-m,m}$$

in the D-Hoffkin model and the term

$$\sum_{f \in \Phi_t^{\text{arr}}} y^q_{t-\delta(f),f}$$

in the MH model both represent total number of flights that would arrive at the airport in time period t.

In general, we expect the size of the D-Hoffkin model to scale better with the number of flights in the problem. The MH model has $O(T|\Phi||\Theta|)$ variables and $O((T + |\Phi|)|\Theta|)$ constraints, while the D-Hoffkin model has $O(T|\Delta||\Theta|)$ variables and $O(T|\Delta||\Theta|)$ constraints. If the flight durations of two flights are similar, then these flights may be treated as if their flight duration is the same. One possible way of grouping the durations is to divide the range of possible flight durations into equal-length bins, for example 10-minute bins. Following a scheme such as this, the number of distinct flight durations does not change much as the number of flights increase, so for large instances of this problem $|\Phi|$ is much greater than $|\Delta|$. Once every flight duration bin has at least one flight whose duration falls into that bin, adding additional flights does not increase the number of variables or constraints in the D-Hoffkin model. Such an addition would, on the other hand, increase the number of variables and constraints in the MH model. Thus, for large instances of the problem, the D-Hoffkin model will have fewer variables and constraints than the MH model. There are some cases in which the MH model could have fewer constraints than the D-Hoffkin model. In particular, this can occur if there are many time periods and the number of flights does not greatly exceed the number of flight duration categories. However, for a wide range of problems, we expect that

the D-Hoffkin model would be more computational efficient than the MH model. Computational experiments confirming this intuition are provided in Section 5.4.

5.3.1 Model equivalence

The D-Hoffkin model is equivalent to the MH model. Given a feasible solution to the MH model, it is possible to create a corresponding solution to the D-Hoffkin model by aggregating the flights' assigned departure times. For example, if the solution to the MH model allows k flights with flight duration of m to depart at time t under the scenarios of node n, then the variable $z^n_{t,m}$ would take the value k in the corresponding solution to the D-Hoffkin model. This process can be reversed to construct a feasible solution to the MH model from a solution to the D-Hoffkin model. If k flights with flight duration of m are allowed to depart at time t under the scenarios of node n in some solution to the D-Hoffkin model, then it is possible to select k flights with duration m and to schedule these flights' departure to time period t in the scenarios of node n.

More rigorously, let $\mathscr{P}_{MH}$ be the space of feasible solutions to the MH model, and let $\mathscr{P}_{DH}$ be the space of feasible solutions to the D-Hoffkin model. For a node n of the scenario tree, let $q(n)$ be some scenario of n; the manner of selection is not important for our purposes. Conversely, for a scenario q in Θ, let $n(t, q)$ be the corresponding node of the scenario tree in time period t. Let us define a mapping

$$\gamma_1 : \mathscr{P}_{MH} \to \mathscr{P}_{DH}$$

by

$$z^n_{t,m}(\gamma_1(\boldsymbol{\lambda})) = \sum_{s=1}^{t} \sum_{f \in D_{s,m}} y^{q(n)}_{t,f}(\boldsymbol{\lambda}) \qquad \forall m \in \Delta, t \in \{1, ..., T\}, n \in \mathscr{N}_t;$$

$$x^n_{t,m}(\gamma_1(\boldsymbol{\lambda})) = \sum_{\sigma=1}^{t} \left(|D_{\sigma,m}| - \sum_{s=1}^{\sigma} \sum_{f \in D_{s,m}} y^{q(a(\sigma,n))}_{\sigma,f}(\boldsymbol{\lambda}) \right) \qquad \forall m \in \Delta, t \in \{1, ..., T\}, n \in \mathscr{N}_t;$$

$$w^n_t(\gamma_1(\boldsymbol{\lambda})) = w^{q(n)}_t(\boldsymbol{\lambda}) \qquad t \in \{1, ..., T\}, n \in \mathscr{N}_t.$$

This mapping is well-defined, which is to say that the resulting output is always feasible. Furthermore, this mapping preserves costs, so that each solution in $\mathscr{P}_{MH}$ is mapped to a solution $\mathscr{P}_{DH}$ with the same objective value.

Proposition 2. *The function γ_1 is a well-defined function from $\mathscr{P}_{MH}$ to $\mathscr{P}_{DH}$, and $C(\boldsymbol{\lambda}) = C(\gamma_1(\boldsymbol{\lambda}))$*

In the other direction, we can define a mapping $\gamma_2 : \mathscr{P}_{DH} \to \mathscr{P}_{MH}$ by

$$y^q_{t,f}(\gamma_2(\boldsymbol{\lambda})) = Y[t, f, q] \qquad \forall t \in \{1, ..., T\}, f \in \Phi, q \in \Theta;$$

$$y^q_{T+1,f}(\gamma_2(\boldsymbol{\lambda})) = \sum_{t=\tau(f)}^{T} Y[t, f, q] \qquad \forall f \in \Phi, q \in \Theta;$$

$$w^q_t(\gamma_2(\boldsymbol{\lambda})) = w^{n(t,q)}_t(\boldsymbol{\lambda}) \qquad \forall t \in \{1, ..., T\}, q \in \Theta;$$

where $\mathbf{Y}$ is the output of Algorithm 3.

Note that Step 7 of this algorithm is not uniquely defined, as there may be multiple ways of selecting the specified flights. Different implementations of this step can result in different corresponding solutions. Thus, while each solution to the MH model has a single corresponding solution to the D-Hoffkin model, the D-Hoffkin model may have multiple corresponding solutions to the MH model. However, once

Algorithm 3: Procedure for constructing a feasible solution to $\mathscr{P}_{MH}$ from a feasible solution in $\mathscr{P}_{DH}$.

1: Inputs: $\boldsymbol{\lambda}$, a feasible solution in $\mathscr{P}_{MH}$.
2: For each f in Φ, q in Θ and $t \in \{\tau(f), ..., T+1\}$ initialize $Y[t, f, q] = 0$.
3: **for** each duration m in Δ **do**
4: **for** each time t in $\{1, .., T\}$ **do**
5: **for** each node n in $\mathscr{N}_t$ **do**
6: Let $k = z^n_{t(n),m}(\boldsymbol{\lambda})$.
7: Choose some set $H^{q(n)}_{t,m}$ of k flights from $(\cup^t_{s=1} D_{s,m}) \setminus \left(\cup^{t-1}_{s=1} H^{q(n)}_{s,m}\right)$.
8: **for** each flight f in $H^{q(n)}_{t,m}$ **do**
9: Set $Y[t, f, q] = 1$ for all q in n.
10: **end for**
11: **end for**
12: **end for**
13: **end for**
14: return $\mathbf{Y}$.

a manner of selecting these flights has been chosen, then the function γ_2 is well-defined. This function also preserves the objective value of the solution.

Proposition 3. *Given any method for selecting flights in Step 7 of Algorithm 3, γ_2 is a function from $\mathscr{P}_{DH}$ to $\mathscr{P}_{MH}$, and $C(\boldsymbol{\lambda}) = C(\gamma_2(\boldsymbol{\lambda}))$ for any $\boldsymbol{\lambda}$ in $\mathscr{P}_{MH}$.*

Proofs of Propositions 2 and 3 are given in Sections D.1 and D.2 respectively of Appendix D.

5.3.2 D-Hoffkin Variants

In a similar fashion to the MH model, we can show that there are closely related models whose constraints are all facet-defining or redundant. The D-Hoffkin model also does not strictly enforce conservation flow at the airport. We define a variant of the D-Hoffkin model in which this constraint is enforced in a stricter fashion, which we call the D-Hoffkin-Flow model. The D-Hoffkin-Flow model has

the same objective as the D-Hoffkin model, and has constraints defined by:

$$
\begin{aligned}
x^n_{1,m} + z^n_{1,m} &= |D_{1,m}| && \forall m \in \Delta, n \in N_1, \\
x^n_{t,m} + z^n_{t,m} - x^{a(t-1,n)}_{t-1,m} &= |D_{t,m}| && \forall m \in \Delta, t \in \{2, ..., T\}, n \in N_t, \\
E_t + w^{a(t-1,n)}_{t-1} + \sum_{\substack{m \in \Delta: \\ m < t}} z^{a(t-m,n)}_{t-m,m} &= l^n_t + w^n_t && \forall t \in \{1, ..., T\}, n \in \mathscr{N}_t, \qquad (5.15) \\
l^n_t &\leq M^n_t && \forall t \in \{1, ..., T\}, n \in \mathscr{N}_t, \\
w^n_0 &= 0, \\
w^n_t, l^n_t, x^n_{t,m}, z^n_{t,m} &\in \mathbb{Z}^+_0.
\end{aligned}
$$

Similarly, we also propose a variant called the D-Hoffkin-Diversion model in which the constraints 5.15 are replaced by

$$
E_t + w^{a(t-1,n)}_{t-1} + \sum_{\substack{m \in \Delta: \\ m < t}} z^n_{t-m,m} = l^n_t + w^n_t + d^n_t \qquad \forall t \in \{1, ..., T\}, n \in \mathscr{N}_t,
$$

and in which we reintroduce the constraints

$$
w^n_t \leq W^{\max} \qquad \forall t \in \{1, ..., T\}, n \in N_t.
$$

As before, the results of Section 4.8 can be applied to these models to prove that all of the inequality constraints are either facet-defining or redundant, and full-dimensional formulations for the D-Hoffkin-Flow and D-Hoffkin-Diversion model are provided in Sections C.3 and C.4 respectively of Appendix C. It is possible to show that the MH-Flow model is equivalent to the D-Hoffkin-Flow model, and that the MH-Diversion is equivalent D-Hoffkin-Diversion model. These proofs are similar to the proof of equivalence between the MH model and the D-Hoffkin model.

5.4 Computational Experiments

The setup for our computational experiments is similar to that used in [5]. The main differences between our methodology and that of the existing work are that we obtained flight schedules from a different data source, and we varied the length of the time periods in the division of the planning horizon.

We selected a single representative day, specifically July 15th, 2017. We acquired the scheduled flights for this day for six airports, specifically Hartsfield-Jackson Atlanta International Airport (ATL), O'Hare International Airport (ORD), Dallas/Fort Worth International Airport (DFW), LaGuardia Airport (LGA), San Francisco International Airport (SFO), and Ronald Reagan Washington National Airport (DCA). These schedules were taken from the database of airline on-time performance data maintained by the Bureau of Transportation Statistics (`www.transtats.bts.gov`). Flights whose departure times occurred before the initiation of the GDP and that arrived within the planning horizon were treated as exempt. For each of these airports, we selected ten time horizons in which to run a hypothetical GDP. Five of these horizons begin at 7:00 a.m. while the remaining five begin at 5:00 p.m. In either of these cases, the duration of the five planning horizons varies between five, six, seven, eight and nine hours. Each possible time horizon at each airport was divided in four different ways. Specifically, we divided each possible time horizon into equally-spaced two-minute, five-minute, ten-minute, and fifteen-minute intervals. To summarize, there are six airports, each with ten possible planning horizons, each of which is divided in four different ways, resulting in a total of 240

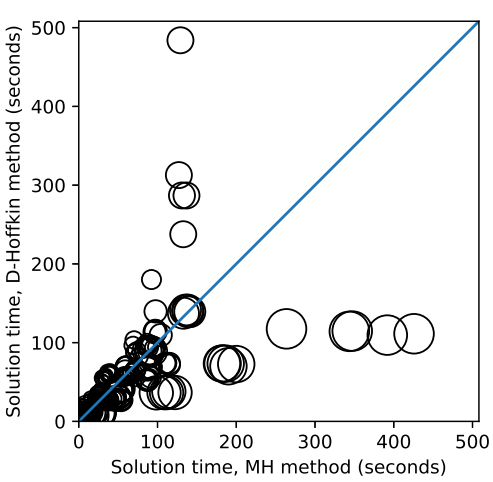
different cases. In each of these cases, we ran six different trials with varying choices of parameters, so that 1440 trials were run.

We will refer to one of these choices of parameters as the "baseline", and it is constructed as follows. The capacity scenarios were formed under the assumption that the capacity would be low at the beginning of the time horizon, but would increase at some time period. The capacity will always increase before the final three hours of the planning horizon. For example, if the planning horizon is 5 hours then the latest possible time at which the capacity could increase is 2 hours from the start of the GDP. The capacity has an equal probability of increasing during each time period from the initiation of the GDP to the aforementioned latest possible time, and each of these time periods has a corresponding capacity scenario. Branches form in the scenario tree exactly when the capacities of scenarios differ. That is, if the airport experiences the same capacities under scenario ω_1 as it does under ω_2 up until (and including) time period t, then these scenarios would share a scenario node at time period t. Otherwise, these scenarios would be placed in different nodes at that time. Capacity values were taken from [67]. For the lower initial capacity values, we used the estimated admissible arrivals per hour under instrument weather conditions. For the increased values, we used the estimated admissible arrivals per hour under visual weather conditions. At ATL, there are two possible configurations with corresponding capacities. For these experiments, we used the values reported for the "arrival priority" configuration. The cost of ground delay was set to 1 per time unit, while the cost of air delay was set to 3 per time unit, and the maximum number of flights allowed to take air holding was set to the difference between the

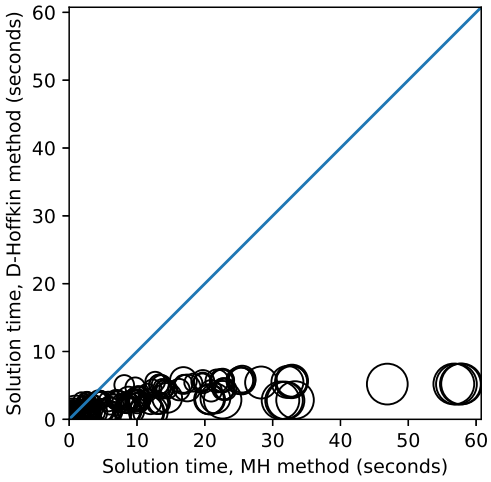
increased capacity and the initial capacity.

The remaining five choices of parameters are formed by altering the baseline. Note that alterations are independent and not cumulative, so that the alterations used to describe the second choice of parameters do not apply for the third choice, etc. Under the second choice of parameters, the constraints on the maximum number of flights is removed. In the third choice, the cost of air delay is set to 2 per time unit. In the fourth choice of parameters, the probabilities of the scenarios are altered so that there is a 60% chance that the capacity will improve within the first hour of the planning horizon. The scenarios in which the capacity improves within the first hour are all given equal probability, and the remaining scenarios are assigned equal probability. For example, suppose that the planning horizon is six hours, and that this horizon is divided into fifteen-minute time periods. Then, the first hour has four time periods, and four corresponding scenarios. The combined probability of these scenarios is 60%, so each of these scenario must have a 15% chance of occurring. There are eight more scenarios corresponding to time periods in the second and third hours of the planning horizon. Since the combined probability of these scenarios is 40%, then each of these scenarios must have a 5% chance of occurring. In the fifth choice of parameters, no airborne holding is allowed, i.e. $W^{\max}$ is set to zero. For the final choice of parameters, information on the increased capacity arrives thirty minutes before the change actually occurs. This affects the scenario tree. Suppose that the capacity of the airport is different in time period t under scenario ω_1 than under ω_2. Then thirty minutes before the time period t, these scenarios would be placed in a different scenario node.

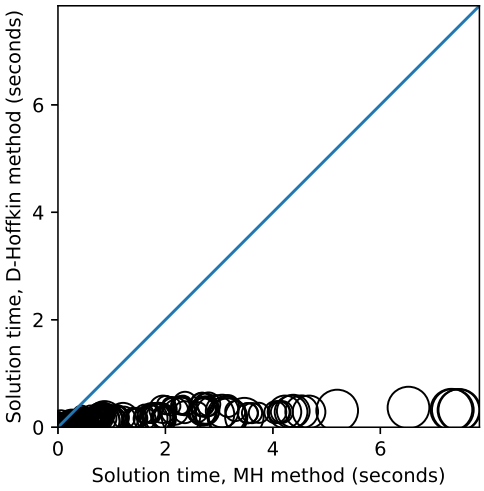
These instances were solved using the Gurobi commercial IP solver (version 5) on a desktop computer with an Intel Core i7-4790 dual-core CPU (3.6 GHz) and with 8 GB of RAM. The default settings were used for the solver. The amount of time required to solve each model in each instance is plotted in Figure 5.1. Figs. 5.1(a), 5.1(b), 5.1(c), 5.1(d) display the results for the instances with two-minute, five-minute, ten-minute, and fifteen-minute time periods respectively. Each point of the scatter plot provides the solution times required by each model to solve a single instance. The size of the point is proportional to the number of non-exempt flights within the planning period. The points falling to the right of the diagonal line correspond to instances in which the D-Hoffkin required less computational time than the MH model, while the opposite holds for the points falling to the left of the diagonal line. The D-Hoffkin has clearly faster performance on instances where the time periods are five, ten or fifteen minutes long. The MH model often requires much longer solution times. There are a few of these instances for which the solution time of the D-Hoffkin model is longer than that the MH model, but even in these instances the D-Hoffkin model requires only slightly more computational time. When two-minute time periods are used, the benefits of the D-Hoffkin model are not as pronounced. For the instances with the largest quantities of flights, the D-Hoffkin model still results in considerably faster solution times than the MH model. However, in instances with moderate quantities of flights, the MH model can provide considerably faster solution time. Overall, while the MH model does result in faster solution times in a few specific circumstances, the proposed model is the more efficient model for a wide range of instances and choices of parameters.



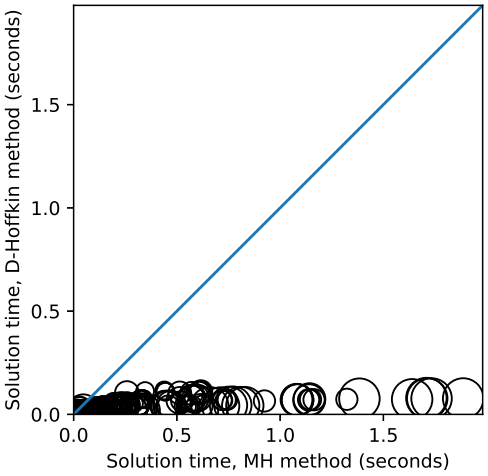
(a) Two-minute time periods



(b) Five-minute time periods



(c) Ten-minute time periods



(d) Fifteen-minute time periods

Figure 5.1: Solution times of D-Hoffkin method vs. MH method.

5.5 Equitable Assignments

There are existing results that provide conditions under which a static model for the SAGHP demonstrates a type of equity property, called the first-scheduled-first-served (FSFS) rule [37]. This rule states that if flight f_1 has a later scheduled arrival time than f_2, then f_1 should not have an earlier assigned arrival time than f_2 does. We can show that the MH and D-Hoffkin model exhibit a slightly weaker property, namely that if two flights have the same flight duration, then the flight with earlier scheduled time is allowed to depart before a flight with the later scheduled departure. We call this the First-Scheduled First-Served within Duration (FSFS-D) property. For solutions of the MH model, this can be defined as follows.

Definition 6 (FSFS-D)**.** *A feasible solution* $\boldsymbol{\lambda}$ *to the MH model is* ***FSFS-D*** *if the following is satisfied: For any time period* t*, for any flight* f*, and for any scenario* q *such that*

$$y^q_{t,f}(\boldsymbol{\lambda}) = 1$$

then

$$y^q_{t',f'}(\boldsymbol{\lambda}) = 0$$

for all $t' < t$ *and for any flight* f' *such that* $\delta(f') = \delta(f)$ *and* $\tau(f') > \tau(f)$*.*

Intuitively, this states that if some flight f departs in a time period t then there should not be any flights f' that have a later scheduled departure and an earlier assigned departure time. Such a property is more difficult to define for the

D-Hoffkin model, since the D-Hoffkin model does not directly assign a departure to each flight but instead determines the quantity of flights permitted to depart in each time period. However, it is possible to take a feasible solution to the D-Hoffkin model and produce a set of individual flight assignments that agree with the designated quantities of departures. This was demonstrated in Section 5.3.1 when we showed that there is a mapping from solutions to the D-Hoffkin model to solutions of the MH model (Proposition 3). We can further provide such a mapping where every feasible solution to the D-Hoffkin model corresponds to an FSFS-D feasible solution to the Hoffkin model. This indicates that solutions to the D-Hoffkin model can be implemented in a manner that is consistent with the FSFS-D property.

Let us define a mapping $\gamma_2^{\text{RBS}} : \mathscr{P}_{DH} \rightarrow \mathscr{P}_{MH}$ as follows:

$$y^q_{t,f}(\gamma_2^{\text{RBS}}(\boldsymbol{\lambda})) = Y[t,f,q] \qquad \forall t \in \{1,...,T\}, f \in \Phi, q \in \Theta;$$

$$y^q_{T+1,f}(\gamma_2^{\text{RBS}}(\boldsymbol{\lambda})) = \sum_{t=\tau(f)}^{T} Y[t,f,q] \qquad \forall f \in \Phi, q \in \Theta;$$

$$w^q_t(\gamma_2^{\text{RBS}}(\boldsymbol{\lambda})) = w_t^{n(t,q)}(\boldsymbol{\lambda}) \qquad \forall t \in \{1,...,T\}, q \in \Theta;$$

where $\mathbf{Y}$ is the output of Algorithm 4.

Note that this is a special case of the definition given in Section 5.3.1; while Algorithm 3 selects some flights arbitrarily in each iteration, Algorithm 4 selects those flights with the earliest scheduled departure time. In each iteration, this selection is made from a set of flights that all have the same flight duration. Thus, we could equivalently state that the algorithm selects those flights with the earliest scheduled arrival time. This is analogous to the Ration-By-Schedule (RBS) procedure used in practice, which assigns arrival times to flights in order of their scheduled arrival

Algorithm 4: Procedure for constructing a feasible solution to $\mathscr{P}_{MH}$ from a feasible solution in $\mathscr{P}^*_{GDH}$.

Inputs: $\boldsymbol{\lambda}$, a feasible, FSFS-D solution in $\mathscr{P}^*_{GDH}$.
For each f in Φ, q in Θ and $t \in \{\tau(f), ..., T+1\}$ initialize $Y[t, f, q] = 0$.
for each duration m in Δ **do**
  Order the flights of Φ^{dur}_m in any total ordering such that $f < f'$ whenever $\tau(f) < \tau(f')$.
  for each time t in $\{1, .., T\}$ **do**
    for each node n in $\mathscr{N}_t$ **do**
      Let $k = z^n_{t,m}(\boldsymbol{\lambda})$.
      Let $H^n_{t,m}$ be the first k flights from $(\cup^t_{s=1} D_{s,m}) \setminus \left(\cup^{t-1}_{s=1} H^n_{t,m}\right)$.
      for each flight f in $H^n_{s,m}$ **do**
        Set $Y[t, f, q] = 1$ for all q in n.
      end for
    end for
  end for
end for
return $\mathbf{Y}$.

times.

Proposition 4. *For any solution $\boldsymbol{\lambda}$ in $\mathscr{P}_{DH}$, $\gamma_2^{RBS}(\boldsymbol{\lambda})$ is an FSFS-D solution in $\mathscr{P}_{MH}$ and $C(\gamma_2^{RBS}(\boldsymbol{\lambda})) = C(\boldsymbol{\lambda})$.*

The proof of Proposition 4 is given Section D.3 of Appendix D. This implies that for any feasible solution $\boldsymbol{\lambda}$ to the MH model, there is an FSFS-D solution that achieves the same objective value; the solution $\gamma_2^{\text{RBS}}(\gamma_1(\boldsymbol{\lambda}))$ is one such solution. This also implies that is always an optimal solution to the MH model that satisfies the FSFS-D property. Such a solution can be constructed by applying the mapping γ_2^{RBS} to any optimal solution of the D-Hoffkin model.

Theorem 11. *There exists an optimal solution to the MH model that is an FSFS-D solution.*

Similar results hold for the previously-discussed variants of the MH model and

the D-Hoffkin model.

5.6 Other Objective Functions

The MH model can admit a broader class of objective functions than the one that we provided in Section 5.2.1. Some such objective functions were discussed in [5]. Rather than having a fixed cost of ground delay, it is possible to specify a cost $c_g(t, f)$ of allowing each flight f to depart in each time period t, so that the objective would be:

$$\min \sum_{q \in \Theta} p(q) \left(\sum_{f \in \Phi} \sum_{t=\tau(f)}^{T+1} c_g(t, f) y_{t,f}^q + \sum_{t=1}^{T} c_a w_t^q \right).$$

Since the variables D-Hoffkin model describe aggregated quantities of flights and do not specify the departure times of each individual flight, such an objective function is incompatible with the D-Hoffkin model.

However, many objective functions of the above form would not be used in practice. For example, suppose that the costs $c_g(t, f)$ were specified so that the cost of holding some flight f_1 on the ground for k time periods was much larger than the cost of holding another flight f_2 on the ground for the same amount of time. Then the Air Navigation Service Provider (ANSP) implementing this delays would be giving much higher priority to flight f_1 and f_2. In current practice, the ANSP is expected to act as a neutral party that treats all flights equally. It is therefore reasonable to restrict our attention to costs of the form:

$$c_g(t, f) = \Gamma(t - \tau(f)).$$

That is, the cost of assigning the flight f to the departure time t is a function of the amount of time that the flight is held on the ground. We will assume that $\Gamma(0) = 0$, so that there is no cost for assigning no ground delay to a flight.

If some flight f_1 has been waiting on the ground longer than some other flight f_2, it seems that it would be more fair to allow flight f_1 to depart than f_2. This is essentially the first-scheduled first-served (FSFS) equity rule mentioned in Section 5.5. The cost function Γ is compatible with the FSFS equity rule when

$$\Gamma(t) - \Gamma(t-1) < \Gamma(t+1) - \Gamma(t),$$

for all t, so that the cost of assigning additional delays to a flight increases with the amount of delay assigned to the flight. A function that satisfies this property is called ***marginally increasing***.

Under the previously-defined cost function, we showed that the MH model has an optimal solution that satisfies the FSFS-D property (Theorem 12 of Section 5.5). A stronger statement holds when the cost of ground delays are marginally increasing. In this case, every optimal solution satisfies the FSFS-D property.

Theorem 12. *If Γ is marginally increasing, then every optimal solution to the MH model satisfies the FSFS-D property.*

Proof. Let $\boldsymbol{\lambda}$ be a solution to the MH model such that $\boldsymbol{\lambda}$ does not satisfy the FSFS-D property. Then we can construct a solution $\boldsymbol{\lambda}^*$ such that $C(\boldsymbol{\lambda}^*) < C(\boldsymbol{\lambda})$. Since $\boldsymbol{\lambda}$ does not satisfy the FSFS-D property, then there exists some flights ϕ and ψ, some scenario ρ, some time period s, and some time period $\sigma < s$ such that $\delta(\phi) = \delta(\psi)$,

$\tau(\phi) < \tau(\psi)$, such that

$$y^{\rho}_{s,\phi}(\boldsymbol{\lambda}) = 1$$

and such that

$$y^{\rho}_{\sigma,\psi}(\boldsymbol{\lambda}) = 1.$$

Define the solution $\boldsymbol{\lambda}^*$ as follows. Let

$$\begin{aligned}
y^{r}_{t,\psi}(\boldsymbol{\lambda}^*) &= y^{r}_{t,\phi}(\boldsymbol{\lambda}) && \text{for all } r \in n(\sigma,\rho), t \in \{\sigma, T+1\};\\
y^{r}_{t,\phi}(\boldsymbol{\lambda}^*) &= y^{r}_{t,\psi}(\boldsymbol{\lambda}) && \text{for all } r \in n(\sigma,\rho), t \in \{\sigma, T+1\};\\
v(\boldsymbol{\lambda}^*) &= v(\boldsymbol{\lambda}) && \text{for all other variables } v.
\end{aligned}$$

In other words, we simply swap the assignment of flight ϕ and ψ in the scenarios of the node $n(\sigma,\rho)$. It is straightforward to verify that $\boldsymbol{\lambda}^*$ is feasible.

Constraint 5.1 of the MH model implies that for each scenario ω in $n(\sigma,\rho)$, there is a unique time period $t(\omega)$ such that

$$y^{\omega}_{t(\omega),\phi}(\boldsymbol{\lambda}) = 1.$$

Furthermore, it must be true that $t(\omega) > \sigma$ for all scenarios ω in $n(\omega,\rho)$. If there were some scenario ω' such that $t(\omega') \leq \sigma$, then the anti-anticipatory constraints (5.5) would require that $y^{\omega}_{t(\omega'),\phi} = 1$ for all scenarios ω in $n(\omega', t(\omega'))$. However, the scenario ρ is in $n(\omega', t(\omega'))$, and $y^{\rho}_{t(\omega'),\phi} = 0$. Then, the cost of $\boldsymbol{\lambda}^*$ is given by

$$\begin{aligned}
C(\boldsymbol{\lambda}^*) &= C(\boldsymbol{\lambda})\\
&+ \sum_{\omega \in n(\sigma,\rho)} p(\omega)\left[\Gamma(\sigma - \tau(\phi)) - \Gamma(t(\omega) - \tau(\phi)) + \Gamma(t(\omega) - \tau(\psi)) - \Gamma(\sigma - \tau(\psi))\right].
\end{aligned}$$

Since Γ is marginally increasing, each $t(\omega) > \sigma$, and $\tau(\phi) < \tau(\psi)$, then Lemma 1 from [37] implies that

$$\Gamma(t(\omega) - \tau(\psi)) - \Gamma(\sigma - \tau(\psi)) < \Gamma(t(\omega) - \tau(\phi)) - \Gamma(\sigma - \tau(\phi)).$$

Then

$$\Gamma(\sigma - \tau(\phi)) - \Gamma(t(\omega) - \tau(\phi)) + \Gamma(t(\omega) - \tau(\psi)) - \Gamma(\sigma - \tau(\psi)) < 0$$

and

$$C(\boldsymbol{\lambda}^*) < C(\boldsymbol{\lambda}).$$

Thus, $\boldsymbol{\lambda}$ is not optimal. □

This is analogous to the results provided in [37], where it was shown that a static model for the SAGHP produced assignments in accordance to the FSFS equity rule whenever the cost of ground delays were marginally increasing.

5.6.1 The GD-Hoffkin Model

With some minor modifications, the D-Hoffkin model can also admit ground delay costs that are marginally increasing. We will refer to this model as the Generalized D-Hoffkin model, or GD-Hoffkin model for short. The GD-Hoffkin model tracks ground delays in a more detailed fashion. The variables $x^n_{t,m}$ are replaced by the decision variables:

$x^n_{t,\tau,m}$:= the number of flights with flight duration m and scheduled departure τ taking ground delay in time period t under scenario node n.

For notational convenience, we also define a set of parameters,

$$\mu(t) := \Gamma(t+1) - \Gamma(t)$$

the marginal cost of assigning the $(t+1)^{\text{th}}$ unit of ground delay to a flight that has already taken t units of ground delay. Then Γ is marginally increasing if and only if μ is increasing.

The objective of the GD-Hoffkin model is replaced by:

$$\min \sum_{n \in \mathcal{N}} p(n) \left[\sum_{m \in \Delta} \sum_{s=1}^{t(n)} \mu(t(n)-s) x^n_{t(n),s,m} + c_a w^n_{t(n)} \right].$$

The constraints 5.11 and 5.12 of the Hoffkin model, which ensure that all flights either depart or take delay, are replaced by

$$x^n_{1,1,m} + z^n_{1,1,m} = |D_{1,m}| \quad \forall m \in \Delta, n \in N_1, \tag{5.16}$$

$$z^n_{t,m} - \sum_{s=1}^{t-1} x^{a(t-1,n)}_{t-1,s,m} + \sum_{s=1}^{t} x^n_{t,s,m} = |D_{t,m}| \quad \forall m \in \Delta, t \in \{2,...,T\}, n \in N_t. \tag{5.17}$$

These constraints serve the same function, but describe the propagation of delay in more detail. In addition, we introduce the constraints

$$x^n_{t,s,m} \leq |D_{s,m}| \qquad \forall m \in \Delta, t \in \{1,...,T\}, s \in \{2,...,t\}, n \in N_t. \tag{5.18}$$

These constraints ensure that flights with some scheduled departure time τ receiving delays do not contribute to the delay variable $x^n_{t,s,m}$ for some other scheduled departure time s that is later than τ. This model admits feasible solutions in which some flight f contributes to the delay variable $x^n_{t,s,m}$ for some s that is earlier that the scheduled departure time of f. Such solutions would be inconsistent with our interpretation of the variables, and could not be implemented in practice. However,

we can show that this problem can be avoided when delay costs are marginally increasing. In this case, flights with earlier departure times would have been waiting longer, and would have a higher marginal cost of delay. This means that the variable $x^n_{t,s,m}$ would have a greater associated cost than $x^n_{t,\tau,m}$, and that any optimal solution would assign flights to $x^n_{t,\tau,m}$ before assigning flights to $x^n_{t,s,m}$. In essence, optimal solutions must follow a type of FSFS-D equity property in order to ensure that this model is valid, and we can prove that this is the case whenever delay costs are marginally increasing.

We formalize the above arguments as follows. We define the FSFS-D equity property for the GD-Hoffkin in Definition 7. The interpretation of this definition is that amongst flights with the same duration, those flights with an earlier scheduled departure time can only take delay if the flights with later scheduled departure time are also taking delay.

Definition 7 (FSFS-D)**.** *A solution* $\boldsymbol{\lambda}$ *to the GD-Hoffkin model is* ***FSFS-D*** *if the following is satisfied. For any time period* t*, for any scheduled departure time* τ*, for any flight duration* m*, and for any scenario node* n *of* $\mathcal{N}_t$ *such that*

$$x^n_{t,\tau,m}(\lambda) > 0$$

then

$$x^n_{t,\tau',m}(\lambda) = |D_{\tau',m}|$$

for all τ' *where* $\tau < \tau' \leq t$*.*

If the ground delay costs are marginally increasing, then any optimal solution to the GD-Hoffkin model is FSFS-D.

Theorem 13. *If μ is increasing then every optimal solution $\boldsymbol{\lambda}^*$ to the GD-Hoffkin is FSFS-D.*

Proof. Let $\boldsymbol{\lambda}$ be a solution that is not FSFS-D. Then there exists some time period t, some scenario n in $\mathscr{N}_t$, some flight duration m, and some departure times τ and τ' such that $\tau < \tau'$, such that $x^n_{t,\tau,m}(\boldsymbol{\lambda}) > 0$, and such that $x^n_{t,\tau',m}(\lambda) < |D_{\tau',m}|$. Define another solution $\boldsymbol{\lambda}^*$ by

$$\begin{aligned} x^n_{t,\tau,m}(\boldsymbol{\lambda}^*) &= x^n_{t,\tau,m}(\boldsymbol{\lambda}^*) - 1, \\ x^n_{t,\tau',m}(\boldsymbol{\lambda}) &= x^n_{t,\tau',m}(\boldsymbol{\lambda}) + 1, \\ v(\boldsymbol{\lambda}^*) &= v(\boldsymbol{\lambda}) \qquad \text{for all other variables } v. \end{aligned}$$

It is straightforward to verify that $\boldsymbol{\lambda}^*$ is a feasible solution, and that the cost of $\boldsymbol{\lambda}^*$ is given by:

$$\begin{aligned} C(\boldsymbol{\lambda}^*) &= \mu(t-\tau') - \mu(t-\tau) + C(\boldsymbol{\lambda}) \\ &< C(\boldsymbol{\lambda}). \end{aligned}$$

This proves that $\boldsymbol{\lambda}$ is not an optimal solution. □

5.6.2 Model Equivalence.

The GD-Hoffkin model is also equivalent to the MH model, but in a weaker sense than the D-Hoffkin model. We do not believe that there is a correspondence between the feasible solutions of the two models that preserves objective values, but there is such a correspondence between the FSFS-D solutions of the two models. Theorems 12 and 13 show that the optimal solutions of these models are always

FSFS-D when the ground delay costs are marginally increasing. Thus, under those costs, there is a correspondence between the optimal solutions of the two models.

Let $\mathscr{P}^*_{MH}$ be the set of feasible FSFS-D solutions to the MH model, and let $\mathscr{P}^*_{GDH}$ be the set of feasible FSFS-D solutions to the GD-Hoffkin model. We define a mapping from $\mathscr{P}^*_{MH}$ to $\mathscr{P}^*_{GDH}$ in a similar fashion to the way that γ_1 was defined in Section 5.6. Let γ_1^* be given by:

$$z^n_{t,m}(\gamma_1^*(\boldsymbol{\lambda})) = \sum_{s=1}^{t} \sum_{f \in D_{s,m}} y^{q(n)}_{t,f}(\boldsymbol{\lambda}) \qquad \forall m \in \Delta, t \in \{1,...,T\}, n \in \mathscr{N}_t,$$

$$x^n_{t,\tau,m}(\gamma_1^*(\boldsymbol{\lambda})) = |D_{\tau,m}| - \sum_{s=\tau}^{t} \sum_{f \in D_{\tau,m}} y^{q(a(s,n))}_{s,f}(\boldsymbol{\lambda}) \quad \forall m \in \Delta, t \in \{1,...,T\},$$

$$\tau \in \{1,...,t\}, n \in \mathscr{N}_t,$$

$$w^n_t(\gamma_1^*(\boldsymbol{\lambda})) = w^{q(n)}_t(\boldsymbol{\lambda}) \qquad t \in \{1,...,T\}, n \in \mathscr{N}_t.$$

Proposition 5. *The map γ_1^* is a well-defined function from $\mathscr{P}^*_{MH}$ to $\mathscr{P}^*_{GDH}$, and it satisfies the property that $C(\gamma_1^*(\boldsymbol{\lambda})) = C(\boldsymbol{\lambda})$ for any $\boldsymbol{\lambda}$ in $\mathscr{P}_{MH}$.*

In the other direction, the mapping γ_2^{RBS} can be applied (defined in Section 5.5). This is a slight abuse of notation since γ_2^{RBS} was defined to be a function on the feasible solutions of the D-Hoffkin model. However, the intent should be clear since this function does not depend on any variables of the D-Hoffkin model that were modified in the GD-Hoffkin model.

Proposition 6. *The map γ_2^{RBS} is a well-defined function from $\mathscr{P}^*_{GDH} \to \mathscr{P}^*_{MH}$ such that $C(\gamma_2^{RBS}(\boldsymbol{\lambda})) = C(\boldsymbol{\lambda})$.*

Proofs of Propositions 5 and 6 are provided in Sections D.4 and D.5 of Appendix D. Again, similar statements can be made for the variants of the models.

5.7 Conclusion

In this chapter, we have proposed a new formulation for the dynamic single airport ground holding problem, and we provided computational results that indicate that this model is usually more efficient than the existing model. We have shown that closely related models to the new model and the existing model have very strong polyhedral properties. We also demonstrated an equity property for the optimal solutions from both the existing and new models.

While our theoretical results demonstrate the strength of some closely related formulations, further work would be required to show that the existing model and newly proposed model have strong polyhedral properties themselves. Furthermore, while these models seem to be strong, there are instances of the problem for which the optimal solution of the linear programming relaxation is not integral. This indicates that there is an opportunity to identify valid inequalities.

These models neglect many sources of uncertainty present in real problems. For example, these models assume that flights have fixed flight durations, while actual flight durations can deviate from the expected durations. Future work could improve the realism of these models. There is also further work to be done in developing mechanisms that would apply these models in practice. There are many ways to integrate these models into air traffic management decision-making. Care must be taken to ensure that such an implementation has desirable properties such as equitable treatment of participants, compatibility with our existing traffic management processes, and robustness to errors in model parameters.

Part III

Representative Regions and Representative Traffic Management Initiatives

Chapter 6: Representative Region Selection

This chapter was adapted from the paper "Data Exploration with Selection of Representative Regions: Formulation, Axioms, Methods, and Consistency" written by Alexander Estes, Michael Ball, and David Lovell. The paper is available on SSRN, abstract number 3005997.

In this chapter, we present a new type of unsupervised learning problem in which we find a small set of representative regions that approximates a larger dataset. These regions may be presented to a practitioner along with additional information in order to help the practitioner explore the data set. An advantage of this approach is that it does not require the presence of cluster structure in the data. We formally define this problem, and we present axioms that should be satisfied by functions that measure the quality of representatives. We provide a quality function that satisfies all of these axioms. Using this quality function, we construct two methods for finding representatives. We provide convergence results for a general class of methods, and we show that these results apply to several specific methods, including the two methods proposed in this chapter. We provide an example of how representative regions may be used to explore a data set.

6.1 Introduction

We present a general problem in data exploration that we call *representative region selection*. The broad premise is that we have a large space of data observations that are of interest to a human, but too voluminous to be considered directly. The broad goal is to help that human by extracting from this space a manageable number of representative regions, which can then be considered in the context of the original decision. Specifically, in representative region selection (RRS), these several regions are selected and information about the locations of these regions and the observations that occur in these regions is presented. The goal is to help the human understand the distribution of the original observations.

There are existing methods that could be described as RRS methods, although these methods were designed for more specific contexts. One exemplar is the Density Estimation Tree method [68]. This method creates a binary decision tree where each leaf of the tree corresponds to a region of the data. For each region, the proportion of observations that fall in the region is provided. This method was developed as an interpretable method for density estimation. Any RRS method can be used as an interpretable method for density estimation; this is discussed in detail in Section 6.6.4. However, RRS forms a more general class of problems, where the goal is to select regions such that these regions and their accompanying information describe the data set as well as possible. The determination of how well a set of regions describes a data set is dependent on how the data will be used; this is discussed in Section 6.4. Similarly, the accompanying information that is provided should be

relevant to the application. In some cases, the most relevant information may be the proportion of observations that fall within the region, as in the density estimation application. In other cases, other information such as the distribution of class labels or some response variable may be more pertinent. As far as we know, we are the first to describe this problem in generality.

The RRS problem bears resemblance to the problem of clustering. In clustering, the goal is to partition the data into sets, where observations placed in the same set are similar to each other, while observations that fall in different sets are dissimilar. However, the purpose of RRS is generally distinct from that in clustering. The goal of clustering methods is to identify cluster structure within the data set, while the goal of representative region identification methods is to summarize the distribution of the data set. Some data sets do not have strong cluster structure, in which case it is unreasonable to apply a clustering method. However, RRS methods may still be used to describe the distribution of such data sets. Clustering methods do not always produce clusters that are easily viewed or understood by humans. Many methods (e.g. [69] and [70]) can produce clusters of irregular shape. However, in the representative region problem, the regions are intended to be presented to a human. Thus, care should be taken to select regions that can be described in a way that a human can understand. If the data consist of a relatively small number of regularly shaped clusters, then it may be reasonable to use these clusters as representative regions. If the data have strong cluster structure but the shapes of these clusters are irregular, then it may be sensible to first cluster the data and subsequently apply an RRS method to each cluster in order to describe these clusters to a human.

6.1.1 Layout and Contributions

The layout of the remainder of this chapter is as follows. Section 6.2 describes existing literature on related problems, including clustering, dimensional reduction, density estimation, other methods for approximating data sets, and other methods for data exploration. We highlight how the method described herein differs from the closest analogues amongst these existing works. A formal mathematical definition of representative-finding methods is given in Section 6.3. Like clustering methods, representative-finding methods can be put on a stronger foundation when bolstered by an axiomatic approach to describing the quality of their results; such an approach is shown in Section 6.4, with appropriate parallels between the axioms of clustering and representative-finding showcased. In addition to merely describing the quality of the results, the axiomatic approach can be used constructively, to induce representative-finding methods. Section 6.5 shows this for two different induced methods. Section 6.6 provides general convergence results for RRS methods. In particular, it provides conditions under which the proportion of observations falling in each region of the RRS method converge to the true probability of receiving an observation. We also show convergence of the coverage to the true support of the observed data. The section also shows how RRS methods can be used naturally for density estimation, and provides conditions under which these estimates are consistent. All of these results make fairly benign assumptions about the true distribution of the observations and minimal assumptions about the RRS method used. Section 6.7 applies the convergence results to several specific RRS methods.

These include the two induced RRS methods introduced in this chapter. We also apply these results to two existing methods in order to demonstrate the breadth of the RRS framework and its relevance to existing models. Finally, in Section 6.8, we offer conclusions and suggestions for additional research.

The main contributions of the chapter are:

- A formal definition of representative region selection (RRS) methods.
- A set of axioms that any manner of measuring the quality of RRS methods should follow, and a measure of quality that satisfies these axioms.
- Two RRS methods constructed from the axiomatic quality measure.
- Convergence results for a very broad class of RRS methods. These results consist of:
  - Conditions under which the proportion of observations in each region converges to the probability of receiving an observation in that region
  - Conditions under which the coverage of the regions converges to the support of the observations.
  - A method for constructing density estimates from RRS methods and conditions under which these estimates are consistent.
- A demonstration that our convergence results are applicable to four specific RRS methods, including our two proposed methods.

6.2 Related Literature

In this section, we discuss relevant existing work. We focus on those methods in unsupervised learning and data exploration that are most closely related to the RRS problem.

6.2.1 Clustering and Clustering Axioms

Clustering methods, similarly to RRS methods, provide a description of a dataset by dividing it into regions. See [71] or [72] for an overview of these methods. However, as discussed in Section 6.1, the regions produced by clustering algorithms are not necessarily easily understood by a human, and clustering methods serve a different purpose than RRS methods. A similarity shared between clustering methods and RRS methods is that there may be many reasonable ways to define the performance of the method, but there are some requirements that such a definition should satisfy. To give an example, consider a data set X with a clustering C. Suppose that another data set X' is created by permuting the points of X, and that a corresponding clustering C' is created by applying the same permutation to C. Then, the measured qualities of these two clusterings should be the same. In other words, the measure of quality should not be affected by the order in which the points of the data set are arranged. A set of axioms that should be met by measures of cluster quality were provided in [6], as were some quality measures that satisfied all of these axioms. A set of similar axioms has also been developed for clusters on graphs by [73]. We provide a set of axioms for quality measures of representative sets

of regions. The axioms provided by [6] are used as a starting point, and our axioms parallel those in the previous work. All of the axioms need some modification in order to be applicable to the RRS problem as we define it. For some axioms, these modifications are small, while in other cases drastic changes are required.

6.2.2 Approximating a Data Set with Data Points

There is a substantial body of literature that involves producing a small set of observations that approximates a larger data set. Core set methods (e.g. [74, 75, 76]) are used to estimate the result of applying an algorithm to a data set when it would be too computationally expensive to apply the algorithm directly. In these methods, a relatively small subset of data points is selected and the algorithm is applied only to these points. Core set methods attempt to select these points in a manner that would preserve the value of the original algorithm. In a similar vein, prototype reduction methods decrease the computational expense of nearest-neighbor classification algorithms by creating a relatively small data set and applying these algorithms to the smaller data set (e.g. [77, 78]). In the context of experimental design, there may be many potential experiments while the number of experiments that can be run is limited by some budget. In this case, representative subset selection algorithms (e.g. [79, 80, 81]) can be used to choose a diverse set of experiments.

The above methods differ from the RRS problem in that the selected subset is not necessarily intended to be viewed by human observers, and these methods produce points instead of regions. Furthermore, prototype reduction methods gen-

erally require a labeled dataset whereas RRS methods should work equally well on an unlabeled data set. For some core set methods or representative subset selection methods, it may be possible to create a corresponding RRS method by associating a region with each data point. However, the results may be difficult to interpret since these methods were not designed for this context.

6.2.3 Domain-Specific Data Summarization

There is a wide variety of research on data summarization for certain specific domains. Two particularly active fields of research include text summarization (e.g. [82, 83, 84]) and summarization of datasets in which each observed data point is a subset or "basket" from some set of items (e.g. [85, 86, 87, 88]). These methods tend to be highly specialized to their respective domains and it would be difficult to apply these methods to other domains. The methods discussed in this chapter are designed for more general types of data sets.

6.2.4 Density Estimation

As previously mentioned, any RRS method can be used as an interpretable measure for density estimation. This is also discussed in more detail in Section 6.6.4. Some existing density estimation methods can be placed in the framework of RRS method. These include Density Estimation Trees [68] and multivariate histograms. There is a large body of research on density estimation; see for example [89] or [90] for overviews. The vast majority of these methods were not designed with

interpretability in mind. As a result, for datasets in more than 2 or 3 dimensions, it is very difficult to display the resulting density estimates in a manner that a human could understand. By design, RRS methods produce estimates that a human can understand. RRS methods are also more general, as they can be used to provide other types of information instead of density estimates.

6.2.5 Other Methods for Data Exploration

There are many existing methods for data exploration. Dimension reduction and manifold learning techniques map the data into a lower-dimensional space. For the purposes of data exploration, this is usually a 2-dimensional space so that the data may be plotted and examined visually. There is a well-developed body of research on these methods, and there are several surveys available. See for example [91], [92] or [93]. These techniques attempt to perform this mapping in a manner that preserves relationships between data points as well as possible. A difference between the purpose of dimension reduction and that of RRS methods is that dimension reduction techniques attempt to explain the relationships between data points by displaying them in a new space, while RRS methods attempt to describe the distribution of data points in the original space. Furthermore, there is no guarantee that a given data set can be displayed in a low dimension without severely distorting the relationships between points.

In bump hunting (also referred to as mode hunting) the goal is to identify regions of the data in which observations occur with high probability (e.g. [94, 95]).

These methods have a similar goal to RRS methods in that both attempt to provide information about the distribution of the data. However, bump hunting focuses on high probability regions only, while RRS methods attempt to provide a more complete description of the data set.

6.3 Formal Definition of Representative-Finding Methods

The goal in Representative Region Selection (RRS) is to take a data set and from it produce a small set of representative regions that may be presented to a practitioner. In this section, we formally define a class of methods that produce representatives so that we may analyze the properties of such methods. It is important to note that some methods might satisfy the definitions given here while not being terribly useful. For example, a RRS method could always choose one large region that encompasses the entire set of possible observations. Methods that produce the same set of representatives regardless of the observed data set can also meet this definition. In section 6.4, we discuss properties that methods for finding representatives should have in order to be useful.

6.3.1 Representative-Finding Methods

Let us assume that there is some set M (i.e., sample space) such that any observed element will be an element of M. An observed data set is then a finite subset of M. For any set S, we will use the notation $\mathrm{Fin}\,(S)$ for the set of finite subsets of S and we will denote the set of all subsets (i.e., the power set) of S by

2^S. An RRS method takes a finite data set and produces a finite set $A_1, ..., A_n$ of representative regions. If these regions are intended to help a human understand the data, then the regions themselves must be intuitive. Thus, we assume that the set of possible regions is restricted to some collection $\mathscr{C}$, where each element $A \in \mathscr{C}$ is a subset of M. The collection $\mathscr{C}$ of allowed regions is chosen for the particular application. For example, if the space M in which observations occur has a natural distance measure, then the collection $\mathscr{C}$ could consist of the set of balls in M according to that distance measure. For observations in $\mathbb{R}^n$, another potential choice for $\mathscr{C}$ is the set of hyper-rectangles, i.e., the sets of the form $[a_1, b_1] \times ... \times [a_n, b_n]$. Then, we can formally define an RRS method as follows:

Definition 8 (RRS Method)**.** *Let M be some set. An **RRS method on M** is an ordered pair $(\mathscr{C}, f)$ where $\mathscr{C} \subseteq 2^M$ and f is a function $f : Fin(M) \to Fin(\mathscr{C})$.*

For a RRS method $(\mathscr{C}, f)$, we refer to the collection $\mathscr{C}$ as the *allowed regions* for f. Note that there is no requirement that the regions produced by the RRS method should be disjoint. We allow for representative regions to overlap. There is also no requirement that the function f is surjective, so it is possible that a certain region is in the set of allowed regions, but this region will never be chosen as one of the representative regions.

We will say that a point x of the data set is *covered* by a set of regions $A_1, ..., A_n$ if the observation falls in at least one of these regions. Then, let us define the coverage region produced by an RRS method on some data set S to be the set of points covered by the resulting representative regions.

Definition 9 (Coverage Region)**.** *For an RRS method* $(\mathscr{C}, f)$ *on* M*, the* ***coverage region*** *of* $(\mathscr{C}, f)$ *on the set* S *is*

$$C(S; f) := \bigcup_{A \in f(S)} A.$$

We expect that in many cases, a user would want the set of representatives to cover all of the observations that occur in the data set. If an RRS method always satisfies this requirement, then we will say that this method is *sample-covering*.

Definition 10 (Sample Covering)**.** *An RRS method* $(\mathscr{C}, f)$ *on* M *is* ***sample-covering*** *if*

$$C(S; f) \supseteq S$$

for any $S \in Fin(M)$*.*

We also expect that a user would usually be more interested in regions where observations occur than regions where there are no observations. Then, it may be reasonable to require that each selected representative region should cover at least one data point. We call RRS methods with this property *sample-intersecting* methods.

Definition 11 (Sample Intersecting)**.** *An RRS method* $(\mathscr{C}, f)$ *on* M *is* ***sample-intersecting*** *if* $A_i \cap S$ *is non-empty for any* $A_i \in f(S)$ *and for any* $S \in Fin(M)$*.*

An RRS method may depend on some parameters chosen by a user. This will result in a family of methods $\{(\mathscr{C}_\theta, f_\theta) : \theta \in \Theta\}$ for some set of parameters Θ. For example, a user may wish to limit the maximum number of representatives that are produced. This would ensure that the set of representatives is of reasonable

size for manual examination. In this case, we would be interested in a family of representative-generating functions $\{f_k : k \in \mathbb{N}\}$ where $(\mathscr{C}_k, f_k)$ is guaranteed to produce no more that k representative regions. Alternatively, a user may want to place limits on the size of the regions in order to guarantee that all the points covered by a representative region are sufficiently similar. Then it may be appropriate to consider a family $\{(\mathscr{C}_\alpha, f_\alpha) : \alpha \in \mathbb{R}^+\}$ where $(\mathscr{C}_\alpha, f_\alpha)$ is guaranteed to produce regions of diameter at most α.

6.4 Axioms of RRS Methods

Axioms for clustering functions were first considered by Kleinberg [96]. The main idea is that any function that produces a clustering for a set of data points should satisfy some properties that agree with our intuition of what a cluster should look like. The chosen axioms were shown to be inconsistent, so there is no clustering function that could satisfy all the axioms. Ben-David and and Ackerman [6] made some alterations to the axioms proposed by Kleinberg. Instead of examining functions that produce a clustering for a set of data points, Ben-David and Ackerman instead considered functions that produce a quality score for any given clustering and any given set of data points. These functions are cluster quality measures (CQMs). They also argued that one of the axioms proposed by Kleinberg was too strict and proposed a weaker version of this axiom. The new set of axioms is consistent, and they provided some CQMs that meet all the axioms. In this section, we discuss the axioms for CQMs and how they may be adapted to the context of

representative finding functions.

6.4.1 Existing Work: Cluster Quality Measures

Assume that there is a data set S consisting of n data points. Then, a clustering is simply a set partition or equivalence relation on X. Assume also that there is a distance measure d where $d(x, y)$ gives the distance between two points x and y of S. A CQM judges the quality of a clustering for a set of points with a distance measure. That is, let us define $\mathcal{C}$ to be the set of triples (C, S, d) where C is a clustering of the sample S and d is a distance measure. A clustering quality measure is a function $Q : \mathcal{C} \to \mathbb{R}$ that satisfies a set of axioms. The interpretation is that $Q(C, S, d)$ is the quality of the clustering C for the set S under the distance measure d.

6.4.1.1 Scale-invariance

The scale-invariance axiom states that rescaling the data should not affect the quality of the clustering. A function $Q : \mathcal{C} \to \mathbb{R}$ is *scale-invariant* if and only if for any distance measure d, for every positive λ, and for every clustering $C \in \mathcal{C}$, then

$$Q(C, d) = Q(C, \lambda d).$$

Here, λd is defined to be the distance measure such that

$$(\lambda d)(x, y) = \lambda d(x, y).$$

6.4.1.2 Consistency

The consistency axiom states that if distances between elements of the same cluster are decreased while distances between elements of different clusters are increased, then the quality of the cluster should not decrease. This is formalized as follows. For a clustering C of some set S, we say that d' is a *C-consistent variation* of d if

$$d'(x, y) \leq d(x, y)$$

for any pair of points x, y that are placed in the same cluster of C and

$$d'(x', y') \geq d(x', y')$$

for any pair of points x, y that are placed in different clusters. Then we say that a function $Q : \mathcal{C} \to \mathbb{R}$ is *consistent* if for any clustering C on some set S, for any distance measure d, and for any distance measure d' that is a C-consistent variation of d, then

$$Q(C, S, d') \geq Q(C, S, d).$$

6.4.1.3 Richness

The richness axiom states that for any clustering, there should be a distance measure that makes the clustering optimal for our quality measure. A function $Q : \mathcal{C} \to \mathbb{R}$ is *rich* if for every clustering C on some set S, there exists a distance

measure d such that

$$C = \arg\max_{C'} Q\left(C', S, d\right).$$

Note that the set of possible clusterings on S is finite, so this is well-defined.

6.4.1.4 Isometry Invariance

The isometry invariance axiom states that relabeling points without changing the distance between them should not affect the quality of the clustering. Let us say that the clusterings C and C' are isomorphic with respect to the distance measure d if there exists some bijection $\phi : S \to S$ such that

$$d\left(x, y\right) = d\left(\phi\left(x\right), \phi\left(y\right)\right)$$

and such that x and y are in same cluster of C if and only if $\phi\left(x\right)$ and $\phi\left(y\right)$ are in the same cluster in the clustering C'. This is denoted by $C \approx_d C'$. Then, we say that a quality function is *isomorphism-invariant* if for any distance measure d and for any clusterings C, C' such that $C \approx_d C'$ then

$$Q\left(C, S, d\right) = Q\left(C', S, d\right).$$

These axioms provide a set of guidelines for cluster quality functions.

6.4.2 Axioms for Quality Measures of Representative-Finding Functions

We would like to adapt the clustering axioms for our setting. However, some alterations are necessary. The clustering axioms assume that the quality of the

clustering depends only on the distances between the observed data points in each cluster. However, the quality of representative regions may depend on factors other than distances. We must consider the space in which the observations and representative regions occur, rather than just the distances between the observed points. Furthermore, the desired properties of representative regions are different from those of clusters, as discussed in Section 6.1. In some cases, the axioms can be transferred to the new setting with relatively few modifications. However, in other cases, drastic changes are needed in order to make the axioms suitable for the RRS problem.

6.4.2.1 Preliminaries

We assume that the observed data points and all representatives come from some metric space (M, d), and that the representatives are selected from some allowed set $\mathscr{C}$. A quality measure is a function that takes an observed data set and a set of representative regions and returns a positive real number that describes how well the regions approximate the data set.

Definition 12 (Representative Quality Measure)**.** *Let (M, d) be a metric space, and let $\mathscr{C}$ be a subset of 2^M. Then a* ***representative quality measure*** *(RQM) for $(M, d; \mathscr{C})$ is a function $Q : Fin(\mathscr{C}) \times Fin(M) \to \mathbb{R}$.*

6.4.2.2 Scale-isometry invariance

The isomorphism invariance and scale invariance axioms can be transferred fairly naturally into the RRS setting. These can be combined into one axiom. We

define a λ-isometry on (M, d) to be a mapping that scales the distance between points by the same factor.

Definition 13 (λ-isometry)**.** *Let (M, d) be a metric space, and let λ be a positive real number. A function $\phi : M \to M$ is a* ***λ-isometry*** *if*

$$\lambda d(x, y) = d(\phi(x), \phi(y)) \tag{6.1}$$

for any $x, y \in M$.

For a set $A \subseteq M$, we define $\phi(A)$ to be the image of A under the mapping ϕ,

$$\phi(A) := \{\phi(x) : x \in A\}.$$

Similarly, if R is a set of representatives

$$R = \{A_1, ..., A_n\}$$

we let

$$\phi(R) := \{\phi(A_1), ..., \phi(A_n)\}.$$

Then, we will define a quality function to be *scale-isometry* invariant as follows.

Definition 14 (Scale-Isometry Invariance.)**.** *Let there be some metric space (M, d) and let $\mathscr{C}$ be a subset of 2^M. Let Q be an RQM for $(M, d, \mathscr{C})$. Then Q is* ***scale-isometry invariant*** *if*

$$Q(\phi(R), \phi(S)) = Q(R, S) \tag{6.2}$$

for any $\lambda > 0$, for any λ-isometry ϕ, for any $R \subseteq \mathscr{C}$ and for any $S \in Fin(M)$.

The intuition behind this definition is that there is some set of representative regions R for a sample S from some metric space M, and the space M is rescaled by a transformation ϕ. If the representatives and sample are transformed accordingly, then the quality should remain the same. This single axiom captures both the ideas of scale invariance and isomorphism invariance.

6.4.2.3 Richness

The richness axiom requires a large amount of adjustment in order to be applicable to the RRS application. It is reasonable to expect that there is an optimal number of clusters for a given data set. However, for any given k representative regions, there is likely to be a set of $k+1$ representatives that better approximates the data. A similar concept for clustering called refinement preference is discussed in [6], who note that refinement-preferring measures are never rich if the preference is strict. However, it is possible that a particular set of k representatives is the best choice when compared against other sets of possible representatives with cardinality k. In this case, the quality measure should allow the number of points covered by each representative region to adjust with the data set. For example, suppose that the data may be separated into two regions where the behavior of the observed data is similar within each region, but is different between the two regions. Regardless of whether these regions have similar or drastically different numbers of observations, we would want to choose two representative regions that correspond with these two regions of data. Thus, the quality measure should allow the number of observations

covered by each representative region to vary. It should not require, for example, the representative regions to cover the same numbers of points. We formalize this axiom as follows.

Definition 15 (Cardinality-Constrained Rich)**.** *An RQM function Q on $(M, d, \mathscr{C})$ is **cardinality-constrained rich** if for any k positive integers $n_1, ..., n_k$ there exists a finite subset S of M, a partition $\{T_1, ..., T_k\}$ of S, and a selection of representative regions $R = \{A_1, ..., A_k\} \subseteq \mathscr{C}$ such that the following hold:*

- $|T_i| = n_i$;
- $S \cap A_i = T_i$;
- $Q(R, S) \geq Q(R', S)$ *for any selection of* $R' \subseteq Fin(\mathscr{C})$ *such that* $|R'| \leq k$;
- $Q(R, S) > Q(R', S)$ *for any selection of* $R' \subseteq Fin(\mathscr{C})$ *such that* $|R'| < k$.

Instead of restricting the number of representatives and maximizing quality, it is possible to choose a minimum acceptable level of quality and find the smallest set of representatives that achieves that level of quality. As before, we would like the number of observations covered by each representative to be able to vary. This inspires another definition of richness that we call *quality-constrained richness*.

Definition 16 (Quality-Constrained Rich)**.** *An RQM function Q on $(M, d, \mathscr{C})$ is **quality-constrained rich** if for any k positive integers $n_1, ..., n_k$ there exists a finite subset S of M, a partition $\{T_1, ..., T_k\}$ of S, and a selection of representative regions $R = \{A_1, ..., A_k\} \subseteq Fin(\mathscr{C})$ such that the following hold:*

- $|T_i| = n_i$;
- $S \cap A_i = T_i$;
- $|R'| > k$ *for any selection of regions* $R' \subseteq Fin(\mathscr{C})$ *such that* $Q(R, S) > Q(R', S)$;
- $|R'| \geq k$ *for any selection* $R' \subseteq Fin(\mathscr{C})$ *such that* $Q(R, S) \geq Q(R', S)$.

It is easy to see that these two notions of richness are equivalent.

Proposition 7. *An RQM* Q *on* $(M, d, \mathscr{C})$ *is cardinality-constrained rich if and only if it is quality-constrained rich.*

Thus, we can say that an RQM is *rich* if it is either cardinality-constrained rich or if it is quality-constrained rich. Note that richness is a property not only of the quality function Q, but also of the set of allowed regions $\mathscr{C}$. If the set of allowed regions is not flexible enough, then it is not possible to define a rich quality function.

6.4.2.4 Refinement preference

As mentioned above, we expect that with more representative regions, the original data points can be presented more accurately. We can formalize this notion as follows. We say that a set of representative regions R' is a proper refinement of another selection R if R' contains all of the regions of R and also contains more regions.

Definition 17 (Proper Refinement)**.** *Let M be a set, and let $\mathscr{C} \subseteq 2^M$ be a set of allowed regions of M. Let R and R' be finite subsets of $\mathscr{C}$. Then R' is a* ***proper refinement*** *of R if $R \subsetneq R'$.*

We will say that a quality function Q is refinement-preferring if for any set of representatives R, then there exists a proper refinement R' of higher quality.

Definition 18 (Refinement-Preferring)**.** *Let Q be an RQM on $(M, d, \mathscr{C})$. Then Q is* ***refinement-preferring*** *if for any $R \subseteq \mathscr{C}$ and for any $S \subseteq M$ there exists a proper refinement R' of R such that $Q(R', S) \geq Q(R, S)$.*

It could be argued that a set of representatives could so perfectly capture a set of data that adding any more representatives would harm a user's understanding of the data. However, we posit that the set of data could be nearly perfectly approximated if for each point of the sample there is a corresponding, very small representative region that contains this point. Refining a set of representative regions by adding such points would describe the original data more accurately. Naturally, such a set of representatives would also generally be too large to be examined by a human. Thus, selecting representative regions through unconstrained optimization of a quality function is unlikely to be useful. Instead, constraints can be defined that ensure the set of representatives is usable by humans, and the quality function can be optimized subject to these constraints. For example, one approach may be to maximize a quality function subject to a maximum number of representative regions.

6.4.2.5 Consistency

The consistency axiom for clustering states that if the clustering structure is strengthened, then its quality should not get worse. This axiom requires significant modification since the representative regions do not necessarily follow cluster structure. We define a type of consistency for RRS methods as follows. Consider some representative region A. Suppose there were another region A' that was smaller than A, but that still contained all of the points of the data set that are contained in A. Then if A were replaced with A', the representatives would be more specific in their description of the data, but the coverage of the data set would not be harmed. Thus, the quality should not decrease. More generally, consider two sets of representative regions R and R'. If for each region A in R, there is a smaller region A' in R' that covers the same points as A, then the quality of R' should be at least as good as the quality of R. This can be formalized as follows.

Definition 19 (S-Consistent Variation)**.** *Let M be a set, and let $\mathscr{C} \subseteq 2^M$ be a set of allowed regions of M. Let R and R' be finite subsets of $\mathscr{C}$. Let S be a finite subset of M. Then R' is an* ***S-consistent variation*** *of R if there is a bijection $\phi : R \to R'$ such that for each $A \in R$,*

$$A \supseteq \phi(A)$$

and

$$A \cap S = \phi(A) \cap S.$$

The property that R' is an S-consistent variation of R formalizes the idea discussed above that R' is more specific than R without sacrificing coverage of the data.

Definition 20 (Consistent)**.** *Let Q be an RQM on $(M, d, \mathscr{C})$. Then Q is* ***consistent*** *if for any finite subset S of M, for any finite set of representative regions $R \subseteq \mathscr{C}$, and for any set of regions R' that is an S-consistent variation of R, then*

$$Q(R', S) \geq Q(R, S).$$

6.4.3 Axiomatic Representative Quality Functions

We will define an *axiomatic representative quality measure* (ARQM) to be an RQM that meets all of the axioms described above.

Definition 21 (Axiomatic Representative Quality Function)**.** *Let Q be an RQM on $(M, d, \mathscr{C})$. Then Q is an* ***axiomatic representative quality function*** *(ARQM) if Q is scale-isometry invariant, rich, refinement-preferring, and consistent.*

In the preceding discussion, we assumed that Q increases with the quality of the representatives. However, it is equally possible to define a quality function that decreases when the quality of representatives increases. Some slight and intuitive changes to the axioms would be necessary. For example, the consistency axiom would need to be modified so that if R' is an S-consistent variation of R then $Q(R', S) \leq Q(R, S)$. Similar modifications can be made to the richness and refinement preference axioms.

We can show that it is possible to satisfy all of these axioms simultaneously. We provide an example of an ARQM function for a case where observations occur in $\mathbb{R}^n$, distance is induced from a p-norm, and the set of allowed representatives is the set of closed balls. Here, the distance d induced from the p-norm is defined to be

$$d(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}}.$$

We will denote by $B[y, r]$ the closed ball centered at y with radius r. for some $p \geq 1$. For some set R of representatives, let $C(R)$ be the coverage region of R. Then we define a quality measure as follows.

Definition 22 (Radius and Coverage Measure)**.** *We define the* ***radius and coverage measure****, denoted Q_{rc}, by*

$$Q_{rc}(R, S) = \frac{1}{\max\limits_{w,x \in S} d(w, x)} \left(\max_{B[y,r] \in R} r \right) + \gamma \left| S \setminus C(R) \right|$$

where γ is some constant.

This quality measure decreases as the quality of representatives increases. The left term penalizes impreciseness of the representatives, measured by the maximum of their radii. This term is scaled by the maximum distance between points so that the scale-isometry axiom is not violated. The right term penalizes uncovered points. We claim that Q_{rc} is an ARQM.

Theorem 14. *The Q_{rc} function is an ARQM.*

Proof. For scale-isometry invariance, simply note that any λ-isometry will not affect

the points that are covered, and will scale all distances equally, leaving the ratio of distances unaffected.

In order to prove refinement preference, we create a refinement R' as follows. Choose any point x' of S and add a representative $B[x', r]$ with radius smaller than any existing radius. The term that penalizes the radius would not change, and the coverage region cannot decrease so the term that penalizes uncovered points must decrease or stay the same.

For richness, let there be a subset of $\mathbb{R}^n$ of size k. Let there be some positive integers $n_1, ..., n_k$. Then define the sets T_i so that each set T_i has n_i points very close to each other and the points in each pair of sets T_i and T_j are very far apart. Let $B[x_i, \epsilon_i]$ be the ball of minimum radius that covers T_i. Note that $\{B[x_1, \epsilon_1], ..., B[x_k, \epsilon_k]\}$ is the set of k representatives that minimizes the quality function, and that any set of $k-1$ representatives must have achieve a strictly higher value.

For consistency, let there be some set of representatives R for some set S. Let R' be an S-consistent variation of R. Then the coverage region of R and R' are the same, and the radii-penalizing term cannot be higher for R' than R. □

Thus, it is possible to satisfy all of these axioms simultaneously.

6.5 Constructing RRS Methods from an ARQM

Suppose that there is some ARQM Q. This can be used to induce two families of methods for finding representatives. If there is a desired quality q, then the RRS

method can find the smallest set of representatives that achieves the quality level q. Alternatively, if there is a bound k on the number of representatives, then the RRS method can find the set of k representatives of maximum quality. If desired, other conditions can be imposed. For example, it may be desired to require that the RRS method is sample-covering, sample-intersecting, or both.

In this section, we suppose that observations occur in $\mathbb{R}^n$ and that the set of allowed regions is the set of balls in $\mathbb{R}^n$. We consider the Q_{rc} quality measure as defined in Definition 22. As stated in Theorem 14, if the distance is induced by a p-norm, then Q_{rc} is an ARQM. Regardless of what distance measure d is used, and regardless of whether the quality measure satisfies the axioms, it is possible to use this quality measure to construct RRS methods in the aforementioned manner. We describe two RRS methods that are induced by the Q_{rc} function with additional constraints. First, we require that the RRS method is sample-covering. In many applications, this requirement is reasonable, as it ensures that no observed data points are neglected. Second, we require that the RRS chooses representatives from balls that are centered on members of the data set S. This aids in the presentation and interpretation of the representative regions, since each ball can be presented with the original data point at its center.

6.5.1 Minimum Dominating Set Method

First, we consider the case that there is a desired level of quality, and we construct a sample-covering RRS that produces a smallest set of representatives

that achieves that quality level. Recall that Q_{rc} assigns lower values to higher-quality representatives, so the desired level of quality q is an upper bound on Q_{rc}. Let S be a data set. Let $\mathscr{B}(S)$ be the set of closed balls $B[x, r]$ for $x \in S$ and $r > 0$. For a set of balls R, let $C(R)$ be the set of points in S that are covered by R. The problem of choosing the smallest sample-covering set of balls with centers in S that meet the quality requirement can be written as:

$$\min_{R \in \mathrm{Fin}(\mathscr{B}(S))} |R|$$

such that

$$\frac{1}{\max\limits_{w,x \in S} d(w, x)} \left(\max_{B[y,r] \in R} r \right) + \gamma |S \setminus C(R)| \leq q,$$

$$R \text{ is sample-covering.}$$

If R is sample-covering, then the term $\gamma |S \setminus C(R)|$ will always be zero. Then, by removing this term and rescaling the constraint, the problem can be written as

$$\min_{R \in \mathrm{Fin}(\mathscr{B}(S))} |R|$$

such that

$$\max_{B[y,r] \in R} r \leq q',$$

$$R \text{ is sample-covering,}$$

where

$$q' = q \left(\max_{w,x \in S} d(w, x) \right).$$

Note that there always exists an optimal solution to this problem such that the radius of each representative region is equal to q'. Then we can see that for any

optimal solution to the following problem, there is a corresponding optimal solution to the above optimization problem:

$$\min_{Y \subseteq S} |Y|$$

such that

$$\text{for all } x \in S, \text{ there exists a } y \in Y \text{ such that } x \in B[y, q'].$$

That is, the problem reduces to the problem of finding the smallest set of closed balls of radius q' whose centers are in S and that cover S. This can also be expressed as a problem on a graph. Consider the graph G whose nodes are the points of S and that has an edge between two points if their distance is at most q'. This graph is known as the q'-*neighborhood* graph of S. A set of nodes R is a dominating set if every node of G is either a node of R or is adjacent to a node of our R. Then, the optimization problem is to find the minimum dominating set of nodes R in G. This well-known problem is called the minimum dominating set (MDS) problem. The MDS problem is NP-hard [97]. See [98] for a thorough graph-theoretical treatment of dominating sets in graphs, or [99] for a bibliography of this topic. The problem may be formulated as an integer program as follows.

$$\min \sum_{v \in V(G)} x_v$$

such that

$$x_v + \sum_{u:\{u,v\} \in E(G)} x_u \geq 1 \qquad \forall v \in V(G),$$

$$x \in \{0, 1\} \qquad \forall v \in V(G).$$

Here, x_v is a binary decision variable that takes a value of 1 if the node v is included in the minimum dominating set and takes a value of zero otherwise. The single constraint enforces that each node is dominated. Since the problem is NP-hard, this IP is likely intractable for larger data sets. In that case, the problem may be solved approximately. There are existing approximation algorithms and heuristics for the MDS problem [100, 101, 102]. Some of these methods are designed to produce a dominating set, albeit perhaps not one of minimum size. Therefore, if these methods are used then the quality requirement will be satisfied, although the set that is produced may not be the smallest set that satisfies the quality requirement.

6.5.2 K-Center Method.

Consider instead a case where a constraint is placed on the number of representatives, whose quality is maximized. Again, we include the requirements that the center of each ball is taken from the original data set and that the representatives cover the data set. For the Q_{rc} RQM, this gives the optimization problem:

$$\min_{R \in \mathrm{Fin}(\mathscr{B}(S))} \left(\frac{1}{\max\limits_{w,x \in S} d(w,x)} \left(\max_{B[y,r] \in R} r \right) + \gamma |S \setminus C(R)| \right)$$

such that

$$|R| \leq k,$$

$$R \text{ is sample-covering.}$$

As before, if the representatives are sample-covering, then the coverage penalty will be zero. Multiplying the objective by a constant does not change the optimization

problem, so the scale factor on the term penalizing radii can be removed. Then the optimization problem is reduced to

$$\min_{R \in \mathrm{Fin}(\mathscr{B}(S))} \left(\max_{B[y,r] \in R} r \right)$$

such that

$$|R| \leq k,$$

$$R \text{ is sample-covering .}$$

In any optimal solution, the maximum radius is the maximum distance from any point x to the center of the nearest representative region. If there were some solution in which the maximum radius was greater than this maximum distance, then the radius could be reduced without damaging the coverage, resulting in a better solution. There is no solution in which the maximum radius is less than this maximum distance, as such a solution would not be sample-covering. Then, any solution to the problem

$$\min_{Y \subseteq S, |Y| \leq k} \left(\max_{x \in S} \left(\min_{y \in Y} d(x, y) \right) \right).$$

will also provide a solution to the original problem. This is another well-known NP-hard problem known as the k-center problem, and has previously been proposed as a method for finding cluster centers [103]. There exist several exact solution methods for this problem [104, 105, 106, 107, 108]. Again, for large data sets, these methods may not be computationally tractable, but there exist heuristics and approximation algorithms [103, 106, 109, 110, 111, 112, 113, 114]. These can be forced to produce a set of k points, but this set of k points may not be the set that minimizes the quality function.

6.6 General Convergence Properties

In this section, we provide convergence results related to RRS methods. We focus on three types of convergence. In many cases, we believe that it is useful to present the proportions of observations that fall in each representative region. This can be interpreted as an approximation of the probability that an observation occurs in the region. In our first set of convergence results, we provide conditions under which this proportion converges to the true probability. The coverage region of the RRS method can be thought of as an approximation of the set of values that the variable takes. In other words, the coverage region approximates the support of the probability measure induced by the observations. In our second set of convergence results, we provide conditions under which the coverage region converges to the true support. Finally, we demonstrate how any RRS method can be used to generate an estimate of the density of the underlying population, and we provide conditions for the consistency of this estimate.

In some cases, our results apply to a single, fixed RRS method. However, we would typically expect that an RRS method includes constraints to ensure that a human could understand the results. For example, a method might have a parameter k such that the method always produces at most k representatives. Then, the practitioner could specify the number of representatives that he or she is willing to examine. Some types of convergence are possible even with these constraints. We provide conditions under which the proportion of observations falling in each region converges to the true probability and we provide conditions under which the

probability of receiving an observation within the coverage region converges to 1. These results immediately indicate that these types of information provided by the RRS method are accurate if there is sufficient data.

Other types of convergence are not possible with such constraints. For example, consider the case where an RRS method is used as an interpretable density estimation method. This density usually will not converge if there is a constraint on the quantity of representative regions produced. However, if the number of regions was increased, then the density estimate might converge. In these situations, we provide convergence results for a parametrized family of methods. Admittedly, these results are not as immediately relevant to the use of RRS methods with a single fixed set of constraints. However, the results do provide conceptual support that these methods are well-founded. Intuitively, the results ensure that RRS methods could be used to approximate the data set arbitrarily well if the methods were allowed to produce a sufficiently large number of representatives.

6.6.1 Preliminaries

We begin by stating some assumptions and definitions that we use in our analysis. The conditions required by our results can be separated into two categories: conditions concerning the process by which observations arrive, and conditions on the RRS process.

6.6.1.1 Conditions on the Sampling Process

We make the typical assumption that the observed data set consists of a set of i.i.d. samples. This assumption could potentially be weakened. Our proofs will generally hold as long as the law of large numbers can be applied to ensure that the proportions of observations falling in any single fixed region converge to the probability of receiving an observation in that region. We assume that the samples have values in some metric space (M, d). Let $(\Omega, \mathcal{F}, \mathcal{P})$ be a probability space, and let $\mathcal{E}$ be a sigma-algebra on M. It will be assumed that any closed ball of the metric space (M, d) is an element of $\mathcal{E}$. It will also be assumed that any potential representative region is an element of $\mathcal{E}$. Let $(X_1, X_2, ...)$ be a sequence of i.i.d. $(M, \mathcal{E})$-valued random variables. Let P_X be the probability measure on $(M, \mathcal{E})$ induced by any X_i. We will use the notation that S_n is the sample of the first n observations; that is,

$$S_n = \{X_1, ..., X_n\},$$

and we will use the notation

$$C_n(f) = C(S_n; f).$$

For any subset T of M, we use $\hat{P}_{X,n}$ to denote the empirical measure produced by the first n observations,

$$\hat{P}_{X,n}(T) := \frac{|T \cap S_n|}{n}.$$

If P_X has a corresponding probability density function with respect to some reference density μ, we refer to that function as g_X. In the rest of section 6.6, whenever we

refer to M, d, Ω, $\mathcal{F}$, $\mathcal{P}$, $\mathcal{E}$, X_n, S_n, P_X, g_X or μ, it will be in reference to these quantities as defined here. Any mentions of the sampling process are in reference to this random process.

We occasionally mention the support of P_X. By this, we mean the set of points $y \in M$ such that $P_X(B[y;r]) > 0$ for all positive radii r. Many results will require that the support of P_X be compact and have full measure. Some commonly used distributions would not result in a compact support. For example, the support resulting from a variable with a normal distribution would be the entire real number line. However, in almost any practical situation, there are bounds on the values that a variable can take, if for no other reason than because of the physical limitations of the devices used to collect and store the data. Thus, we believe this to be a reasonable assumption. While this assumption is not commonly made in the density estimation literature, similar assumptions are used in the broader field of machine learning. For example, many works analyzing forest-based regression or classification methods make the stronger assumption that the feature vector is supported on $[0,1]^k$ (e.g. [115, 116, 117]).

6.6.1.2 Conditions on Representative-Generating Functions

For these convergence results, we place restrictions on the class $\mathscr{C}$ of allowed regions. We assume that each element of $\mathscr{C}$ is characterized by some parameters. One of these parameters should be related to the size of the region. This is called the *scale parameter*, and we reserve the symbol λ for this parameter. We assume that

for any choice of scale parameter λ and any choice θ of the parameters excluding the scale parameter, then there is a well-defined region $R(\theta, \lambda)$. However, we do not require that every such region $R(\theta, \lambda)$ be included in the set $\mathscr{C}$ of possible regions. For example, the set of allowed regions could be chosen to be the balls with radii less than some number L. In this case there are two parameters: the radius of the ball and the center of the ball. The scale parameter would be the radius of the ball. For each x, and for any radius $r > L$, the ball $B(x, r)$ is well-defined but is not included in the set $\mathscr{C}$. We formulate these requirements in the following manner.

Definition 23 (Scalable Class of Regions)**.** *A* ***scalable class of regions*** *in a space M is an ordered triple $(\mathscr{C}, \Theta, R)$ where $\mathscr{C}$ is a subset of 2^M, Θ is a set, and R is a function $R : \Theta \times \mathbb{R}^+ \to 2^M$ such that*

$$\mathscr{C} = \{R(\theta, \lambda) : (\theta, \lambda) \in \Xi\} \tag{6.3}$$

for some set $\Xi \subseteq \Theta \times \mathbb{R}^+$ and such that

$$R(\theta, \lambda) \subseteq R(\theta, \lambda') \tag{6.4}$$

whenever $\lambda' \geq \lambda$.

The regions should behave well with respect to changes in the scale parameter. Intuitively, the requirement is that when the scale parameter increases, the region expands, and all parts of the region expand at a sufficiently similar rate. We refer to classes of regions with this property as *uniform-scale* classes, and this is formalized in Definition 24.

Definition 24 (Uniform-Scale Set of Regions)**.** *Let $(\mathscr{C}, \Theta, R)$ be a scalable class of regions in a metric space (M, d). Then $(\mathscr{C}, \Theta, R)$ is* ***uniform-scale*** *if for any $\epsilon > 0$ there exists some $\delta > 0$ such that for any region $R(\theta, \lambda) \in \mathscr{C}$ the following hold:*

- *For any $x \in R(\theta, \lambda)$ then $B[x, \delta] \subseteq R(\theta, \lambda + \epsilon)$.*
- *For any $x \in R(\theta, \lambda - \epsilon)$ then $B[x, \delta] \subseteq R(\theta, \lambda)$.*

We further require that as the scale parameter changes, then the probability of receiving an observation in the region changes in a regular fashion.

Definition 25 (Uniform Continuity of Class with Respect to Probability Measure)**.** *Let $(\mathscr{C}, \Theta, R)$ be a scalable class of regions in a space M. Let $\mathcal{A}$ be a σ-algebra on M and let μ be a measure on $(M, \mathcal{A})$. Assume that $R(\theta, \lambda) \in \mathcal{A}$ for any $\theta \in \Theta$ and $\lambda \in \mathbb{R}^+$. Then $(\mathscr{C}, \Theta, R)$ is* ***uniformly continuous with respect to μ*** *if for any $\epsilon > 0$ there exists some $\delta > 0$ such that for any $R(\theta, \lambda) \in \mathscr{C}$ then*

$$\mu(R(\theta, \lambda + \delta)) - \mu(R(\theta, \lambda)) \leq \epsilon$$

and

$$\mu(R(\theta, \lambda)) - \mu(R(\theta, \lambda - \delta)) \leq \epsilon.$$

For example, consider the class of balls in a metric space, with the scale parameter defined to be the radius of the ball. This class is uniform-scale. Further suppose that the metric space under consideration is $\mathbb{R}^n$, that there is an upper bound L on the radius of the ball, and that there is a Borel probability measure P with a bounded density with respect to the usual measure on $\mathbb{R}^n$. Then the class

is uniformly continuous with respect to P. Consider alternatively the set of hyper-rectangles in $\mathbb{R}^n$, with side lengths bounded by some number L. This class may be parameterized as the set

$$\left\{ \prod_{i=1}^{n} [x_i - r_i\lambda, x_i + r_i\lambda] : \mathbf{x} \in \mathbb{R}^n, \lambda \in (0, L), r \in \left[1, \frac{L}{\lambda}\right]^n, \text{ and } r_j = \lambda \text{ for some } j \right\}.$$

Here, the scale parameter λ provides the length of the smallest side of the hyper-rectangle. The parameter r_i gives the ratio between the width of the hyper-rectangle in the ith dimension and the smallest side. The parameter $\mathbf{x}$ provides the center. Under this parameterization, the class is uniformly scaled. This class is also uniformly continuous with respect to P for any Borel probability measure P with a bounded density with respect to the usual measure.

Some results require a bound on the number of representative regions produced or on the sizes of these regions. We will say that an RRS method has bounded diameter if there is some distance L such that the representative regions always have diameter of at most L.

Definition 26 (Bounded Diameter)**.** *An RRS method $(\mathscr{C}, f)$ on a metric space (M, d) has **bounded diameter** if there exists an L such that for any $S \in Fin(M)$, then $d(x, y) \leq L$ for all $x, y \in A$ for each $A \in f(S)$.*

We say that a method is uniformly positive (with respect to some measure) if there is a uniform positive lower bound on the measure of any region produced by the method.

Definition 27 (Uniformly Positive)**.** *Let $(\mathscr{C}, f)$ be an RRS method on M. Let $\mathcal{A}$ be a σ-algebra on M and let μ be a measure on $(M, \mathcal{A})$. Assume that $\mathscr{C} \subseteq \mathcal{A}$. The*

method f is ***uniformly positive with respect to*** μ *if there exists some positive real number γ such that $\mu(f(S)) \geq \gamma$ for any $S \in Fin(M)$.*

We say that an RRS method has *bounded cardinality* if it always produces at most u representatives for some finite u.

Definition 28 (Bounded Cardinality)**.** *An RRS method $(\mathscr{C}, f)$ has* ***bounded cardinality*** *if there exists some u such that $|f(S)| \leq u$ for any $S \in Fin(M)$.*

In many cases, these properties need only hold asymptotically.

Definition 29 (Asymptotically bounded Diameter)**.** *Let $X_1, X_2, ...$ be a sequence of random variables and let $S_n = \{X_1, ..., X_n\}$. The RRS method $(\mathcal{C}, f)$ has* ***asymptotically bounded diameter with respect to the sequence $X_1, X_2, ...$*** *if there exists an L and there almost surely exists an N such that for all $n > N$, $d(x, y) \leq L$ for all $x, y \in A$ for each $A \in f(S_n)$.*

Throughout this section and the rest of the chapter, we simply state that an RRS method has asymptotically bounded diameter when there is no ambiguity as to the relevant sequence of variables. Typically, the sequence of observed values is the relevant sequence. We define *asymptotically bounded cardinality* similarly.

A user may want to use an RRS method with bounded cardinality in order to guarantee that the number of representative regions is small enough to be easily examined. However, if the number of regions is restricted, we expect that there is some limit on how well an RRS method can approximate a data set. On the other hand, if the number of representatives were increased, we expect that the

approximation should improve in quality, and as this bound becomes very large, we would want the approximation to become very good. Following this intuition, some convergence results involve a parameterized family of RRS methods of the form $\{(\mathscr{C}, f_m) : m \in \mathbb{N}\}$. For our results, we consider families where the parameter m has some relation to the maximum diameter of the representative regions. We say that a family is *arbitrarily precise* if this maximum diameter becomes arbitrarily small as the parameter m increases.

Definition 30 (Arbitrarily Precise)**.** *A family $\{(\mathscr{C}, f_m) : m \in \mathbb{N}\}$ of RRS methods on some metric space (M, d) is* ***arbitrarily precise*** *(with respect to some sequence $X_1, X_2, ...$ of random variables) if for any $\epsilon > 0$ there exists some natural number N such that for all $m > N$ the RRS method $(\mathscr{C}, f_m)$ has diameter asymptotically bounded by ϵ.*

Intuitively, the maximum diameter of the regions produced by the RRS method is a measure of how much detail the method provides. More regions of smaller size provide a finer level of detail, while fewer regions of larger radii provide a higher-level picture of the dataset. A family of representatives is arbitrarily precise if the level of detail provided by the representatives becomes indefinitely fine as the parameter increases. Note that while we only provide a definition for families of the form $\{(\mathscr{C}, f_m) : m \in \mathbb{N}\}$, it is easy to define a similar property for more general families. For example, the family may be parameterized by a positive real number α. This form was chosen for simplicity, but it is easy to produce a more general definition under which all of the results still hold.

6.6.2 Convergence of Proportions

We can provide conditions under which the proportion of observations in each representative region converges to the probability of receiving an observation in that region. If a finite set of representatives is fixed and the sample size is increased, then the law of large numbers would imply that the proportion of observations in each region would converge almost certainly to the corresponding true probability. However, the representative regions from an RRS method are not fixed. Their selection depends on the sample, and the regions need not converge to any particular set of regions as the sample increases in size. Regardless, we can provide conditions under which the proportions will converge.

First, we show that if a set of regions is taken from a uniform-scale class that is uniformly continuous with respect to P_X, then the proportion of observations in each of these regions will converge uniformly to the true probability. There is a broad body of research that provides conditions under which empirical probabilities of a class converge uniformly to the true probabilities. For comprehensive overviews, see for example [118] or [119]. To the best of the author's knowledge, the most relevant result provides sufficient and necessary conditions that a class of regions achieves convergence that is uniform both over the elements of the class and over a set of probability measures. See [120] for an English translation of this result. We present part of this result briefly. Let there be a class $\mathscr{C}$ of regions in some space M. For a finite sample $S \subseteq M$, the class $\mathscr{C}$ is said to *shatter* S if for any subset $T \subseteq S$, there

exists an element C in $\mathscr{C}$ such that C covers exactly the points in T; that is,

$$C \cap S = T.$$

For example, let $\mathscr{C}$ be the set of intervals in $\mathbb{R}$. Then $\mathscr{C}$ shatters any two-point subset of $\mathbb{R}$, because there is always an interval that contains one point but not the other, there is an interval that contains both the points, and there is an interval that contains none of the points. On the other hand, $\mathscr{C}$ does not shatter any three distinct points in $\mathbb{R}$, because there is no interval that contains the largest and smallest points and that excludes the middle point. It is said that a class $\mathscr{C}$ has *Vapnik-Chervonenkis (VC) dimension* of k if $\mathscr{C}$ shatters at least one set of size k and does not shatter any larger set. Continuing the example, the set of intervals on $\mathbb{R}$ would have VC dimension of two. If a class $\mathscr{C}$ has finite VC dimension, then the empirical probability converges to the true probability uniformly across this class [120].

For classes of regions known to have finite VC dimension, then the existing result would provide convergence stronger than that described in Theorem 15. For example, the hyper-rectangles of $\mathbb{R}^n$ and the balls of $\mathbb{R}^n$ under the Euclidean distance are known to have finite dimension, so in these cases the existing result could be applied. It is not generally a trivial problem to identify the VC dimension of a class of regions or to establish whether the dimension is finite. The results produced here depend on properties that are simpler to demonstrate, and we believe that this will allow easier application for a wide choice of representative regions. The author is not aware of any existing result that would demonstrate that a uniform-

scale class would have finite VC dimension, nor is the author aware of any existing result that would immediately provide the convergence described in Theorem 15 for uniform-scale classes of regions.

Theorem 15. *Let $(\mathscr{C}, \Theta, R)$ be a uniform-scale class on M that is uniformly continuous with respect to P_X, and suppose P_X has compact support K with $P_X(K) = 1$. Then*

$$\sup_{R(\theta,\lambda)\in\mathscr{C}} \left| \hat{P}_{X,n}(R(\theta,\lambda)) - P_X(R(\theta,\lambda)) \right| \underset{a.s.}{\to} 0.$$

This is exactly the statement that $\mathscr{C}$ forms a Glivenko-Cantelli class with respect to P_X.

Proof. It is sufficient to prove the case that $M = K$. Suppose that the theorem holds in this case. Then, for the case that $K \subset M$, simply define a new class $(\mathscr{C}', \Theta, R')$ on K by $R'(\theta, \lambda) = R(\theta, \lambda) \cap K$. Then, by our supposition, it is true that

$$\sup_{R'(\theta,\lambda)\in\mathscr{C}'} \left| \hat{P}_{X,n}(R'(\theta,\lambda)) - P_X(R'(\theta,\lambda)) \right| \underset{a.s.}{\to} 0.$$

Then, since $P_X(K) = 1$, it is true that $P_X(R(\theta,\lambda)) = P_X(R'(\theta,\lambda))$ and that $\hat{P}_{X,n}(R(\theta,\lambda)) = \hat{P}_{X,n}(R'(\theta,\lambda))$, so it must also be true that

$$\sup_{R(\theta,\lambda)\in\mathscr{C}} \left| \hat{P}_{X,n}(R(\theta,\lambda)) - P_X(R(\theta,\lambda)) \right| \underset{a.s.}{\to} 0.$$

Thus, let us suppose that $M = K$. Let there be $\epsilon > 0$. Since $(\mathscr{C}, \Theta, R)$ is uniform scale and is uniformly continuous with respect to P_X, then there exists a

$\delta > 0$ and a $\nu > 0$ such that for any θ and λ the following hold:

$$P_X(R(\theta, \lambda + \delta)) - P_X(R(\theta, \lambda)) < \frac{\epsilon}{2},$$

$$P_X(R(\theta, \lambda)) - P_X(R(\theta, \lambda - \delta)) < \frac{\epsilon}{2},$$

$$B[x, \nu] \subseteq R(\theta, \lambda + \delta) \text{ for all } x \in R(\theta, \lambda),$$

$$B[x, \nu] \subseteq R(\theta, \lambda) \text{ for all } x \in R(\theta, \lambda - \delta).$$

Since M is compact, there exists a finite set $A \subseteq M$ such that

$$M \subseteq \bigcup_{a \in A} B\left[a; \nu\right].$$

Let $\mathscr{B}$ be the set of balls:

$$\mathscr{B} = \{B[a; \nu] : a \in A\}.$$

For each set $R(\theta, \lambda) \in \mathscr{C}$, let us define:

$$\hat{R}^{\text{lower}}(\theta, \lambda) = \bigcup_{\substack{\beta \in \mathscr{B}: \\ \beta \subseteq R(\theta, \lambda)}} \beta,$$

$$\hat{R}^{\text{upper}}(\theta, \lambda) = \bigcup_{\substack{\beta \in \mathscr{B}: \\ \beta \cap R(\theta, \lambda) \neq \emptyset}} \beta.$$

That is, $\hat{R}^{\text{lower}}(\theta, \lambda)$ is an approximation of the region $R(\theta, \lambda)$ formed by taking the union of all balls β that fall completely within $R(\theta, \lambda)$, while $\hat{R}^{\text{upper}}(\theta, \lambda)$ is an approximation formed by taking the union of all the balls β that intersect with $R(\theta, \lambda)$. Note that

$$\left\{\hat{R}^{\text{lower}}(\theta, \lambda) : R(\theta, \lambda) \in \mathscr{C}\right\}$$

and

$$\left\{\hat{R}^{\text{upper}}(\theta,\lambda) : R(\theta,\lambda) \in \mathscr{C}\right\}$$

are finite sets, since there are only finitely many unions of a finite collection of sets. Then, by the strong law of large numbers we know that almost surely there exists an N such that for all $n > N$ and for all $R(\theta,\lambda) \in \mathscr{C}$,

$$\begin{aligned}
\hat{P}_{X,n}\left(\hat{R}^{\text{lower}}(\theta,\lambda)\right) &\geq P_X(\hat{R}^{\text{lower}}(\theta,\lambda)) - \frac{\epsilon}{2}, \\
\hat{P}_{X,n}\left(\hat{R}^{\text{upper}}(\theta,\lambda)\right) &\leq P_X(\hat{R}^{\text{upper}}(\theta,\lambda)) + \frac{\epsilon}{2}.
\end{aligned}$$

Further note that

$$R(\theta,\lambda-\delta) \subseteq \hat{R}^{\text{lower}}(\theta,\lambda) \subseteq R(\theta,\lambda) \subseteq \hat{R}^{\text{upper}}(\theta,\lambda) \subseteq R(\theta,\lambda+\delta).$$

Then, we have for any $n > N$

$$\begin{aligned}
\hat{P}_{X,n}\left(R(\theta,\lambda)\right) &\geq \hat{P}_{X,n}\left(\hat{R}^{\text{lower}}(\theta,\lambda)\right) \\
&\geq P_X\left(\hat{R}^{\text{lower}}(\theta,\lambda)\right) - \frac{\epsilon}{2} \\
&\geq P_X\left(R(\theta,\lambda-\delta)\right) - \frac{\epsilon}{2} \\
&\geq P_X(R(\theta,\lambda)) - \epsilon.
\end{aligned}$$

and similarly,

$$\begin{aligned}
\hat{P}_{X,n}(R(\theta,\lambda)) &\leq \hat{P}_{X,n}\left(\hat{R}^{\text{upper}}(\theta,\lambda)\right) \\
&\leq P_X\left(\hat{R}^{\text{upper}}(\theta,\lambda)\right) + \frac{\epsilon}{2} \\
&\leq P_X\left(R(\theta,\lambda+\delta)\right) + \frac{\epsilon}{2} \\
&\leq P_X(R(\theta,\lambda)) + \epsilon.
\end{aligned}$$

This completes the proof that $\mathscr{C}$ is a Glivenko-Cantelli class.

□

This immediately implies that if an RRS method chooses its regions from a class that satisfies the given regularity conditions, then proportions of observations falling in the representative regions will converge to the true probabilities.

Corollary 3. *Let $(f, \mathscr{C})$ be a representative-generating function on (M, d) where $(\mathscr{C}, \Theta, R)$ is a uniform-scale class that is uniformly continuous with respect to P_X. If P_X has compact support K with $P_X(K) = 1$ then*

$$\sup_{A \in f(S_n)} \left| \hat{P}_{X,n}(A) - P_X(A) \right| \to_{a.s.} 0.$$

6.6.3 Convergence of the Coverage Region

Here, we discuss convergence results that compare the coverage region of the RRS method with the support of the probability measure P_X. We show that if the method is sample-covering, then the coverage region will eventually have full measure under P_X. This guarantees that the representatives will eventually cover the entire region in which observations may occur. As a lemma, we first show that the distance from any point of positive measure to the closest point covered by the representatives will almost surely converge to zero.

Lemma 14. *Let $(\mathscr{C}, f)$ be a sample-covering representative-generating function. Let K be any compact subset of the support of P_X. Then*

$$\sup \{\inf \{d(x, z) : z \in C_n(f)\} : x \in K\} \overset{a.s.}{\to} 0.$$

Proof. Let there be $\epsilon > 0$. Since K is compact, there is a finite set $\{a_1, ..., a_m\} \subseteq K$ such that the set of balls

$$\left\{ B\left[a_i; \frac{\epsilon}{2}\right] : i = 1, ..., m \right\}$$

covers K. Since K is a subset of the support of P_X, then for each i we know that $P_X\left(B\left[a_i, \epsilon/2\right]\right) > 0$. Thus, almost surely, eventually every ball $B\left[a_i; \epsilon/2\right]$ will receive at least one observation $\hat{x}_i$. Now, consider some $x \in K$. We know that x falls into a ball $B\left[a_i; \epsilon/2\right]$ for some i. The distance from x to $\hat{x}_i$ must be less that ϵ, since that is the diameter of the ball and $\hat{x}_i$ also falls into the ball. Since $\hat{x}_i$ is an observed data point and the RRS method is sample-covering, then $\hat{x}_i$ is within the coverage region of the RRS method. Thus, the point x can be a distance of at most ϵ from the coverage region of the representatives. □

If a set K will eventually be covered by our representatives, then we would expect that the probability of receiving an observation that is not covered by the representatives but is in the set K should go to zero. Theorem 16 confirms this intuition.

Theorem 16. *Let $(\mathscr{C}, f)$ be a sample-covering representative-generating function on (M, d) where $(\mathscr{C}, \Theta, R)$ is a uniform-scale class that is uniformly continuous with respect to P_X. If $(\mathscr{C}, f)$ has asymptotically bounded cardinality, then*

$$P_X(K \setminus C_n(f)) \overset{a.s.}{\to} 0.$$

for any compact set K in the support of P_X.

Proof. Since $(\mathscr{C}, f)$ has asymptotically bounded cardinality, then there exists some N_0 such that $|f(S_n)| \leq u$ for any $n > N_0$. Let there be $\epsilon > 0$. Since $(\mathscr{C}, \Theta, R)$ is uniform-scale and is uniformly continuous with respect to P_X then we know that there is a δ such that

$$P_X(R(\theta, \lambda + \delta)) - P_X(R(\theta, \lambda)) < \frac{\epsilon}{u}.$$

for any $R(\theta, \lambda) \in \mathscr{C}$ and there exists a ν such that $B(x, \nu) \subseteq R(\theta, \lambda + \delta)$ for any $x \in R(\theta, \lambda)$.

By the above property and by Lemma 14, we know that almost surely there exists some N_1 such that for all $n > N_1$,

$$K \subseteq \bigcup_{R(\theta,\lambda) \in f(S_n)} R(\theta, \lambda + \delta).$$

Using the definition of $C_n(f)$,

$$\begin{aligned} K \setminus C_n(f) &\subseteq \left(\bigcup_{R(\theta,\lambda) \in f(S_n)} R(\theta, \lambda + \delta) \right) \setminus \left(\bigcup_{R(\theta,\lambda) \in f(S_n)} R(\theta, \lambda) \right) \\ &\subseteq \bigcup_{R(\theta,\lambda) \in f(S_n)} R(\theta, \lambda + \delta) \setminus R(\theta, \lambda). \end{aligned}$$

Then,

$$\begin{aligned} P_X(K \setminus C_n(f)) &\leq \sum_{R(\theta,\lambda) \in f(S_n)} P_X(R(\theta, \lambda + \delta)) - P_X(R(\theta, \lambda)) \\ &< \sum_{R(\theta,\lambda) \in f(S_n)} \frac{\epsilon}{u} \\ &= \frac{\epsilon |f(S_n)|}{u} \\ &\leq \epsilon. \end{aligned}$$

This completes the proof. □

We have shown that any compact subset of the support will be covered by the representatives. Thus, if the support itself is compact, then it will be covered.

Corollary 4. *Suppose that P_X has compact support K with $P_X(K) = 1$. Let $(\mathscr{C}, f)$ be a sample-covering representative-generating function on (M, d) where $(\mathscr{C}, \Theta, R)$ is a uniform-scale class that is uniformly continuous with respect to P_X. If f has asymptotically bounded cardinality, then*

$$P_X(C_n(f)) \xrightarrow{a.s.} 1.$$

We have shown that under reasonable assumptions, the coverage region will become large enough that the probability of an observation falling in the coverage region will become arbitrarily close to one. However, this leaves open the possibility that the coverage region may be larger than necessary. In general, RRS methods may have constraints that ensure their usability by a practitioner. For example, there are limits on the number of representative regions that a human can practically look at, so an RRS method may have bounds on the number of regions. These constraints may prevent the coverage region from converging to the support of the probability measure. On the other hand, if these constraints were gradually loosened, then these coverage regions can become increasingly accurate. We use arbitrarily precise families of functions to make this idea rigorous. These families can achieve a type of convergence.

Theorem 17. *Suppose P_X has compact support K with $P_X(K) = 1$. Let $\{(\mathscr{C}, f_m) : m \in \mathbb{N}_0\}$ be an arbitrarily precise family. If each method $(\mathscr{C}, f_m)$ is sample-covering*

and sample-intersecting, then

$$\lim_{m\to\infty}\left(\limsup_{n\to\infty} d_H(C_n(f_m), K)\right) = 0$$

almost surely, where d_H is the Hausdorff distance of the metric space (M, d).

Proof. Let there be $\epsilon > 0$. From the definition of arbitrarily precise, we know that there exists some N_1 such that for all $m > N_1$, f_m has diameter asymptotically bounded by ϵ. Consider some $m > N_1$ and some $x \in C_n(f_m)$. We will show that

$$\limsup_{n\to\infty} d_H\left(C_n(f_m), K\right) \leq \epsilon.$$

Using Lemma 14 and the fact that $(\mathscr{C}, f_m)$ has asymptotically bounded diameter, we know that there almost surely exists an N_2 such that for all $n > N_2$

$$\sup\{\inf\{d(x, z) : z \in C_n(f_m)\} : x \in K\} \leq \epsilon.$$

and such that $d(x, y) \leq \epsilon$ for any $d(x, y) \in A$ and for any $A \in f_m(S_n)$. Consider some $n > N_2$ and $z \in C_n(f_m)$. Then $x \in A$ for some A in $f_m(S_n)$. Since f_m is sample-intersecting, then there exists some $x \in S_n \cap A$. Since $P_X(K) = 1$, then the set S_n is almost surely a subset of K. Since $z \in A$ and $x \in A$, then $d(x, z) \leq \epsilon$. Thus, for any $z \in C_n(f_m)$ there exists a $x \in K$ such that $d(x, z) \leq \epsilon$, which implies that

$$\sup\{\inf\{d(x, z) : x \in K\} : z \in C_n(f_m)\} \leq \epsilon.$$

This completes the proof. □

While a single representative-generating function may have a coverage region that is larger than necessary, this corollary states that it is possible to achieve

a coverage region that is arbitrarily close to the support of P_X by choosing a representative-generating function that is sufficiently far along in the family.

6.6.4 Density Estimation with RRS methods

Any RRS method can be used to create an estimate of the density of our population. For some representative-generating function f, some sample S and some $x \in M$, let $c(S, x; f)$ be the set of representatives that cover x when f is applied to the sample S. That is,

$$c(S, x; f) = \{A \in f(S) : x \in A\}.$$

We will use the notation

$$c_n(x; f) = c(S_n, x; f).$$

Suppose that we have the sample S_n of first n observations, and we want to estimate the value of the p.d.f. $g_X(x)$ for some point x in M. We can use the representative-generating function f to provide an estimate $\hat{g}_n(x; f)$ for the value of $g_X(x)$ with

$$\hat{g}_n(x; f) = \begin{cases} \dfrac{1}{|c_n(x; f)|} \displaystyle\sum_{A \in c_n(x;f)} \frac{\hat{P}_{X,n}(A)}{\mu(A)} & \text{if } c_n(x; f) \neq \emptyset; \\ 0 & \text{if } c_n(x; f) = \emptyset. \end{cases}$$

In general, instead of taking the unweighted average of the terms summed in the definition, any convex combination could be used. This density estimate is not necessarily a proper density in that it might not integrate to 1. This is not uncommon in density estimation, and there is existing work on generating a bona fide density from such an estimate ([121, 122, 123]).

As before, an RRS method may have constraints on the number of representatives produced or may have other constraints, since these methods are designed to produce descriptions that can be viewed by a human. For this reason, a particular representative-generating function may not produce a consistent estimation of the density. Again, families of such RRS methods can be shown to produce this consistent estimation. Also, there may be uncovered regions of the support. The estimate of the density in these regions is zero under the provided method. From the results in Section 6.6.3, the measure of these regions will approach zero eventually. Nevertheless, the existence of these regions contributes to a lack of uniform convergence of the density. While the density may not converge uniformly over the support of the random variable, we can show that the densities provided by our method will exhibit a type of uniform convergence within the coverage region.

Theorem 18. *Let the density g_X have compact support K with $P_X(K) = 1$. Let $\{(\mathscr{C}, f_m) : m \in \mathbb{N}\}$ be an arbitrarily precise family of RRS methods on M for some $\mathscr{C}$ where $(\mathscr{C}, \Theta, R)$ is a uniform-scale class and is uniformly continuous with respect to P_X. Suppose that each $(\mathscr{C}, f_m)$ is uniformly positive with respect to μ. If g_X is continuous on M, then*

$$\lim_{m\to\infty}\left(\limsup_{n\to\infty}\left(\sup_{x\in C_n(f_m)}|g_X(x)-\hat{g}_n(x;f_m)|\right)\right)=0$$

almost surely.

Proof. Since g_X is continuous and K is compact, then g_X is uniformly continuous

on K. Then there exists some δ such that if $d(x,y) < \delta$ then

$$|g_X(x) - g_X(y)| \leq \frac{\epsilon}{2}.$$

Since f_m is arbitrarily precise, there exists some N_1 such that for all $m > N_1$, each f_m has diameter asymptotically bounded by δ. Take some $m > N_1$ and $x \in C_n(f_m)$. Then

$$\begin{aligned}
|\hat{g}_n(x; f_m) - g_X(x)| &= \left| \frac{1}{|c_n(x; f_m)|} \sum_{A \in c_n(x; f_m)} \frac{\hat{P}_{X,n}(A)}{\mu(A)} - g_X(x) \right| \\
&\leq \left| \left(\frac{1}{|c_n(x; f_m)|} \sum_{A \in c_n(x; f_m)} \frac{\hat{P}_{X,n}(A)}{\mu(A)} \right) - \left(\frac{1}{|c_n(x; f_m)|} \sum_{A \in c_n(x; f_m)} \frac{\hat{P}_X(A)}{\mu(A)} \right) \right| \\
&+ \left| \frac{1}{|c_n(x; f_m)|} \sum_{A \in c_n(x; f_m)} \frac{P_X(A)}{\mu(A)} - g(x) \right|. \qquad (6.5)
\end{aligned}$$

Let's consider the first term of the right-hand side of equation 6.5:

$$\begin{aligned}
\text{Term 1} &= \left| \left(\frac{1}{|c_n(x; f_m)|} \sum_{A \in c_n(x; f_m)} \frac{\hat{P}_{X,n}(A)}{\mu(A)} \right) - \left(\frac{1}{|c_n(x; f_m)|} \sum_{A \in c_n(x; f_m)} \frac{\hat{P}_X(A)}{\mu(A)} \right) \right| \\
&\leq \frac{1}{|c_n(x; f_m)|} \sum_{A \in c_n(x; f_m)} \frac{\left| \hat{P}_{X,n}(A) - P_X(A) \right|}{\mu(A)}.
\end{aligned}$$

Since $(\mathscr{C}, f_m)$ is uniformly positive with respect to μ, then there exists some constant β such that $\mu(A) \geq \beta$ for all $A \in \mathscr{C}$. Then,

$$\text{Term 1} \leq \frac{1}{|c_n(x; f_m)|} \sum_{A \in c_n(x; f_m)} \frac{\left| \hat{P}_{X,n}(A) - P_X(A) \right|}{\beta}.$$

By Corollary 3, there almost surely exists some N_2 such that for any $n > N_2$,

$$\sup_{A \in \mathscr{C}} \left| \hat{P}_{X,n}(A) - P_X(A) \right| < \frac{\epsilon\beta}{2}.$$

so that

$$\begin{aligned}\text{Term 1} &\leq \frac{1}{|c_n(x; f_m)|} \sum_{A \in c_n(x; f_m)} \frac{\epsilon}{2} \\ &= \frac{\epsilon}{2}.\end{aligned}$$

Now, let's look at the second term:

$$\begin{aligned}\text{Term 2} &= \left| \frac{1}{|c_n(x; f_m)|} \sum_{A \in c_n(x; f_m)} \frac{P_X(A)}{\mu(A)} - g_X(x) \right| \\ &= \left| \frac{1}{|c_n(x; f_m)|} \sum_{A \in c_n(x; f_m)} \frac{1}{\mu(A)} \left(P_X(A) - g_X(x) \int_A d\mu \right) \right|.\end{aligned}$$

Now, using the definition of a density,

$$\begin{aligned}\text{Term 2} &= \left| \frac{1}{|c_n(x; f_m)|} \sum_{A \in c_n(x; f_m)} \frac{1}{\mu(A)} \int_A (g_X - g_X(x))\, d\mu \right| \\ &\leq \frac{1}{|c_n(x; f_m)|} \sum_{A \in c_n(x; f_m)} \frac{1}{\mu(A)} \int_A |g_X - g_X(x)|\, d\mu.\end{aligned}$$

Since $(\mathscr{C}, f_m)$ has diameter asymptotically bounded by δ, then there exists an $n > N_3$ such that $d(y, z) < \delta$ for any $y, z \in A$ and any $A \in f_m(S_n)$. Then, $|g_X(y) - g_X(x)| \leq \frac{\epsilon}{2}$ for any $y \in A$ and for any region $A \in c_n(x, f_m)$. Then,

$$\begin{aligned}\text{Term 2} &\leq \frac{1}{|c_n(x; f_m)|} \sum_{A \in c_n(x; f_m)} \frac{1}{\mu(A)} \int_A \frac{\epsilon}{2} du \\ &= \frac{\epsilon}{2}.\end{aligned}$$

Combining these bounds on terms 1 and 2 of equation 6.5 gives us that

$$|\hat{g}_n(x; f_m) - g_X(x)| \leq \epsilon,$$

which completes the proof. □

Without too much additional work, Theorem 18 and Corollary 4 can be used to provide some sufficient conditions under which the asymptotic total integrated error can be made arbitrarily small.

Theorem 19. *Let the density g_X have support K, and let g_X be continuous on K. Suppose that K is compact with $P_X(K) = 1$ and that there exists some $\delta > 0$ such that*

$$\mu(\{x : x \in M, d(x, y) \leq \delta \text{ for some } y \in K\})$$

is finite. Let $\{(\mathscr{C}, f_m) : m \in \mathbb{N}\}$ be an arbitrarily precise family of representative-generating functions for some $\mathscr{C}$ where $(\mathscr{C}, \Theta, R)$ is a uniform-scale class and is uniformly continuous with respect to P_X. Suppose that each $(\mathscr{C}, f_m)$ is sample-covering, sample-intersecting, has asymptotically bounded cardinality, and is uniformly positive with respect to μ. Let $\hat{g}_{n,m}$ be the function $\hat{g}_n(\cdot; f_m)$.

$$\lim_{m\to\infty} \left(\limsup_{n\to\infty} \int_M |g_X - \hat{g}_{n,m}| \, d\mu \right) = \epsilon$$

almost surely.

Proof. From the assumptions, there exists some δ such that

$$\mu\left(\{x : x \in M, d(x, y) \leq \delta \text{ for some } y \in K\}\right) = C^*$$

for some finite constant C^*. Let there be some $\epsilon > 0$. Using Theorem 18 and the fact that $\{(\mathscr{C}, f_m) : m \in \mathbb{N}\}$ is arbitrarily precise, there exists some N_1 such that for all $m > N_1$,

$$\limsup_{n\to\infty} \left(\sup_{x \in C_n(f_\theta)} |g_X(x) - \hat{g}_{n,m}(x)| \right) < \frac{\epsilon}{C^*}$$

almost surely and f_m is asymptotically bounded by $\frac{\delta}{2}$. Since $(\mathscr{C}, f_m)$ is sample-intersecting and has diameter asymptotically bounded by $\frac{\delta}{2}$, then there almost surely exists some N_2 such that for all $n > N_2$,

$$C_n(f_m) \subseteq \mu\left(\{x : x \in M, d(x,y) \leq \delta \text{ for some } y \in K\}\right)$$

and therefore,

$$\mu\left(C_n(f_m)\right) \leq C^*.$$

Then consider the integrated absolute error:

$$\begin{aligned}\int_M |g_X - \hat{g}_{n,m}| d\mu = &\int_{M \setminus (C_n(f_m) \cup K)} |g_X - \hat{g}_{n,m}| d\mu + \int_{C_n(f_m)} |g_X - \hat{g}_{n,m}| d\mu \\ &+ \int_{K \setminus C_n(f_m)} |g_X - \hat{g}_{n,m}| d\mu.\end{aligned}$$

Note that g_X is zero on the set $M \setminus K$, that $\hat{g}_{n,m}$ is zero on the set $M \setminus C_n(f_m)$, and that both of these functions are zero on $M \setminus (C_n(f_m) \cup K))$. Thus,

$$\begin{aligned}\int_M |g_X - \hat{g}_{n,m}| d\mu &= \int_{C_n(f_m)} |g_X - \hat{g}_{n,m}| d\mu + \int_{K \setminus C_n(f_m)} g_X d\mu \\ &< \int_{C_n(f_m)} \frac{\epsilon}{C^*} d\mu + \int_{K \setminus C_n(f_m)} g_X d\mu \\ &= \epsilon + \int_{K \setminus C_n(f_m)} g_X d\mu.\end{aligned}$$

Let G^* be the maximum value that g_X takes on K. Since K is compact and g_X is continuous on K, such a value exists by the extreme value theorem. Then,

$$\int_M |g_X - \hat{g}_{n,m}| < \epsilon + G^* P_X(K \setminus C_n(f_m)).$$

By Corollary 4, we know that

$$\lim_{n \to \infty} P_X(K \setminus C_n(f_m)) = 0.$$

Thus,

$$\limsup_{n\to\infty} \int_M |g_X - \hat{g}_{n,m}| d\mu < \epsilon.$$

□

Thus, if we have an arbitrarily precise family of representative-generating functions, we can make our integrated absolute error arbitrarily small by choosing a representative-generating function that is sufficiently far along in the family.

6.7 Convergence of Specific Methods

In this section, we apply the results in Section 6.6 to specific methods that fall within the RRS framework. Specifically, we discuss the application of these results to multivariate histograms, density estimation trees, and the two methods constructed in Section 6.5. Convergence results are already known for multivariate histograms and density estimation trees. Our results in these cases admittedly do not greatly improve the results already known in the literature. However, we include these methods as a demonstration of the general applicability of the RRS framework. The convergence results for the two remaining methods are new. Throughout this section, we will use similar notation and assumptions as in Section 6.6.1.

6.7.1 Multivariate Histograms

Multivariate histograms are a well-known and well-studied method for density estimation in multiple dimensions. See for example Chapter 3 of [89]. Multivariate histograms partition the space of observations into subsets called bins. The

estimated density is constant within each bin. For a given bin, the corresponding density is defined to be the proportion of observations that fall within that bin divided by the volume of the bin.

This method can be placed within the RRS framework. Specifically, for a histogram with a given choice of bins, the corresponding RRS method selects those bins of the histogram in which observations occur. We do not especially recommend this as an RRS method because it is difficult to control the number of representative regions produced by this method, and the method is sensitive to the choice of bins. Regardless, it does serve as an example of the breadth of our results. The histogram RRS method can be defined as follows. Suppose that the space in which observations occur is a hyper-rectangle. A binning is a partition of the hyper-rectangle into many smaller hyper-rectangles formed by separating each interval $[l_i, u_i]$ into smaller intervals. This can be formalized by the following definition.

Definition 31 (Binning)**.** *Let $\mathscr{R}$ be a hyper-rectangle*

$$\mathscr{R} := \prod_{i=1}^{n} [l_i, u_i]$$

*for some $l_1, ..., l_n$ and $u_1, ..., u_n$. A **binning** $\mathcal{B}$ is a set of the form*

$$\mathcal{B} := \left\{ \prod_{i=1}^{n} A_i : (A_1, ..., A_n) \in I_1 \times ... \times I_n \right\}$$

for n sets $I_1, ..., I_n$ where each I_i is of the form

$$I_i = \{[l_i, a(i,1)], [a(i,1), a(i,2)], ...[a(i,k_i), u_i]\}$$

for some $a(i,1), ..., a(i,k_i)$.

Each binning induces a different RRS method, where the selected regions are the bins of the histogram in which observations occur. This method is

Definition 32 (Histogram RRS)**.** *Let $\mathcal{B}$ be a binning of the hyper-rectangle $\mathscr{R}$. Then the* ***histogram RRS*** *with binning $\mathcal{B}$ is the RRS method $(\mathcal{B}, f_{\mathcal{B}})$ defined by*

$$f_{\mathcal{B}}(S) = \{\beta \in \mathcal{B} : \beta \bigcap S \neq \emptyset\}.$$

Suppose that the same assumptions are made concerning the random observations as were made in Section 6.6. Let μ be the standard reference measure on $\mathbb{R}^n$, and suppose that P_X has a bounded density g_X that is continuous on $\mathscr{R}$. Then, as discussed in Section 6.6.1 the set of hyper-rectangles in $\mathscr{R}$ is a uniform-scale class that is uniformly continuous with respect to P_X. Thus, the proportion of observations falling in each bin of the histogram will converge to the true probability by Corollary 3. Suppose there is a sequence $\mathcal{F} = \{f_n : n \in \mathbb{N}\}$ where f_n is the histogram RRS corresponding to some binning $\mathcal{B}_n$. Further suppose that the maximum size of a bin in $\mathcal{B}_n$ approaches 0 as n approaches infinity. Then, the family $\mathcal{F}$ is an arbitrarily precise family. Furthermore, each f_n is sample-covering, sample-intersecting and has bounded cardinality by definition. Thus, we can apply Theorem 17 and it is true that the union of the non-empty bins of the histogram will converge to the true support of the random variable. Assuming that each bin of each binning has positive measure, then Theorem 18 guarantees that as the bin size is reduced, the density estimate of the histogram becomes consistent in terms of integrated absolute error, and Theorem 19 guarantees that the density estimate converges uniformly over the non-empty bins of the histogram.

6.7.2 Density Estimation Trees

Similarly to a histogram, a density estimation tree (DET) partitions the dataset into hyper-rectangles. In a DET, each of these regions corresponds to a leaf in a binary tree. As in a histogram, the estimated density within a leaf region is defined to be the proportion of observations that fall within the leaf divided by the volume. The details of how the tree is built are not important for our analysis, so we omit a presentation of these details. There are some parameters that affect how the tree is constructed; one such parameter is the minimum number of observations that must fall within each leaf region.

As with multivariate histograms, a DET method can be defined to be an RRS method $(\mathscr{C}, f)$ where $\mathscr{C}$ is the set of hyper-rectangles and the representative regions are simply the leaf regions of the tree. Again, if the observations are i.i.d. variables with a bounded, continuous density and with full compact support, then the regularity conditions will be satisfied, and the proportion of observations falling within each leaf will converge to the true probability. Since a DET partitions the region of possible values, then this method must be sample-covering. If a positive minimum number of leaf observations is set, then the method is also sample-intersecting. Let $\{(\mathscr{C}, f_m) : m \in \mathbb{N}\}$ be a family of density estimation tree methods, where each f_m has a bound $[l_m, u_m]$ on the diameter of each leaf region and where each l_m is positive. This implies that each f_m has bounded cardinality. Assume that u_m and l_m both approach zero as m approaches infinity. In other words, the leaf regions must become arbitrarily small, which is an assumption similar to the one made in [68].

Then by Theorem 19 the density estimate of this family will converge.

6.7.3 MDS method

In Section 7.4.1, we discuss an RRS method that involves solving a minimum dominating set (MDS) problem. This MDS problem may be solved exactly or with any approximation algorithm. First, we define a *DS method* to be a method that produces representatives that form a (not necessarily minimal) dominating set.

Definition 33 (DS method)**.** *Let (M, d) be a metric space, and let $\mathscr{C}$ be the set of balls on this space. Let α be a positive real number. An RRS method $(\mathscr{C}, f)$ is a* ***DS(α)*** *method if for any sample S,*

$$f(S) = \{B(y, \alpha) : y \in Y\}$$

where Y is a dominating set for the α-neighborhood graph on S.

We can now define a MDS method to be one that approximately solves an MDS problem.

Definition 34 (MDS method)**.** *Let (M, d) be a metric space, and let $\mathscr{C}$ be the set of balls on this space. Let α and ν be positive real numbers. For any finite sample S from M, let $\gamma(S; \alpha)$ be the size of a minimum dominating set for the α-neighborhood graph of S. An RRS method $(\mathscr{C}, f)$ is an* ***MDS(ν, α) method*** *if $(\mathscr{C}, f)$ is a DS(α) method and*

$$|f(S)| \leq \nu\gamma(S; \alpha)$$

for any finite sample S from M.

Under mild regularity conditions, then by Corollary 3 the proportion of observations falling in each of these balls will converge to the true probability. As noted in Section 6.6.1 these regularity conditions hold if the metric space is $\mathbb{R}^d$ under any p-norm and if the observed variables have a bounded, continuous density with a full, compact support. Let there be a sequence $(\alpha_1, \alpha_2, ...)$ where $\lim_{n\to\infty} \alpha_n = 0$. Let there be some $\nu > 0$ and let $\{f_m : m \in \mathbb{N}\}$ be a family of representative-generating functions where f_m is a MDS(ν, α_m) method. By construction, each f_m is sample-intersecting and sample-covering, and this family is arbitrarily precise. Thus, by Theorem 17 the coverage region of this method will converge to the support of the observations. An additional condition required by Theorem 19 is that each f_i has cardinality bounded by some finite number. We can show that this is the case.

Theorem 20. *Let (M, d) be a metric space, let $\mathscr{C}$ be the set of balls on this space, and let $(\mathscr{C}, f)$ be an MDS(ν, α). Suppose that P_X has compact support K with $P_X(K) = 1$. Let*

$$k^*(r) = \min\left\{k : \textit{There exist } k \textit{ points } a_1, ..., a_k \textit{ in } K \textit{ such that } \cup_{i=1}^{k} B\left[a_i; r\right] \supseteq K\right\}.$$

Then f has cardinality that is asymptotically bounded by $\nu k^*(\frac{\alpha}{2})$.

Proof. Since K is compact, then $k^*(\frac{\alpha}{2})$ exists. Let $a_1, ...a_k$ be $k^*(\frac{\alpha}{2})$ points that cover K. By definition of support, there will almost surely be some N such that for $n > N$, the sample S_n will have some point a_i' in each of balls $B\left[a_i; \frac{\alpha}{2}\right]$. Then, note that $B\left[a_i'; \alpha\right]$ covers the entirety of S_n. Thus, a_i' provides a dominating set on the α-neighborhood graph of S_n. The size of this solution is $k^*(\frac{\alpha}{2})$. Thus, the optimal solution of the MDS problem on the α-neighborhood graph of S_n is at most

$k^*(\frac{\alpha}{2})$. By definition of the MDS(ν, α) method, then $|f(S_n)|$ must have size of at most $\nu k^*(\frac{\alpha}{2})$. □

We have shown that each of our representative-generating functions has cardinality that is asymptotically bounded if we are using the MDS method. We may then apply our convergence results from Theorem 19. Since each method produces a set of balls with some fixed radius α, this method will be uniformly positive with respect to most reasonable choices for a reference measure. In particular, this is true using the usual measure on $\mathbb{R}^n$. The other regularity constraints will also hold in this setting as long as the observations have compact support.

Corollary 5. *Let the density g_X have support K, and let g_X be continuous on K. Suppose that K is compact with $P_X(K) = 1$. Let $\mathscr{C}$ be the set of balls formed by the standard Euclidean norm on $\mathbb{R}^n$, and let μ be the usual measure on $\mathbb{R}^n$. Let $(\alpha_1, \alpha_2, ...)$ be a sequence of positive real numbers such that $\alpha_m \to 0$, and let $\{(\mathscr{C}, f_m) : m \in \mathbb{N}\}$ be a sequence of RRS methods where each $(\mathscr{C}, f_m)$ is an MDS(ν, α_m) function. Let $\hat{g}_{n,j}$ refer to the function $\hat{g}_n(\cdot; f_j)$. Then*

$$\lim_{m\to\infty}\left(\limsup_{n\to\infty}\int_{\mathbb{R}^n}|g_X - \hat{g}_{n,j}|\,d\mu\right) = 0$$

almost surely.

Thus, the MDS method in $\mathbb{R}^n$ using Euclidean distance is a consistent density estimation method.

6.7.4 The K-Center Method

In Section 6.5.2, we defined an RRS method that produces representative methods by solving a k-center problem. Again, it is not necessary to solve this problem exactly.

Definition 35 (KC method)**.** *Let (M, d) be a metric space, and let $\mathscr{C}$ be the set of balls on this space. Let k be a positive integer. For any finite sample S from M, let $\gamma(S)$ be the optimal distance in the solution to a k-center problem on S. Then, a $(\mathscr{C}, f)$ RRS method is a **KC(ν,k)** method if $(\mathscr{C}, f)$ is sample-covering and for any finite sample S in M*

$$f(S) = \{B[y, \alpha] : y \in Y\}$$

for some set $Y \subseteq S$ such that $|Y| = k$ and some α such that

$$\alpha \leq \nu\gamma(S).$$

As with the MDS function, with some mild regularity assumptions, the proportion of observations in the representative regions of a KC(ν,k) method will converge to the true proportions.

Consider a family of methods $\{f_m : m \in \mathbb{N}\}$ where each f_m is a KC(ν,m) method. In order to apply our results that guarantee the coverage regions converges to the support and that the density converges, it must be true that the maximum size of the representative regions decreases as the number of allowed representatives increases. We can show that this is true. First, we show that the size of the representative regions for a fixed k is asymptotically bounded.

Lemma 15. *Let (M, d) be a metric space, and let $\mathscr{C}$ be the set of balls on this space. Let k be a positive integer, and let $(\mathscr{C}, f)$ be a KC(ν,k) method. Suppose that P_X has a compact support K with $P_X(K) = 1$. Let*

$$r_k^* = \inf \left\{ r : \textit{There exist } k \textit{ points } a_1, ..., a_k \textit{ in } K \textit{ such that } \cup_{i=1}^k B\left[a_i; r\right] \supseteq K \right\}.$$

For any $\epsilon > 0$, $(\mathscr{C}, f)$ has diameter asymptotically bounded by $[2r_k^* - \epsilon, \nu 2r_k^* + \epsilon]$.

Proof. The existence of r_k^* follows from the fact that K is compact. Let there be $\epsilon > 0$. First, consider the lower bound. Lemma 14 implies that there almost certainly exists some N such that for all $n \geq N$,

$$\bigcup_{A \in f(S_n)} A \supseteq K$$

By our definition of f, the set of representatives $f(S_n)$ must consist of a set of k balls $B[y_1, r'], ..., B[r_k, r']$ where each y_i is in S_n and r' is some positive radius. Since $P_X(K) = 1$, then these representatives will almost surely all be elements of K. Then, using the definition of r_k^*, we have that

$$r' + \frac{\epsilon}{2} \geq r_k^*,$$
$$r' \geq r_k^* - \frac{\epsilon}{2}.$$

The diameter of the ball is at most twice the radius, so this provides the described lower bound. Now, consider the upper bound. By definition of r_k^*, there must exist some k points $a_1, ..., a_k$ in K such that

$$\bigcup_{i=1}^k B\left[a_i; r_k^* + \frac{\epsilon}{4\nu}\right] \supseteq K.$$

Since K is compact, there exists some set of points $b_1, ..., b_l$ in K such that

$$\bigcup_{i=1}^{l} B\left[b_i; \frac{\epsilon}{4\nu}\right] \supseteq K.$$

Since K is the support of X_i, it must be true that each ball $B\left[b_i; \frac{\epsilon}{4\nu}\right]$ has a non-zero chance of receiving an observation. Thus, almost surely there exists some N such that for all $n > N$, there has been an observation in each of these balls. Consider some $n > N$, and consider some a_i. Since the balls $B\left[b_j; \frac{\epsilon}{4\nu}\right]$ cover K, then a_i must fall into one of these balls. As previously mentioned, there must exist at least one observation $x \in S_n$ that falls into the same ball as a_i. Then, the distance between x and a_i is at most $\frac{\epsilon}{4\nu}$. Thus, for each a_i, there is a corresponding observation $x_i \in S_n$ such that $d(a_i, x_i) < \frac{\epsilon}{4\nu}$. Now consider any other observation $x' \in S_n$. By definition of $a_1, ..., a_k$, we know that there exists some a_i such that $d(x', a_i)$ is at most $r_k^* + \frac{\epsilon}{4\nu}$. Then,

$$\begin{aligned} d(x', x_i) &\leq d(x_i, a_i) + d(a_i, x') \\ &\leq \frac{\epsilon}{4\nu} + r_k^* + \frac{\epsilon}{4\nu} \\ &\leq r_k^* + \frac{\epsilon}{2\nu}. \end{aligned}$$

Thus, x_i provides a feasible (but not necessarily optimal) solution to the k-center problem for the sample of the first n observations with optimal distance of at most $r_k^* + \frac{\epsilon}{2\nu}$. Then, since $f(S_n)$ provides a ν-approximation of the k-center problem on S_n, we know that the radius of any representative in $f(S_n)$ is bounded above by $\nu r_k^* + \frac{\epsilon}{2}$. This completes the proof. □

This lemma lets us show that family $\{f_m : m \in \mathbb{N}\}$ is arbitrarily precise.

Theorem 21. *Let (M, d) be a metric space, and let $\mathscr{C}$ be the set of balls on this space. Let there be $\nu > 0$, and let $\{(\mathscr{C}, f_m) : m \in \mathbb{N}\}$ be a family of RRS methods where each f_m is a KC(ν, m) method. Suppose that P_X has a compact support K with $P_X(K) = 1$. Then $\{(\mathscr{C}, f_m) : m \in \mathbb{N}\}$ is arbitrarily precise.*

Proof. The compactness of K implies that for any $\epsilon > 0$, there exists a_1,...,a_m such that

$$\bigcup_{i=1}^{m} B\left[a_i; \epsilon\right] \subseteq K.$$

This in turn implies that for any $\epsilon > 0$, there exists a k such that K can be covered by k balls of radius ϵ. Thus, as k increases, r_k^* approaches zero. The upper bound given in Theorem 15 can therefore be made arbitrarily small as k increases. and the family $\{f_i : i \in \mathbb{N}\}$ is arbitrarily precise. □

By definition, each f_m in the family of approximate k-center methods is sample-covering and sample-intersecting. Since Theorem 21 demonstrates that the family of k-center functions is arbitrarily precise, the convergence results from Section 6.6 apply. In the case that the metric space is $\mathbb{R}^n$ under the usual Euclidean distance and the reference measure is the usual measure, then most of the regularity conditions are also satisfied, though we still require some conditions on the distribution of the observations.

Corollary 6. *Let the density g_X have support K, and let g_X be continuous on K. Let $\mathscr{C}$ be the set of balls on $\mathbb{R}^n$ under the Euclidean norm, and let μ be the usual reference measure on $\mathbb{R}^n$. Let there be $\nu > 0$ and let $\{(\mathscr{C}, f_m) : m \in \mathbb{N}\}$ be a sequence*

of RRS methods where f_m is an KC(ν, k) function. Let $\hat{g}_{n,j}$ refer to the function $\hat{g}_n(\cdot; f_j)$. Then

$$\lim_{m\to\infty}\left(\limsup_{n\to\infty}\int_{\mathbb{R}^n}|g_X-\hat{g}_{n,j}|\,d\mu\right)=0$$

almost surely.

As in the MDS method, the density estimated produced by the family of k-center methods is consistent.

6.8 Conclusion and Further Work

We define and explore a new type of unsupervised learning problem that involves summarizing a data set with a much smaller set of regions that are representative of the original data set. This problem is referred to as the representative region selection (RRS) problem. The notion of a representative quality measure (RQM) is discussed, and axioms for these quality measures are discussed. These axioms provide properties that should be satisfied by any function that measures the quality of a set of representatives. We provide an example of an axiomatic RQM (ARQM), and we propose two methods of generating representative regions based on the ARQM. We provided convergence results for several properties of these regions. We provided conditions under which the proportion of observations in each region converges to the probability of receiving an observation and conditions under which the area covered by the regions converges to the support of the observations. We demonstrated that any RRS method can be used as a method for density estimation, and we provided conditions under which this estimate is consistent. It was shown

(under mild assumptions concerning the distribution of observations) that the two methods proposed here satisfy the conditions for convergence. It was further shown that two existing methods for density estimation can be framed as RRS methods, and that these methods also satisfied the conditions for convergence. Finally, we provided an example demonstrating the use of these methods to explore a dataset.

We show only one function that satisfies the RQM axioms. Many variants of this function would also satisfy the RQM axioms, so this does provide practitioners with some breadth of choice. However, it would be worth producing larger families of functions that satisfy these axioms in order to give more choices for practitioners wishing to create representative-generating functions from an ARQM and apply these functions to a dataset. It would also be worth exploring the theoretical connections between RQM axioms and the consistency results that we provide. We have shown that two methods induced by one particular ARQM satisfy the requirements for convergence. It would be worth examining whether all methods constructed in a similar manner from some ARQM would meet these conditions.

There are also some theoretical questions remaining related to the consistency results provided. While we provide sufficient conditions under which several types of consistency are achieved, it would be beneficial to see whether or not these conditions can be weakened. We also do not provide any results concerning the rate at which convergence occurs. Further work could provide such results.

Chapter 7: Representative Air Traffic Management Initiatives

The material in this chapter appears in the paper, "Unsupervised prototype reduction for data exploration and an application to air traffic management initiatives". This paper is available on SSRN, abstract number 3067091.

In this chapter, we discuss a new approach to unsupervised learning and data exploration that involves summarizing a large data set using a small set of "representative" elements. These representatives may be presented to a user in order to provide intuition regarding the distribution of observations. Alternatively, these representatives can be used as cases for more detailed analysis. We call the problem of selecting the representatives the Unsupervised Prototype Reduction problem. We discuss the KC-UPR method for this problem and compare it to other existing methods that may be applied to this problem. We propose a new type of distance measure that allows for more interpretable presentation of results from the KC-UPR method. We demonstrate how solutions from the Unsupervised Prototype Reduction problem may be used to provide decision support for the planning of air traffic management initiatives, and we produce computational results that compare the effectiveness of several methods in this application. We also provide an example of how the KC-UPR method can be used for data exploration, using data from air

traffic management initiatives at Newark Liberty International Airport.

7.1 Introduction

Congestion in an air transportation system arises when air traffic demand exceeds available airspace capacity. This occurs most often when airspace capacity is reduced due to weather. In the U.S., air traffic management is typically exercised by instituting specific traffic management initiatives (TMIs). For example, a ground delay program (GDP) is a TMI that seeks to balance the arrival flow into an airport with the airport's maximum possible arrival rate. An airspace flow program (AFP) performs a similar function relative to a volume of airspace. Most often the target airport or airspace has a weather-induced capacity reduction. While there is a significant amount of research on optimization models to support TMI planning, e.g. [1, 5, 37, 38, 124], this body of work has so far struggled to find acceptance among personnel that plan traffic management initiatives, and for the most part has not been implemented. There are several reasons for this. Existing stochastic models are idealized and do not fully capture uncertainty in demand and weather. For example, most existing models assume that flight times are certain and that flights depart at the scheduled time. Another difficulty is that while the TMI plan is created by an air navigation service provider (ANSP), e.g. the Federal Aviation Administration (FAA) in the U.S. or Eurocontrol in Europe, flight operators often respond to TMIs by reorganizing their operations. While some TMI planning models seek to take this into account, it is not an easy task to predict flight operator behavior. A further

complication is that some performance criteria implicitly considered by planners may be difficult to formally model. For example, if an airport experiences a high amount of airborne congestion on one day, then FAA decision-makers may choose to implement a more conservative TMI on the next day so that air traffic controllers at the airport do not face an extremely stressful situation two days in a row. It is difficult to integrate these considerations into existing optimization models. Finally, in current practice, a discussion is held between the relevant flight operators and those implementing the traffic management initiatives. This provides an opportunity for flight operators to provide input to the FAA, which existing optimization approaches do not allow.

A more recent body of research has sought to enhance decision support by providing decision makers with relevant historical data, e.g. [125, 126, 127]. These historical data can take the form of "similar days": each such day has demand, capacity, and weather characteristics likely to be similar to the current day within the geographic region for which a TMI plan is sought. Allowing a TMI planner to examine the characteristics of similar historical days, including the impact of any TMI employed, can help improve the quality and objectivity of decision made by the planner. While work on optimization models is ongoing and may eventually lead to a method that overcomes the aforementioned difficulties, methods that focus on improving the availability of data could provide a more palatable manner of improving TMI decision-making in the nearer term.

The research described in this chapter addresses a specific problem that arises in creating the decision support framework described above. Specifically, we seek

to identify a small set of "representative days" from a longer list of "similar days". For example, the number of similar days could exceed 100 or even several hundred but the representative days should be small enough for detailed examination by the TMI planner, e.g. 5 to 8 days. Each of the representative days should represent a (potentially large) subset of similar days in the sense that each day in the subset employed a TMI similar to the one employed by its representative. In this way, the planner can see an example of each type of TMI that might be appropriate on the current day and also the outcome of that TMI.

More generally, the problem we address can be viewed as a new approach to unsupervised learning and data exploration. The problem we address takes as input a "large" dataset and produces a "small" set of representatives that should approximately describe the dataset. This is a similar problem to that of prototype reduction in supervised learning. In that context, the methods are applied to data that have class labels, and the goal is usually to reduce the size of the training set in order to improve computational performance while maintaining the ability of a classifier to predict class labels of new observations [77]. However, unsupervised prototype reduction can be applied to data that lack class labels, and the goal is to help a human to understand the data. The prototype elements can be presented to a practitioner in order to help him or her understand in which general regions of the feature space observations will occur. The practitioner could also use these prototype elements as candidates for more detailed types of analyses, such as case studies or simulations. We refer to this problem as the Unsupervised Prototype Reduction (UPR) problem. Chapter 6 provides theoretical results for a broader

class of methods. The goal of this chapter is to implement these methods in the context of air traffic management, thereby demonstrating both the practicality of such an implementation and the particular effectiveness of one specific method.

7.2 Background and Contributions

The focus of the chapter is on two concepts. The first is how to produce a small set of data and present this set to a human in order to help her or him understand a large set of data. This is the general UPR problem. The second is how the UPR problem can inform air traffic management decisions. In particular, we demonstrate the applicability of a UPR method based on solving a k-center facility location problem. This method is referred to as the KC-UPR method.

7.2.1 Methods For Prototype Reduction.

There is a substantial body of literature involving methods for producing a small data set that approximates a larger data set. This done is in order to reduce computational burden or to decrease sensitivity to noise when a classification technique is applied. These methods have been referred to as *instance selection*, *prototype reduction* or *representative subset selection* methods. The vast majority of existing methods are designed for labeled data sets and are not applicable to unlabeled data. See [77], [78], or [128] for overviews of prototype reduction methods for supervised learning problems.

There is a more limited body of literature that may be applied to unsupervised

learning problems. Several such methods has been used in the context of experimental design [79]. The methods are divided into two categories: cluster-based designs and uniform designs. Cluster-based designs first attempt to partition the data into clusters so that observations within the same cluster are similar while observations in different clusters are dissimilar. The resulting clusters are used to generate representatives. These methods have the disadvantage that they are only sensible if the data exhibit cluster structure. Uniform designs, on the other hand, attempt to select representatives that are distributed uniformly across the observed data set. Such methods are more widely applicable as they do not rely on the existence of cluster structure. Popular uniform design methods include the Kennard-Stone [81] and OptiSim [80] methods. These methods are similar, and each forms a set by selecting prototypes one at a time. Both methods attempt to choose new elements that are dissimilar to those that have already been chosen. A more recently proposed method first identifies outliers and then randomly samples the remaining points [129]. This method was developed for a supervised learning setting, but may be applied in an unsupervised setting.

This work applies the work in Chapter 6. In that chapter, the selected representatives are regions of the data, while here we focus on methods whose representatives can be identified by individual points. In the previous chapter, a mathematical framework is defined that can be used to analyze a broader class of methods and conditions are provided under which these methods have statistical consistency properties. That chapter also defines a more general form of the KC-UPR method and shows that it satisfies those consistency conditions. Once the theoretical be-

havior is established, however, there is still significant effort left in constructing a working practical implementation of the methods. In this chapter, we provide guidance concerning the application of these methods, and we apply these methods to a problem in air traffic management. We also conducted computational experiments that compare the KC-UPR method with the other existing methods. These experiments provide both qualitative and quantitative insight into the performance of these methods.

7.2.2 Methods Resembling Prototype Reduction in Air Traffic Management.

In the air traffic management literature, there have been several instances where a small set of elements has been selected from a larger data set so that the smaller set of elements in some way summarizes the larger set. The most popular selection method has been to use the centroids produced by the k-means clustering algorithm. In [130], the k-means centroids were used to produce a small set of capacity vectors. These were intended for use as scenarios in a stochastic optimization problem. Similarly, [131] and [132] used methods based on k-means centroids in order to produce scenarios for simulation studies. In order to identify "typical" days in the national airspace with respect to the causes and locations of GDPs, [133] also used k-means centroids.

However, this approach has some downsides. The k-means centroids are generally not members of the original data set. In many applications, this makes the

representatives harder to interpret. This is true in the application of TMI decision support. Most TMIs have restrictions on the values that their parameters may take. The k-means clustering algorithm will not respect these restrictions, and may generate TMIs that could not be implemented. A decision maker viewing these representatives would likely find them unusual and may doubt their relevance. Another related downside is that the k-means method requires averaging members of a dataset. This is not always possible, as some features of the data set may be categorical or may have restricted values as previously discussed. Many methods mentioned in Section 7.2.1, including the KC-UPR method, do not have this downside. The author is aware of one work that does not use k-means centroids, as [134] proposed a method for choosing days for detailed case studies using an integer program. This method is heavily tailored to a specific setting, and would require substantial modification to be used in other applications. The KC-UPR method and the other existing UPR methods are more generally applicable. From the existing body of work, it is difficult to say which method is the most appropriate. We provide computational results that provide both quantitative and qualitative information concerning how these methods perform in this setting.

7.2.3 Methods Resembling Prototype Selection in Energy

The need to identify a representative set from a data set has arisen in domains other than air transportation. In particular, there has been some research concerning the selection of a set of sample time periods for simulations of energy systems.

Approaches proposed in [135] and [136] attempt to find sample days such that duration curves of certain features across the sample days are similar to those of the entire data set. These approaches would generally be useful in other domains where all relevant features can be expressed as duration curves. This may not be the case in all settings. Indeed, it is not the case for the air traffic management application that we consider in this chapter. For example, there is a single scope associated with each ground delay program, and it does not make sense to aggregate this feature into a time series. Other authors, for example [137, 138, 139], have proposed clustering-based approaches. These approaches can only be reasonably applied if the data exhibit cluster structure. The UPR methods discussed in this chapter are applicable to a broader set of problems and do not rely on cluster structure.

7.2.4 Related Methods for Decision Support

There is a well-established body of research suggesting that experts exhibit systematic biases when producing predictions or making decisions, and these biases may be curbed by asking the experts to consider analogous situations from the past; see for example [140, 141, 142]). Our proposed decision support tool would generate analogous situations by means of unsupervised prototype reduction. The author is aware of only two existing methods for producing sets of analogous situations from data sets. In [143] the three most similar analogues are selected, while in [144] the data set is clustered and the most similar cluster of analogues is selected. The former method may neglect informative observations, as there may be more than

three historical situations similar to the current situation. The latter method may be reasonably applied only if the data exhibit cluster structure. Even in this case, the method may select a set of analogues that is too large to be examined by a human decision maker in a reasonable amount of time. Our approach avoids these downsides as it allows the user to specify the number of cases produced, does not rely on cluster structure, and attempts to choose the analogous situations so as to capture as much useful information as possible.

7.2.5 Other Methods for Data Exploration

There are many existing methods for data exploration, and UPR methods can be used alongside existing techniques. Mode hunting methods, e.g. [94, 95], address a similar problem, but attempt only to find combinations of features that are especially likely to occur. UPR methods may produce some representatives that represent areas where observations are likely to occur, but these methods can also describe regions where observations are less likely to occur. In this way, UPR methods may describe the data more completely than mode hunting methods. Clustering methods also have a somewhat similar goal, attempting to divide the data into subsets such that elements within a subset are similar and elements in separate subsets are dissimilar. See [145] or [72] for an overview of this subject. The goal of a clustering method is to find structure within the data, while the goal of UPR methods is to describe the distribution of the data. If the data under consideration do not have cluster structure, then clustering methods cannot be reasonably applied, but UPR

methods can be used. Many clustering methods can produce clusters of irregular shape, e.g. [69, 70]. Such methods can identify structure, but it is a non-trivial task to present this structure in a way that a practitioner could understand. If a data set has strong cluster structure, it may be appropriate to divide the data into clusters and then apply a UPR method to each cluster to provide a description of that cluster.

Summary statistics are useful for giving a high-level description of the data, but more detailed understanding is often desired. UPR methods provide details that would not be captured by summary statistics. Existing density estimation techniques, for example kernel density estimation [146, 147], could be used to produce an estimated density of the data. See [89] for an overview of this topic. However, the resulting densities tend to be difficult to present to a practitioner, so these methods are not immediately useful in helping a practitioner to understand data. Marginal density estimates can be plotted for each feature separately. This provides values of each single feature that are more or less likely to occur, but does not provide any information about combinations of multiple features. Parallel coordinates plots, see for example [148], or plots showing pairs of features help to show relationships between pairs of variables, but similarly do not provide information concerning combinations of three or more variables. UPR methods would provide information on observed combinations of all variables. There are various existing dimension-reduction techniques that present high-dimensional data in a lower-dimensional space while attempting to preserve relationships between the observations as well as possible. For a survey of the topic, see for example [91] or [92]. However, there is no guarantee

that a data set can be presented accurately in a dimension that is low enough that it may easily be presented visually [91]. UPR methods do not require the dimension of the data to be small enough to be represented visually. Instead, these methods require only that the data set has low enough dimension that a single observation or a small set of observations can be presented.

7.2.6 Layout and Contributions

In Section 7.3, we provide an informal definition of a new problem in unsupervised learning, which we call the UPR problem. This problem involves producing a small set that approximates a large data set, but that is also a subset of the large set. We discuss many characteristics of the problem, including properties that are desirable in UPR methods, and how to measure the quality of results for the UPR problem. Section 7.4 contains a description of how a facility location problem, known as the k-center problem, can be used to solve the UPR problem. We use the term *KC-UPR method* to refer to the specialized use of the k-center method to solve the UPR problem. Some properties of this method are discussed in Section 7.4. In Section 7.5, several other algorithms that can be used to produce solutions to the UPR problem are provided. These methods are qualitatively compared with the KC-UPR method. We propose in Section 7.6 a new type of distance measure that is defined so that results from the KC-UPR method are more interpretable. In Section 7.7, we discuss how UPR methods may be used to provide decision support in the context of air traffic management. Section 7.8 contains computational results

comparing the effectiveness of several UPR methods in this context. An example using the KC-UPR method to explore traffic management data from Newark Liberty International Airport is given in Section 7.9. Conclusions and suggestions for further work are given in Section 7.10.

The main contributions of this chapter are as follows:

- A discussion of a new type of unsupervised learning problem, called the UPR problem. While a theoretical treatment of the problem was given in Chapter 6, we focus on aspects arising in the implementation of these methods to real datasets.
- The definition of a new type of distance measure that allows for easier interpretation of results from the KC-UPR method.
- Computational results comparing the effectiveness of several UPR methods in the context of air traffic decision support.
- A demonstration of how results from the KC-UPR method may be used to explore a data set, using air traffic management data from Newark Liberty International Airport.

7.3 Unsupervised Prototype Reduction

In this section, we define the problem of unsupervised prototype reduction (UPR) and discuss some characteristics. A formal definition of a closely-related problem was given in Chapter 6. Here, we take an informal approach that treats

the problem more generally.

7.3.1 Definition of Unsupervised Prototype Reduction

UPR methods are relevant when a practitioner wishes to interact with data in some way, but is hindered by the size of the data set. In our intended application, TMI decision-makers would like to examine the TMIs that have been used in similar situations, but this may be a large set. It is easy to imagine many other settings where data may be informative for some decision but the quantity of data is overwhelming. More generally, a practitioner may want to gain intuition for the distribution of observations in a data set. UPR methods can be used to help provide this intuition.

The input of a UPR problem is a set of data. We assume that the data are unlabeled, so there is no class associated with each observation. We also assume that a distance measure is available. That is, for any two data points, we assume there is a non-negative real number that describes the distance between those points. The output from a UPR method is a new data set, which we refer to as the *representative set* or *set of representatives*. Elements of the representative set are called *representatives*. The representative set should be much smaller than the original data set, and should approximate the data as well as possible. Discussion of how to measure how well the representative set approximates the data set is given in Section 7.3.3.

7.3.2 Desirable Properties for UPR Methods

While we define a UPR method simply as any method that takes a data set and returns a smaller representative set, there are many properties that are generally useful.

Representative Set is Subset of Data Set. It is useful for the representative set to be a subset of the original data set. The k-means centroids provide an example where this condition may not be met, since these are constructed by averaging multiple observations. This condition is especially important if some features have a restricted set of values that they can take, as these restrictions may be violated by a method that produces representatives that are not members of the original data set. This is also important if observations have associated secondary data. For example, in the application to TMI planning, each TMI in the original data set has associated performance metrics from when it was implemented. If the representative set is a subset of the original data set, these performance metrics can be provided to aid decision-making.

Quantity of Representatives Can be Controlled. There are often constraints on the number of representatives that can be selected. For the purposes of data exploration, there is a limit on the quantity of data that a human can easily examine, and the screen space of an interface may limit the number of representatives that can be displayed simultaneously. If the representatives are to be used as scenarios in human-in-the-loop trials or other expensive simulations, then the quantity

is constrained by a budget. It is desirable to be able to control the number of representatives in order to guarantee that these constraints will not be violated.

Representatives Are Diverse. If a representative set contains two elements that are very similar to each other, then a human will likely consider one of these to be redundant. Similarly, if the representatives are to be used as scenarios for simulation or as case studies for detailed analysis, then much of the insight provided by analyzing one of the representatives is likely to be captured by analyzing the similar representative. For these reasons, it is beneficial for the representatives to be diverse.

7.3.3 Quality Measures.

When prototype reduction is used in supervised learning, quality is often measured by the accuracy of a classifier that uses the representatives as a training set. In an unsupervised setting, the notion of how well a set of points represents the entire dataset is both subjective and dependent on how the representatives will be used. There are many reasonable quality measures and a practitioner should choose the measure most appropriate for their application. In Chapter 6, a list of desirable properties for quality measures was given and a quality measure was defined that satisfies all of the properties. We provide a much wider variety of possible quality measures, but do not attempt to satisfy those properties. Indeed, the properties are not applicable to some of the discussed measures.

Distance-based Measures. Quality can be determined by the distance from observations to the representatives. One possible measure is the maximum distance from any point to the closest representative. There are many variations on this objective. For example, the average distance or average squared distance from each point to its closest representative may be used instead of the maximum distance.

Statistical Measures. It can be important to approximately preserve statistical properties of the entire dataset. For example, in [134] a set of sample days is selected such that the mean of some features across the sample is approximately equal to the mean across the entire set of observations. Similarly, [135] attempt to preserve correlation of certain features. The median or any other measure of central tendency could be used instead of the mean. Other types of statistical measures, such as measures of dispersion or skewness, may also be included.

Problem-based Measures. If the representatives are used as an intermediate step of some process, then the resulting outcomes can be used as a measure of quality. For example, suppose the representatives will be used in a decision-support tool, such as the proposed tool mentioned in Section 7.2. If the results of a selected decision can be simulated accurately, then human-in-the-loop trials could be run. Experts would be asked to make decisions with and without the aid of the support tool. The decisions would be simulated, and the UPR method would be judged by the resulting outcomes. For example, if the UPR method would be used to support air traffic management decisions, then the results may be judged based on

performance indicators such as the quantities of air delays and ground delays issued to flights.

Multi-objective Measures. The aforementioned quality measures may be combined with each other. This can be accomplished by taking a weighted combination of multiple measures. Alternatively, a vector of several quality measures can be used. This could be applicable to a lexicographical approach to generating representatives. For instance, if there are multiple solutions that minimize some quality measure, a second quality measure can be used to discriminate between these solutions.

7.3.4 Differentiating Representatives

Some representatives may be similar to many observations in the original data set while other representatives may be similar to few of the original observations. The former can be interpreted as representing more typical regions of the data, while the latter represent unusual regions. In some applications, the more typical representatives may be more useful, while in other applications, the more unusual representatives may be more useful. In the case of TMI planning, both the typical and atypical will be useful. The typical representatives give an idea of the most common types of TMIs that have been run in the past. However, it is possible that a seldom-run TMI would be the best TMI for the given situation. Indeed, the proposed decision-support tool would be especially useful in this situation because decision makers are less likely to be aware of a strategy if it has not been used frequently. In any case, information concerning how typical or atypical a representative is is

important in order to help a practitioner to treat the representative appropriately.

A measure called *prevalence* can help to provide this information. Let D^* be the maximum distance from any point in the data to its closest representative, that is,

$$D^* = \max_{x \in X} \left(\min_{r \in R} d(x, r) \right)$$

where R is the set of representatives and X is the set of data. The prevalence of a representative r is defined to be the proportion of data points that are within D^* of r,

$$\rho(r) := \frac{|\{x : d(x, r) \leq D^*\}|}{|X|}.$$

Intuitively, the prevalence of a representative is an estimate of the probability that observations similar to the representative will occur. Results from Section 6.6.2 show that under general conditions this estimate will converge to the true probability. A higher prevalence indicates that observations similar to a representative are likely to occur, while a lower prevalence indicates that such observations are rare.

7.4 The KC-UPR Method

In this section, we describe the KC-UPR method for the UPR problem. This method is a specialized application of the k-center method presented in Section 6.5.2. Here, we provide an alternate derivation for the method in the context of unsupervised prototype reduction.

7.4.1 Data Coverage and Minimum Dominating Sets

In order to guarantee that the representatives represent the whole set of data well, a coverage requirement is enforced. Specifically, for any point of our data some representative must be within a distance threshold D of that data point. For the purposes of the present discussion, assume that D is given. In Section 7.4.2, we discuss how to produce this threshold. There may be many choices of sets of representatives that satisfy the coverage requirement; as an extreme example, the entire data set would work. In a trivial sense, this is the choice that best represents the entire data set, but this choice does nothing to reduce the amount of effort required to make use of the data. If the intent is to reduce this burden as much as possible, then it would make sense to choose the smallest possible set that still meets the coverage requirement.

This problem can be expressed as a problem on a graph. Define the D-neighborhood graph of the data set to be the graph where each node corresponds to a data point and where there is an edge between two nodes if and only if the distance between the data points is at most D. Let G be the D-similarity graph, and let V and E be the nodes and edges, respectively. Then, the problem of finding the smallest set of representatives that meets the coverage requirements is exactly the problem of finding a subset R of V such that:

1. For any v in V, either v is an element of R or v is adjacent to an element of R.

2. For any subset R' that satisfies condition 1, $|R'| \geq |R|$.

This is exactly the minimum dominating set problem, discussed in Section 6.5.1. As mentioned in Section 6.5.1, this problem can be formulated and solved as an integer program, and there are several existing heuristics for the problem [100, 101, 102].

7.4.2 The KC-UPR Method

As discussed in Section 7.3.2, it is a useful property to be able to determine how many representatives will be produced. The KC-UPR method achieves this in the following manner. Let k be the number of desired representatives. The distance threshold D is chosen so that the MDS of the D-neighborhood graph will be of size k. The resulting set is used as the representative set. Let $d_1, ..., d_m$ be the set of observed distance between pairs of points, ordered from smallest to largest. Note that for any distance thresholds D and D' such that

$$d_i \leq D, D' < d_{i+1}$$

the similarity graph produced by D is the same as that produced by D'. Thus, the threshold can be restricted to values in the finite set of observed distance values $d_1, ..., d_m$. The choice of the threshold determines how strict the coverage requirement is, where a lower value requires more coverage than a higher value does. This intuition suggests that a lower threshold is preferable to a higher threshold. If there are multiple values for the threshold that will produce k representatives, then the threshold is chosen to be the minimum of these possible values. There may also be instances such that there is no distance threshold for which the MDS has exactly

k representatives. Then, k is treated as an upper bound and D is chosen to be the minimum distance threshold such that the MDS has at most k representatives. Note that if the distance threshold is chosen to be sufficiently large, then the MDS will have size 1. Thus, the defined selection process is valid for any non-zero natural number k.

The problem of finding the smallest distance threshold such that the MDS method will produce at most k representatives is equivalent to the k-center facility location problem, which was discussed in Section 6.5.2. This relationship was first shown by [108]. In the k-center problem, k locations are chosen in order to minimize the maximum distance that any customer must travel to reach the closest facility. Let X be the set of data points, and let $d(x, y)$ be the distance between x and y. Then, the KC-UPR method chooses a set of k representatives R that minimize the objective

$$\max_{x \in X} \left(\min_{r \in R} d(x, r) \right),$$

which was discussed in Section 7.3.3. As mentioned in Section 6.5.2, this problem is NP-Hard, but there exist many exact methods and heuristics that can be used to solve the problem [103, 104, 105, 106, 106, 107, 108, 109, 110, 111, 112, 113, 114].

7.4.3 Properties of the KC-UPR Method

The KC-UPR method has many favorable properties. Statistical consistency properties of this method in a more general context are discussed in 6.7. The KC-UPR satisfies many of the desirable properties for UPR methods that were

discussed in Section 7.3.2. Specifically, it is desirable for the method to produce a representative set that is a subset of the original data set, to allow control of the quantity of representatives, and to produce a diverse set of representatives. By definition, the KC-UPR method produces representatives that are members of the original data set and allows the user to specify the number of representatives. While the KC-UPR does not explicitly attempt to produce a diverse set of representatives, this goal is complementary to the objective of minimizing the maximum distance from any point to its closest representative. This objective encourages uniform coverage of the data, making it unlikely that the KC-UPR method will place two representatives close together. There are some additional beneficial properties of the KC-UPR method.

7.4.3.1 Choice of Distance.

The KC-UPR method can be applied with any distance measure. This property is critical for our application because we propose a distance measure between TMIs that is not the typical Euclidean distance. In general, this property is important for applications where distances other than Euclidean distance are often used, such as time-series data, genomic data, text data, categorical data or any multivariate data with mixed units.

7.4.3.2 Outlier Resistance.

The KC-UPR method offers some resistance to outliers. Let us consider what happens if an outlying point is added to a data set. Since the number of representatives is fixed, then one of the available representatives must be used to cover that outlier. This leaves only $k-1$ representatives to cover the remaining data. However, once the outlier is covered it no longer has any impact on the objective, and the placement of the remaining representatives will not be affected by the location of the outlier. This property is expressed formally in Remark 2.

Remark 2. *Let there be a data set X, and let z be a point of X such that the distance between z and each point in $X \setminus \{z\}$ is greater than the optimal distance in the solution to a $(k-1)$-center problem on $X \setminus z$. Then a subset R is an optimal solution to the k-center problem on X if and only if $R = R' + \{z\}$ where R' is an optimal solution to the $(k-1)$-center problem on $X \setminus \{z\}$.*

Remark 2. Let X be a data set with a point z such that the distance between z and each point of $X \setminus \{z\}$ is greater than the optimal distance in the solution of the $(k-1)$-center problem on $X \setminus \{z\}$. First, we will prove the forward direction. Let R be an optimal solution to the k-center problem on X. Let R^* be an optimal solution to the k-center problem on $X \setminus \{z\}$ with corresponding optimal distance D. Note that $R^* + \{z\}$ is a feasible solution to the k-center problem on X with corresponding distance D. Since R is optimal, the distance threshold corresponding to R must also be no more than D. However, the distance from z to any other point is more than D, so R must include z. Then note that $R \setminus \{z\}$ provides a solution

to the $(k-1)$-center problem on $X \setminus \{z\}$ with distance D.

Conversely, let $R = R' + \{z\}$ where R' is an optimal solution to the $(k-1)$-center problem on $X \setminus \{z\}$. Let D be the distance corresponding to R'. Then, note that R provides a feasible solution to the k-center problem with corresponding distance D. Let R^* be an optimal solution to the k-center problem on X with corresponding distance D^*. Since R^* is optimal, we know that $D^* \leq D$. The distance between z and any other point is more than D, so R^* must include z. Thus, $R^* \setminus \{z\}$ is a feasible solution to the $(k-1)$-center problem on $X \setminus \{z\}$. Note that the corresponding distance is D^*. Since D is the optimal distance threshold of this problem, we have $D \leq D^*$. Thus, $D = D^*$ and R is an optimal solution to the k-center problem on X. □

By definition, outliers must be far away from almost all other points, although there may be a few outlying points that are close together. This suggests that outliers may be identified by running the KC-UPR method and finding small connected components of the D-neighborhood graph, where D is the resulting distance threshold. If it would be undesirable for some representatives to be outliers in a particular application, then these outliers could be removed and the KC-UPR method could be rerun. The prevalence (as defined in Section 7.3.4) of representatives that are outliers will also be extremely low, so the representatives may be presented with their prevalence and the user will be able to tell which representatives represent outliers.

7.5 Other Methods For Prototype Selection

In this section we discuss other UPR methods. Each method is presented briefly. Some algorithmic details have been omitted for the sake of brevity and clarity. The outputs from these methods and the KC-UPR method applied to an example data set are presented to illustrate the differences between these methods.

7.5.1 Kennard-Stone Algorithm

The Kennard-Stone algorithm [81] chooses elements one-by-one. This is carried out by a greedy selection process that aims to maximize the diversity of the selected elements. The algorithm may be described as follows:

1. Choose the first two representatives to be the data points that are farthest apart.

2. Choose the next representative to be the data point with maximum distance to the nearest existing representative.

3. Repeat step 2. until the desired number of representatives have been chosen.

This can be thought of as a heuristic that attempts to maximize the minimum of the distances between representatives. If few representatives have been chosen, then the point that maximizes the distance to the closest representative likely falls on the boundary of the data. After there are a sufficient number of points on the boundary, then this point of maximum distance is likely to fall on the interior of the data. So,

the Kennard-Stone algorithm will first choose representatives along the boundary of the data and will subsequently cover the interior.

7.5.2 OptiSim

The OptiSim algorithm [80] generalizes the Kennard-Stone algorithm. OptiSim also chooses elements using a greedy selection process that values diversity. However, OptiSim attempts to balance the diversity of the selected elements with how well these elements represent the data set. This algorithm has two parameters, M and D, and follows the steps:

1. Choose the first representative at random.
2. Let A be the set of remaining points x such that the distance from x to the closest existing representative is at least D.
3. Let B be a set of M points randomly sampled from A.
4. Choose the next representative to be the data point in B with maximum distance to the closest existing representative.
5. Repeat steps 2-4 until the desired number of representatives have been chosen.

The parameter D is a distance threshold, and the algorithm guarantees that the distance between any two representatives is at least D. This can be used if there is a preexisting, known requirement on the diversity of the representatives. If such a requirement is not known, then this parameter may be set to zero. When the distance threshold D is set to zero, then OptiSim may be thought of as a variant of

Kennard-Stone where the next representative is chosen from a sample of the data points. Choosing from a sample instead of the whole data set increases the likelihood of choosing a representative from a region of the data where many observations fall. The parameter M controls the size of this sample. If M is set to one, then the OptiSim algorithm selects the representatives almost completely at random. As M increases, the variance of the algorithm decreases and the algorithm behaves more like the Kennard-Stone algorithm. Indeed, the Kennard-Stone algorithm is a special case where D is set to zero and M is set to the size of the data set.

7.5.3 Isolation Forest for Representative Subset Selection

The Isolation Forest method was originally designed for outlier detection [149] but has been adapted for use as a UPR method [129]. The Isolation Forest method assigns a score between 0 and 1 to each point in the dataset. A higher score indicates that a point is more likely to be an outlier. This was adapted to the UPR problem in the following manner:

1. Apply the Isolation Forest method to determine an outlier score for each point.

2. If there are some points with scores significantly larger than all other points, then declare these points to be anomalous and remove them from consideration.

3. For each non-anomalous point with a score greater than 0.5, declare these points to be uncommon and select them as representatives.

4. Select the remaining representatives by taking a uniform sample from the points with score no larger than 0.5. Declare these representatives to be common representatives.

The authors of the method do not provide an objective criteria to determine whether a point has an outlier score significantly larger than all other points, as required in step 2. For the purposes of this chapter, we will declare a point to be an anomaly if its outlier score is 0.8 or higher. The authors also do not specify what to do if the number of non-anomalous points with outlier score greater than 0.5 exceeds the desired number of representatives. In this case, we sort these points by outlier score and select the points with the k highest scores, where k is the number of desired representatives.

7.5.4 K-Medoids

As discussed in Section 7.2.2, the k-means method has been proposed for several problems in air traffic management that resemble UPR problems. This requires averaging several members of the dataset, which has aforementioned downsides. The k-medoids algorithm [150] is a similar algorithm, but does not require such averaging. Similarly to k-means, k-medoids also forms clusters of points, each of which has an associated centroid element. For the purposes of UPR, the clusters are calculated as a step of the algorithm, but only the centroids are used in the end. The algorithm has the following steps:

1. Choose an initial set of k points. Form k clusters by assigning each point to

its nearest representative. Calculate the sum of distances from each point to its nearest representative.

2. For each cluster, find the point x in the cluster that minimizes the sum of distances from x to the other points of the cluster. Select these k cluster centers to be the new set of representatives.

3. Form k new clusters by assigning each point its nearest representative.

4. Recalculate the sum of distances from each point to its nearest representative. If this sum has changed since the last iteration, return to step 2. Otherwise, stop.

This algorithm can be interpreted as a heuristic that attempts to minimize the sum of distances objective,

$$\sum_{x \in X} \min_{r \in R} d(x, r),$$

which was discussed in Section 7.3.3.

7.5.5 Computational Costs

The Kennard-Stone and the OptiSim method both have computational costs that are at worst $O(n^2) + O(nk^2)$, where n is the number of data points and k is the number of centers. The cost of the Isolation Forest method is $O(tn \log(n))$, where t is the number of trees in the forest and n is the number of data points. If t is much smaller than n, then this method has the lowest computational complexity. The k-medoids method is more computationally expensive than the preceding

methods, as each iteration can require at worst $O(n^2)$ operations, and it may take many iterations to reach the halting criteria. The KC-UPR method is more computationally expensive than the other methods, and requires exponential time unless NP = P. We use the exact solution method proposed by [105]. Experiments conducted in the existing literature suggest that the required computational time can vary depending on the instance, but instances with up to 1000 nodes were solved in seconds, while an instance with 1817 nodes required more than 10 minutes to solve exactly. These instances could likely be solved much more quickly with modern computing equipment, but this remains a relatively computationally expensive method. Heuristic approaches can be much cheaper. For example, [109] propose an iterative improvement procedure whose cost is $O(nk^2)$ per iteration.

7.5.6 Qualitative Comparison

Here, we demonstrate the differences between these methods with the use of an example dataset. We use a two-dimensional dataset so that it is easy to present the results visually. This example focuses on qualitative properties of the methods. In Section 7.8, we provide details of a more realistic computational experiment focusing on quantitative results. The dataset consists of two features taken from 581 traffic management initiatives implemented at Newark Liberty International Airport (EWR) between the dates of July 21st, 2010 and December 31st, 2014. These features were normalized so that the mean of each feature is zero and the standard deviation is one. A scatter plot of the data is shown in Fig. 7.1. The distance

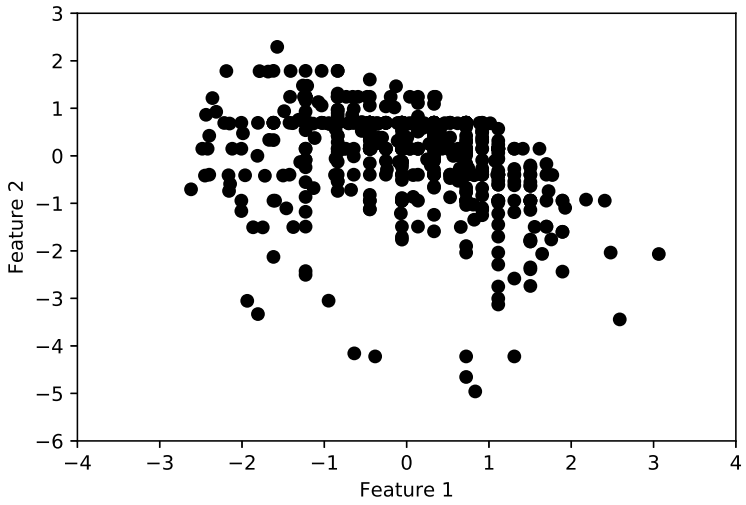


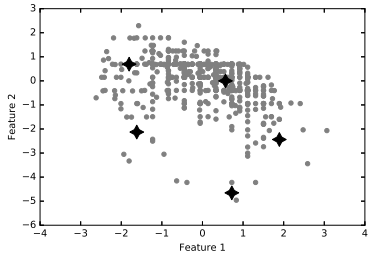
Figure 7.1: Scatter Plot of Example Data Set.

measure used throughout this example is standard Euclidean distance on the normalized data. The KC-UPR, k-medoids, Kennard-Stone, OptiSim, and Isolation Forest methods were each applied to this data set. Each method was applied once with the number of representatives chosen to be 5. For the OptiSim method, the radius parameter D was set to 0 and results were produced with the sample size parameter set to 10, 50 and 100. In this section, the OptiSim method with sample size of 10, 50 or 100 is referred to as OptiSim 10, OptiSim 50 or OptiSim 100 respectively.

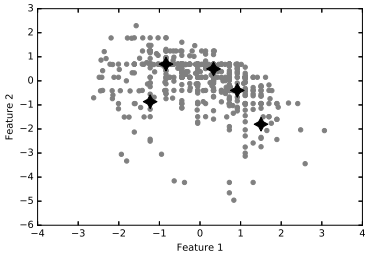
The representative sets of size 5 produced by all methods are shown in Fig. 7.2. The scatter plot of the data is shown in the background as gray points, and the representatives are shown in the foreground as larger black points. For this dataset, the Isolation Forest method identified 92 observations as uncommon. Thus, the rep-

resentatives selected by this method are all labeled as uncommon, and are among the most extreme points of the data set. This highlights a downside of the method: if the number of uncommon points is large relative to the requested quantity of representatives, then the selected representatives will tend to be extreme or unusual points. The Isolation Forest incorporates random elements when computing the outlier scores. In this example, the final set of representatives is entirely determined by the outlier scores because the number of uncommon points was greater than the requested number of representatives. In other cases, it is also possible for the selection of the final set of points to involve some randomness. These two sources of randomness can cause the results from this method to exhibit some variability, although the existing literature on the method indicates that its results are quite stable [129]. While we did not conduct rigorous experiments confirming this property, our experience with the method provides some anecdotal support, as the outputs that we observed from the method did not seem to change much in repeated applications of the method. An alternative set of results from the Isolation Forest method is shown in Fig. 7.3(a). The two resulting sets of representatives differ only in a single representative.

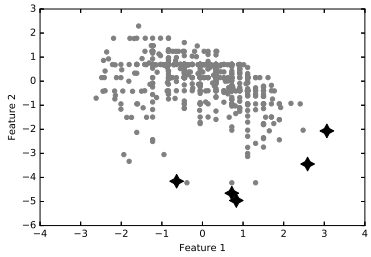
The k-medoids method differs from KC-UPR and Kennard-Stone methods in that the objective of the former method is affected by the density of points in a region. Since the k-medoids method attempts to minimize the sum of distances, this method prioritizes coverage of denser regions rather than less dense regions. In contrast, the max-min objective forces the KC-UPR method to attempt to cover all the regions where observations occur as well as possible. This leads the KC-



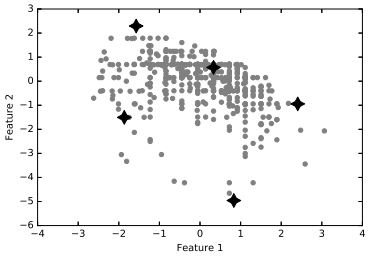
(a) K-center



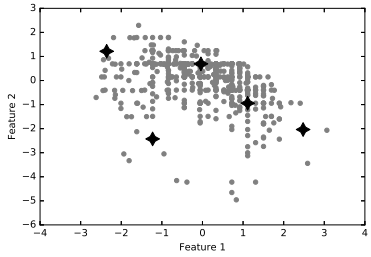
(b) K-medoids



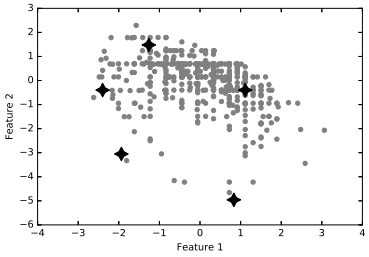
(c) Isolation Forest



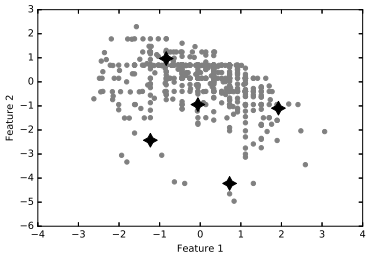
(d) Kennard-Stone



(e) OptiSim 10



(f) OptiSim 50



(g) OptiSim 100

Figure 7.2: UPR methods applied to example data set, 5 representatives

UPR method to cover more of the regions of the dataset with moderate numbers of representatives, while the k-medoids method places more observations in the most central regions of the dataset. The Kennard-Stone method places its representatives in even less central locations than the KC-UPR method, as its goal is to place as much distance between representatives as possible.

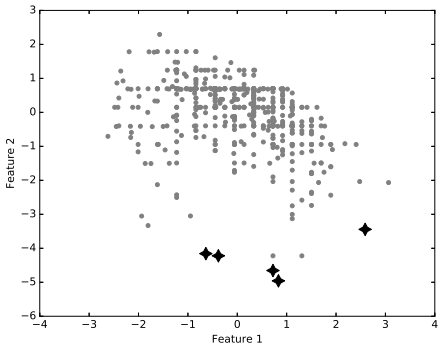
If the sample size is low, then OptiSim is more likely to select representatives from higher density regions. As the sample size increases, the behavior becomes more like the Kennard-Stone method and representatives are more likely to be spaced further apart. However, this trend is obscured by a high degree of variability in the results, due to the random sampling that occurs in the method. An alternative set of results for the OptiSim methods is shown in Figs. 7.3(b)-7.3(d). We observe that for all three choices of sample size, the sets of representatives shown in Figs. 7.2(e)-7.2(g) differ significantly from those shown in Figs. 7.3(b)-7.3(d).

7.6 An Interpretable Distance Measure for Data Exploration

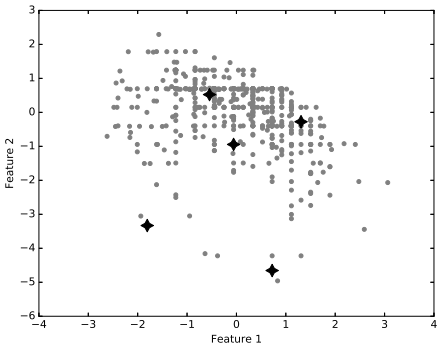
If the KC-UPR method is to be used for data exploration, it is important to choose a distance measure that allows for the representatives to be easily interpreted. We propose one such distance measure.

7.6.1 Maximum Pairwise-Quantile Distances

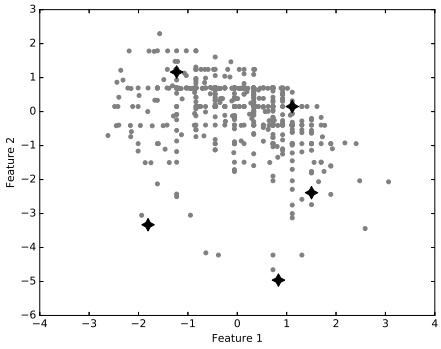
Let X be the set of observed data points. We assume that for each feature f and any pair of observations x and y there is a known distance measure $d_f(x, y)$ with



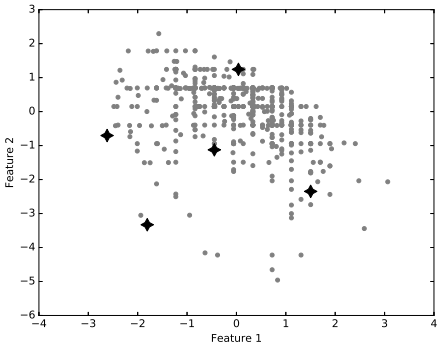
(a) Isolation Forest



(b) OptiSim 10



(c) OptiSim 50



(d) OptiSim 100

Figure 7.3: Alternative Outputs of UPR methods, 5 representatives

respect to the feature f. Generally, it is much easier and more intuitive to define a distance measure for a single feature than it is to define distance for all features simultaneously. If all features had the same units and took a similar range of values, then it would be reasonable to compare the feature-wise distances directly. This is often not the case. For example, consider an application involving TMI data. A possible feature of a TMI is the number of flights allowed to access a resource in some hour. The unit would be flights per hour, and this would typically take a value between zero and 120. Another possible feature is the duration that the TMI is in effect. The duration of a typical GDP would usually be between 240 and 960 minutes. Thus, it is prudent to normalize the feature-wise distances. We suggest the following normalized distance measure for a feature f:

$$\tilde{d}_f(x,y) = \frac{|\{u,v\} : u,v \in X, d_f(x,y) \geq d_f(u,v)|}{|\{u,v\} : u,v \in X|}.$$

The normalized distance in the feature f for a pair of points (x,y) is the proportion of pairs of points whose distance is at most as large as the distance between x and y in the feature f. We will refer to this as the pairwise-quantile normalization procedure. This is similar to a standard normalization procedure where the data are transformed by subtracting the mean of each feature and dividing by the standard deviation. However, our proposed method performs a normalization on the distances instead of on the data itself.

Once the normalized distances have been computed for each feature, these distances must be combined to get an overall distance. We suggest taking the

maximum of the pairwise-quantile distances,

$$d(x, y) = \max_{f \in F} \tilde{d}_f(x, y),$$

where F is the set of features. We refer to the distance measure produced in this manner as *maximum pairwise-quantile distance*.

7.6.2 Advantages and Disadvantages of Maximum Pairwise-Quantile Distances

The maximum pairwise-quantile distance measure can be described in two steps: first, a normalization procedure is applied to the pairwise distances in each feature; second, the pairwise distances in each feature are combined by taking the maximum. The advantages and disadvantages of this can be attributed to either the first or the second step of the process.

7.6.2.1 Advantages and Disadvantages of the Normalization Procedure.

A benefit of the normalization procedure is that it is invariant to any transformation that preserves the ranking of distances between pairs of points.

Remark 3. *Let there be two data sets X_1 and X_2. Let d_1 and d_2 be distance measures on X_1 and X_2 respectively. Let ϕ be a bijection between $X_1 :\to X_2$ such that*

$$d_1(x, y) \leq d_1(u, v)$$

if and only if

$$d_2(\phi(x), \phi(y)) \leq d_2(\phi(u), \phi(v))$$

for any (not necessarily distinct) points $x, y, u, v \in X_1$. *Let* $\tilde{d}_1$ *and* $\tilde{d}_2$ *be the normalized distance measures of* d_1 *and* d_2. *Then it will be true that*

$$\tilde{d}_1(x, y) = \tilde{d}_2(\phi(x), \phi(y))$$

for any pair of points (x, y) *in* X.

This is easily observed by noting that since the ranking of pairwise distances is unaffected by the transformation, then the proportion of pairs of points whose distance is at most the distance between a given pair of points will remain the same. As an example, consider a numerical feature where the distance $d_f(x, y)$ is defined to be the absolute value of the difference of x and y in feature f. Then, any linear or affine transformation of the data will not affect the normalized distance in that feature.

This normalization approach is also resistant to outliers because it is based on the rankings rather than the magnitudes of pairwise distances, and inclusion of outliers will not greatly affect these rankings. In contrast, consider the commonly-used "z-score" method of normalization where the sample mean is subtracted from each observation and the result is divided by the sample standard deviation. Using the z-score normalization, features with outliers will tend to have smaller distances because outliers will inflate the sample standard deviation.

A disadvantage of the normalization procedure is that if the distance between some pair of points is desired, then it is necessary to calculate the feature-wise

distance between every pair of points in all features. The most straightforward approach to finding the pairwise-quantile distance for a single pair of points (x, y) is by examining all of the pairwise distances in each feature and by counting how many of these distances are smaller than the distance between x and y in that feature. This approach requires $O(mn^2)$ observations to find the distance between x and y, where m is the number of features and n is the number of observations. A more efficient approach can be taken to compute the distances between all pairs of points. For each feature, it is possible to calculate all the pairwise distances and sort them in $O(n^2 \log(n))$ operations. Then, the computational complexity of performing this sorting for all features is $O(mn^2 \log(n))$. The position of a pair of points in the sorted list of distances for a given feature immediately provides the normalized distance for that pair in that feature. It is possible to mitigate this disadvantage by using an approximation to this distance measure, which is described in Section 7.6.3.

7.6.2.2 Advantages and Disadvantages of Taking Maximum of Feature-wise Distances.

An advantage of taking the maximum of the feature-wise distances that makes it especially well-suited for use with the KC-UPR method is that the results from the method are especially easy to interpret when using this distance measure. Along with the k representatives, the KC-UPR method also produces a distance threshold D where every observed data point is guaranteed to be within a distance of D from a representative. Since the distance is the maximum of the pairwise-quantile distances

in each feature, then two points have a distance of at most D if and only if their pairwise-quantile distance in each feature f is at most D. For each feature, there exists a number D_f such that the pairwise-quantile distance for a pair of points in that feature is at most D if and only if the distance between those points is at most D_f. To be more specific, this value D_f is given by:

$$D_f := \min \left\{ r : \frac{|\{x, y\} : x, y \in X, d_f(x, y) \leq r|}{|\{x, y\} : x, y \in X|} \leq D \right\}.$$

When the representatives are presented to the user, these feature-wise distance thresholds may also be presented. This would inform the user that for any point x in the data, there exists a representative y such that $d_f(x, y)$ is at most D_f in every feature f. For example, we might solve the KC-UPR method using the maximum pairwise-quantile distance and find that the optimal distance is 0.3. Suppose that we have two features f_1 and f_2. Further suppose that exactly 30% of all pairs of points have a distance in feature f_1 of at most 10 and 30% of all pairs have a distance in feature f_2 of at most 17. Then, we can say that for any point x there exists a representative r such that the distance between x and r in the feature f_1 is at most 10 and the distance between x and r in the feature f_2 is at most 17.

A disadvantage of taking the maximum is that the distances between pairs of points become more similar as the number of features increases. To illustrate, suppose that the pairwise distances for each feature are distributed uniformly between 0 and 1, and suppose that the distances between pairs in each feature are independent of the distances in all other features. Let m be the number of features. Then, the maximum of the pairwise distances across the features is the maximum of

m i.i.d. variables uniformly distributed on $[0, 1]$. As a result, while the interquantile range for distances in a single feature is $\frac{1}{2}$, the interquantile range of the distances resulting from taking the maximum across m features is

$$\left(\frac{3}{4}\right)^{1/m} - \left(\frac{1}{4}\right)^{1/m},$$

which approaches zero as m gets very large. This particular disadvantage, known as "concentration of distances", is present in many distance measures and is well-documented, e.g. [151].

7.6.3 Approximate Maximum Pairwise-Quantile Distances

As discussed in Section 7.6.2.1, the costs of computing the maximum pairwise-quantile distance for a single pair of data points and for all data points are $O(mn^2)$ operations and $O(mn^2 \log(n))$ operations respectively, where m is the number of features and n is the number of data points. For some large data sets, it may be computationally intractable to perform this many operations. In this case, we may perform an approximate version of the normalization process as follows. We create a sample Y of size l from X by any resampling method, such as bootstrapping. We then define the approximate pairwise-quantile distance to be

$$\hat{d}_f(x, y) = \frac{|\{u, v\} : u, v \in Y, d_f(x, y) \geq d_f(u, v)|}{|\{u, v\} : u, v \in Y|}.$$

In this case, to find the distance between any pair of points (x, y), we need only compare the distance between x and y in each feature to the pairwise distances from the sample Y in that feature. This will require $O(ml^2)$ steps. Similar to before, if

the distances between a large number of pairs are desired, it will be more efficient to first sort the distances between pairs of Y in each feature. This initial sorting requires $O(ml^2 \log(l))$ steps. Then the distance between two data points x and y is produced by finding the number of pairs in Y whose distance is at most $d_f(x, y)$ for each feature f. This can be accomplished by conducting a binary search for each feature on the sorted pairs of differences, which requires $O(m \log(l))$ operations.

7.7 Representative TMIs at an Airport

We describe how UPR methods can be applied to find representative TMIs for the purpose of aiding analysis or decision-making of TMIs. It is expected that this will be the second step of a multi-step process. The first step would be a filtering process that selects only days that are similar to a reference day in terms of weather and traffic conditions. This narrows the information under consideration to that which is most relevant to problem. For example, if the goal is to aid the planning of a TMI, then the current day would be selected as the reference day, while if the goal is to evaluate the performance of a TMI on a past day, then the reference day would be that past day. In the second step, the TMI actions taken on the similar days would be summarized with a UPR method. In the final step, this summarized set of TMIs would be presented to the decision-maker along with additional information, such as the performance that the TMI achieved when it was run. For the moment, we will focus on the TMIs issued by the FAA that affect the terminal airspace of a single given airport. These are ground stops (GSs) and ground delay programs

(GDPs).

7.7.1 Terminal Traffic Management Initiatives

In a GDP, the number of flights allowed to arrive at an airport is restricted. A decision maker in the FAA command center decides a time interval that the program will be in effect and decides the program rate, which is the number of flights that can arrive, in each hour. Flights that are still on the ground and are scheduled to arrive at the airport while the GDP is in effect will receive a controlled time of departure (CTD). These CTDs are assigned in such a way that if all flights left at their CTD and if their flight duration matched the scheduled flight duration, then the number of flights arriving at the airport in each time period would match the specified program rate. The FAA often exempts some longer-distance flights from the GDP. The geographic region of flights that are not exempt from the GDP is called the scope. This is sometimes specified as a radius, in which case a flight will be exempt if its flight distance is greater than the radius. Alternatively, this is sometimes specified as a set of predefined geographic regions corresponding to Air Route Traffic Control Centers (ARTCCs), and flights not departing from these geographic regions will be exempt. A GS has a similarly-defined time of effect and scope, but is a more severe action, as any flight that is controlled by a GS is not allowed to depart until the GS has been cancelled.

As time progresses, the FAA can issue revisions to a GDP that alter its rate, scope, or duration. The FAA can also extend or cancel a GS. Based on our con-

versations with FAA personnel, the initiation of a GDP is usually an anticipatory measure while GSs and revisions to GDPs are largely reactive measures. A decision maker would never issue a GDP with the intent of altering it later, and if the GDP is working as planned then the decision maker would not revise it. Similarly, a decision maker will only issue a GS to ease existing congestion at an airport, and would not plan on issuing it beforehand. There is perhaps one exception. Sometimes, a decision maker will decide to take a "wait-and-see" approach. In this strategy, no action is taken initially. If conditions become worse and congestion begins to form at the airport, then a GS will be issued to ease the congestion. Upon termination of the GS, a GDP will be issued in order to prevent congestion from forming again. In this case, we consider the GS to be part of the general strategy taken by the decision makers. Examination of the data indicates that this strategy occurs frequently. Out of 582 GDPs initiated between July 20th, 2010 and December 31st, 2014, 100 of these GDPs were immediately preceded by a GS.

Since our goal is to produce a tool to aid in the analysis of anticipatory TMI measures, we consider only three types of TMI strategies. The first strategy is to run a GDP with no GS preceding it. The second strategy is to run a GS immediately followed by a GDP. The third strategy is to issue no TMI action. It is important to include this last option, as there is often discussion about whether or not a GDP should be run at all. We consider only the first GDP initiated on each day. We define a day at an airport to begin at 4:00 a.m. local time and to continue for 24 hours. This better reflects the natural cycle of an airport than a definition where days begin at 12:00 a.m., as a larger volume of traffic arrives at an airport close to

midnight than arrives at a time close to 4:00 in the morning.

7.7.2 Features of TMIs

We would like our set of chosen features to reflect all aspects of TMI planning. [152] and [62] discuss the parameters relevant to the planning of a GDP. These authors were interested in producing methods that could choose these parameters optimally. The timing features mentioned in [152] are the *file time*, *start time* and the *end time*. The file time is the time at which the GDP is declared and put into effect. The start and end times are the earliest and latest times, respectively, in the interval of ETAs that are controlled by the GDP. The start and end times were also considered by Cook and Wood, although the file time was not. Both pairs of authors discussed the planned airport acceptance rate. Both made simplifying assumptions about the form of the acceptance rates declared in a GDP, although their assumptions differed. These authors also made simplifying assumptions about the manner in which the scope of the program was declared.

Our choice of features reflecting the timing of the TMI is similar to the parameters discussed by Ball and Lulli and by Cook and Wood. However, some of our GDPs have an associated GS, which neither pair of authors considered. We include the *GS duration* in order to describe the associated GS. This is defined to be the length of time between the time at which the GS is declared and the start time of the GDP. If the TMI is a GDP that is not preceded by a GS, then we define this duration to be zero. We include the file time and start time, as defined in Ball and

Lulli. However, in order to make our treatment of the GDP consistent with our treatment of the GS, we use the *GDP duration* (the difference between the start and end times) instead of the end time of the GDP. In order to represent the file time and start time as numerical features, we define the file time and start time by the number of minutes after 4:00 a.m. that the time occurs. We use minutes as the unit for the duration of the GDP and the GS. The choice of units is not particularly important since we use the maximum pairwise-quantile distance discussed in Section 7.6, and linear or affine rescaling of features does not affect the resulting distances under this measure.

Similar to Ball and Lulli or Cook and Wood, we include a feature that reflects the scope of the GDP. The scopes of some TMIs are specified as a radius while the scopes of others are specified as a set of ARTCCs. In order to render the scope as a purely numeric feature that is consistent across all TMIs, we code it as the number of CORE 30 airports (those 30 airports with the highest traffic volumes) included in the defined scope. If the scope of some TMI was specified as a radius, then this proxy describes the scope almost completely, as only a relatively small range of radii would result in the inclusion of a given number of CORE 30 airports. If the scope was instead given as a combination of ARTCCs, then any combination of these ARTCCs is possible in principle. However, only a small set of combinations tend to be used in practice. Furthermore, the combinations used tend to correspond to the severity of restriction, so that the more restrictive combination of ARTCCs contains all of the ARTCCs of the less restrictive combination. For example, if there is a GDP at a New York airport, a relatively unrestrictive scope might include only the

eastern half of the United States in the scope, while a more restrictive scope might include all of the United States except for the West coast. Thus, at a given airport it would be rare for there to be two declared scopes that include similar numbers of CORE 30 airports but that describe significantly different geographic regions. For these reasons, the number of CORE 30 airports generally provides a good idea of the scope that was selected.

We also consider features that reflect the rate of the GDP. Unlike Ball and Lulli or Cook and Wood, we do not make simplifying assumptions about the types of rates that may be declared in a GDP. The rate is a vector that specifies the number of flights that would be allowed to arrive in each hour of the program. Since the timing of the GDPs may be different, there may be an hour in which one GDP has a defined rate while another GDP does not. This makes direct comparison of rate vectors difficult. One way to solve this problem is to use the average of the rates declared in the rate vector. This does not contain all of the information about the rate, but does give a general idea of how restrictive the GDP is. Another approach is to use the proportion of the flights wishing to arrive at the airport that would be allowed to do so according to the declared rate of the GDP. This also captures how restrictive the program is, but measures the restrictions relative to the demand. Consider some observed GDP x. Let n_{xi} be the number of flights scheduled to arrive at the airport in the i^{th} hour of the day that the GDP x occurred on. Let r_{xi} be the rate of the GDP x in the i^{th} hour of this day if this hour is included in the GDP.

Then, we define the relative rate r'_{xi} to be

$$r'_{xi} := \begin{cases} 1 & \text{if } r_{xi} \text{ is not defined,} \\ \min\left\{1, \dfrac{r_{xi}}{n_{xi}}\right\} & \text{if } r_{xi} \text{ is defined.} \end{cases}$$

For each GDP, the relative rate is then a vector in $[0,1]^{24}$. This gives a total of seven features: file time, start time, GDP duration, GS duration, scope (number of CORE 30 airports), average rate, and relative rate vector.

7.7.3 Feature Combinations.

Some of the features mentioned above have some overlap in terms of the information that they provide. The relative rate vector includes information about the start time and duration of the GDP. If the number of scheduled arrivals is higher than the planned rate for the GDP throughout the time of effect, then the value of the relative rate vector will be less than one in exactly those time periods that the GDP is scheduled for. In this case, the start time and duration are described entirely by the relative rate vector. We generally expect the number of scheduled arrivals to be higher than the planned rate of the GDP, since the purpose is to restrict the number of flights. Thus, it may be unnecessary to include the duration or start time if the relative rate is included. The average rate and relative rate vector also provide similar information, so it might not be necessary to include both of these. The remaining GDP parameters, namely ground stop duration, file time, and scope serve more distinct purposes. We would like to include a set of parameters that is comprehensive. Based on the preceding discussion, we enforce the following

Table 7.1: Sets of Chosen Features.

Feature Set	Start Time	GDP Duration	Avg. Rate	Relative Rate
0	Yes	Yes	Yes	Yes
1	No	Yes	Yes	Yes
2	Yes	No	Yes	Yes
3	Yes	Yes	No	Yes
4	Yes	No	No	Yes
5	No	Yes	No	Yes
6	No	No	Yes	Yes
7	Yes	Yes	Yes	No
8	No	No	No	Yes

requirements in order to ensure that the parameters provide a detailed description of the GDP:

1. The ground stop duration, file time, and scope are always selected.

2. Either the relative rate is selected, or all of the start time, duration and average rate are selected.

There are nine combinations of parameters satisfying these requirements, summarized in Table 7.1. The ground stop duration, file time, and scope are omitted from Table 7.1 because these are always included in the feature set. There is a risk that including too many parameters may result in "concentration of distances" as discussed in Section 7.6.2.2, making it more difficult to differentiate between observations. On the other hand, a higher level of detail may allow for a higher degree of differentiation between observations. Ultimately, the best choice of features is that which provides the highest quality representatives. We make this determination through computational experiments, described in Section 7.8.

7.7.4 Distances Between TMIs.

We define distance between TMIs by the maximum pairwise-quantile distance measure, as discussed in Section 7.6. This requires defining a distance for each individual feature. The file time, start time, GDP duration, GS duration, scope (number of CORE 30 airports) and average rate are all numerical features. For these features, we use the absolute difference in the feature. The only remaining feature is the relative rate vector, which is a vector of length 24. For this feature, we define distance to be the sum of the absolute differences between the vectors. That is,

$$d_r(x, y) := \sum_{i=1}^{24} \left| r'_{xi} - r'_{yi} \right| .$$

7.7.5 Generation of Representative GDPs.

In the application of a decision-support tool, the maximum feature-wise quantile distance is calculated for the set of all GDPs that have occurred. This provides the distance measure that is used. When a decision maker is facing a situation, a set of GDPs that occurred in similar situations is identified. The decision maker specifies the number of representatives that he or she is willing to examine, and the KC-UPR method is used to produce the representative TMIs. The representatives can be presented to the user along with the corresponding prevalences. The optimal distance D from the solution to the KC-UPR is also presented, so that the user will know that every data point is within a distance of D from the representatives. As

mentioned in section 7.6.2.2, we can express this distance in terms of a distance in each feature to make this more easily interpreted.

7.7.6 Measuring Quality of Representative GDPs.

Many ways of measuring the quality of representative GDPs were discussed in Section 7.3.3. Since the proposed representatives are to be used in a decision-support tool, one way of measuring the quality of the representatives is to run human-in-the-loop trials in which decision makers are asked to use these representatives to support their decision, as mentioned in Section 7.3.3. We believe that this would be the best way to determine how useful these representatives would be in practice. However, due to the expense of running such trials, we leave the design and implementation of these trials as an area for future work. Instead, we use several quality measures that do not directly measure provided benefit, but instead measure to what extent the representatives exhibit desirable properties.

We use all three of the distance-based quality measures mentioned in 7.3.3. We also use another quality measure more specific to the problem of TMI planning. Recent research recognizes that a successful TMI must balance several aspects of performance. For example, one aspect of performance is throughput, which describes how well the restricted resource was utilized. Another aspect is predictability, which describes how well and how far in advance flight operators are able to know what actions the FAA will take. There are trade-offs between these aspects. For example, if the FAA alters their plans more frequently in response to changing conditions,

this may allow them to make more complete use of the restricted resource. This would increase throughput but would decrease predictability. Flight operators may have preferences as to which aspects of performance should be prioritized, and these preferences may change over time. Existing research in this direction includes the definition of performance metrics to capture different aspects of TMI performance [153] and a proposed mechanism that would solicit preferences from flight operators in order to suggest a balance of performance metrics [154, 155]. This suggests that if a set of representatives is of high quality, then it should vary in terms of the types of performance achieved by its representatives. We consider three performance objectives: predictability, efficiency, and throughput. As a metric of predictability, we use the number of reactive TMI actions that take place during the day; i.e., the number of ground stops or GDP revisions made. A higher number of these reactive actions indicates that the FAA behaved in a less predictable manner from the perspective of flight operators. For efficiency, we use the average number of minutes of airborne holding per flight for the day. For throughput we use the average arrival delay per flight for the day. We normalize each metric in the typical "z-score" fashion. That is, we calculate the mean μ and standard deviation σ of the metric, and then replace each observation x with the normalized observation

$$\hat{x} := \frac{x - \mu}{\sigma}.$$

Then, the performance of a TMI can be described as a vector in $\mathbb{R}^3$. A measure of how varied a set of performance vectors is can be provided by taking the sum of the Euclidean distances between these vectors. We will call this measure the *perfor-*

mance variety. Formally, let there be N selected vectors, each of which consist of observations from M normalized performance metrics. Let z_{ij} be the jth normalized performance metric corresponding to the ith performance. Then, the performance variety is given by

$$\sum_{i=1}^{N} \sum_{i'=i+1}^{N} \sqrt{\sum_{j=1}^{M} (z_{ij} - z_{i'j})^2}.$$

7.8 Computational Results

In this section, we provide computational results comparing various UPR methods in application to TMI decision support and evaluation.

7.8.1 Methodology

Our data set consisted of the days between July 21st, 2010 to December 31st, 2014 on which GDPs were run at Newark Liberty International Airport, a total of 581 days. Our data source was the National Traffic Management Logs (NTML), a product of the FAA. Out of these days, 100 were randomly selected to be reference days. Existing research provides a distance score for any pair of days in terms of the traffic and weather conditions [156]. We found the 50 days most similar to the reference day according to the distance score provided by that research. Our envisioned decision-support tool would carry out this filtering step in order to identify the most relevant historical information for planning a TMI on the reference day. We then used a UPR method to identify the representative TMIs. We chose the number of representatives to be equal to 5 in this step, which we believe to be a

practical number for the intended application. Finally, the performance variety and all three distance-based measures (discussed in Section 7.3.3) were calculated for the representatives produced. This entire procedure was repeated for each set of possible features discussed in Section 7.7.2 and for each potential UPR method. The UPR methods that we consider are the KC-UPR, k-medoids, OptiSim and Isolation Forest methods. We also ran this procedure with a method that selected the representatives completely at random. The KC-UPR method was solved exactly using the method in [105]. For the OptiSim method, the distance threshold is set to zero, and we used three different values for the sample size, specifically 5, 10 and 25. In the tables and discussion below, we used the terms "OptiSim 5", "OptiSim 10" and "OptiSim 25" to refer to the OptiSim method with these respective choices of parameters. Note that for those methods that incorporate some randomness into the selection process (namely the Isolation Forest method, the OptiSim variants, and the method that makes its selection uniformly at random) there are two sources of variance that affect the results. The first source is the variability in the characteristics of the days, while the second source is the variability in the performance of the method on a given day. We assume that if one of these methods were implemented, it would only be run a single time on each day, and that the resulting representatives from this single run would be used. Under this assumption, the standard deviations of the quality measures observed in our experiments are indicative of the variability in performance that would be observed in practice. However, these results do not provide much insight regarding the variability of the performance that would be observed if a method were repeatedly run in a single, given situation. We leave this

as an avenue for future research.

7.8.2 Performance By Feature Set

We use the performance variety averaged across the 100 reference days as the criteria to determine which feature set produced the best set of representatives. The distance-based measures are not used for this purpose because different choices of features result in different measures of distance, so distance-based measures cannot be reasonably compared between feature sets. The average performance variety for each set of features and each UPR method is shown in Table 7.2. For each UPR method, the value corresponding to the feature set that results in the highest performance variety is shown in bold. The values shown in parentheses are the standard deviations across the sample of 100 reference days. In almost all cases, the variance of the resulting performance variety is large enough to explain the differences in performance. Since the feature sets do not seem to differ greatly by this measure, any of the feature sets seems to be reasonable for our application.

7.8.3 Performance By Method

The results for all UPR methods and all four quality measures in the computational trials for feature set 0 are shown in Table 7.3. The Kennard-Stone method is abbreviated as KS and the Isolation Forest method is abbreviated as IF. For the sake of brevity, the results for the other combinations of features are not included here, but are available in Appendix E. The observations that we make here tend to

Table 7.2: Performance Variety of Unsupervised Prototype Reduction Methods.

Features	KC-UPR		K-medoids		Isolation Forest		Random	
0	25.25	(10.15)	23.88	(10.60)	27.57	(9.28)	16.48	(7.50)
1	**25.32**	(10.34)	24.01	(10.49)	**28.25**	(10.08)	15.19	(6.96)
2	24.66	(10.70)	**24.14**	(10.47)	27.55	(9.49)	15.78	(7.99)
3	24.82	(10.39)	23.33	(10.31)	26.99	(9.46)	**16.67**	(7.39)
4	24.35	(10.61)	23.79	(10.42)	26.89	(9.14)	15.40	(6.89)
5	25.12	(10.55)	23.87	(10.55)	27.39	(9.61)	14.88	(6.44)
6	24.77	(10.23)	24.12	(10.14)	27.62	(9.70)	15.40	(6.98)
7	24.18	(10.00)	21.65	(10.16)	27.45	(10.39)	17.17	(7.84)
8	23.14	(9.81)	23.13	(9.56)	23.84	(7.89)	16.63	(7.88)
Features	Kennard-Stone		OptiSim 5		OptiSim 10		OptiSim 25	
0	28.47	(11.28)	19.39	(9.29)	24.66	(10.75)	26.71	(10.72)
1	28.19	(11.27)	20.62	(9.83)	22.96	(10.29)	**28.00**	(10.39)
2	27.87	(10.66)	19.79	(8.51)	23.19	(9.86)	26.99	(11.24)
3	**28.79**	(11.45)	21.04	(9.38)	23.61	(10.03)	27.00	(10.90)
4	28.48	(10.90)	20.60	(9.65)	24.40	(10.33)	27.79	(11.06)
5	28.37	(11.61)	20.65	(9.55)	23.24	(10.32)	27.39	(10.59)
6	28.21	(10.75)	19.74	(9.37)	24.60	(11.54)	26.14	(10.15)
7	28.03	(11.14)	**21.87**	(9.95)	**24.68**	(10.26)	26.38	(10.33)
8	23.68	(9.31)	18.08	(8.09)	23.86	(9.59)	24.11	(9.64)

be true for the eight other combinations of features. For the performance variety, a higher number is better, while for the three distance-based measures a lower number is better. For each quality measure, the value corresponding to the UPR method that produced the best result is shown in bold, and the values shown in parentheses are sample standard deviations. Recall that the Kennard-Stone method is a special case of the OptiSim method. In this example, the Kennard-Stone method would be equivalent to an OptiSim method with distance threshold D set to 0 and with subsample size M set to 50. We note that the performance of the OptiSim method does seem to approach that of the Kennard-Stone as the parameter M is increased. The Kennard-Stone method seems clearly preferable to the Isolation Forest method, as the former outperformed the latter by all performance measures. The KC-UPR

Table 7.3: Quality Measures of UPR Methods, 5 Representatives, Feature Set 0.

Method	Perf. Variety		Max-Min Dist.		Sum Dist.		Sum Sq. Dist.	
KC-UPR	25.25	(10.15)	**0.804**	(0.043)	32.93	(1.27)	24.18	(1.91)
K-medoids	23.88	(10.60)	0.901	(0.061)	**32.13**	(0.62)	**23.08**	(0.97)
KS	**28.47**	(11.28)	0.904	(0.045)	33.99	(1.37)	25.92	(2.19)
OptiSim 5	19.39	(9.29)	0.941	(0.054)	32.90	(1.00)	24.26	(1.57)
OptiSim 10	24.66	(10.75)	0.924	(0.055)	33.32	(1.59)	24.90	(2.55)
OptiSim 25	26.71	(10.72)	0.899	(0.053)	33.54	(1.46)	25.22	(2.32)
IF	27.57	(9.28)	0.941	(0.054)	36.18	(2.30)	29.43	(3.77)
Random	16.48	(7.50)	0.966	(0.035)	33.19	(0.95)	24.75	(1.50)

method similarly dominates the OptiSim 10 method. There also seems to be little reason to use the OptiSim 5 method over the KC-UPR method. The KC-UPR method performed significantly better than the OptiSim 5 method in terms of performance variety and max-min distances, while the performance of the two methods was nearly identical in terms of sum of distances and sum of squared distances. The random selection method, predictably, performed poorly, and is dominated by the KC-UPR, k-medoids, and OptiSim 5 methods.

It is less clear which method is the best among the KC-UPR, k-medoids, Kennard-Stone and OptiSim 25 methods. None of these four methods dominated all of the other methods in all performance criteria. Kennard-Stone exhibited high performance variety but exceedingly poor performance in terms of sum of distances and sum of squared distances. In fact, this method was worse than random sampling in terms of these last two measures. The OptiSim 25 method shares similar strengths and weaknesses to the Kennard-Stone method, but these are less exaggerated. The performance variety of the OptiSim 25 method was still relatively high, but not as high as the Kennard-Stone method. Likewise, the sum of distance and sum of squared distances produced by the OptiSim 25 method were relatively high, but

not quite as high as the Kennard-Stone method. The advantages and disadvantages of the k-medoids method contrast with those of OptiSim 25 and the Kennard-Stone method. The k-medoids method produced by far the lowest sums of distances and sums of squared distance, but also produced the lowest performance variety. The OptiSim 25, Kennard-Stone, and k-medoids methods all performed similarly in terms of max-min distances. The KC-UPR method provided by far the lowest max-min distances, and gave the most balanced performance of all the methods. While the performance variety of the KC-UPR method is not quite as high as the OptiSim 25 or Kennard-Stone methods, it did produce higher performance variety than the k-medoids method. Similarly, while the KC-UPR method did not perform quite as well as the k-medoids method in terms of sums of distances or sums of squared distance, it performed better than the OptiSim 25 method and the Kennard-Stone method under these objectives. Based on these observations, we make the following recommendations:

- If high performance variety is desired but good coverage as measured by the sum of distances or sum of squared distances measures is not critical, then we recommend the Kennard-Stone method.
- If the sum of distances or sum of squared distances are especially pertinent and high performance variety is not, then we recommend the k-medoids method.
- If a low max-min distance is desired or if the method needs to perform adequately by all measures, then we recommend the KC-UPR method.

We believe that for our application, both the performance variety and the distance-

based measures capture important aspects of the problem. While these results provide some insight into quality of the summarizations produced by these methods, we would again like to emphasize that the ability of these methods to support air traffic management decisions in practice can only be determined through carefully constructed human-in-the-loop trials.

In general, the performance criteria given here cannot be simultaneously optimized, and we do not believe that it is possible to develop a method that would dominate all of the existing methods. However, it may be possible to produce methods that perform more consistently across multiple criteria. One possible manner of developing such a method would be to use a multi-criteria optimization approach. The KC-UPR could easily be modified to accommodate multiple criteria with a hierarchical optimization technique. For example, the sum of distance or sum of square distance objective could be minimized subject to the constraint that the selected points must achieve the minimum max-min distance. We leave the development of such techniques as an avenue for future research.

7.8.4 Sensitivity to Number of Representatives

We conducted two sets of experiments similar to the ones described above, but changed the number of representatives that were selected. In one set of experiments we identified sets of 3 representatives while in the other set of experiments we identified sets of 10 representatives. The results from the former experiments for feature set 0 are shown in Table 7.4, while the results for the latter experiments are

Table 7.4: Quality Measures of UPR Methods, 3 Representatives, Feature Set 0

Method	Perf.	Variety	Max-Min Dist.		Sum Dist.		Sum Sq. Dist.	
KC-UPR	8.27	(4.98)	**0.87**	(0.05)	36.28	(1.80)	28.22	(2.87)
K-medoids	4.96	(3.15)	0.95	(0.05)	**34.21**	(0.82)	**25.14**	(1.29)
KS	7.24	(4.21)	0.96	(0.03)	37.49	(2.57)	30.37	(4.23)
OptiSim 5	5.60	(3.55)	0.97	(0.04)	35.65	(1.79)	27.39	(2.89)
OptiSim 10	5.72	(3.64)	0.96	(0.04)	36.06	(1.81)	28.02	(2.93)
OptiSim 25	7.59	(4.49)	0.95	(0.04)	36.34	(2.02)	28.48	(3.31)
IF	**9.22**	(4.65)	0.98	(0.03)	41.05	(2.55)	36.20	(4.41)
Random	4.27	(2.66)	0.98	(0.03)	35.73	(1.49)	27.55	(2.41)

shown in Table 7.5. The results for all of these experiments with all sets of features are provided in Appendix E.

First, let us examine the results from the experiments with 3 representatives. There are some significant differences in these results. When there are fewer representatives, the relative performance of most methods by the sum of distances and sum of square distances decreases. The k-medoids and OptiSim 5 methods are the only methods that perform better than the random allocation by these criteria. The Kennard-Stone method is less competitive than in the previous experiments. The method still exhibits poor performance under the distance-based criteria, but no longer provides the highest performance variety. Both the OptiSim 25 and KC-UPR method dominate the Kennard-Stone method under all performance criteria in this set of experiments. In a similar fashion, the OptiSim 25 method is not as competitive, and is dominated by KC-UPR method. However, many of the observations we made for the experiments with 5 representatives hold. The Isolation Forest still provides the highest performance variety and the worst distance-based measures, and the k-medoids method still produces the lowest performance variety and the best sum of distance and sum of squared distance measures. It remains

Table 7.5: Quality Measures of UPR Methods, 10 Representatives, Feature Set 0

Method	Perf. Variety		Max-Min Dist.		Sum Dist.		Sum Sq. Dist.	
KC-UPR	94.34	(27.25)	**0.72**	(0.03)	**27.88**	(0.55)	**19.35**	(0.61)
K-medoids	96.26	(24.47)	0.79	(0.05)	27.90	(0.30)	19.50	(0.44)
KS	**103.36**	(28.42)	0.77	(0.05)	28.09	(0.62)	19.77	(0.92)
OptiSim 5	95.86	(29.03)	0.84	(0.08)	28.07	(0.38)	19.76	(0.57)
OptiSim 10	100.96	(25.83)	0.79	(0.05)	28.03	(0.49)	19.68	(0.73)
OptiSim 25	102.17	(26.54)	0.77	(0.04)	28.05	(0.59)	19.70	(0.87)
IF	99.84	(25.76)	0.85	(0.07)	28.58	(0.77)	20.52	(1.17)
Random	70.63	(25.31)	0.94	(0.06)	28.73	(0.57)	20.81	(0.90)

true that OptiSim 5 provides worse performance variety and max-min distance than the KC-UPR method while providing similar performance by the sum of distances and sum of squared distance measure.The KC-UPR method is still one of the better balanced methods. This method was second-best in performance variety, best in max-min distance, and is comparable to or better than all methods except for k-medoids by the sum of distance and sum of square-distance measure.

Next, let us examine the results from the experiments with 10 representatives. Recall that after filtering, each dataset only has 50 data points, so the number of representatives in these experiments is quite large relative to the quantity of data. Again, there are some significant differences in these results. With more representatives, most methods perform roughly the same under the sum of distances and sum of squared distances measure. For this reason, the quality of the methods is determined primarily by the performance variety and the max-min distance. In this set of experiments, the Kennard-Stone method is more competitive than in the previous experiments, and dominates all other methods except for the KC-UPR method. The KC-UPR method still performs significantly better than all

other methods by the max-min distance, but in these experiments it has the lowest performance variety of all methods except for the random method. Thus, if the number of representatives is large relative to the size of the dataset, we recommend the use of the Kennard-Stone method or the KC-UPR method. If variety of the representatives is more important than coverage of the data, then Kennard-Stone should be used; if coverage is more important than variety, then KC-UPR should be used.

7.9 An Example of Exploring a Data Set with Representatives

As an example of how the KC-UPR method can be used for data exploration, we consider a data set of TMIs that have been run at Newark Liberty International Airport (EWR) from July 20th, 2010 to December 31st, 2014. We select the features to be those in feature set 0 as described in Section 7.7.3 and then we compute the maximum pairwise-quantile distance measure as described in Section 7.6. We choose 10 to be the number of desired representatives, as this is a reasonable number for human examination. We use a larger number of representatives in this example than in Section 7.8 because this example does not include the filtering step, so the data set is much larger. The KC-UPR method was solved using the exact method in [105]. As mentioned in Section 7.6.2, due to the construction of the maximum pairwise-quantile distance, the resulting optimal distance can be described in terms of each individual feature. The resulting feature distances are shown in Table 7.6. These indicate that for any TMI in our data set there is a representative TMI such

Table 7.6: Feature Distances in K-Center Solution.

Feature	Distance	
File Time	267	minutes
Start Time	210	minutes
GDP Duration	330	minutes
GS Duration	49	minutes
Scope (number of CORE 30 airports)	10	airports
Average Rate	7.21	flights per hour
Vector of Relative Rates	0.705	(unitless)

that the difference in file times is at most 267 minutes, the difference in start times is at most 210 minutes, and so on.

The features of the representatives and the prevalence may also be presented. Since the KC-UPR method generates representatives that are members of the original data set, it is possible to present features of the TMI that were not used in the generation of the representatives. For example, recall that the scope of a GDP is usually specified as either a list of regional traffic control centers or a numerical radius. In order to make these scopes comparable, we instead use the number of included CORE 30 airports to generate the representatives. In the presentation of the representatives, the scope can be displayed as originally specified. Every TMI is assigned an "Impacting Condition" which describes the cause of the TMI. This feature was not used at all when generating representatives, but it can be displayed with the results. The features and prevalence of each representative are shown in Tables 7.7. Some features are omitted for the sake of brevity. All times shown are local times.

By examining the prevalences of the representatives, one can immediately get a rough idea of how the data set looks. The TMI that occurred on 10/29/2010

Table 7.7: 10 Representative TMIs from EWR

Date	Start Time	GDP Duration	GS Duration	Impacting Condition	Scope	Prev.
10/29/2010	12:00	659 min	0 min	Wind	1425 nm	0.8210
02/27/2014	14:46	433 min	46 min	Wind	1400 nm	0.4596
02/11/2013	12:00	599 min	0 min	Low Ceilings	1175 nm	0.3993
09/25/2014	15:00	599 min	0 min	Construction	All	0.2358
09/16/2010	17:36	383 min	73 min	Thunderstorms	1400 nm	0.0688
04/20/2011	12:07	777 min	51 min	Low Ceilings	All	0.0310
12/10/2013	08:13	1006 min	46 min	Snow-Ice	All	0.0224
02/15/2014	12:29	510 min	33 min	Snow-Ice	All	0.0138
01/27/2011	13:00	719 min	0 min	Snow-Ice	All	0.0034
01/26/2011	11:43	736 min	103 min	Snow-Ice	All	0.0017

has extremely high prevalence. More than 80% of the TMIs are within the optimal distance threshold from the KC-UPR problem. Roughly 45% and 40% of the TMIs are similar to the TMIs that occurred on 02/27/2014 and 02/11/2013, respectively. If there were a need to produce typical examples of TMIs run at EWR then these three TMIs would be good choices, especially the one that occurred on 10/29/2010. Slightly less than 25% of the TMIs are similar to the representative with the fourth-highest prevalence. This is still a significant portion of the data, but is considerably less than the prevalence of the preceding three representatives. Following this representative, the remaining representatives all have fairly low prevalence. This suggests that there is a single group of TMIs that are all relatively similar to each other, while there are several smaller, less typical groups of TMIs.

These results can lead us to further avenues of data exploration. For example, the four lowest-prevalence representatives all appear to have an impacting condition of "Snow-Ice", while the two highest-prevalence representatives both have an impacting condition of "Wind:. This suggests that TMIs initiated because of snow or

ice tend to be rarer and more varied than those initiated because of wind. We can confirm this by looking more closely at the data. Indeed, only 20 of the TMIs in our data set list Snow-Ice as the impacting condition, and manual examination of these TMIs reveals that this group exhibits a wide variety of rates, file times, start times and durations. As another example, consider the representative TMI that occurred on 09/25/2014. The impacting condition of this TMI is "Construction". In our discussions with TMI decision makers, wind and low ceilings were frequently mentioned as important causes of TMIs, but construction was rarely mentioned. Since our method generated a representative of moderate prevalence caused by construction, we reexamined the data to determine whether this was a significant catalyst of TMIs. We found that 79 TMIs in our data set were caused by runway construction or maintenance. This does represent a significant portion of the 581 TMIs in the data set.

While we present simply the features and prevalence here, it might be useful to present other information. For example, in order to use the representatives to help plan or evaluate TMIs, it would be helpful to produce estimates of the performance of the representatives and present these along with the features. It might also be useful to provide information that provides context to the TMI decision, such as the current and forecast weather or traffic at the time the TMI was initiated.

Table 7.8: Feature Distances in K-Center Solution, 5 Representatives.

Feature	Distance	
File Time	408	minutes
Start Time	322	minutes
GDP Duration	443	minutes
GS Duration	70	minutes
Scope (number of CORE 30 airports)	10	airports
Average Rate	11.07	flights per hour
Vector of Relative Rates	2.459	(unitless)

7.9.1 Sensitivity to Number of Representatives

In order to show how these results can be affected by the number of representatives, we also used the KC-UPR method to generate a set of 5 representatives for this data set. The optimal distance, again described in terms of each individual feature, is shown in Table 7.8. Naturally, a set of 5 representatives cannot summarize a dataset as well as 10 representatives can, so this distance must be larger than that achieved with 10 representatives. In fact, the distance achieved by the set of 5 representatives is significantly larger than that achieved by the set of 10 representatives. This indicates that 10 representatives can provide a much more accurate summary of the data than 5 representatives can.

The set of 5 representatives produced by the KC-UPR method are shown in Table 7.9. The larger optimal distance tends to inflate the prevalences, since each representative has more points within the distance threshold. Despite the fact that this summarization is much coarser, there is still some information that can be provided by the representatives. The highest-prevalence representative in the set of 5 representatives (07/24/2013) has parameters that fall between the two representa-

tives with the highest prevalence from the set of 10 representatives, and remains a reasonable choice for a typical TMI at EWR. The lowest-prevalence representative in the set of 5 representatives is the third-lowest-prevalence representative from the set of 10 representatives (02/15/2014), and this still accurately conveys that this type of TMI is rare.

The other representatives are somewhat misleading. The TMI that occurred on 08/31/2014 in the set of 5 representatives is similar to the one that occurred on 09/16/2010 in the set of 10 representatives. In the set of 5 representatives, the prevalence of this TMI does accurately indicate that this type of TMI has been used more frequently than TMIs similar to the one that occurred on 02/15/2014, but less frequently than TMIs similar to the one implemented on 07/24/2013. However, the prevalence of this representative seems to suggest that this type of TMI has been run very frequently, while investigation of the data shows that TMIs run in response to thunderstorms constitute a relatively small portion of the TMIs. Similarly, the representative TMI that occurred on 02/03/2014 has lower prevalence relative to that of 07/24/2013 and 08/31/2014, which accurately describes the relative frequency of these types of actions. However, examining the prevalence of 02/03/2013 in isolation would suggest that it is more common than it actually is. The representative 01/28/2013 is extremely misleading. This TMI has the 11th-highest GDP duration and the 6th-earliest start time of any TMI. This would not be considered a typical TMI by any reasonable measure, but the high distance score greatly inflates the prevalence of this TMI.

Overall, this sensitivity analysis demonstrates that there are limits to the

Table 7.9: 5 Representative TMIs from EWR

Date	Start Time	GDP Duration	GS Duration	Impacting Condition	Scope	Prev.
07/24/2013	13:21	578 min	6 min	Wind	1400 nm	0.9380
01/28/2013	10:30	869 min	0 min	Snow-Ice	All	0.7263
08/31/2014	15:23	276 min	53 min	Thunderstorms	1400 nm	0.6799
02/03/2014	12:00	779 min	0 min	Snow-Ice	All	0.2530
02/15/2014	12:29	510 min	33 min	Snow-Ice	All	0.0723

ability of this method to summarize the dataset accurately. The optimal distance reported by the method does provide information concerning how well the representatives summarize the data, and users can use this to determine whether or not the representatives describe the data sufficiently well for their purposes. A possible future avenue of work is to develop an automated method that selects the quantity of representatives in a manner that balances the accuracy of the description with its succinctness.

7.10 Conclusions and Further Work

We investigated a new problem in data exploration called the Unsupervised Prototype Reduction (UPR) problem, which seeks to find a "representative" set of data points from a much larger data set. An application in air traffic management motivated our work but a broader set of applications exist. While there are several known close relatives of UPR, certain applications benefit from the special features incorporated into its definition. We identified several algorithms for solving this problem and focused particularly on the KC-UPR method. We compared all identified algorithms both qualitatively and with a set of computational experiments.

A distance measure was proposed that aids in the interpretation of results from the KC-UPR method. While no method clearly outperformed all others according to a set of performance criteria, the KC-UPR method exhibited the most balanced performance. For this reason, we recommend its use for our target application. We provided an example in which the KC-UPR method was used to explore air traffic management data.

The original motivation for this research was the development of a novel approach to supporting TMI planning. That approach involved maintaining historical data on previous TMIs and the associated day's salient weather and demand characteristics. To support the planning of a TMI, the decision support tool would assemble a set of "similar" days to provide the decision maker with insights into the impact of possible plans for the current day. This architecture led to the need to produce a representative set of days from the much larger set of similar days. The KC-UPR method was constructed to solve this problem and the results of this chapter identified it as the best choice among a set of alternatives. As discussed above, here making the "best" choice took into account multiple desirable metrics for measuring a set of representative days and the fact that a single method could not be best in all of them. The TMI planning architecture envisioned seeks to take advantage of virtually all available data and is highly user centric. It enables a user to see the actual impact of prior plans on days with similar characteristics. To the extent that models can produce alternate plans it gives those plans context and also provides a basis for comparing a new plan with plans used in the past. The hope is that such a decision support tool will overcome the various impediments, e.g. as

discussed in Section 7.2, that have prevented adoption of advanced tools including optimization-based approaches. The very nature of this approach to TMI planning makes it very difficult to evaluate its impact without a human-in-loop experiment or actual implementation. Our experimental results have shown that the KC-UPR method is superior to alternatives for the specific need of the desired decision support tool. The further development of the broader TMI planning architecture and its benefits assessment are beyond the scope of this chapter and represent attractive future research topics.

It is noteworthy that this work could immediately be used in similar related applications. For example, it could be used by traffic management specialists to identify comparable days for post-decision analysis or to aid in the development of TMI "playbooks" that describe standard traffic management strategies. This could also aid researchers studying air traffic management, as it could help them to understand the TMI strategies currently used in practice. This method could also be used to select sample TMIs for simulations.

In the examples we considered here, we used an exact method [105] to solve the KC-UPR problem. For larger problems, this method may not be tractable. There exist many heuristics as mentioned in Section 7.4.2. However, these heuristics have mostly been developed for the context of facility location problems. Instances of these problems tend to be relatively small when compared to the size of large data sets. As a result, the existing heuristics would likely not be computationally tractable for especially large data sets. For example, most of these heuristics assume that the pairwise distances between points can be sorted in a reasonable amount of

time. This would not be true for extremely large data sets. Thus, if the KC-UPR method is to be applied to big data sets, new heuristics for the problem will be necessary.

Appendix A: Proofs for Chapter 2

This appendix contains the proofs of results in Chapter 2.

A.1 Proof of Proposition 1

This section contains the proof of Proposition 1. Oracle optimality implies state-by-state dominance because the value of the deterministic problem for an instance ω is a lower bound on the cost that any policy can achieve for that instance. The fact that state-by-state dominance implies robust dominance follows immediately from the definitions. Let π be a robustly dominant policy. Let ω be some instance of some arrival process Ω. Define an arrival process Ω' for which the instance ω is the only possible instance. Note that we can create a policy ψ_ω such that

$$\mathbf{C}(\psi_\omega; \omega) = \mathbf{C}^*(\omega).$$

Then,

$$\begin{aligned}\mathbf{C}(\pi;\omega) &= \sup_{\nu\in\Omega'} \mathbf{C}(\pi;\nu)\\ &\leq \sup_{\nu\in\Omega'} \mathbf{C}(\psi_\omega;\nu)\\ &= \mathbf{C}^*(\omega).\end{aligned}$$

So, π is oracle optimal.

A.2 Proof of Lemma 2

This section contains the proof of Lemma 2. First, we prove another lemma. Assuming that the DSTP is time-partitioned, then if an arc is included in the subset U_t of some subset-greedy policy in the time period t, the policy will never assign flow on that arc in any later time period.

Lemma 16. *Let there be a time-partitioned dynamic transportation problem. Let there be a subset-greedy policy π with subsets $U_0, U_1, ..., U_T$ and corresponding orderings $\prec_0, \prec_1, ..., \prec_T$. Let there be some time period t, $a \in \mathscr{A}$ and $b \in \mathscr{B}$ such that*

$$(a,b) \in U_t.$$

Then

$$x(t', a, b; \pi; \omega) = 0$$

for all $t' > t$ and all ω in any arrival process Ω.

Proof. Proof of Lemma 16. First note that

$$\min\{s^x(t, a; \pi; \omega), d^x(t, b; \pi; \omega)\} = 0.$$

That is, after the allocation in time t there must either be no more remaining supply at a or no more remaining demand at b. This is true because the arc (a, b) is in the subset U_t of arcs, and thus the greedy policy will attempt to move as many resources as possible along this arc. Since the problem is time-partitioned and $(a, b) \subseteq \mathscr{F}_t$, then a is a supply node in $\mathscr{A}_\tau$ and b is a demand node in $\mathscr{B}_{\tau'}$ for some $\tau, \tau' \leq t$. Then, from the definition of the time-partitioned transportation problem, the nodes a and b will receive no supply and demand respectively at the beginning of time period $t + 1$. Thus,

$$s(t + 1, a; \pi; \omega) = s^x(t, a; \pi; \omega);$$

$$d(t + 1, b; \pi; \omega) = d^x(t, b; \pi; \omega).$$

From Lemma 1, we have that

$$s(t', a; \pi; \omega) \leq s(t + 1, a; \pi; \omega),$$

$$d(t', b; \pi; \omega) \leq d(t + 1, b; \pi; \omega),$$

for any $t' \geq t + 1$. Thus,

$$\min\{s(t', a; \pi; \omega), d(t', b; \pi; \omega)\} = 0.$$

This implies that

$$x(t', a, b; \pi; \omega) = 0.$$

□

Now we are ready to prove Lemma 2. Consider some time period t. Let

$$V = \left(\bigcup_{t'=0}^{t-1} U_{t'} \right).$$

If V is disjoint from U_t, then $U'_t = U_t$, and so the policy π' will perform exactly the same as the policy π in time period t. In the case that V is not disjoint from U_t, then consider some arc (a, b) in $V \cap U_t$. From Lemma 16, we know that the policy π will not allocate any resources along the arc (a, b) in time t since this arc was present in a subset U_τ corresponding to an time period τ earlier than t. Thus, we can remove this arc from the set U_t, and it will not affect the allocation made in time t. Extending this argument, we can remove the entire set $V \cap U_t$ from U_t and it will not affect the allocation. This completes the proof. □

A.3 Proof of Theorem 2

In this section, we prove Theorem 2. It is important to note that in the formulation of the deterministic problem, it is assumed that all allocations are made at the time that minimizes the cost. A subset-greedy policy does not necessarily make allocations at that time. However, if the subset-greedy policy does make its allocations at the appropriate times, then the incurred cost is the same as that of a greedy algorithm applied to the deterministic problem. This gives us the following lemma:

Lemma 17. *Let there be a time-partitioned dynamic transportation problem. Let π be a subset-greedy policy with disjoints subsets $U_0, U_1, ..., U_T$ and corresponding orderings $\prec_0, \prec_1, ..., \prec_T$, where the subsets $U_0, .., U_T$ form a partition of $\mathscr{F}^*$. Let $\prec^*$*

be the corresponding ordering on $\mathscr{F}^*$ as described in Lemma 3. Let $\mathbf{C}^G(\omega)$ be the cost obtained by applying the greedy algorithm with ordering $\prec^*$ to the deterministic problem for instance ω. Then

$$\mathbf{C}(\pi;\omega) \geq \mathbf{C}^G(\omega).$$

for any instances ω. Furthermore, if $U_t \subseteq \mathscr{M}_t$ for all t then

$$\mathbf{C}(\pi;\omega) = \mathbf{C}^G(\omega)$$

for any instance ω, while if $U_t \not\subseteq \mathscr{M}_t$ then there exists some instance ω for which

$$\mathbf{C}(\pi;\omega) > \mathbf{C}^G(\omega).$$

This lemma follows easily from Lemma 3. From Lemma 3, it is known that the total amount allocated along each arc is equal to the amount allocated in the greedy solution to the deterministic problem. The cost of each arc in the deterministic problem is defined to be the minimum possible cost of assigning resources along that arc. Thus, if the subset-greedy policy makes its allocations at the minimum cost times, the incurred costs will be the same as the greedy solution to the deterministic problem. On the other hand, if the subset-greedy policy makes some allocation at a time when the cost is not minimized, then the policy will incur a greater cost.

Now, we can prove the theorem. First, we prove the contrapositive of the forward direction. Suppose it is not true that $U_t \subseteq \mathscr{M}_t$ for some t. Then, by Lemma 17 the subset-greedy policy must incur a cost greater than the deterministic problem for some instance of some arrival process. Thus, π would not be oracle-optimal. Alternatively, suppose that $\prec^*$ does not produce a Monge sequence for the

deterministic problem. Then by Theorem 1, there must be a some set of supplies and demands such that the greedy solution with ordering $\prec^*$ does not produce the optimal solution. Construct an instance of the arrival process ω that has these supplies and demands. By Lemma 17, the performance of the subset-greedy policy π is no better than the performance of the greedy solution corresponding to $\prec^*$. Thus, π would also not produce the optimal solution to the deterministic problem, and therefore would not be oracle-optimal.

In the opposite direction, let $U_t \subseteq \mathscr{M}_t$ for all t and let $\prec^*$ produce a Monge sequence for the deterministic problem. Let ω be some instance of some arrival process Ω. Then by Lemma 17, we know that π achieves the same cost as the greedy solution with ordering $\prec^*$ in instance ω. Since $\prec^*$ is a Monge sequence for the deterministic problem, then this greedy solution is optimal. Thus, π is oracle-optimal.

A.4 Proof of Theorem 3.

In this appendix, we provide a proof of Theorem 3. It is sufficient to prove a special case of this theorem in which the extra assignments $\mathbf{r}$ use all of the supply and demand at the nodes of the Monge star. We will show that this proof can be extended to the case where some supply or demand is left at these nodes. Our proof is constructive. We first provide the algorithm, and then we show that it satisfies the conditions of the theorem.

A.4.1 Algorithmic Construction

Let $(\aleph, B)$ be a Monge star. Let π be a policy, and let $\mathbf{r}$ be any vector of assignments on the arcs of $\mathscr{F}_0$ such that

$$\sum_{b \in B} r_{\aleph b} = s^x(0, \aleph; \pi),$$

$$r_{\aleph\beta} = d^x(0, \beta; \pi) \qquad \forall \beta \in B,$$

$$r_{ab} = 0 \qquad \forall (a, b) \text{ such that } a \neq \aleph \text{ or } b \notin B,$$

$$r_{\aleph b} \geq 0.$$

We provide an algorithm for implementing a policy ϕ that makes the additional assignments $\mathbf{r}$ and that dominates the policy π. This algorithm is designed to facilitate the proof of the theorem, and little care is taken towards making the algorithm computationally efficient.

An assignment of resources from a supply node to a demand node can be written as a tuple (t, a, b, q) where t is the time period in which the assignment is made, a is the supply node, b is the demand node, and q is the quantity moved from a to b. The algorithm attempts to match assignments made by the policy π with those made by the policy ϕ. These assignments are matched in three different ways. First, if π makes an assignment in some time period t that is along the same as one of the excess assignments that the policy ϕ makes in time period zero, then the new assignment can be matched with the old one. Second, ϕ can make an assignment along the same arc as π, but ϕ makes the assignment in an earlier time period. In the third and final case, pairs of assignments are matched. This occurs when ϕ

makes two assignments along some arcs $(\aleph, \beta)$ and (a, b) while π instead makes this assignment along some arcs $(\aleph, b)$ and (a, β). We will call this a *swap*.

The algorithm maintains four sets of assignments. The first set U^ϕ contains assignments that were made by ϕ that have not yet been matched to assignments made by π. The second set U^π conversely contains assignments made by π that have not yet been matched to assignments made by ϕ. The third set M^π contains the assignments made by π that have been matched to those made by ϕ, while the fourth set M^ϕ contains those assignments made by ϕ that have been matched.

The policy ϕ in the initial time period is defined by

$$x(0, a, b; \phi) = x(0, a, b; \pi) + r_{ab}.$$

In the initial time period, every assignment that π makes is imitated by ϕ. Thus, at the end of time period zero, we define the sets M^π and M^ϕ to consist of the assignments $(0, a, b, x(0, a, b; \pi))$ on the feasible arcs (a, b) in $\mathscr{F}_0$. For a similar reason, we define the set U^π of assignments made by π to be empty at the end of time period zero. The policy ϕ does make some additional assignments that ϕ does not, specified by the vector $\mathbf{r}$. Thus, we initialize the set U^ϕ with an assignment $(0, \aleph, \beta, r_{\aleph\beta})$ for each arc $(\aleph, \beta)$ such that $r_{\aleph\beta} > 0$. In summary, at the end of time period zero,

$$\begin{aligned} U^\pi &= \emptyset; \\ M^\pi = M^\phi &= \{(0, a, b, x(0, a, b; \pi)) : (a, b) \in \mathscr{F}_0, x(0, a, b; \pi) > 0\}; \\ U^\phi &= \{(0, a, b, r_{a,b}) : (a, b) \in \mathscr{F}_0, r_{a,b} > 0\}. \end{aligned}$$

For each time period greater than zero, the policy is defined by Algorithm 5. Note that this algorithm takes as an input the sets U^π, U^ϕ, M^π, M^ϕ, and that the algorithm alters these sets. These sets are maintained for the entire duration of the problem, so that any change made to these sets in some time period persists into the next time period. For example, if an assignment is removed from U^ϕ in time period t, then that assignment will still be absent from the list in the beginning of time period $t+1$.

The algorithm is divided into four sections. The first section (lines 2 to 4) simply initializes the variables X^π and X^ϕ. The variable $X^\pi[a,b]$ stores the quantity of resource allocated by π along the arc (a,b) that has been matched by the algorithm, while $X^\phi[a,b]$ stores the quantity of resources that the policy ϕ will allocate along the arc (a,b). The second section (lines 6 to 28) consists of a single for-loop. We will refer to this for-loop as "the main for-loop". This loop iterates over the set of arcs that would receive a positive quantity of resources under the policy π in time period t and attempts to match this assignment. This for-loop is further divided into two subsections. The first subsection (lines 7 to 17) checks to see if some portion of the assignment can be matched with an assignment that ϕ has already made. If this is possible, then this matching is made. The second subsection (lines 19 to 27) handles the remaining quantity of the assignment. If there is sufficient supply and demand remaining at the appropriate nodes, then the policy ϕ can simply make an assignment in the current time period that matches the remaining quantity. If not, then some of the quantity will remain unmatched, and will be stored in the list U^π. The final section (lines 29 to 37) consists of a while-loop that attempts to make

swaps. We often refer to this loop simply as "the while-loop". The final section (lines 39 to 41) simply defines the output.

A.4.2 Proof Layout and Notation.

Our proof strategy is as follows. First, we prove some general properties of the algorithm. We show that the set U^ϕ and U^π never contain assignments along the same arc. We also prove that the nodes satisfy a type of conservation property. This allows us to relate the amounts remaining at these nodes with the amounts of unmatched assignments involving those nodes. We also show that each assignment in the set U^ϕ involves at least one node of the Monge star. These properties allow us to prove that the algorithm provides a well-defined policy ϕ. That is, ϕ never assigns more supply and demand than exists. Our second goal is to prove that the algorithm records assignments and matches them correctly. We prove that every assignment made by each policy is recorded on either the list of unmatched assignments or the list of matched assignments, and never on both lists. We also show that the total cost of the matched assignments is always less for ϕ than it is for π. The third goal is to prove that by the end of problem, no assignments are left unmatched. We use our conservation property to show that if π is feasible then the list of unmatched assignments made by ϕ is empty at the end of the problem, and then we use these properties and the balance condition to show that this implies that U^π is empty as well. Finally, we combine these results to show that ϕ satisfies the properties described in the theorem. Since each assignment is either matched or unmatched

Algorithm 5: Policy ϕ for time period t with Monge star $(\aleph, B)$

Inputs: supply node $\aleph$ and set of demand nodes B from the Monge star; sets U^ϕ, U^π, M^ϕ, M^π; remaining supplies $s(t, a; \phi; \omega)$ for each $a \in \mathscr{A}$; remaining demands $d(t, b; \phi; \omega)$ for each $b \in \mathscr{B}$; assignment $x(t, a, b; \pi; \omega)$ that π makes on each arc $(a, b) \in \mathscr{F}_0$.
for each $(a, b) \in \mathscr{F}_t$ **do**
  Initialize $X^\phi[a, b] = X^\pi[a, b] = 0$.
end for
---
for each arc (a, b) such that $x(t, a, b; \pi; \omega) > 0$ **do**
  if there is some assignment $(0, a, b, q_1)$ in U^ϕ **then**
    Set $q = \min\{q_1, x(t, \aleph, \beta; \pi; \omega)\}$.
    Remove $(0, \aleph, \beta, q_1)$ from U^ϕ; if $q_1 > q$ replace with $(0, \aleph, \beta, q_1 - q)$.
    Add $(t, \aleph, \beta, q)$ to the list M^π.
    Add $(0, \aleph, \beta, q)$ to the list M^ϕ.
  else
    Set $q = 0$.
    Set $r = x(t, a, b; \pi; \omega)$.
  end if
  Set $X^\pi[a, b] = q$.
  Set $r = x(t, a, b; \pi; \omega) - q$.
  . . . . . .
  Let

$$q' := \left(\min \left\{ r, s(t, a; \phi, \omega) - \sum_b X^\phi[a, b], d(t, b; \phi, \omega) - \sum_a X^\phi[a, b] \right\} \right)^+$$

  Set $X^\pi[a, b] = x(t, a, b; \pi; \omega)$.
  if $q' > 0$ **then**
    Set $X^\phi[a, b] = q'$
    Add (t, a, b, q') to M^π and to M^ϕ.
  end if
  if $r > q'$ **then**
    Add $(t, a, b, r - q')$ to the list U^π.
  end if
end for

while there is an assignment of the form $(0, \aleph, \beta, q_1)$ in U^ϕ and a pair of assignments $(\tau, \aleph, b, q_2)$ and (τ', a, β, q_3) in U^π such that $(a, b) \in \mathscr{F}_t$ and $c(0, \aleph, \beta) + c(t, a, b) \leq c(\tau, \aleph, b) + c(\tau', a, \beta)$ **do**
  Let $q = \min\{q_1, q_2, q_3\}$.
  Increase $X^\phi[a, b]$ by q.
  Remove $(0, \aleph, \beta, q_1)$ from U^ϕ; if $q_1 > q$ replace with $(0, \aleph, \beta, q_1 - q)$.
  Remove $(\tau, \aleph, b, q_2)$ from U^π; if $q_2 > q$ replace with $(\tau, \aleph, b, q_2 - q)$.
  Remove (τ', a, β, q_3) from U^π; if $q_3 > q$ replace with $(\tau', a, \beta, q_3 - q)$.
  Add $(0, \aleph, \beta, q)$ and (t, a, b, q) to M^ϕ.
  Add $(\tau, \aleph, b, q)$ and (τ', a, β, q) to M^π.
end while

---

for $(a, b) \in \mathscr{F}_t$ **do**
  Output: set $x(t, a, b; \phi; \omega) = X^\phi[a, b]$
end for

and no assignments are unmatched at the end of the problem, then the cost of each policy can be calculated solely by looking at the list of matched assignments. The total cost of the assignments in this list is always less for ϕ than it is for π, so ϕ achieves a lower cost.

For a set of assignments L, we use $Q(L)$ to refer to the total quantity of assignments in the set. For a set L and supply node a, we let $L(a)$ to refer to the subset of L that involves the node a. We define $L(b)$, $L(a, b)$ and $L(t, a, b)$ similarly for a demand node b, arc (a, b) or arc (a, b) in time period t. That is,

$$Q(L) = \sum_{(t,a,b,q) \in L} q,$$

$$L(a) = \{(\tau, a', b', q) \in L : a' = a\},$$

$$L(b) = \{(\tau, a', b', q) \in L : b' = b\},$$

$$L(a, b) = \{(\tau, a', b', q) \in L : a' = a, b' = b\},$$

$$L(t, a, b) = \{(\tau, a', b', q) \in L : \tau = t, a' = a, b' = b\}.$$

Note that by definition of the algorithm, quantities of assignments placed in the sets U^π and U^ϕ are always strictly positive. We will make frequent use of these facts in the proof.

A.4.3 Properties of Algorithm 5.

By definition, the quantities of assignments placed in the sets U^π and U^ϕ are always strictly positive. We also note that any time an assignment (t, a, b, q) is placed in the list M^π or M^ϕ, the quantity q is chosen by taking the minimum of some quantities from assignments present in the lists U^π and U^ϕ. Thus, these assignments will also have strictly positive quantities. This implies that the total quantity in the list is always non-negative, and is zero if and only if the set is empty. Note also that every unmatched assignment in the list U^ϕ involves the supply node $\aleph$ and one of the demand nodes β from the Monge star. Furthermore, no assignments are ever added to this list after initialization, so this is true at any point in time. It is also true that there is at most one assignment corresponding to an arc in each time period in each set. These facts will be used frequently throughout this proof.

A useful property is that there cannot simultaneously exist an unmatched assignment made by π and an unmatched assignment made by ϕ along the same arc.

Lemma 18. *At any point in the execution of Algorithm 5 in any time period t, the following holds for any arc (a, b):*

$$Q(U^\pi(a, b))Q(U^\phi(a, b)) = 0.$$

Proof. Proof of Lemma 18. This is true at the beginning of the execution of the algorithm in the first time period because at that point U^π is empty. New assignments are never added to U^ϕ, and the quantities of existing assignments in U^ϕ can only decrease. Thus, the only way that this equality could be violated is if there were to exist some assignment $A_1 = (0, a, b, q_0)$ in U^ϕ when a new assignment $A_2 = (t, a, b, q^*)$ is added to U^π. We show that if such an assignment A_1 exists in U^ϕ, then the new assignment A_2 will not be added to U^π.

A new assignment A_2 can only be added to U^π in the second subsection of an iteration of the main for-loop corresponding to the arc (a, b) during some time period t. Suppose that when the second subsection of the main for-loop is executed, there still exists some assignment $(0, a, b, q_0)$ in U^ϕ. Since quantities of assignments in U^ϕ can only decrease in the execution of the algorithm, then when the if-statement on line 7 was executed, there must have been some assignment $A_1' = (0, a, b, q_1)$ in U^ϕ for some $q_1 \geq q_0$. Furthermore, if $q_1 \leq q$ then the assignment A_1' would have been removed in line 9, so it must be true that $q_1 > q$. This implies that $q = x(t, a, b; \pi; \omega)$. This in turn implies that r is set to zero when it is calculated on line 17, and that the value of q' calculated on line 19 is also zero. Thus, line 26 is not executed, and no assignment will be added to U^π in this iteration of the for-loop. □

The variables follow a conservation property that relates the unmatched assignments by each policy and the amount of remaining supplies and demands resulting from these policies.

Lemma 19. *In the execution of Algorithm 5 in time period t, at the beginning and*

end of each iteration of the for-loop beginning on line 6 the following hold:

1. *For any $a \in \mathscr{A}$,*

$$Q(U^{\phi}(a)) + s(t, a; \phi; \omega) - \sum_{b} X^{\phi}[a, b] =$$

$$Q(U^{\pi}(a)) + s(t, a; \pi; \omega) - \sum_{b} X^{\pi}[a, b].$$

2. *For any $b \in \mathscr{B}$,*

$$Q(U^{\phi}(b)) + d(t, b; \phi; \omega) - \sum_{a} X^{\phi}[a, b] =$$

$$Q(U^{\pi}(b)) + d(t, b; \pi; \omega) - \sum_{a} X^{\pi}[a, b].$$

These statements also hold at the beginning of the second section of the for-loop (line 19) and at the beginning and end of the while-loop beginning on lines 29.

Proof. Proof of Lemma 19. Here, we prove that property 1 holds. The proof of property 2 is nearly identical.

The property is true after initialization. First, we prove that this is true at the beginning of the first iteration of the for-loop on line 6 during time period 1. By definition of how we initialized U^{ϕ} and M^{ϕ},

$$Q(U^{\phi}(a)) = \sum_{b:(a,b)\in\mathscr{F}_0} r_{ab},$$

and

$$s(1,a;\phi;\omega) = s^{\text{new}}(0,a) + s^{\text{new}}(1,a) - \sum_b x(t,a,b;\phi;\omega)$$
$$= s^{\text{new}}(0,a) + s^{\text{new}}(1,a) - \sum_b x(t,a,b;\pi;\omega) - r_{ab}.$$

Then

$$Q(U^\phi(a)) + s(t,a,\phi;\omega) = s^{\text{new}}(0,a) + s^{\text{new}}(1,a) - \sum_b x(t,a,b;\pi;\omega)$$
$$= s(t,a,\pi;\omega).$$

At this point in execution, $X^\phi[a,b] = 0$ and $X^\pi[a,b] = 0$ for any arc (a,b). The list U^π is empty at that point in execution, so $Q(U^\pi(a)) = 0$. Thus, this property holds at the beginning of the first iteration of the for-loop on line 6.

The main for-loop preserves the property. We now show that the main for-loop preserves property 1. Assume that the statement holds at the beginning of some iteration of the for-loop. Let (a,b) be the arc that corresponds to this iteration of the for-loop. Consider some equality in property 1 corresponding to some supply node a'. If $a \neq a'$, then none of the terms of the equality will be altered in this for-loop. Thus, it is sufficient to consider the case in which $a = a'$.

First, we show that the first subsection of the for-loop will preserve the equality. Consider the case that the if-statement on line 7 is executed. If $a = a'$, then $Q(U^\phi(a))$ will decrease by q on line 9 and $X^\pi[a,b]$ will increase by q. None of the other terms will be changed. Thus, the quantities on either side of the equality will both be preserved by this section. In the case that the else-statement is executed,

then $X^{\pi}[a, b]$ will increase by q while no other terms will change. However, $q = 0$ in this case, so in fact none of the terms of the equality will change. Thus, in either case, the first subsection of the main for-loop will preserve the equality.

Next, we show that the second subsection of the main for-loop will preserve the equality. In line 22, the value of $X^{\phi}[a, b]$ increases by the amount q. No other lines affect the values of any of the terms on the left-hand-side. Thus, within this section the left-hand-side will decrease by exactly q. In line 20, the value of $X^{\pi}[a, b]$ increases by $x(t, a, b; \pi; \omega) - q$. In line 26, the value of $Q(U^{\pi}(a))$ increases by $r - q'$, which is equal to $x(t, a, b; \pi; \omega) - q - q'$. No other lines affect the values of any of the right-hand-side terms. Thus, the right-hand-side will also decrease by the amount q, and the equality will be preserved.

The while-loop preserves the property. The while-loop beginning on line 29 also preserves this property. The argument is similar to the above argument. Let there be an iteration of this while-loop corresponding to some assignment $(0, \aleph, \beta, q_1)$ from U^{ϕ} and assignments $(\tau, \aleph, b, q_2)$ and (τ', a, β, q_3) in U^{π}. Consider some equality in property 1 corresponding to some supply node a'. Note that if $a' \neq \aleph$ and $a' \neq a$ then none of the terms of the equality will be altered by the for-loop. Consider the case that $a' = \aleph$. In line 32, the value of the left-hand side decreases by q, while in line 33, the value of the right-hand side decreases by q. No other lines alter any terms of this equality. Since the expressions on both side of the equality decrease by q, then the equality is preserved. Similarly, if $a' = a$, then the value of the left-hand decreases by q in line 31, while the value of the right-hand side decreases by q in

line 34. Thus, this equality is preserved as well.

The transition to the next time period preserves the property. Finally, we assume that the statement holds at the end of execution of the algorithm in some time period, and we show that it is preserved by the transition to the next time period. We first prove this for equation 1. Let X_0^π and X_0^ϕ be the respective values of X^π and X^ϕ at the end of execution of the algorithm in the current time period t, and let X_1^π and X_1^ϕ be the respective values of X^π and X^ϕ at beginning of the main for-loop in the next time period $t+1$. Note that by definition, $x(t, a, b; \phi; \omega) = X_0^\phi[a, b]$ and $x(t, a, b; \pi; \omega) = X_0^\pi[a, b]$ at the end of the algorithm for the time period t. At the beginning of the main for-loop in the next time period, $X_1^\pi[a, b] = 0$ for any a and b. The lists U^π and U^ϕ will not change during the transition. Then, using these

observations and the induction hypothesis,

$$
\begin{aligned}
&Q(U^{\phi}(a)) + s(t+1, a; \phi; \omega) - \sum_{b} X_1^{\phi}[a, b] \\
&\quad = Q(U^{\phi}(a)) + s(t+1, a; \phi; \omega) \\
&\quad = Q(U^{\phi}(a)) + s^{\text{new}}(t+1, a; \omega) + s(t, a, \phi; \omega) - \sum_{b} x(t, a, b; \phi; \omega) \\
&\quad = Q(U^{\phi}(a)) + s^{\text{new}}(t+1, a; \omega) + s(t, a, \phi; \omega) - \sum_{b} X_0^{\phi}[a, b] \\
&\quad = Q(U^{\pi}(a)) + s^{\text{new}}(t+1, a; \omega) + s(t, a, \pi; \omega) - \sum_{b} X_0^{\pi}[a, b] \\
&\quad = Q(U^{\pi}(a)) + s^{\text{new}}(t+1, a; \omega) + s(t, a, \pi; \omega) - \sum_{b} x(t, a, b; \pi; \omega) \\
&\quad = Q(U^{\pi}(a)) + s(t+1, a, \pi; \omega) \\
&\quad = Q(U^{\pi}(a)) + s(t+1, a, \pi; \omega) - \sum_{b} X_0^{\pi}[a, b].
\end{aligned}
$$

This completes the proof. □

Next, we show that every unmatched assignment made by π involves at least one node of the Monge star.

Lemma 20. *At any point in the execution of Algorithm 5 during any time period,* $Q(U^{\pi}(a, b)) = 0$ *for any arc* (a, b) *such that* $a \neq \aleph$ *and* $b \notin B$.

Proof. Proof of Lema 20. This holds true at the beginning of the first time period because U^{π} is initially empty. The only place in which assignments are added to U^{π} is on line 26 in the for-loop beginning on line 6. If this line is executed in some iteration of the for-loop corresponding to some arc (a, b), then an assignment of the form (t, a, b, q_1) is added to U^{π}. Thus, we need to show that line 26 will not

be executed in iterations of the loop corresponding to arcs (a, b) where $a \neq \aleph$ and $b \notin B$. This line only executes if

$$r > q'$$

where r is defined on line 17, and q' is defined on line 19. Note that when $a \neq \aleph$ and $b \notin B$, then the if-statement on line 7 is not executed, since every unmatched assignment in U^ϕ involves an arc of the Monge star. Thus, when q' is calculated, we can see that

$$X^\pi[a, b] = 0$$

and

$$r = x(t, a, b; \pi; \omega).$$

Then,

$$q' = \min \left\{ x(t, a, b; \pi; \omega), s(t, a; \phi, \omega) - \sum_b X^\phi[a, b], d(t, b; \phi, \omega) - \sum_a X^\phi[a, b] \right\}.$$

By property 1 of Lemma 19,

$$s(t, a; \phi; \omega) - \sum_{b'} X^\phi[a, b'] = s(t, a; \pi; \omega) - \sum_{b'} X^\pi[a, b'] + Q(U^\pi(a)) - Q(U^\phi(a)).$$

Note that it is always true that $X^\pi[a', b'] \leq x(t, a', b'; \pi; \omega)$ for any arc (a', b') and (as mentioned above) that $X^\pi[a, b] = 0$ when q' is calculated in this iteration. Thus,

assuming that π is well-defined,

$$\begin{aligned} s(t,a;\pi;\omega) - \sum_{b'} X^{\pi}[a,b'] &= s(t,a;\pi;\omega) - \sum_{b' \neq b} X^{\pi}[a,b'] \\ &\geq s(t,a;\pi;\omega) - \sum_{b' \neq b} x(t,a,b;\pi;\omega) \\ &\geq x(t,a,b;\pi;\omega). \end{aligned}$$

Thus,

$$s(t,a;\phi;\omega) - \sum_{b'} X^{\phi}[a,b'] \geq x(t,a,b;\pi;\omega) + Q(U_0^{\pi}(a)) - Q(U^{\phi}(a)).$$

Now, since $a \neq \aleph$, $Q(U^{\phi}(a)) = 0$, so

$$\begin{aligned} s(t,a;\phi;\omega) - \sum_{b'} X^{\phi}[a,b'] &\geq x(t,a,b;\pi;\omega) + Q(U_0^{\pi}(a)) \\ &\geq x(t,a,b;\pi;\omega). \end{aligned}$$

By a nearly identical argument, using property 2 from Lemma 19,

$$d(t,b;\phi;\omega) - \sum_{a'} X^{\phi}[a',b] \geq x(t,a,b;\pi;\omega).$$

Then, using the definition of q',

$$\begin{aligned} q' &\geq x(t,a,b;\pi;\omega) \\ &= r. \end{aligned}$$

This implies that line 26 will not be executed. Thus, no assignment of the form (t,a,b,q) for $a \neq \aleph$ and $b \notin B$ will ever be added to U^{π}. □

A.4.4 Well-definedness of ϕ.

Using the properties developed in Subsection A.4.3, we can show that the policy ϕ is well-defined.

Theorem 22. *Let ϕ be the policy defined by Algorithm 5. If π is a well-defined policy, then for any time period t,*

$$\sum_b x(t,a,b;\phi;\omega) \le s(t,a;\phi;\omega) \qquad \text{for all } a \in \mathscr{A};$$

$$\sum_a x(t,a,b;\phi;\omega) \le d(t,a;\phi;\omega) \qquad \text{for all } b \in \mathscr{B}.$$

Proof. Proof of Theorem 22. We show this for property for a supply node a; the proof is nearly identical for a demand node b. Note that the variables $X^\phi[a,b]$ are defined to be zero at the beginning of the execution of the algorithm in time period t, that these variables' values never decrease, and that $x(t,a,b;\phi;\omega)$ is assigned to $X^\phi[a,b]$ at the end of the algorithm. Thus, proving that

$$s(t,a;\phi;\omega) - \sum_b X^\phi[a,b] \ge 0 \qquad \text{for all } a \in \mathscr{A},$$

$$d(t,b;\pi;\omega) - \sum_b X^\phi[a,b] \ge 0 \qquad \text{for all } b \in \mathscr{B}$$

at all points in the algorithm is sufficient to prove this theorem. This is clearly true at the beginning of the execution of the algorithm in time period 1, because the assignment of ϕ in the first time period is defined to be feasible and the variables $X^\phi[a,b] = 0$ for any arc (a,b) at that point in execution.

The main for-loop preserves the property. We will show that the main for-loop preserves this property. Assume that the property holds at the beginning of

some iteration of the loop, and let (a, b) be the arc corresponding to this iteration. Let X_0^ϕ denote the value of X^ϕ at the beginning of this for-loop and let X_1^ϕ denote the value of X^ϕ at the end of this for-loop. Note that the first subsection of the for-loop does not alter the values of X^ϕ, so the property is preserved by the first subsection of the for-loop. In the second section of the for-loop,

$$\sum_{b'} X_1^\phi[a', b'] = \begin{cases} \sum_{b'} X_0^\phi[a', b'] & \text{if } a' \neq a, \\ q' + \sum_{b'} X_0^\phi[a', b'] & \text{if } a' = a. \end{cases}$$

where q' is defined on line 19. In the case that $a' \neq a$, then applying an induction argument,

$$\begin{aligned} s(t, a', b; \phi; \omega) - \sum_{b'} X_1^\phi[a', b'] &= s(t, a'; \phi; \omega) - \sum_{b'}, X_0^\phi[a', b'] \\ &\geq 0. \end{aligned}$$

Consider the case that $a' = a$. Note that by definition, the value of r calculated on line 17 is non-negative. By the induction hypothesis,

$$\left(s(t, a; \phi; \omega) - \sum_{b'} X_0^\phi[a, b'] \right) \geq 0$$

and

$$\left(d(t, b; \phi; \omega) - \sum_{a'} X_0^\phi[a', b] \right) \geq 0.$$

Thus,

$$q = \min \left\{ r, s(t, a; \phi; \omega) - \sum_{b'} X_0^\phi[a, b'], d(t, b; \phi; \omega) - \sum_{a'} X_0^\phi[a', b] \right\}.$$

Now,

$$\sum_{b'} X_1^{\phi}[a, b'] = q' + \sum_{b'} X_0^{\phi}[a, b'].$$

By the definition of q',

$$\begin{aligned}\sum_{b'} X_1^{\phi}[a, b'] &\leq \left(s(t, a; \phi; \omega) - \sum_{b'} X_0^{\phi}[a, b'] \right) + \sum_{b'} X_0^{\phi}[a, b'] \\ &= s(t, a; \phi; \omega).\end{aligned}$$

Thus, the main for-loop will preserve the property.

The while-loop preserves the property. Next, we will show that the while-loop also preserves this property. As before, let X_0^{ϕ} refer to the value of X^{ϕ} at the beginning of this iteration of the loop, and let X_1^{ϕ} refer to the value at the end. For all other variables, the values referred to here are the values at the beginning of the iteration immediately after q is calculated in this loop. Each iteration of this while-loop corresponds to a some assignment $(0, \aleph, \beta, q_1)$ in U^{ϕ} and some assignments $(\tau, \aleph, b, q_2)$ and (τ', a, β, q_3) in U^{π}. We observe that $a \neq \aleph$ and $b \neq \beta$ by Lemma 18. This while-loop only alters the value of $X^{\phi}[a, b]$, and leaves $X^{\phi}[a', b']$ unchanged for all other a' and b'. Thus, the inequalities in Lemma 22 will be preserved for all $a' \neq a$ and for all $b' \neq b$. Now,

$$s(t, a; \pi; \omega) - \sum_{b} X_1^{\phi}[a, b] = s(t, a; \pi; \omega) - q - \sum_{b} X_0^{\phi}[a, b].$$

Now, using property 1 from Lemma 19, we have

$$s(t,a;\phi;\omega) - \sum_b X_0^\phi[a,b] =$$

$$s(t,a;\pi;\omega) - \sum_b X^\pi[a,b] + Q(U^\pi(a)) - Q(U^\phi(a)).$$

From the definition of the algorithm $X^\pi[a,b]$ is always equal to $x(t,a,b;\pi;\omega)$ at this point in execution. Thus, assuming π is well-defined,

$$s(t,a;\pi;\omega) - \sum_b X^\pi[a,b] = s(t,a;\pi;\omega) - \sum_b x(t,a,b;\pi;\omega)$$

$$\geq 0.$$

Thus,

$$s(t,a;\phi;\omega) - \sum_b X_0^\phi[a,b] \geq Q(U^\pi(a)) - Q(U^\phi(a)).$$

Finally, note that when U^ϕ is initialized all of the assignments occur with supply node $\aleph$, and no new assignments are ever added to the list U^ϕ. As we observed above, $a \neq \aleph$, so $Q(U^\phi(a)) = 0$. Note also that U^π contains the assignment (τ', a, β, q_3), so $Q(U^\pi(a)) > q_3$ This gives us

$$s(t,a;\phi;\omega) - \sum_b X_0^\phi[a,b] \geq q_3.$$

From the definition, $q \leq q_3$, so

$$s(t,a;\phi;\omega) - \sum_b X_1^\phi[a,b] = s(t,a;\phi;\omega) - q - \sum_b X_0^\phi[a,b]$$

$$\geq s(t,a;\phi;\omega) - q_3 - \sum_b X_0^\phi[a,b]$$

$$\geq 0.$$

This completes the proof. □

A.4.5 The Algorithm Makes Matchings Correctly.

The assignments made by the policies π and ϕ correspond exactly to those stored in the lists, so that every assignment made by the policy π is recorded either in the list U^π or M^π, and every assignment made by ϕ is either in the list U^ϕ or M^ϕ.

Lemma 21. *At the beginning (and end) of execution of Algorithm 5 in time period t, the following hold:*

$$x(\tau, a, b; \phi; \omega) = Q(U^\phi(\tau, a, b)) + Q(M^\phi(\tau, a, b)),$$

$$x(\tau, a, b; \pi; \omega) = Q(U^\pi(\tau, a, b)) + Q(M^\pi(\tau, a, b))$$

for any $\tau < t$, $(a, b) \in \mathcal{F}_\tau$.

Proof. Proof of Lemma 21. First, we will show that this is true at the beginning of time period 1. Let (a, b) be any arc. From the definition of the initialization,

$$Q(U^\phi(\tau, a, b)) = r_{ab};$$

$$Q(M^\phi(\tau, a, b)) = x(\tau, a, b; \pi);$$

so,

$$\begin{aligned} Q(U^\phi(\tau, a, b)) + Q(M^\phi(\tau, a, b)) &= r_{ab} + x(\tau, a, b; \pi) \\ &= x(\tau, a, b; \phi; \omega). \end{aligned}$$

Similarly,

$$Q(U^\pi(\tau, a, b)) = 0;$$

$$Q(M^\pi(\tau, a, b)) = x(\tau, a, b; \pi);$$

so,

$$Q(U^\pi(\tau, a, b)) + Q(M^\pi(\tau, a, b)) = x(\tau, a, b; \pi).$$

Next, we show that if this property holds at the beginning of time period t, then it will still be true at the beginning of time period $t+1$. We will in fact show something slightly stronger. Specifically, at the beginning (and end) of each section of the main for-loop, and at the beginning (and end) of each iteration of the while-loop,

- For any $\tau < t$ and for any $(a, b) \in \mathcal{F}_\tau$,

$$x(\tau, a, b; \phi; \omega) = Q(U^\phi(\tau, a, b)) + Q(M^\phi(\tau, a, b));$$

- For any $(a, b) \in \mathcal{F}_t$,

$$X^\phi[a, b] = Q(U^\phi(t, a, b)) + Q(M^\phi(t, a, b));$$

- For any $\tau < t$ and for any $(a, b) \in \mathcal{F}_\tau$,

$$x(\tau, a, b; \pi; \omega) = Q(U^\pi(\tau, a, b)) + Q(M^\pi(\tau, a, b));$$

- For any $(a, b) \in \mathcal{F}_t$,

$$X^\pi[a, b] = Q(U^\pi(\tau, a, b)) + Q(M^\pi(\tau, a, b)).$$

This implies the desired result because at the end of execution of the algorithm in time period t, $X^{\phi}[a,b] = x(t,a,b;\phi;\omega)$ and $X^{\pi}[a,b] = x(t,a,b;\pi;\omega)$.

We show that the main for-loop will preserve these properties. Consider an iteration of this for-loop corresponding to some arc (a,b). Consider the first subsection of the for-loop. In the case where the if-statement on line 7 is executed, then $Q(U^{\phi}(0,a,b))$ will decrease by q, $Q(M^{\phi}(0,a,b))$ will increase by q, $Q(M^{\pi}(t,a,b))$ will increase by q, and $X^{\pi}[a,b]$ will increase by q. None of the other terms of the above equalities will be modified. From this observation, it is easy to verify that the equalities will be maintained. If the else-statement is executed instead, then none of the terms in the above equalities will be modified. In the second subsection of the for-loop, then $X^{\pi}[a,b]$ will increase by $x(t,a,b;\pi;\omega) - q$, $X^{\phi}[a,b]$ will increase by q', $Q(M^{\phi}(t,a,b))$ will increase by q', $Q(M^{\pi}(t,a,b))$ will increase by q', and U^{π} will increase by $x(t,a,b;\pi;\omega) - q - q'$, and no other terms will change. Again, it can be verified that the equalities will be preserved. Finally, let there be some iteration of the while-loop corresponding to an assignment $(0,\aleph,\beta,q_1)$ in U^{ϕ} and assignments

$(\tau, \aleph, b, q_2)$ and (τ', a, β, q_3) in U^π. Then the following changes occur:

$$X^\phi[a, b] \text{ increases by } q,$$

$$Q(U^\phi(0, \aleph, \beta)) \text{ decreases by } q,$$

$$Q(U^\pi(\tau, \aleph, b)) \text{ decreases by } q,$$

$$Q(U^\pi(\tau', a, \beta)) \text{ decreases by } q,$$

$$Q(M^\phi(0, \aleph, \beta)) \text{ increases by } q,$$

$$Q(M^\phi(t, a, b)) \text{ increases by } q,$$

$$Q(M^\pi(\tau, \aleph, b)) \text{ increases by } q,$$

$$Q(M^\pi(\tau', a, \beta)) \text{ increases by } q,$$

and again it is straightforward to verify that the equalities are preserved. □

We can also show that the costs present in the list M^ϕ are always less than the costs present in the list M^π.

Lemma 22. *At the beginning (and end) of each section of the main for-loop and the while-loop of Algorithm 5,*

$$\sum_{(\tau,a,b,q)\in M^\pi} qc(\tau, a, b) \geq \sum_{(\tau,a,b,q)\in M^\phi} qc(\tau, a, b).$$

Proof. Proof of Lemma 22. This is true at the beginning of the first time period because the sets M^π and M^ϕ are set equal in the initialization. Consider some iteration of the main for-loop corresponding to some arc (a, b). If the if-statement is executed in the first subsection, then an assignment of the form $(0, \aleph, \beta, q)$ is added to M^ϕ while one of the form $(\tau, \aleph, \beta, q)$ is added to M^π. From the definition of

Monge star, $c(0, \aleph, \beta) \leq c(\tau, \aleph, \beta)$ for any τ, so these additions will preserve this property. If the else-statement is executed instead, then sets will not be modified. The second subsection of the main for-loop preserves this property because the exact same assignment is added to both of the lists M^π and M^ϕ. The while-loop adds two assignments $(0, \aleph, \beta, q)$ and (t, a, b, q) to the set M^ϕ and adds two assignments $(\tau, \aleph, b, q)$ and (τ', a, β, q) to the list M^π. From the definition of the while-loop,

$$c(0, \aleph, \beta, q) + c(t, a, b) \leq c(\tau, \aleph, b) + c(\tau', a, \beta),$$

so the total cost added to M^ϕ is at most that added to M^π. Thus, this while-loop also preserves this property. □

Since the assignments made by the policies correspond to assignments placed in the lists maintained by the algorithm, then the costs incurred by the policy can be calculated by adding the costs in the lists.

Lemma 23. *At the end of execution of Algorithm 5 in the final time period T, the following hold:*

$$\boldsymbol{C}(\phi; \omega) = \sum_{(t,a,b,q) \in U^\phi} qc(t, a, b) + \sum_{(t,a,b,q) \in M^\phi} qc(t, a, b),$$

$$\boldsymbol{C}(\pi; \omega) = \sum_{(t,a,b,q) \in U^\pi} qc(t, a, b) + \sum_{(t,a,b,q) \in M^\pi} qc(t, a, b).$$

A.4.6 The Algorithm Matches All Assignments.

Now we can show that if π is feasible at the end of the problem, then the list U^π will be empty at the end of problem.

Lemma 24. *If π is feasible for some instance ω, then the list U^{ϕ} contains no assignments at the end of execution of Algorithm 5 in the final time period T in the instance ω.*

Proof. Proof of Lemma 24. We prove this by contradiction. Suppose π is feasible, and suppose that there exists some unmatched assignment at the end of execution. Let $s(T+1, \aleph; \phi; \omega)$ and $s(T+1, \aleph; \pi; \omega)$ be the amount of excess supply at node $\aleph$ left by the policies ϕ and π respectively at the end of the problem. Since π is feasible in instance ω, $s(T+1, \aleph; \pi; \omega) = 0$. It is implied by Lemma 19 that at the end of execution,

$$s(T+1, \aleph; \phi; \omega) = Q(U^{\pi}(\aleph)) - Q(U^{\phi}(\aleph)).$$

By assumption, U^{ϕ} has some unmatched assignment $A_1 = (0, \aleph, \beta, q_1)$, so $Q(U^{\phi}(\aleph)) > 0$. From Lemma 22,

$$s(T+1, a; \phi; \omega) \geq 0,$$

so it must be true that $Q(U^{\pi}(\aleph)) > 0$. So, there exists some unmatched assignment $A_2 = (\tau, \aleph, b, q_2)$ in U^{π}. From Lemma 18, it must be true that $b \neq \beta$. Applying a similar argument to the demand node β, there must exist some unmatched assignment $A_3 = (\tau_2, a, \beta, q_3)$ in U^{π} where $a \neq \aleph$. Then, by the Monge star condition, there would exist some $\tau^* \geq \tau, \tau'$ such that $(a, b) \in \mathscr{F}_t$ and $c(0, \aleph, \beta) + c(\tau^*, a, b) \leq c(\tau, \aleph, b) + c(\tau', a, \beta)$. Thus, the while-loop on line 29 would have identified a swap involving the assignments A_1, A_2 and A_3 in the time period τ^*. At least one of these assignments would have been removed. Therefore, these three assignments cannot

all exist. This is a contradiction, so if π is feasible, then U^ϕ must have no unmatched assignments at the end of execution of the algorithm in the final time period. □

We can use this fact to show that U^π is also empty at the end of the problem.

Lemma 25. *If π is feasible for some instance ω, then the set U^π contains no assignments at the end of execution of Algorithm 5 in the final time period T under the instance ω.*

Proof. Proof of Lemma 25. As before, let $s(T+1, \aleph; \phi; \omega)$ and $s(T+1, \aleph; \pi; \omega)$ be the amount of excess supply at node $\aleph$ left by the policies ϕ and π at the end of the problem. Since π is feasible, $s(T+1, \aleph; \pi; \omega) = 0$. It is implied by Lemma 19 that at the end of execution,

$$s(T+1, \aleph; \phi; \omega) = Q(U^\pi(\aleph)) - Q(U^\phi(\aleph)).$$

Using Lemma 24, then

$$s(T+1, \aleph; \phi; \omega) = Q(U^\pi(\aleph)).$$

Similarly,

$$\sum_{\beta \in B} s(T+1, \beta; \phi; \omega) = \sum_{b \in \beta} Q(U^\pi(\beta)).$$

By the definition of the initialization of ϕ,

$$s(1, \aleph; \phi; \omega) = \sum_{\beta \in B} s(1, \beta; \phi; \omega) = 0,$$

so by Lemma 1,

$$s(T+1, \aleph; \phi; \omega) = \sum_{\beta \in B} s(T+1, \beta; \phi; \omega) = 0.$$

Thus,

$$\sum_{b \in \beta} Q(U^{\pi}(\beta)) = Q(U^{\pi}(\aleph)) = 0.$$

Furthermore, by Lemma 20, the set U^{π} does not contain any assignments (τ, a, b, q) where $a \neq \aleph$ and $b \notin B$. Thus, it must be true that U^{π} is empty. □

A.4.7 The Policy ϕ is Dominant Over π.

Using the properties developed in the preceding sections, we can prove that if π is feasible, then ϕ is feasible and incurs a lower cost. First, we show that if π is feasible then ϕ is feasible.

Lemma 26. *If π is feasible for some instance ω, then ϕ is feasible for the instance ω.*

Proof. Proof of Lemma 26. In this proof we use similar notation as in the proof of Lemma 24. Let a be some supply node. If π is feasible then it is implied by Lemma 19 that

$$s(T+1, a; \phi; \omega) = Q(U^{\pi}(a)) - Q(U^{\phi}(a)).$$

Using Lemma 26 and Lemma 25, U^{ϕ} and U^{π} must be empty. Thus,

$$s(T+1, a; \phi; \omega) = 0.$$

A nearly identical argument shows that

$$d(T+1, b; \phi; \omega) = 0.$$

Thus, ϕ is feasible for the instance ω. □

Now we prove that ϕ incurs less cost than π.

Theorem 23. *If π is feasible for some instance ω, then ϕ is feasible for this instance and*

$$\mathbf{C}(\phi;\omega) \leq \mathbf{C}(\pi;\omega).$$

Proof. Proof of Theorem 23. By Lemma 26, if π is feasible then ϕ is feasible. At the end of execution in the final time period, $Q(U^{\phi}) = 0$ by Lemma 24 and $Q(U^{\pi}) = 0$ by Lemma 25. Then, using Lemma 23,

$$\mathbf{C}(\phi;\omega) = \sum_{(t,a,b,q)\in M^{\phi}} qc(t,a,b),$$

$$\mathbf{C}(\pi;\omega) = \sum_{(t,a,b,q)\in M^{\pi}} qc(t,a,b),$$

and by Lemma 22,

$$\mathbf{C}(\phi;\omega) \leq \mathbf{C}(\pi;\omega).$$

□

This completes the proof that when the vectors of extra assignments use all of the supply and demand of the Monge star, an appropriate policy can be constructed. Finally, we extend this proof to the case where supply or demand is left in the Monge star.

We are finally ready to prove Theorem 3. The proof involves the application of Theorem 23 to an equivalent problem, which is constructed as follows. For each $\beta \in B$ such that $r_{\aleph\beta} < d(0,\beta)$, reduce the demand of β to $r_{\aleph\beta}$ and create a duplicate

node β' with demand

$$d(0, \beta') = d(0, \beta) - r_{\aleph\beta}.$$

Let the node β' otherwise behave identically to β. That is, for some supply node a, the arc (a, β') is feasible in time period t if and only if (a, β) is feasible in the original problem, and $c(t, a, \beta')$ is identical to $c(t, a, \beta)$. Since the problem is time-partitioned, neither β nor β' will receive additional demand in later time periods. Similarly, if

$$s(0, \aleph) < \sum_{\beta \in B} r_{\aleph\beta},$$

then reduce the supply at the node $\aleph$ to

$$\sum_{\beta \in B} r_{\aleph\beta}$$

and create a duplicate node $\aleph'$ with supply

$$s(0, \aleph') = s(0, \aleph) - \sum_{\beta \in B} r_{\aleph\beta}.$$

Similarly to before, the feasible arcs from $\aleph'$ are defined to correspond to the feasible arcs of $\aleph$. This includes those arcs between $\aleph$ and the newly created duplicate nodes β'. The costs of the arcs from $\aleph'$ are again defined to be the same as those from $\aleph$. It is straightforward to verify that the new problem is equivalent to the original problem, and the pair $(\aleph, B)$ (where B excludes the newly created duplicate nodes) is still a Monge star in the new problem. Thus, Theorem 23 may be applied to this new problem to guarantee the existence of a policy ϕ with the properties given in Theorem 23. Since the problems are equivalent, then this also provides a policy for the original problem.

Appendix B: Proofs for Chapter 3

This appendix contains the proofs of results in Chapter 3.

B.1 Proof of Theorem 4.

First, we demonstrate that the problem can be formulated as a time-partitioned DSTP. We let the set of supply nodes that have supply at the beginning of the problem be the set $\mathscr{A}_0$ of scheduled flights at the beginning of the planning period, while the set of supply nodes that may receive supply in time t corresponds to the set $\mathscr{A}_t$ of potential pop-up flights that could be announced in time period t. The set of demand nodes of the DSTP that have demand in the initial time period then correspond to slots $\mathscr{B}_0$ that are certain to be available when the planning period begins, and the set of demand nodes receiving demand in time period t is given by the set of slots $\mathscr{B}_t$ whose existence (or non-existence) is revealed in time period t. The arrival process is defined as follows. In the initial time period, let there be a single unit of supply at each node corresponding to a flight of $\mathscr{A}_0$, and let there be a single unit of demand at each node b corresponding to a slot in $\mathscr{B}_0$. In each subsequent time period t, for each pop-flight a of $\mathscr{A}_t$ that announces its intent to fly, let a single unit of supply arrive at the corresponding node a. Let a single unit of demand arrive at

each node corresponding to slot b of $\mathscr{B}_t$ whose existence is revealed in time period t.

This ground holding problem allows tentative assignments. Such assignments do not occur in the DSTP framework. In a DSTP, once some supply has been matched with some demand then both the supply and demand are removed from the problem. However, for the ground holding problem, the objective function depends on tentative assignments only if they become realized assignments. That is, delay is determined by the slot that the flight is assigned to when it departs, and any previous tentative slot assignments do not matter. Because of this, it is not necessary to model the tentative assignments. In order to formulate the ground holding problem as a DSTP, we let an assignment between a supply node and a demand node correspond to the departure of a flight for a slot. A flight a can depart in time period τ for a slot b only if this departure time would allow the flight to arrive at the slot time. Thus, for a flight a to depart in time t for a slot b it must be true that

$$t + \tau^{\text{fly}}(a) = \tau^{\text{slot}}(b).$$

This is a stronger requirement than the constraint in the ground holding program that ensures each flight can use its assigned slot. We must also enforce the requirement that ensures flights are not given slot times earlier than their arrival times. We implement both of these constraints through the definition of the feasible arcs, which we define to be:

$$\mathscr{F}_t = \{(a, b) : t + \tau^{\text{fly}}(a) = \tau^{\text{slot}}(b) \text{ and } \tau^{\text{OTA}}(a) \leq \tau^{\text{slot}}(b)\}.$$

The remaining constraints that each flight is assigned to at most one slot and each slot is assigned to at most flight are implied by the constraints of the DSTP ensuring that the amount assigned on an arc does not exceed the available supply and demand. The cost $c(t, a, b)$ is again given by the incurred ground delay

$$\tau^{\text{slot}}(b) - \tau^{\text{OTA}}(a).$$

The DSTP as defined requires that all the supplies and demands are matched. In the ground holding problem, it is acceptable to have slots that are not assigned to any flight. This can be accommodated within the the DSTP framework as follows. Let there be an extra time period $T + 1$, and let there be a single supply node that can receive supply in this time period. Define the arrival process so that the amount of arriving supply is equal to the excess number of slots. The extra node represents the option of leaving slots open, and any slot that is not assigned to a flight will be assigned to this extra supply node.

Next, we show that the policy is equivalent to a subset greedy policy π. In particular, the corresponding subset U_t of the policy π consists of the entire set of feasible arcs, and these arcs are ordered by increasing slot time and decreasing flight time.

Definition 36. *An ordering $\prec^{RBD}$ on the arcs of the dynamic ground-holding problem is a ration-by-distance ordering if the following is true:*

- $(a, b) \prec^{RBD} (a', b')$ *whenever* $\tau^{slot}(b) < \tau^{slot}(b')$*;*
- $(a, b) \prec^{RBD} (a', b')$ *whenever* $\tau^{slot}(b) = \tau^{slot}(b')$ *and* $\tau^{fly}(a) > \tau^{fly}(a')$.

Note that there is not a unique $\prec^{\text{RBD}}$ ordering, since there may be multiple slots with the same slot time and there may be multiple flights with the same flight time. Any tie-breaking rule may be used and this result holds as long as this tie-breaking rule is compatible with the tie-breaking rules that the D-RBD policy uses when it finds the minimal slot and when it finds the minimal flight. In the results in the rest of this section, we will assume that the these same tie-breaking rules are used.

Lemma 27. *The D-RBD policy is equivalent to a subset greedy policy where the subset U_t is chosen to be the set $\mathscr{F}_t$ of all feasible arcs in time period t and where the ordering $\prec^{RBD}$ is used in every time period.*

Proof. Proof of Lemma 27. We will prove this by contradiction. Let $\pi^{\text{D-RBD}}$ refer to the D-RBD policy, and let π^G refer to the greedy policy described above. Suppose that in some instance ω, the D-RBD policy assigns a different quantity than the subset greedy policy along an arc (a, b) in some time period t. That is,

$$x\left(t, a, b; \pi^{\text{D-RBD}}; \omega\right) \neq x\left(t, a, b; \pi^{G}; \omega\right)$$

for some time period t, flight a and slot b. Let τ be the earliest time period at which this occurs, and let (a^*, b^*) be the minimum arc in the ordering $\prec^{\text{RBD}}$ at which this occurs.

Consider the case that $x(\tau, a^*, b^*; \pi^{\text{D-RBD}}; \omega) = 1$ and $x(\tau, a^*, b^*; \pi^{G}; \omega) = 0$. This would imply that under the policy $\pi^{\text{D-RBD}}$, both the flight a^* and the slot b^* were unassigned at the beginning of the time period τ in instance ω. The time period τ is the earliest time period in which π^G makes different assignments from

$\pi^{\text{D-RBD}}$, so this flight and slot must also be unassigned at the beginning of time period τ under the policy π^G. The arc (a^*, b^*) is included in the subset of arcs that π^G makes assignments on in time period t. Since π^G does not make the assignment (a^*, b^*) in this time period, it must instead make an assignment on some arc (a', b') such that $(a', b') \prec^{\text{RBD}} (a^*, b^*)$ and such that either $a' = a^*$ or $b' = b^*$. Since $\pi^{\text{D-RBD}}$ makes the assignment (a^*, b^*) it cannot make the assignment (a', b'), so

$$x\left(\tau, a^*, b^*; \pi^{\text{D-RBD}}; \omega\right) \neq x\left(\tau, a', b'; \pi^G; \omega\right).$$

However, since $(a^*, b^*) \prec^{\text{RBD}} (a', b')$ then the arc (a', b') is not the minimal arc under which the policies differ. This is a contradiction.

Consider the alternate case that $x\left(\tau, a^*, b^*; \pi^{\text{D-RBD}}\right) = 0$ and $x\left(\tau, a', b'; \pi^G; \omega\right) = 1$. As before, since the policies do not differ before τ, the flight a^* and the slot b^* must still be unassigned by policy $\pi^{\text{D-RBD}}$ at the beginning of this time period. The assignment of a^* to b^* is clearly feasible since π^G makes this assignment. Thus, if $\pi^{\text{D-RBD}}$ makes a tentative assignment between a^* and b^* in the time period τ, this assignment must be realized in the time period τ. By assumption, this assignment is not made in time τ, so $\pi^{\text{D-RBD}}$ does not tentatively pair flight a^* with slot b^*.

When the $\pi^{\text{D-RBD}}$ searches for a flight to match with slot b^*, either it chooses another flight a' over a^*, or the flight a^* is already unavailable. If flight a' is chosen over flight a^*, then the flight time of a' must be no greater than that of a^*, and if they are equal, then the flight a' precedes a^* under the tie-breaking rule. If the flight a^* is unavailable, then the flight a^* must have been matched with a slot b'. Since slot b' was paired before b^*, then either b' has an earlier slot-time than b^* or b' precedes

b^* under the tie-breaking rule. In any case, the policy $\pi^{\text{D-RBD}}$ makes a tentative (not necessarily realized) assignment (a', b') where $(a', b') \prec^{\text{RBD}} (a^*, b^*)$ and $a' = a^*$ or $b' = b^*$. There are two cases: this assignment is realized in the time period τ, or this assignment is a tentative assignment that is not realized in time period τ. The former case is similar to the argument above; we reach a contradiction because

$$x\left(\tau, a', b'; \pi^{\text{D-RBD}}; \omega\right) \neq x\left(\tau, a^*, b^*; \pi^G; \omega\right)$$

and $(a', b') \prec^{\text{RBD}} (a^*, b^*)$ so (a^*, b^*) is not minimal.

Consider the case where the assignment is a tentative assignment. There are three sub-cases. In the first sub-case, the slot time of b' is strictly less than that of b^*. Then $b' \neq b^*$, so it must be true that $a' = a^*$. Since the assignment of a^* to b^* is feasible in time period τ,

$$\begin{aligned} \tau + \tau^{\text{fly}}(a') &= \tau^{\text{slot}}(b^*) \\ &> \tau^{\text{slot}}(b'). \end{aligned}$$

However, this indicates that the flight time of flight a' is too long, and this flight could not make the slot b'. This tentative assignment would not be allowed under the D-RBD policy. In the second sub-case, the slots b^* and b' have the exact same slot time and the flight time of a' is strictly greater than that of a^*. Then,

$$\begin{aligned} \tau^{\text{slot}}(b') &= \tau^{\text{slot}}(b^*) \\ &= \tau + \tau^{\text{fly}}(a^*) \\ &< \tau + \tau^{\text{fly}}(a'). \end{aligned}$$

Again, this tentative assignment would not be allowed under the D-RBD policy. In the final sub-case, the slots b^* and b' have the exact same slot time and the flight time of a' is equal to that of a^*. In this case,

$$\tau^{\text{slot}}(b') = \tau + \tau^{\text{fly}}(a').$$

However, this assignment would be a realized, permanent assignment under the D-RBD policy. Thus, in any case, we reach a contradiction, and these policies must be equivalent. □

Now we can make use of the results of Theorem 2 to show that the D-RBD policy is optimal. By Lemma 27, the D-RBD policy is equivalent to a subset-greedy policy π, such that the subset U_t consists of the set $\mathscr{F}_t$ of feasible arcs and a ration-by-distance ordering $\prec^{\text{RBD}}$ is used. Each arc (a, b) has at most one time at which it may be feasible, so the sets $U_0, ..., U_T$ are disjoint. This is true because if

$$t + \tau^{\text{fly}}(a) = \tau^{\text{fly}}(b)$$

for some time period t, flight a and slot b, then

$$t' + \tau^{\text{fly}}(a) \neq \tau^{\text{fly}}(b)$$

for any $t' \neq t$. By definition, each feasible arc must be included in at least one of the subsets, so these do form a partition of the set of all feasible arcs $\mathscr{F}^*$. Let $\prec^*$ be the corresponding ordering of the arcs of the deterministic problem in some instance ω. In this case, the ordering $\prec^*$ place arcs in increasing order of the time at which the arc is feasible, then by increasing slot time, and then by decreasing flight time. We

claim that $\prec^*$ forms a Monge sequence. Let there be some feasible arcs (a, b), (a, b'), (b', a) in $\mathscr{F}^*$ such that $(a, b) \prec^* (a, b')$ and such that $(a, b) \prec^* (a', b)$. First, we must show that (a', b') is feasible. Each arc (a^*, b^*) is feasible in at most one time period, specifically the time period

$$\tau^{\text{slot}}(b^*) - \tau^{\text{fly}}(a^*),$$

and this arc is feasible if and only if

$$\tau^{\text{slot}}(b^*) - \tau^{\text{fly}}(a^*) \geq \tau^{\text{new}}(a^*),$$

$$\tau^{\text{slot}}(b^*) - \tau^{\text{fly}}(a^*) \geq \tau^{\text{new}}(b^*),$$

and

$$\tau^{\text{slot}}(b^*) \geq \tau^{\text{OTA}}(a^*).$$

We claim that $\tau^{\text{fly}}(a') \leq \tau^{\text{fly}}(a)$. By definition of the ordering $\prec^*$, the arc (a, b) must be feasible in a time period t and the arc (a', b) must be feasible in a time period t' where $t' \geq t$. Thus,

$$\tau^{\text{slot}}(b) - \tau^{\text{fly}}(a') \geq \tau^{\text{slot}}(b) - \tau^{\text{fly}}(a);$$

$$\tau^{\text{slot}}(a) \geq \tau^{\text{slot}}(a').$$

By a similar argument, it must be true that

$$\tau^{\text{slot}}(b') \geq \tau^{\text{slot}}(b).$$

Thus,

$$\tau^{\text{slot}}(b') - \tau^{\text{fly}}(a') \geq \tau^{\text{slot}}(b) - \tau^{\text{fly}}(a')$$

$$\geq \tau^{\text{new}}(a').$$

and

$$\begin{aligned}\tau^{\text{slot}}(b') - \tau^{\text{fly}}(a') &\geq \tau^{\text{slot}}(b') - \tau^{\text{fly}}(a) \\ &\geq \tau^{\text{new}}(b').\end{aligned}$$

Also,

$$\begin{aligned}\tau^{\text{OTA}}(a') &\leq \tau^{\text{slot}}(b) \\ &\leq \tau^{\text{slot}}(b').\end{aligned}$$

So the arc (a', b') is feasible. Next, we need to show that

$$c^*(a, b) + c^*(a', b') \leq c^*(a, b') + c^*(a', b).$$

Since each arc is feasible in at most one period, the minimum cost on an arc is achieved in that time period in which the arc is feasible. Then, for an arc (a^*, b^*),

$$c^*(a^*, b^*) = \tau^{\text{slot}}(b^*) - \tau^{\text{OTA}}(a^*).$$

So,

$$\begin{aligned}c^*(a, b) + c^*(a', b') &= \tau^{\text{slot}}(b) - \tau^{\text{OTA}}(a) + \tau^{\text{slot}}(b') - \tau^{\text{OTA}}(a') \\ &= \tau^{\text{slot}}(b) - \tau^{\text{OTA}}(a') + \tau^{\text{slot}}(b') - \tau^{\text{OTA}}(a) \\ &= c^*(a', b) + c^*(a, b').\end{aligned}$$

This proves that the ordering $\prec^*$ produces a Monge sequence. Then, Theorem 2 proves that the D-RBD policy is optimal.

B.2 Proof of Theorem 5.

In this section, we prove Theorem 5.

Proof. Proof of Theorem 5. We assume without loss of generality that the current time period is time period 0. This can be accomplished by relabeling the arrival times of the drivers and passengers. We prove that $(\aleph, B)$ is a Monge star. Note that for any arc (a, b), the cost $c(t, a, b)$ increases with t. This implies that $(\aleph, \beta) \in \mathscr{M}_0$ for all $\beta \in B$. Let there be some $\beta \in B$. Let there be some passenger $a \in \mathscr{A}$ and some driver $b \in \mathscr{B}$. Let there be some τ such that $\tau > 0$ and $(\aleph, b) \in \mathscr{F}_\tau$ (in other words, $\tau \geq t^{\text{arr}}(b)$) and let there be some τ' such that $\tau' > 0$ and $(a, \beta) \in \mathscr{F}_{\tau'}$ (i.e., $\tau' \geq t^{\text{arr}}(a)$). For simplicity, we will assume that $\tau \geq \tau'$. The case where $\tau' < \tau$ is very similar. Note that assumptions 2 and 3 are not used in this proof; these would be used in the case $\tau' < \tau$.

Observe that

$$t^{\text{arr}}(\aleph) \leq t^{\text{arr}}(a).$$

If the passenger a has made a request by time period 0, then by assumption 1, a must have an arrival time no earlier than $\aleph$. If a instead makes his or her request in a later time period, then his or her arrival time is later than $\aleph$ because $\aleph$ makes her or his request no later than time period 0. Now,

$$\begin{aligned} c(0, \aleph, \beta) + c(\tau, a, b) = \delta(\aleph, \beta) + w^p\left(-t^{\text{arr}}(\aleph)\right) + w^d\left(-t^{\text{arr}}(\beta)\right) \\ + \delta\left(a, b\right) + w^p\left(\tau - t^{\text{arr}}(a)\right) + w^d\left(\tau - t^{\text{arr}}(b)\right). \end{aligned}$$

By the triangle inequality,

$$\delta(a,b) \leq \delta(a,\beta) + \delta(\beta,\aleph) + \delta\left(\aleph,b\right).$$

Thus,

$$\begin{aligned} c(0,\aleph,\beta) + c(\tau,a,b) \leq\ & 2\delta(\aleph,\beta) + \delta(a,\beta) + w^p\left(-t^{\text{arr}}(\aleph)\right) + w^d\left(-t^{\text{arr}}(\beta)\right) \\ & + \delta(\aleph,b) + w^p\left(\tau - t^{\text{arr}}(a)\right) + w^d\left(\tau - t^{\text{arr}}(b)\right). \end{aligned}$$

We now introduce several terms whose values are zero,

$$\begin{aligned} c(0,\aleph,\beta) + c(\tau,a,b) \leq\ & 2\delta(\aleph,\beta) + \delta(a,\beta) + w^p\left(-t^{\text{arr}}(\aleph)\right) + w^d\left(-t^{\text{arr}}(\beta)\right) \\ & + \delta\left(\aleph,b\right) + w^p\left(\tau - t^{\text{arr}}(a)\right) + w^d\left(\tau - t^{\text{arr}}(b)\right) \\ & + w^p(\tau' - t^{\text{arr}}(a)) - w^p(\tau' - t^{\text{arr}}(a)) \\ & + w^d(\tau' - t^{\text{arr}}(\beta)) - w^d(\tau' - t^{\text{arr}}(\beta)) \\ & + w^p(\tau - t^{\text{arr}}(\aleph)) - w^p(\tau - t^{\text{arr}}(\aleph)). \end{aligned}$$

Regrouping the right-hand side,

$$\begin{aligned}
c(0,\aleph,\beta)+c(\tau,a,b) \leq\ & 2\delta(\aleph,\beta)-\left[w^d\left(\tau'-t^{\text{arr}}(\beta)\right)-w^d\left(-t^{\text{arr}}(\beta)\right)\right] \\
& +\left[w^p\left(\tau-t^{\text{arr}}(a)\right)-w^p\left(\tau'-t^{\text{arr}}(a)\right)\right] \\
& -\left[w^p(\tau-t^{\text{arr}}(\aleph))-w^p\left(-t^{\text{arr}}(\aleph)\right)\right] \\
& +\delta\left(a,\beta\right)+w^p(\tau'-t^{\text{arr}}(a))+w^d(\tau'-t^{\text{arr}}(\beta)) \\
& +\delta\left(\aleph,b\right)+w^p(\tau-t^{\text{arr}}(\aleph))+w^d\left(\tau-t^{\text{arr}}(b)\right) \\
=\ & 2\delta(\aleph,\beta)-\left[w^d(\tau'-t^{\text{arr}}(\beta))-w^d\left(-t^{\text{arr}}(\beta)\right)\right] \\
& +\left[w^p\left(\tau-t^{\text{arr}}(a)\right)-w^p(\tau'-t^{\text{arr}}(a)\right], \\
& -\left[w^p(\tau-t^{\text{arr}}(\aleph))-w^p\left(-t^{\text{arr}}(\aleph)\right)\right] \\
& +c(\tau',a,\beta) \\
& +c\left(\tau,\aleph,b\right).
\end{aligned}$$

Consider the term

$$\text{Term 1:}=2\delta(\aleph,\beta)-\left[w^d(\tau'-t^{\text{arr}}(\beta))-w^d\left(-t^{\text{arr}}(\beta)\right)\right].$$

Since w^d is increasing (by assumption 5) and $\tau'>0$, then

$$\text{Term 1}\leq 2\delta(\aleph,\beta)-\left[w^d(1-t^{\text{arr}}(\beta)-w^d\left(-t^{\text{arr}}(\beta)\right)\right].$$

Then, by assumption 4 of the Theorem,

$$\text{Term 1}\leq 0.$$

Also consider the term

$$\begin{aligned}
\text{Term 2}:=\ & \left[w^p\left(\tau-t^{\text{arr}}(a)\right)-w^p(\tau'-t^{\text{arr}}(a))\right] \\
& -\left[w^p(\tau-t^{\text{arr}}(\aleph))-w^p\left(-t^{\text{arr}}(\aleph)\right)\right].
\end{aligned}$$

Since w^p is increasing (assumption 5) and $\tau' > 0$, then

$$\begin{aligned}\text{Term 2} \leq & \left[w^p\left(\tau - t^{\text{arr}}(a)\right) - w^p(-t^{\text{arr}}(a))\right] \\ & - \left[w^p(\tau - t^{\text{arr}}(\aleph)) - w^p\left(-t^{\text{arr}}(\aleph)\right)\right] \\ = & \sum_{t'=0}^{\tau-1}\left[w^p\left(t' + 1 - t^{\text{arr}}(a)\right) - w^p(t' - t^{\text{arr}}(a))\right] \\ & - \sum_{t'=0}^{\tau-1}\left[w^p(t' + 1 - t^{\text{arr}}(\aleph)) - w^p\left(t' - t^{\text{arr}}(\aleph)\right)\right],\end{aligned}$$

and since w^p is marginally nondecreasing (assumption 5) and $t^{\text{arr}}(a) \geq t^{\text{arr}}(\aleph)$, we know that

$$\left[w^p\left(t' + 1 - t^{\text{arr}}(a)\right) - w^p(t' - t^{\text{arr}}(a))\right] \leq \left[w^p(t' + 1 - t^{\text{arr}}(\aleph)) - w^p\left(t' - t^{\text{arr}}(\aleph)\right)\right]$$

for any t'. Thus,

$$\text{Term 2} \leq 0.$$

Since Term 1 and Term 2 are both non-positive,

$$c(0, \aleph, \beta) + c(\tau, a, b) \leq c(\tau, \aleph, b) + c(\tau', a, \beta).$$

This completes the proof that $(\aleph, B)$ is a Monge star. Then, the result follows immediately from application of Theorem 3. □

Appendix C: Full-Dimensional Reformulations

Full-dimensional versions of the models discussed in Chapter 5 are provided here.

C.1 Mukherjee-Hansen Flow

A full-dimensional version of the MH-Flow model is given by:

$$\min \sum_{n \in \mathcal{N}} p(n) \left[c_g \left(\sum_{f \in \Phi_t^{\text{dep}}} (t(n) - \tau(f)) y^n_{t(n),f} \right) + c_a \left(\sum_{s=1}^{t(n)} \sum_{f \in \Phi_s^{\text{arr}}} y^{a(s,n)}_{s-\delta(f),f} - \sum_{s=1}^{t(n)} l_s^{a(s,n)} \right) \right]$$

$$\begin{aligned}
&\sum_{s=\tau(f)}^{t} y^{a(s,n)}_{\tau(f),f} \leq 1 && \forall f \in \Phi, t \in \{\tau(f), ..., T+1\}, n \in N_t, \\
&\sum_{s=1}^{t} l_s^{a(s,n)} \leq \sum_{s=1}^{t} \left(E_s + \sum_{f \in \Phi_s^{\text{arr}}} y^{a(s,n)}_{s-\delta(f),f} \right) && \forall f \in \Phi, t \in \{1, ..., T\}, n \in N_t, \\
&l_t^n \leq M_t^n && \forall t \in \{1, ..., T\}, n \in N_t, \\
&y^n_{t,f} \in \{0, 1\}, \\
&l_t^n \in \mathbb{Z}_0^+.
\end{aligned} \tag{C.1}$$

Note that the set of constraints C.1 can be reduced to the set

$$\sum_{s=\tau(f)}^{T+1} y_{\tau(f),f}^{a(s,n)} \leq 1 \qquad \forall f \in \Phi, n \in N_{T+1}$$

since for any f in Φ, q in $\mathscr{N}_{T+1}$, and $t < T+1$, the constraint

$$\sum_{s=\tau(f)}^{T+1} y_{\tau(f),f}^{a(s,q)} \leq 1,$$

dominates the constraint

$$\sum_{s=\tau(f)}^{t} y_{\tau(f),f}^{a(s,a(t,q))} \leq 1.$$

C.2 Mukherjee-Hansen with Diversions

A full-dimensional version of the MH-Diversion model is given by:

$$\min \sum_{n \in \mathscr{N}} p(n) \left[c_g \left(\sum_{f \in \Phi_{t(n)}^{\text{dep}}} (t(n) - \tau(f)) y_{t(n),f}^{n} \right) + c_a w_{t(n)}^{n} \right.$$
$$\left. + c_d \left(w_{t(n)-1}^{a(t(n)-1,n)} + \sum_{f \in \Phi_t^{\text{arr}}} y_{t(n)-\delta(f),f}^{a(t(n)-\delta(f),n)} - l_{t(n)}^{n} - w_{t(n)}^{n} \right) \right]$$

subject to:

$$\sum_{s=\tau(f)}^{T+1} y_{\tau(f),f}^{a(s,n)} \leq 1 \qquad \forall f \in \Phi, n \in N_t,$$

$$E_t + l_t^n + w_t^n \leq w_{t-1}^{a(t-1,n)} + \sum_{f \in \Phi_t^{\text{arr}}} y_{t-\delta(f),f}^{a(t-\delta(f),n)} \qquad \forall t \in \{1,...,T\}, n \in N_t,$$

$$l_t^n \leq M_t^n \qquad \forall t \in \{1,...,T\}, n \in N_t,$$

$$w_t^n \leq W^{\max} \qquad \forall t \in \{1,...,T\}, n \in N_t,$$

$$w_0^n = 0,$$

$$y_{t,f}^n, g_{t,f}^n \in \{0,1\},$$

$$w_t^n, l_t^n \in \mathbb{Z}_0^+.$$

C.3 Dynamic Hoffkin Flow

A full-dimensional version of the D-Hoffkin-Flow model is given by:

$$\min \sum_{n \in \mathcal{N}} p(n) \left[c_g \left(\sum_{s=1}^{t(n)} |D_{s,m}| - z_{s,m}^{a(s,n)} \right) + c_a \left(\sum_{s=1}^{t(n)} \sum_{\substack{m \in \Delta: \\ m<t}} z_{s-m,m}^{a(s-m,n)} - \sum_{s=1}^{t(n)} l_s^{a(s,n)} \right) \right]$$

subject to:

$$\sum_{s=1}^{t} z_{s,m}^{a(s,n)} \leq \sum_{s=1}^{t} |D_{s,m}| \qquad \forall m \in \Delta, t \in \{1,...,T\}, n \in N_t,$$

$$\sum_{s=1}^{t} l_s^{a(s,n)} \leq \sum_{s=1}^{t} \left(E_s + \sum_{\substack{m \in \Delta: \\ m<t}} z_{s-m,m}^{a(s-m,n)} \right) \qquad \forall t \in \{1,...,T\}, n \in \mathcal{N}_t,$$

$$l_t^n \leq M_t^n \qquad \forall t \in \{1,...,T\}, n \in \mathcal{N}_t,$$

$$w_0^n = 0,$$

$$l_t^n, z_{t,m}^n \in \mathbb{Z}_0^+.$$

C.4 Dynamic Hoffkin with Diversions

A full-dimensional version of the D-Hoffkin-Diversion model is given by:

$$\min \sum_{n \in \mathcal{N}} p(n) \left[c_g \left(\sum_{s=1}^{t(n)} |D_{s,m}| - z_{s,m}^{a(s,n)} \right) + c_a w_{t(n)}^n \right.$$
$$\left. + c_d \left(w_{t(n)-1}^{a(t(n)-1,n)} + \sum_{\substack{m \in \Delta: \\ m < t(n)}} z_{t(n)-m,m}^n - l_{t(n)}^n - w_{t(n)}^n \right) \right]$$

subject to:

$$\sum_{s=1}^{t} z_{s,m}^{a(s,n)} \leq \sum_{s=1}^{t} |D_{s,m}| \qquad \forall m \in \Delta, t \in \{1, ..., T\}, n \in N_t,$$

$$E_t + l_t^n + w_t^n \leq w_{t-1}^{a(t-1,n)} + \sum_{\substack{m \in \Delta: \\ m < t}} z_{t-m,m}^n \qquad \forall t \in \{1, ..., T\}, n \in \mathcal{N}_t,$$

$$w_t^n \leq W^{\max} \qquad \forall t \in \{1, ..., T\}, n \in N_t,$$

$$l_t^n \leq M_t^n \qquad \forall t \in \{1, ..., T\}, n \in \mathcal{N}_t,$$

$$w_0^n = 0,$$

$$w_t^n, x_{t,m}^n, z_{t,m}^n \in \mathbb{Z}_0^+.$$

Appendix D: Proofs for Chapter 5

In this appendix, we provide proofs of the results provided in Chapter 5.

D.1 Proof of Proposition 2

The proof of Proposition 2 follows. First, we prove that the mapping is well-defined in the sense that feasible solutions to the MH model are transformed to feasible solutions to the D-Hoffkin model. Next, we prove that the mapping preserves the costs of the solutions.

The mapping produces feasible solutions. Let $\boldsymbol{\lambda}$ be a feasible solution to the MH model. Then, for any node $n \in \mathcal{N}_1$ and for any duration m in Δ,

$$\begin{aligned} z^n_{1,m}(f(\boldsymbol{\lambda})) + x^n_{1,m} &= \sum_{f \in D_{1,m}} y^{q(n)}_{1,f}(\boldsymbol{\lambda}) + \left(|D_{1,m}| - \sum_{f \in D_{1,m}} y^{q(n)}_{1,f}(\boldsymbol{\lambda}) \right) \\ &= |D_{1,m}|. \end{aligned}$$

For any $t \in \{2, ..., T\}$, for any scenario node n in $\mathscr{N}_t$, and for any duration m in Δ,

$$\begin{aligned} z^n_{t,m}(f(\boldsymbol{\lambda})) + x^n_{t,m} &= \sum_{s=1}^{t} \sum_{f \in D_{s,m}} y_{t,f}^{q(a(t,n))}(\boldsymbol{\lambda}) + \sum_{\sigma=1}^{t} \left(|D_{\sigma,m}| - \sum_{s=1}^{\sigma} \sum_{f \in D_{s,m}} y_{\sigma,f}^{q(a(\sigma,n))}(\boldsymbol{\lambda}) \right) \\ &= |D_{t,m}| + \sum_{\sigma=1}^{t-1} \left(|D_{\sigma,m}| - \sum_{s=1}^{\sigma} \sum_{f \in D_{s,m}} y_{\sigma,f}^{q(a(\sigma,n))}(\boldsymbol{\lambda}) \right) \\ &= |D_{t,m}| + x_{t-1,m}^{a(t-1,n)}(\boldsymbol{\lambda}). \end{aligned}$$

For any $t \in \{1, ..., T+1\}$ and for any scenario node n in $\mathscr{N}_t$,

$$\begin{aligned} & E_t + w_{t-1}^{a(t-1,n)}(f(\boldsymbol{\lambda})) - w_t^n(f(\boldsymbol{\lambda})) + \sum_{\substack{m \in \Delta: \\ m<t}} z_{t-m,m}^{a(t-m,m)}(f(\boldsymbol{\lambda})) \\ &\qquad = E_t + w_{t-1}^{a(t-1,n)}(\boldsymbol{\lambda}) - w_t^n(\boldsymbol{\lambda}) + \sum_{\substack{m \in \Delta: \\ m<t}} \sum_{s=1}^{t-m} \sum_{f \in D_{s,m}} y_{t-m,f}^{q(a(t-m,n))}(\boldsymbol{\lambda}) \\ &\qquad = E_t + w_{t-1}^{q(a(t-1,n))}(\boldsymbol{\lambda}) - w_t^{q(n)}(\boldsymbol{\lambda}) + \sum_{f \in \Phi_t^{\text{arr}}} y_{t-d(f),f}^{q(a(t-d(f),n))}(\boldsymbol{\lambda}) \\ &\qquad \leq M_t^n. \end{aligned}$$

For each duration m in Δ, time period t in $\{1, ..., T\}$ and $n \in \mathscr{N}_t$,

$$\begin{aligned} x^n_{t,m}(f(\boldsymbol{\lambda})) &= \sum_{\sigma=1}^{t} \left(|D_{\sigma,m}| - \sum_{s=1}^{\sigma} \sum_{f \in D_{s,m}} y_{\sigma,f}^{q(a(\sigma,n))}(\boldsymbol{\lambda}) \right) \\ &= \sum_{\sigma=1}^{t} |D_{\sigma,m}| - \sum_{\sigma=1}^{t} \sum_{s=1}^{\sigma} \sum_{f \in D_{s,m}} y_{\sigma,f}^{q(a(\sigma,n))}(\boldsymbol{\lambda}) \\ &\geq \sum_{\sigma=1}^{t} |D_{\sigma,m}| - \sum_{s=1}^{t} \sum_{f \in D_{s,m}} \sum_{\sigma=s}^{T+1} y_{\sigma,f}^{q(a(\sigma,n))}(\boldsymbol{\lambda}) \\ &= \sum_{\sigma=1}^{t} |D_{\sigma,m}| - \sum_{s=1}^{t} \sum_{f \in D_{s,m}} 1 \\ &= 0. \end{aligned}$$

The remaining constraints are straightforward to verify.

The mapping preserves costs. For some scenario q, consider the sum:

$$\begin{aligned}
\sum_{t=1}^{T}\sum_{m\in\Delta} x_{t,m}^{n(t,q)}(\gamma_1(\boldsymbol{\lambda})) &= \sum_{t=1}^{T}\sum_{m\in\Delta}\left(\sum_{s=1}^{t}|D_{s,m}| - \sum_{\sigma=1}^{s}\sum_{f\in D_{\sigma,m}} y_{s,f}^{q}(\boldsymbol{\lambda})\right)\\
&= \sum_{t=1}^{T}|\Phi_t^{\text{dep}}| - \sum_{t=1}^{T}\sum_{s=1}^{t}\sum_{m\in\Delta}\sum_{\sigma=1}^{s}\sum_{f\in D_{\sigma,m}} y_{s,f}^{q}(\boldsymbol{\lambda})\\
&= \sum_{t=1}^{T}|\Phi_t^{\text{dep}}| - \sum_{t=1}^{T}\sum_{f\in\Phi_t^{\text{dep}}}\sum_{s=t}^{T} y_{t,f}^{q}(\boldsymbol{\lambda})\\
&= \sum_{t=1}^{T}\sum_{f\in\Phi_t^{\text{dep}}} 1 - \sum_{t=1}^{T}\sum_{f\in\Phi_t^{\text{dep}}}(T+1-t)y_{t,f}^{q}(\boldsymbol{\lambda})\\
&= \sum_{t=1}^{T}\sum_{f\in\Phi_t^{\text{dep}}}\sum_{s=\tau(f)}^{T+1} y_{s,f}^{q}(\boldsymbol{\lambda}) + \sum_{t=1}^{T}\sum_{f\in\Phi_t^{\text{dep}}}(t-(T+1))y_{t,f}^{q}(\boldsymbol{\lambda})\\
&= \sum_{f\in\Phi}\sum_{s=\tau(f)}^{T}\sum_{t=\tau(f)}^{T+1} y_{t,f}^{q}(\boldsymbol{\lambda}) + \sum_{f\in\Phi}\sum_{t=\tau(f)}^{T}(t-(T+1))y_{t,f}^{q}(\boldsymbol{\lambda})\\
&= \sum_{f\in\Phi}\sum_{t=\tau(f)}^{T+1}(T+1-\tau(f))y_{t,f}^{q}(\boldsymbol{\lambda}) + \sum_{f\in\Phi}\sum_{t=\tau(f)}^{T}(t-(T+1))y_{t,f}^{q}(\boldsymbol{\lambda})\\
&= \sum_{f\in\Phi}(T+1-\tau(f))y_{T+1,f}^{q}(\boldsymbol{\lambda}) + \sum_{t=\tau(f)}^{T}(t-\tau(f))y_{t,f}^{q}(\boldsymbol{\lambda})\\
&= \sum_{f\in\Phi}\sum_{t=\tau(f)}^{T+1}(t-\tau(f))y_{t,f}^{q}(\boldsymbol{\lambda}).
\end{aligned}$$

Then,

$$\begin{aligned}
C(\gamma_1(\boldsymbol{\lambda})) &= \sum_{n\in\mathscr{N}} p(n)\left[c_g\left(\sum_{m\in\Delta} x_{t(n),m}^{n}(\gamma_1(\boldsymbol{\lambda}))\right) + c_a w_{t(n)}^{n}(\gamma_1(\boldsymbol{\lambda}))\right]\\
&= \sum_{q\in\Theta} p(q)\left[c_g\left(\sum_{t=1}^{T}\sum_{m\in\Delta} x_{t,m}^{n(t,q)}(\gamma_1(\boldsymbol{\lambda}))\right) + \sum_{t=1}^{T} c_a w_t^{n(t,q)}(\gamma_1(\boldsymbol{\lambda}))\right]\\
&= \sum_{q\in\Theta} p(q)\left[c_g\left(\sum_{f\in\Phi}\sum_{t=\tau(f)}^{T+1}(t-\tau(f))y_{t,f}^{q}(\boldsymbol{\lambda})\right) + \sum_{t=1}^{T} c_a w_t^{q}(\lambda)\right].
\end{aligned}$$

D.2 Proof of Proposition 3

Here, we prove Proposition 3. First, we must show that Step 7 of Algorithm 3 is valid. Then, we must show that the mapping preserves feasibility, and finally we must show that the mapping preserves costs.

Step 7 is valid. We must show there are always at least k flights in the set $(\cup_{s=1}^{t} D_{s,m}) \setminus \left(\cup_{s=1}^{t-1} H_{s,m}^{q(n)}\right)$ whenever Step 7 is reached. Otherwise, this step would not be valid. Consider some iteration corresponding to some m, t, and some n in $\mathscr{N}_t$. By definition, the number of flights remaining in $(\cup_{s=1}^{t} D_{s,m}) \setminus \left(\cup_{s=1}^{t-1} H_{s,m}^{q(n)}\right)$ upon execution of Step 7 is given by

$$\sum_{s=1}^{t} |D_{s,m}| - \sum_{s=1}^{t-1} z_{s,m}^{a(s,n)}(\boldsymbol{\lambda}).$$

Thus, we need to show that

$$\sum_{s=1}^{t} |D_{s,m}| - \sum_{s=1}^{t-1} z_{s,m}^{a(s,n)}(\boldsymbol{\lambda}) \geq z_{t,m}^{n}(\boldsymbol{\lambda}).$$

Since $\boldsymbol{\lambda}$ is a feasible solution to the D-Hoffkin model,

$$\begin{aligned}
\sum_{s=1}^{t} z_{s,m}^{a(s,n)}(\gamma_2(\boldsymbol{\lambda})) &= \sum_{s=1}^{t} \left(D_{s,m} + z_{s-1,m}^{a(s-1,n)}(\gamma_2(\boldsymbol{\lambda})) - z_{s,m}^{a(s,n)}(\gamma_2(\boldsymbol{\lambda})) \right) \\
&= -z_{t(n),m}^{a(t(n),n)}(\gamma_2(\boldsymbol{\lambda})) + \sum_{s=1}^{t(n)} D_{s,m} \\
&\leq \sum_{s=1}^{t} D_{t,m}.
\end{aligned}$$

so this is true.

The mapping produces a feasible solution for the MH model. By definition,

$$\sum_{t=\tau(f)}^{T+1} y_{t,f}^{q}\left(\gamma_2(\boldsymbol{\lambda})\right) = 1$$

for all flights f and scenarios q in Θ, and

$$y_{t,f}^{q}\left(\gamma_2(\boldsymbol{\lambda})\right) \in \{0,1\}$$

for all time periods t in $\{1,...,T\}$, flights f in Φ, and scenarios q in Θ. In order to show that

$$y_{T+1,f}^{q}\left(\gamma_2(\boldsymbol{\lambda})\right) \in \{0,1\}$$

is sufficient to show that

$$\sum_{t=\tau(f)}^{T} Y[t,f,q] \leq 1$$

at the termination of the algorithm. We prove this by proving that if $Y[t_1,f,q] = 1$ for some time period t_1, flight f and scenario q then $Y[t_2,f,q] = 0$ for all $t_2 > t_1$. Suppose that $Y[t_1,f,q] = 1$. By construction, this value was set to one in the iteration corresponding to the flight duration $m = d(f)$, the time period $t = t_1$, and the scenario node $n = n(t_1,q)$. In this same iteration, the flight f is added to $H_{t_1,m}^{q}$. Consider some iteration corresponding to the flight duration $m = d(f)$, some time period $t = t_2 > t_1$ and the scenario node $n = n(t_2,q)$. Since $t_2 > t_1$, then the flight f already appears in the set $H_{t_1,m}^{q}$, and cannot be selected for the set $H_{t_2,m}^{q}$. Therefore, $Y[t_2,f,q] = 0$.

Next, we show that

$$E_t + w_{t-1}^q(\gamma_2(\boldsymbol{\lambda})) - w_t^q(\gamma_2(\boldsymbol{\lambda})) + \sum_{f \in \Phi_t^{\text{arr}}} y_{t-\delta(f),f}^q(\gamma_2(\boldsymbol{\lambda})) \leq M_t^q.$$

Note that

$$\Phi_t^{\text{arr}} = \bigcup_{m \in \Delta} \bigcup_{s=1}^{t-m} D_{s,m}.$$

Re-indexing the summation,

$$E_t + w_{t-1}^q(\boldsymbol{\lambda}) - w_t^q(\boldsymbol{\lambda}) + \sum_{m \in \Delta} \sum_{s=1}^{t} \sum_{f \in D_{s,m}} y_{t-m,f}^q(\gamma_2(\boldsymbol{\lambda})) \leq M_t^q.$$

Note that

$$\begin{aligned}
\sum_{s=1}^{t} \sum_{f \in D_{s,m}} y_{t-m,f}^q(\gamma_2(\boldsymbol{\lambda})) &= \sum_{s=1}^{t} \sum_{f \in D_{s,m}} Y[t-m, f, q] \\
&= \left| H_{t-m,f}^q \right| \\
&= z_{t-m,m}^{n(t-m,q)}(\boldsymbol{\lambda}).
\end{aligned}$$

Thus,

$$\begin{aligned}
&E_t + w_{t-1}^q(\gamma_2(\boldsymbol{\lambda})) - w_t^q(\gamma_2(\boldsymbol{\lambda})) + \sum_{f \in \Phi_t^{\text{arr}}} y_{t-\delta(f),f}^q(\gamma_2(\boldsymbol{\lambda})) \\
&\quad = E_t + w_{t-1}^{n(t-1,q)}(\boldsymbol{\lambda}) - w_t^{n(t,q)}(\boldsymbol{\lambda}) + \sum_{m \in \Delta} z_{t-m,m}^{n(t-m,q)}(\boldsymbol{\lambda}) \\
&\quad \leq M_t^q.
\end{aligned}$$

The proof that $\gamma_2(\boldsymbol{\lambda})$ satisfies the remaining constraints is straightforward.

The transformation preserves costs. In order to show that γ_2 preserves costs, we claim that it is sufficient to prove that $\gamma_1(\gamma_2(\boldsymbol{\lambda})) = \boldsymbol{\lambda}$. If this were the case, then

$$\begin{aligned} C(\boldsymbol{\lambda}) &= C(\gamma_1(\gamma_2(\boldsymbol{\lambda}))) \\ &= C(\gamma_2(\boldsymbol{\lambda})). \end{aligned}$$

since Proposition 2 states that the mapping γ_1 preserves costs.

For any time period t and scenario node n,

$$\begin{aligned} w_t^n(\gamma_1(\gamma_2(\boldsymbol{\lambda}))) &= w_t^{q(n)}(\gamma_2(\boldsymbol{\lambda})) \\ &= w_t^n(\boldsymbol{\lambda}). \end{aligned}$$

For any time period t, flight duration m, and scenario node n,

$$\begin{aligned} z_{t,m}^n(\gamma_1(\gamma_2(\boldsymbol{\lambda}))) &= \sum_{s=1}^{t} \sum_{f \in D_{s,m}} y_{t,f}^{q(n)}(\gamma_1(\boldsymbol{\lambda})) \\ &= \sum_{s=1}^{t} \sum_{f \in D_{s,m}} Y[t, f, q(n)] \\ &= z_{t,m}^n(\boldsymbol{\lambda}). \end{aligned}$$

From here, it is sufficient to show that if $\boldsymbol{\lambda}_1$ and $\boldsymbol{\lambda}_2$ are feasible FSFS-D solutions such that

$$z_{t,m}^n(\boldsymbol{\lambda}_1) = z_{t,m}^n(\boldsymbol{\lambda}_2)$$

for all time periods t, flight durations m and scenario nodes n in $\mathscr{N}_t$ then

$$x_{t,m}^n(\boldsymbol{\lambda}_1) = x_{t,m}^n(\boldsymbol{\lambda}_2)$$

for all time periods t, flight durations m, and scenario nodes n in $\mathscr{N}_t$. In particular, the constraints 5.11 and 5.12 imply that

$$\begin{aligned} x^n_{t,m}(\boldsymbol{\lambda}_1) &= x^n_{t,m}(\boldsymbol{\lambda}_2) \\ &= \sum_{\sigma=1}^{t} |D_{\sigma,m}| - z^{a(\sigma,n)}_{\sigma,m}. \end{aligned}$$

This completes the proof.

D.3 Proof of Proposition 4

Here, we prove Proposition 4. For any feasible FSFS-D solution $\boldsymbol{\lambda}$ in $\mathscr{P}_{DH}$, the proof that $\gamma_2^{\text{RBS}}(\boldsymbol{\lambda})$ is well-defined and feasible is similar to that of γ_2. Then all that is required is to prove that $\gamma_2^{\text{RBS}}(\boldsymbol{\lambda})$ is FSFS-D. Let there be some time period t, some flight f, and some scenario q such that

$$y^q_{t,f}(\gamma_2^{\text{RBS}}(\boldsymbol{\lambda})) = 1.$$

Let there be some time period $t' < t$ and some flight ϕ such that $\delta(\phi) = \delta(f)$ and $\tau(\phi) > \tau(f)$. Consider the iteration of Algorithm 4 corresponding to the duration $\tau(f)$, the time period t' and the scenario node $n(t', q)$. If $t' < \tau(\phi)$, then ϕ is not in the set $\cup_{s=1}^{t'} D_{s,\tau(\phi)}$. Thus, ϕ will not be in the set $H^{n(t',q)}_{t',\tau(\phi)}$. If $t' \leq \tau(\phi)$ then both the flight f and ϕ are in the set $\cup_{s=1}^{t'} D_{s,\tau(\phi)}$. However, f cannot be in the set $H^{n(s,q)}_{t',\tau(f)}$ because it is in the set $H^{n(t,q)}_{t,\tau(f)}$. Thus, f is not one of the first k flights in this iteration. However, ϕ appears in the ordering later than f because $\tau(\phi) > \tau(f)$. Thus, ϕ is also not one of the first k flights in this iteration. Again, ϕ will not be in the set $H^{n(t',q)}_{t',\tau(\phi)}$. Since in either case, ϕ is not in $H^{n(t',q)}_{t',\tau(\phi)}$, then $Y[t', \phi, q] = 0$, and

therefore

$$y^{q}_{t',\phi}(\gamma_2^{\text{RBS}}(\boldsymbol{\lambda})) = 0.$$

This proves that $\gamma_2^{\text{RBS}}(\boldsymbol{\lambda})$ is FSFS-D.

D.4 Proof of Proposition 5

Here, we prove Proposition 5. For any feasible FSFS-D solution $\boldsymbol{\lambda}$, the proof that the solution $\gamma_1^*(\boldsymbol{\lambda})$ is feasible is similar to that of γ_1. We must additionally demonstrate that $\gamma_1^*(\boldsymbol{\lambda})$ is an FSFS-D solution.

The mapping preserves the FSFS-D property. Let $\boldsymbol{\lambda}$ be a feasible FSFS-D solution to the MH model. Let there be a time period t, a scheduled departure time τ, a flight duration m, and a scenario node n of $\mathscr{N}_t$ such that

$$x^{n}_{t,\tau,m}\left(\gamma_1^*(\boldsymbol{\lambda})\right) > 0.$$

Since, by definition,

$$\begin{aligned} x^{n}_{t,\tau,m}\left(\gamma_1^*(\boldsymbol{\lambda})\right) &= |D_{\tau,m}| - \sum_{s=\tau}^{t} \sum_{f \in D_{\tau,m}} y^{q(n)}_{s,f}(\boldsymbol{\lambda}) \\ &> 0. \end{aligned}$$

then there must exist some flight ϕ such that $y^{q(a(s,n))}_{s,\phi} = 0$ for each time period s where $s \leq t$. By constraint 5.1, it must then be true that $y^{q(n)}_{\sigma,\phi} = 1$ for some $\sigma > t$. Thus, since $\boldsymbol{\lambda}$ satisfies the FSFS-D property,

$$y^{q(n)}_{s,\psi} = 0$$

for any flight ψ such that $\tau(\psi) > \tau(\psi)$ and any time period s such that $s < \sigma$. Then, for any time period τ' such that $\tau < \tau' \leq t$,

$$x^n_{t,\tau',m}\left(\gamma_1^*(\boldsymbol{\lambda})\right) = |D_{\tau',m}| - \sum_{s=\tau'}^{t} \sum_{f \in D_{\tau',m}} y^{q(n)}_{s,f}(\boldsymbol{\lambda})$$

$$= |D_{\tau',m}|.$$

This completes the proof that γ_1^* preserves the FSFS-D property.

The mapping preserves cost. Let $\boldsymbol{\lambda}$ be an feasible FSFS-D solution to the MH model. For some time period s, duration m, and scenario q, consider the sum:

$$\begin{aligned}
&\sum_{t=\tau}^{T} \mu(t-\tau) x^{n(t,q)}_{t,\tau,m}\left(\gamma_1^*(\boldsymbol{\lambda})\right) \\
&\quad = \sum_{t=\tau}^{T} \mu(t-\tau) \left(|D_{\tau,m}| - \sum_{s=\tau}^{t} \sum_{f \in D_{\tau,m}} y^q_{s,f}(\boldsymbol{\lambda}) \right) \\
&\quad = \sum_{t=\tau}^{T} \mu(t-\tau) \left(\sum_{f \in D_{\tau,m}} \left(1 - \sum_{s=\tau}^{t} y^q_{s,f}(\boldsymbol{\lambda}) \right) \right) \\
&\quad = \sum_{t=\tau}^{T} \mu(t-\tau) \left(\sum_{f \in D_{\tau,m}} \sum_{s=t+1}^{T+1} y^q_{s,f}(\boldsymbol{\lambda}) \right) \\
&\quad = \sum_{f \in D_{\tau,m}} \sum_{s=\tau+1}^{T+1} y^q_{s,f}(\boldsymbol{\lambda}) \left(\sum_{t=\tau}^{s-1} \mu(t-\tau) \right) \\
&\quad = \sum_{f \in D_{\tau,m}} \sum_{s=\tau+1}^{T+1} y^q_{s,f}(\boldsymbol{\lambda}) \left(\sum_{t=\tau}^{s-1} \Gamma(t+1-\tau) - \Gamma(t-\tau) \right) \\
&\quad = \sum_{f \in D_{\tau,m}} \sum_{s=\tau+1}^{T+1} \Gamma(s-\tau) y^q_{s,f}(\boldsymbol{\lambda}) \\
&\quad = \sum_{f \in D_{\tau,m}} \sum_{s=\tau}^{T+1} \Gamma(s-\tau) y^q_{s,f}(\boldsymbol{\lambda}).
\end{aligned}$$

Then,

$$
\begin{aligned}
C(\gamma_1^*(\boldsymbol{\lambda})) &= \min \sum_{n\in\mathscr{N}} p(n) \left[\sum_{m\in\Delta} \sum_{s=1}^{t(n)} \mu(t(n)-s) x^n_{t(n),s,m}(\gamma_1^*(\boldsymbol{\lambda})) + c_a w^n_{t(n)}(\gamma_1^*(\boldsymbol{\lambda})) \right] \\
&= \sum_{q\in\Theta} p(q) \left(\sum_{t=1}^{T} \sum_{m\in\Delta} \sum_{s=1}^{t} \mu(t-s) x^{n(t,q)}_{t,s,m}(\gamma_1^*(\boldsymbol{\lambda})) + c_a \sum_{t=1}^{T} w_t^{n(t,q)}(\gamma_1^*(\boldsymbol{\lambda})) \right) \\
&= \sum_{q\in\Theta} p(q) \left(\sum_{s=1}^{T} \sum_{m\in\Delta} \sum_{t=s}^{T} \mu(t-s) x^{n(t,q)}_{t,s,m}(\gamma_1^*(\boldsymbol{\lambda})) + c_a \sum_{t=1}^{T} w_t^{n(t,q)}(\gamma_1^*(\boldsymbol{\lambda})) \right) \\
&= \sum_{q\in\Theta} p(q) \left(\sum_{s=1}^{T} \sum_{m\in\Delta} \left(\sum_{f\in D_{s,m}} \sum_{\sigma=s}^{T+1} \Gamma(\sigma-t) y^q_{\sigma,f}(\boldsymbol{\lambda}) \right) + c_a \sum_{t=1}^{T} w_t^q(\boldsymbol{\lambda}) \right) \\
&= \sum_{q\in\Theta} p(q) \left(\sum_{\sigma=1}^{T+1} \sum_{m\in\Delta} \sum_{s=1}^{\sigma} \sum_{f\in D_{s,m}} \Gamma(\sigma-t) y^q_{\sigma,f}(\boldsymbol{\lambda}) + c_a \sum_{t=1}^{T} w_t^q(\boldsymbol{\lambda}) \right) \\
&= \sum_{q\in\Theta} p(q) \left(\sum_{\sigma=1}^{T+1} \sum_{f\in\phi_\sigma^{\text{dep}}} \Gamma(\sigma-\tau(f)) y^q_{\sigma,f}(\boldsymbol{\lambda}) + c_a \sum_{t=1}^{T} w_t^q(\boldsymbol{\lambda}) \right) \\
&= C(\boldsymbol{\lambda}).
\end{aligned}
$$

This completes the proof.

D.5 Proof of Proposition 6

Here, we prove Proposition 6. For any feasible FSFS-D solution $\boldsymbol{\lambda}$ in $\mathscr{P}^*_{GDH}$, the proof that $\gamma_2^{\text{RBS}}(\boldsymbol{\lambda})$ is well-defined, feasible and satisfies the FSFS-D property is identical to the proof for the function γ_2^{RBS} defined on $\mathscr{P}_{DH}$. We must additionally demonstrate that $\gamma_2^{\text{RBS}}(\boldsymbol{\lambda})$ preserves costs.

In order to show that this mapping preserves costs, we claim that it is sufficient

to prove that $\gamma_1^*(\gamma_2^{\text{RBS}}(\boldsymbol{\lambda})) = \boldsymbol{\lambda}$. If this were the case, then

$$\begin{aligned} C(\boldsymbol{\lambda}) &= C(\gamma_1^*(\gamma_2^{\text{RBS}}(\boldsymbol{\lambda}))) \\ &= C(\gamma_2^{\text{RBS}}(\boldsymbol{\lambda})). \end{aligned}$$

since we have already proven (Proposition 5) that the mapping γ_1^* preserves costs.

For any time period t and scenario node n,

$$\begin{aligned} w_t^n(\gamma_1^*(\gamma_2^{\text{RBS}}(\boldsymbol{\lambda}))) &= w_t^{q(n)}\left(\gamma_2^{\text{RBS}}(\boldsymbol{\lambda})\right) \\ &= w_t^n(\boldsymbol{\lambda}). \end{aligned}$$

For any time period t, flight duration m, and scenario node n,

$$\begin{aligned} z_{t,m}^n(\gamma_1^*(\gamma_2^{\text{RBS}}(\boldsymbol{\lambda}))) &= \sum_{s=1}^{t} \sum_{f \in D_{s,m}} y_{t,f}^{q(n)}(\gamma_1^*(\boldsymbol{\lambda})) \\ &= \sum_{s=1}^{t} \sum_{f \in D_{s,m}} Y[t, f, q(n)] \\ &= z_{t,m}^n(\boldsymbol{\lambda}). \end{aligned}$$

From here, it is sufficient to show that if $\boldsymbol{\lambda}_1$ and $\boldsymbol{\lambda}_2$ are feasible FSFS-D solutions such that

$$z_{t,m}^n(\boldsymbol{\lambda}_1) = z_{t,m}^n(\boldsymbol{\lambda}_2)$$

for all time periods t, flight durations m and scenario nodes n in $\mathscr{N}_t$ then

$$x_{t,s,m}^n(\boldsymbol{\lambda}_1) = x_{t,s,m}^n(\boldsymbol{\lambda}_2)$$

for all time periods t and s, flight durations m, and scenario nodes n in $\mathscr{N}_t$. The

constraints 5.16 and 5.17 imply that

$$\sum_{s=1}^{t} x^n_{t,s,m}(\boldsymbol{\lambda}_1) = \sum_{s=1}^{t} x^n_{t,s,m}(\boldsymbol{\lambda}_2)$$
$$= \sum_{\sigma=1}^{t} |D_{\sigma,m}| - z^{a(\sigma,n)}_{\sigma,m}.$$

Let $\eta(1)$ be the maximum scheduled departure time of any flight with duration m that takes delay in time period t under scenario m for the solution $\boldsymbol{\lambda}_1$,

$$\eta(1) = \max\{s \in \{1, ..., t\} : x^n_{t,s,m}(\boldsymbol{\lambda}_1) > 0\}.$$

Define $\eta(2)$ similarly for $\boldsymbol{\lambda}_2$. Then, from the definition of FSFS-D it follows that

$$x^n_{t,s,m}(\boldsymbol{\lambda}_1) = 0$$

for all $s > \eta(1)$,

$$x^n_{t,s,m}(\boldsymbol{\lambda}_1) = |D_{s,m}|$$

for all $s < \eta(1)$, and that

$$\sum_{s=1}^{t} x^n_{t,s,m}(\boldsymbol{\lambda}_1) = x^n_{t,\eta(1),m}(\boldsymbol{\lambda}_1) + \sum_{\sigma=1}^{\eta(1)-1} |D_{\sigma,m}|$$

and a similar statement applies for $\eta(2)$. We claim that $\eta(1) = \eta(2)$. Suppose this were not the case, and assume without loss of generality that $\eta(1) < \eta(2)$. Then,

$$x^n_{t,\eta(1),m}(\boldsymbol{\lambda}_1) + \sum_{\sigma=1}^{\eta(1)-1} |D_{\sigma,m}| = x^n_{t,\eta(2),m}(\boldsymbol{\lambda}_2) + \sum_{\sigma=1}^{\eta(2)-1} |D_{\sigma,m}|$$
$$x^n_{t,\eta(1),m}(\boldsymbol{\lambda}_1) = x^n_{t,\eta(2),m}(\boldsymbol{\lambda}_2) + \sum_{\sigma=\eta(1)}^{\eta(2)-1} |D_{\sigma,m}|$$
$$> |D_{\eta(1),m}|.$$

which would violate the constraint 5.18. Thus, $\eta(1) = \eta(2)$. Then, since

$$x^n_{t,\eta(1),m}(\boldsymbol{\lambda}_1) + \sum_{\sigma=1}^{\eta(1)-1} |D_{\sigma,m}| = x^n_{t,\eta(2),m}(\boldsymbol{\lambda}_2) + \sum_{\sigma=1}^{\eta(2)-1} |D_{\sigma,m}|$$

$$x^n_{t,\eta(1),m}(\boldsymbol{\lambda}_1) = x^n_{t,\eta(2),m}(\boldsymbol{\lambda}_2).$$

This completes the proof.

Appendix E: Additional Computational Results for Chapter 7

Table E.1: Quality Measures of UPR Methods, 3 Representatives, Feature Set 0

Method	Perf.	Variety	Max-Min Dist.		Sum Dist.		Sum Sq. Dist.	
KC-UPR	8.27	(4.98)	**0.87**	(0.05)	36.28	(1.80)	28.22	(2.87)
K-medoids	4.96	(3.15)	0.95	(0.05)	**34.21**	(0.82)	**25.14**	(1.29)
KS	7.24	(4.21)	0.96	(0.03)	37.49	(2.57)	30.37	(4.23)
OptiSim 5	5.60	(3.55)	0.97	(0.04)	35.65	(1.79)	27.39	(2.89)
OptiSim 10	5.72	(3.64)	0.96	(0.04)	36.06	(1.81)	28.02	(2.93)
OptiSim 25	7.59	(4.49)	0.95	(0.04)	36.34	(2.02)	28.48	(3.31)
IF	**9.22**	(4.65)	0.98	(0.03)	41.05	(2.55)	36.20	(4.41)
Random	4.27	(2.66)	0.98	(0.03)	35.73	(1.49)	27.55	(2.41)

Table E.2: Quality Measures of UPR Methods, 3 Representatives, Feature Set 1

Method	Perf.	Variety	Max-Min Dist.		Sum Dist.		Sum Sq. Dist.	
KC-UPR	8.24	(5.05)	**0.87**	(0.05)	35.95	(1.80)	27.69	(2.85)
K-medoids	4.91	(3.22)	0.95	(0.05)	**34.10**	(0.71)	**24.96**	(1.10)
KS	7.46	(4.32)	0.96	(0.04)	37.09	(2.52)	29.72	(4.14)
OptiSim 5	4.91	(2.59)	0.96	(0.04)	35.34	(1.40)	26.88	(2.20)
OptiSim 10	6.14	(3.78)	0.96	(0.04)	35.58	(1.66)	27.27	(2.65)
OptiSim 25	7.92	(4.80)	0.94	(0.04)	35.96	(1.67)	27.86	(2.67)
IF	**9.33**	(4.81)	0.98	(0.03)	40.93	(2.56)	36.01	(4.42)
Random	4.44	(2.61)	0.97	(0.04)	35.52	(1.32)	27.21	(2.15)

Table E.3: Quality Measures of UPR Methods, 3 Representatives, Feature Set 2

Method	Perf. Variety		Max-Min Dist.		Sum Dist.		Sum Sq. Dist.	
KC-UPR	8.19	(5.01)	**0.87**	(0.06)	36.10	(1.72)	27.94	(2.72)
K-medoids	4.86	(3.17)	0.95	(0.05)	**34.15**	(0.80)	**25.05**	(1.26)
KS	6.94	(3.76)	0.96	(0.04)	37.04	(2.65)	29.67	(4.38)
OptiSim 5	5.53	(3.40)	0.97	(0.04)	35.51	(1.47)	27.17	(2.36)
OptiSim 10	5.75	(3.58)	0.96	(0.04)	35.72	(1.65)	27.50	(2.68)
OptiSim 25	7.47	(4.56)	0.95	(0.05)	36.14	(1.93)	28.16	(3.14)
IF	**9.33**	(4.78)	0.98	(0.03)	40.97	(2.50)	36.07	(4.31)
Random	4.56	(2.25)	0.97	(0.03)	35.72	(1.59)	27.53	(2.57)

Table E.4: Quality Measures of UPR Methods, 3 Representatives, Feature Set 3

Method	Perf. Variety		Max-Min Dist.		Sum Dist.		Sum Sq. Dist.	
KC-UPR	8.18	(5.03)	**0.86**	(0.05)	35.78	(1.74)	27.43	(2.75)
K-medoids	5.02	(3.27)	0.94	(0.05)	**34.00**	(0.72)	**24.81**	(1.13)
KS	6.90	(3.95)	0.96	(0.04)	37.17	(2.58)	29.85	(4.24)
OptiSim 5	5.47	(3.55)	0.97	(0.03)	35.22	(1.23)	26.70	(1.95)
OptiSim 10	5.81	(3.64)	0.96	(0.04)	35.59	(1.67)	27.28	(2.69)
OptiSim 25	6.92	(4.14)	0.95	(0.04)	36.06	(1.93)	28.01	(3.13)
IF	**9.06**	(4.64)	0.98	(0.03)	40.24	(2.26)	34.81	(3.87)
Random	4.29	(2.55)	0.98	(0.03)	35.49	(1.20)	27.17	(1.93)

Table E.5: Quality Measures of UPR Methods, 3 Representatives, Feature Set 4

Method	Perf. Variety		Max-Min Dist.		Sum Dist.		Sum Sq. Dist.	
KC-UPR	8.10	(5.12)	**0.85**	(0.05)	35.61	(1.59)	27.16	(2.50)
K-medoids	4.88	(3.19)	0.94	(0.05)	**33.95**	(0.74)	**24.73**	(1.15)
KS	6.60	(3.43)	0.95	(0.04)	36.60	(2.46)	28.95	(4.05)
OptiSim 5	5.74	(4.09)	0.97	(0.04)	35.34	(1.61)	26.90	(2.58)
OptiSim 10	6.28	(3.86)	0.96	(0.04)	35.65	(1.62)	27.39	(2.61)
OptiSim 25	7.15	(4.37)	0.95	(0.04)	35.78	(1.73)	27.58	(2.82)
IF	**9.13**	(4.69)	0.98	(0.04)	40.04	(2.19)	34.47	(3.73)
Random	4.74	(3.13)	0.97	(0.03)	35.76	(1.74)	27.61	(2.84)

Table E.6: Quality Measures of UPR Methods, 3 Representatives, Feature Set 5

Method	Perf.	Variety	Max-Min	Dist.	Sum	Dist.	Sum Sq.	Dist.
KC-UPR	8.20	(5.00)	**0.85**	(0.05)	35.58	(1.81)	27.12	(2.85)
K-medoids	5.17	(3.42)	0.94	(0.05)	**33.94**	(0.68)	**24.71**	(1.04)
KS	7.27	(4.19)	0.95	(0.04)	36.77	(2.52)	29.21	(4.13)
OptiSim 5	4.97	(2.85)	0.96	(0.04)	35.35	(1.58)	26.90	(2.53)
OptiSim 10	5.86	(3.17)	0.94	(0.05)	35.23	(1.49)	26.69	(2.36)
OptiSim 25	6.89	(3.95)	0.94	(0.05)	35.74	(1.81)	27.51	(2.92)
IF	**9.07**	(4.53)	0.98	(0.04)	40.04	(2.32)	34.49	(3.95)
Random	4.23	(2.06)	0.97	(0.03)	35.30	(1.46)	26.86	(2.38)

Table E.7: Quality Measures of UPR Methods, 3 Representatives, Feature Set 6

Method	Perf.	Variety	Max-Min	Dist.	Sum	Dist.	Sum Sq.	Dist.
KC-UPR	8.15	(5.07)	**0.86**	(0.06)	35.80	(1.88)	27.48	(2.99)
K-medoids	5.08	(3.23)	0.94	(0.05)	**34.06**	(0.69)	**24.90**	(1.08)
KS	7.31	(4.26)	0.95	(0.04)	36.39	(2.35)	28.61	(3.85)
OptiSim 5	5.30	(2.94)	0.96	(0.04)	35.21	(1.46)	26.70	(2.32)
OptiSim 10	6.49	(4.02)	0.95	(0.04)	35.55	(2.01)	27.24	(3.23)
OptiSim 25	7.18	(4.46)	0.94	(0.05)	35.65	(1.69)	27.36	(2.70)
IF	**9.12**	(4.57)	0.98	(0.03)	40.52	(2.59)	35.32	(4.45)
Random	4.61	(2.39)	0.98	(0.03)	35.37	(1.55)	27.01	(2.54)

Table E.8: Quality Measures of UPR Methods, 3 Representatives, Feature Set 7

Method	Perf.	Variety	Max-Min	Dist.	Sum	Dist.	Sum Sq.	Dist.
KC-UPR	7.13	(4.52)	**0.86**	(0.06)	35.79	(1.91)	27.47	(3.04)
K-medoids	4.80	(2.67)	0.94	(0.05)	**34.01**	(0.83)	**24.83**	(1.28)
KS	7.24	(4.21)	0.96	(0.04)	37.01	(2.64)	29.61	(4.31)
OptiSim 5	5.68	(3.74)	0.96	(0.04)	35.23	(1.40)	26.73	(2.24)
OptiSim 10	5.75	(3.68)	0.96	(0.04)	35.16	(1.40)	26.60	(2.20)
OptiSim 25	7.77	(4.40)	0.95	(0.04)	35.76	(1.83)	27.56	(2.94)
IF	**8.06**	(4.31)	0.97	(0.03)	42.55	(2.49)	38.81	(4.41)
Random	4.94	(3.35)	0.98	(0.03)	35.64	(1.62)	27.43	(2.67)

Table E.9: Quality Measures of UPR Methods, 3 Representatives, Feature Set 8

Method	Perf. Variety		Max-Min Dist.		Sum Dist.		Sum Sq. Dist.	
KC-UPR	8.11	(5.10)	**0.84**	(0.05)	35.20	(1.69)	26.53	(2.64)
K-medoids	5.23	(3.53)	0.93	(0.06)	**33.82**	(0.64)	**24.53**	(0.99)
KS	7.12	(4.11)	0.95	(0.04)	35.92	(2.05)	27.85	(3.32)
OptiSim 5	5.00	(2.96)	0.95	(0.04)	34.98	(1.61)	26.35	(2.56)
OptiSim 10	5.79	(3.68)	0.95	(0.05)	34.95	(1.44)	26.29	(2.30)
OptiSim 25	7.10	(4.23)	0.93	(0.05)	35.10	(1.44)	26.48	(2.29)
IF	**8.93**	(4.61)	0.98	(0.04)	39.63	(2.30)	33.81	(3.90)
Random	4.36	(2.46)	0.97	(0.04)	35.09	(1.25)	26.53	(2.00)

Table E.10: Quality Measures of UPR Methods, 5 Representatives, Feature Set 0

Method	Perf. Variety		Max-Min Dist.		Sum Dist.		Sum Sq. Dist.	
KC-UPR	25.25	(10.15)	**0.80**	(0.04)	32.93	(1.27)	24.18	(1.91)
K-medoids	23.88	(10.60)	0.90	(0.06)	**32.13**	(0.62)	**23.08**	(0.97)
KS	**28.47**	(11.28)	0.90	(0.05)	33.99	(1.37)	25.92	(2.19)
OptiSim 5	19.39	(9.29)	0.94	(0.05)	32.90	(1.00)	24.26	(1.57)
OptiSim 10	24.66	(10.75)	0.92	(0.06)	33.32	(1.59)	24.90	(2.55)
OptiSim 25	26.71	(10.72)	0.90	(0.05)	33.54	(1.46)	25.22	(2.32)
IF	27.57	(9.28)	0.94	(0.05)	36.18	(2.30)	29.43	(3.77)
Random	16.48	(7.50)	0.97	(0.04)	33.19	(0.95)	24.75	(1.50)

Table E.11: Quality Measures of UPR Methods, 5 Representatives, Feature Set 1

Method	Perf. Variety		Max-Min Dist.		Sum Dist.		Sum Sq. Dist.	
KC-UPR	25.32	(10.34)	**0.80**	(0.04)	32.73	(1.15)	23.88	(1.72)
K-medoids	24.01	(10.49)	0.89	(0.06)	**32.05**	(0.62)	**22.96**	(0.98)
KS	28.19	(11.27)	0.90	(0.05)	33.64	(1.34)	25.38	(2.14)
OptiSim 5	20.62	(9.83)	0.93	(0.05)	32.78	(1.16)	24.08	(1.82)
OptiSim 10	22.96	(10.29)	0.91	(0.06)	32.91	(1.08)	24.25	(1.68)
OptiSim 25	28.00	(10.39)	0.89	(0.05)	33.40	(1.39)	24.99	(2.17)
IF	**28.25**	(10.08)	0.94	(0.06)	36.14	(2.24)	29.37	(3.68)
Random	15.19	(6.96)	0.96	(0.04)	33.13	(1.02)	24.67	(1.64)

Table E.12: Quality Measures of UPR Methods, 5 Representatives, Feature Set 2

.

Method	Perf. Variety		Max-Min Dist.		Sum Dist.		Sum Sq. Dist.	
KC-UPR	24.66	(10.70)	**0.80**	(0.04)	32.93	(1.19)	24.13	(1.75)
K-medoids	24.14	(10.47)	0.89	(0.06)	**32.05**	(0.62)	**22.96**	(0.97)
KS	**27.87**	(10.66)	0.90	(0.05)	33.71	(1.27)	25.50	(2.01)
OptiSim 5	19.79	(8.51)	0.93	(0.06)	32.72	(0.87)	23.98	(1.36)
OptiSim 10	23.19	(9.86)	0.91	(0.06)	33.02	(1.24)	24.43	(1.94)
OptiSim 25	26.99	(11.24)	0.89	(0.06)	33.33	(1.21)	24.88	(1.90)
IF	27.55	(9.49)	0.94	(0.06)	35.91	(2.32)	29.01	(3.81)
Random	15.78	(7.99)	0.97	(0.03)	33.08	(0.97)	24.61	(1.54)

Table E.13: Quality Measures of UPR Methods, 5 Representatives, Feature Set 3

.

Method	Perf. Variety		Max-Min Dist.		Sum Dist.		Sum Sq. Dist.	
KC-UPR	24.82	(10.39)	**0.79**	(0.04)	32.68	(1.23)	23.79	(1.81)
K-medoids	23.33	(10.31)	0.88	(0.06)	**31.98**	(0.59)	**22.85**	(0.92)
KS	**28.79**	(11.45)	0.89	(0.05)	33.67	(1.40)	25.42	(2.21)
OptiSim 5	21.04	(9.38)	0.94	(0.05)	32.73	(1.05)	24.02	(1.65)
OptiSim 10	23.61	(10.03)	0.91	(0.06)	32.87	(1.09)	24.20	(1.68)
OptiSim 25	27.00	(10.90)	0.89	(0.05)	33.20	(1.37)	24.68	(2.17)
IF	26.99	(9.46)	0.94	(0.06)	35.47	(2.17)	28.29	(3.53)
Random	16.67	(7.39)	0.97	(0.03)	33.12	(0.92)	24.65	(1.47)

Table E.14: Quality Measures of UPR Methods, 5 Representatives, Feature Set 4

.

Method	Perf. Variety		Max-Min Dist.		Sum Dist.		Sum Sq. Dist.	
KC-UPR	24.35	(10.61)	**0.79**	(0.04)	32.56	(1.13)	23.58	(1.67)
K-medoids	23.79	(10.42)	0.87	(0.07)	**31.89**	(0.61)	**22.70**	(0.95)
KS	**28.48**	(10.90)	0.88	(0.05)	33.31	(1.26)	24.86	(1.99)
OptiSim 5	20.60	(9.65)	0.93	(0.05)	32.61	(1.04)	23.82	(1.62)
OptiSim 10	24.40	(10.33)	0.90	(0.06)	32.68	(1.08)	23.90	(1.66)
OptiSim 25	27.79	(11.06)	0.88	(0.06)	33.04	(1.53)	24.44	(2.44)
IF	26.89	(9.14)	0.94	(0.06)	35.45	(2.26)	28.26	(3.67)
Random	15.40	(6.89)	0.97	(0.04)	32.92	(0.86)	24.34	(1.36)

Table E.15: Quality Measures of UPR Methods, 5 Representatives, Feature Set 5

.

Method	Perf. Variety		Max-Min Dist.		Sum Dist.		Sum Sq. Dist.	
KC-UPR	25.12	(10.55)	**0.79**	(0.04)	32.52	(1.22)	23.54	(1.76)
K-medoids	23.87	(10.55)	0.88	(0.07)	**31.88**	(0.59)	**22.69**	(0.92)
KS	**28.37**	(11.61)	0.88	(0.05)	33.34	(1.29)	24.90	(2.05)
OptiSim 5	20.65	(9.55)	0.93	(0.06)	32.51	(0.79)	23.66	(1.23)
OptiSim 10	23.24	(10.32)	0.90	(0.06)	32.80	(1.15)	24.08	(1.79)
OptiSim 25	27.39	(10.59)	0.88	(0.05)	33.01	(1.46)	24.40	(2.33)
IF	27.39	(9.61)	0.94	(0.06)	35.47	(2.41)	28.31	(3.94)
Random	14.88	(6.44)	0.96	(0.05)	32.97	(0.97)	24.41	(1.54)

Table E.16: Quality Measures of UPR Methods, 5 Representatives, Feature Set 6

.

Method	Perf. Variety		Max-Min Dist.		Sum Dist.		Sum Sq. Dist.	
KC-UPR	24.77	(10.23)	**0.79**	(0.04)	32.75	(1.30)	23.90	(1.95)
K-medoids	24.13	(10.14)	0.89	(0.06)	**31.95**	(0.62)	**22.80**	(0.96)
KS	**28.21**	(10.75)	0.89	(0.05)	33.34	(1.25)	24.92	(1.99)
OptiSim 5	19.74	(9.37)	0.93	(0.06)	32.51	(0.88)	23.68	(1.36)
OptiSim 10	24.60	(11.54)	0.91	(0.06)	32.67	(1.02)	23.89	(1.57)
OptiSim 25	26.14	(10.15)	0.88	(0.06)	32.98	(1.48)	24.35	(2.33)
IF	27.62	(9.70)	0.94	(0.06)	35.45	(2.18)	28.28	(3.56)
Random	15.40	(6.98)	0.96	(0.04)	32.91	(0.90)	24.32	(1.42)

Table E.17: Quality Measures of UPR Methods, 5 Representatives, Feature Set 7

.

Method	Perf. Variety		Max-Min Dist.		Sum Dist.		Sum Sq. Dist.	
KC-UPR	24.18	(10.00)	**0.78**	(0.04)	32.33	(1.04)	23.26	(1.50)
K-medoids	21.65	(10.16)	0.88	(0.07)	**31.85**	(0.52)	**22.64**	(0.80)
KS	**28.03**	(11.14)	0.88	(0.05)	33.23	(1.20)	24.74	(1.87)
OptiSim 5	21.87	(9.95)	0.93	(0.06)	32.52	(0.99)	23.69	(1.55)
OptiSim 10	24.68	(10.26)	0.90	(0.06)	32.62	(1.06)	23.81	(1.62)
OptiSim 25	26.38	(10.33)	0.87	(0.05)	32.84	(1.09)	24.13	(1.68)
IF	27.45	(10.39)	0.94	(0.04)	38.16	(2.43)	32.67	(4.14)
Random	17.17	(7.84)	0.97	(0.04)	32.87	(0.87)	24.27	(1.39)

Table E.18: Quality Measures of UPR Methods, 5 Representatives, Feature Set 8

.

Method	Perf. Variety		Max-Min Dist.		Sum Dist.		Sum Sq. Dist.	
KC-UPR	23.44	(10.78)	**0.78**	(0.04)	32.28	(1.11)	23.19	(1.62)
K-medoids	22.96	(10.68)	0.85	(0.07)	**31.77**	(0.57)	**22.51**	(0.86)
KS	**26.60**	(11.16)	0.87	(0.05)	32.67	(1.05)	23.87	(1.65)
OptiSim 5	20.64	(10.10)	0.92	(0.06)	32.26	(0.66)	23.28	(1.03)
OptiSim 10	23.22	(10.12)	0.88	(0.07)	32.37	(0.87)	23.41	(1.33)
OptiSim 25	24.81	(10.76)	0.86	(0.06)	32.59	(1.29)	23.76	(2.05)
IF	25.99	(9.66)	0.95	(0.06)	35.54	(2.08)	28.42	(3.40)
Random	15.97	(7.37)	0.96	(0.04)	32.70	(0.88)	23.99	(1.39)

Table E.19: Quality Measures of UPR Methods, 10 Representatives, Feature Set 0

.

Method	Perf. Variety		Max-Min Dist.		Sum Dist.		Sum Sq. Dist.	
KC-UPR	94.34	(27.25)	**0.72**	(0.03)	**27.88**	(0.55)	**19.35**	(0.61)
K-medoids	96.26	(24.47)	0.79	(0.05)	27.90	(0.30)	19.50	(0.44)
KS	**103.36**	(28.42)	0.77	(0.05)	28.09	(0.62)	19.77	(0.92)
OptiSim 5	95.86	(29.03)	0.84	(0.08)	28.07	(0.38)	19.76	(0.57)
OptiSim 10	100.96	(25.83)	0.79	(0.05)	28.03	(0.49)	19.68	(0.73)
OptiSim 25	102.17	(26.54)	0.77	(0.04)	28.05	(0.59)	19.70	(0.87)
IF	99.84	(25.76)	0.85	(0.07)	28.58	(0.77)	20.52	(1.17)
Random	70.63	(25.31)	0.94	(0.06)	28.73	(0.57)	20.81	(0.90)

Table E.20: Quality Measures of UPR Methods, 10 Representatives, Feature Set 1

.

Method	Perf. Variety		Max-Min Dist.		Sum Dist.		Sum Sq. Dist.	
KC-UPR	94.42	(26.93)	**0.71**	(0.03)	**27.82**	(0.53)	**19.27**	(0.56)
K-medoids	96.71	(24.65)	0.78	(0.06)	27.86	(0.30)	19.43	(0.45)
KS	**103.71**	(29.31)	0.76	(0.05)	27.98	(0.53)	19.60	(0.79)
OptiSim 5	96.31	(26.67)	0.82	(0.07)	27.97	(0.35)	19.60	(0.54)
OptiSim 10	103.15	(27.91)	0.77	(0.05)	27.93	(0.45)	19.52	(0.66)
OptiSim 25	102.61	(26.84)	0.76	(0.05)	27.91	(0.43)	19.50	(0.64)
IF	99.97	(25.42)	0.87	(0.07)	28.62	(0.85)	20.59	(1.30)
Random	70.85	(23.48)	0.94	(0.06)	28.65	(0.52)	20.67	(0.82)

Table E.21: Quality Measures of UPR Methods, 10 Representatives, Feature Set 2

.

Method	Perf. Variety		Max-Min Dist.		Sum Dist.		Sum Sq. Dist.	
KC-UPR	96.05	(28.36)	**0.71**	(0.03)	**27.84**	(0.44)	**19.28**	(0.43)
K-medoids	96.87	(24.78)	0.79	(0.05)	27.87	(0.30)	19.44	(0.44)
KS	101.79	(27.74)	0.76	(0.05)	27.96	(0.44)	19.57	(0.65)
OptiSim 5	92.45	(27.77)	0.83	(0.08)	27.99	(0.33)	19.64	(0.50)
OptiSim 10	100.40	(26.31)	0.78	(0.06)	27.92	(0.38)	19.51	(0.56)
OptiSim 25	**102.74**	(26.17)	0.77	(0.05)	27.94	(0.51)	19.55	(0.75)
IF	99.45	(24.26)	0.85	(0.07)	28.61	(0.94)	20.56	(1.46)
Random	72.79	(24.69)	0.94	(0.06)	28.65	(0.58)	20.68	(0.90)

Table E.22: Quality Measures of UPR Methods, 10 Representatives, Feature Set 3

.

Method	Perf. Variety		Max-Min Dist.		Sum Dist.		Sum Sq. Dist.	
KC-UPR	95.21	(27.63)	**0.71**	(0.03)	**27.78**	(0.49)	**19.22**	(0.48)
K-medoids	97.04	(24.58)	0.78	(0.05)	27.85	(0.27)	19.41	(0.41)
KS	**104.87**	(28.39)	0.75	(0.05)	27.95	(0.46)	19.55	(0.69)
OptiSim 5	95.05	(27.80)	0.81	(0.07)	27.94	(0.30)	19.56	(0.45)
OptiSim 10	101.60	(27.17)	0.78	(0.05)	27.86	(0.30)	19.43	(0.44)
OptiSim 25	102.02	(27.96)	0.76	(0.05)	27.91	(0.46)	19.50	(0.68)
IF	99.30	(25.51)	0.85	(0.07)	28.50	(0.62)	20.40	(0.94)
Random	72.46	(26.69)	0.93	(0.06)	28.64	(0.59)	20.66	(0.94)

Table E.23: Quality Measures of UPR Methods, 10 Representatives, Feature Set 4

.

Method	Perf. Variety		Max-Min Dist.		Sum Dist.		Sum Sq. Dist.	
KC-UPR	94.67	(27.90)	**0.71**	(0.02)	**27.78**	(0.39)	**19.18**	(0.32)
K-medoids	95.84	(24.73)	0.78	(0.06)	27.82	(0.27)	19.37	(0.40)
KS	102.92	(27.93)	0.75	(0.04)	27.83	(0.35)	19.39	(0.51)
OptiSim 5	92.11	(26.58)	0.82	(0.08)	27.89	(0.26)	19.48	(0.40)
OptiSim 10	100.11	(27.65)	0.77	(0.05)	27.81	(0.27)	19.36	(0.40)
OptiSim 25	**103.62**	(27.96)	0.75	(0.04)	27.81	(0.33)	19.36	(0.48)
IF	98.00	(25.11)	0.84	(0.07)	28.40	(0.69)	20.26	(1.08)
Random	68.11	(22.92)	0.94	(0.06)	28.58	(0.55)	20.58	(0.86)

Table E.24: Quality Measures of UPR Methods, 10 Representatives, Feature Set 5

.

Method	Perf. Variety		Max-Min Dist.		Sum Dist.		Sum Sq. Dist.	
KC-UPR	94.62	(27.76)	**0.70**	(0.02)	**27.78**	(0.38)	**19.18**	(0.29)
K-medoids	97.15	(25.14)	0.77	(0.06)	27.80	(0.26)	19.35	(0.39)
KS	**105.08**	(30.04)	0.74	(0.04)	27.84	(0.36)	19.39	(0.52)
OptiSim 5	92.32	(26.03)	0.82	(0.08)	27.91	(0.26)	19.51	(0.40)
OptiSim 10	100.06	(27.97)	0.77	(0.05)	27.79	(0.25)	19.33	(0.37)
OptiSim 25	103.34	(28.60)	0.75	(0.04)	27.84	(0.38)	19.40	(0.56)
IF	99.96	(26.13)	0.84	(0.07)	28.40	(0.72)	20.25	(1.11)
Random	70.98	(24.59)	0.93	(0.06)	28.51	(0.49)	20.46	(0.78)

Table E.25: Quality Measures of UPR Methods, 10 Representatives, Feature Set 6

.

Method	Perf. Variety		Max-Min Dist.		Sum Dist.		Sum Sq. Dist.	
KC-UPR	95.23	(28.47)	**0.71**	(0.02)	**27.80**	(0.54)	**19.22**	(0.52)
K-medoids	97.00	(25.44)	0.77	(0.06)	27.81	(0.28)	19.36	(0.41)
KS	**102.82**	(28.83)	0.75	(0.04)	27.83	(0.35)	19.38	(0.51)
OptiSim 5	97.40	(27.51)	0.83	(0.08)	27.93	(0.30)	19.54	(0.45)
OptiSim 10	98.89	(26.85)	0.77	(0.06)	27.83	(0.30)	19.38	(0.44)
OptiSim 25	101.90	(26.80)	0.75	(0.05)	27.83	(0.41)	19.38	(0.60)
IF	101.12	(25.89)	0.85	(0.08)	28.40	(0.74)	20.26	(1.16)
Random	67.73	(27.44)	0.94	(0.05)	28.54	(0.51)	20.51	(0.80)

Table E.26: Quality Measures of UPR Methods, 10 Representatives, Feature Set 7

.

Method	Perf. Variety		Max-Min Dist.		Sum Dist.		Sum Sq. Dist.	
KC-UPR	91.73	(27.16)	**0.71**	(0.03)	27.83	(0.55)	**19.23**	(0.43)
K-medoids	93.06	(25.59)	0.77	(0.06)	**27.79**	(0.26)	19.32	(0.39)
KS	100.67	(26.39)	0.74	(0.04)	27.82	(0.38)	19.36	(0.56)
OptiSim 5	91.98	(29.04)	0.81	(0.07)	27.88	(0.29)	19.47	(0.44)
OptiSim 10	99.09	(26.32)	0.75	(0.05)	27.79	(0.31)	19.32	(0.45)
OptiSim 25	101.20	(26.80)	0.74	(0.05)	27.81	(0.37)	19.35	(0.55)
IF	**101.29**	(26.16)	0.82	(0.07)	29.25	(1.77)	21.54	(2.75)
Random	68.95	(28.00)	0.93	(0.07)	28.60	(0.65)	20.61	(1.03)

Table E.27: Quality Measures of UPR Methods, 10 Representatives, Feature Set 8

.

Method	Perf. Variety		Max-Min Dist.		Sum Dist.		Sum Sq. Dist.	
KC-UPR	92.27	(28.84)	**0.70**	(0.02)	27.84	(0.50)	**19.20**	(0.37)
K-medoids	96.29	(25.32)	0.76	(0.05)	27.77	(0.27)	19.29	(0.39)
KS	103.35	(29.24)	0.73	(0.04)	27.73	(0.25)	19.24	(0.36)
OptiSim 5	92.74	(26.97)	0.80	(0.08)	27.82	(0.21)	19.37	(0.31)
OptiSim 10	99.00	(25.63)	0.76	(0.05)	**27.73**	(0.25)	19.23	(0.37)
OptiSim 25	**103.98**	(27.46)	0.73	(0.04)	27.73	(0.31)	19.23	(0.44)
IF	99.60	(25.10)	0.83	(0.07)	28.27	(0.69)	20.06	(1.09)
Random	67.09	(22.98)	0.93	(0.06)	28.50	(0.56)	20.46	(0.89)

Bibliography

[1] Michael O. Ball, Robert Hoffman, and Avijit Mukherjee. Ground delay program planning under uncertainty based on the ration-by-distance principle. *Transportation Science*, 44(1):1–14, 2010.

[2] Charles N. Glover and Michael O. Ball. Sparse Monge matrices arising from scheduling problems. *Operations Research Letters*, 41(3):246 – 248, 2013.

[3] Alan J. Hoffman. On simple linear programming problems. In Victor Klee, editor, *Convexity: Volume 7 of Proceedings of Symposia in Pure Mathematics*, pages 317–327. American Mathematical Society, 1963.

[4] Michael Z. Spivey and Warren B. Powell. The dynamic assignment problem. *Transportation Science*, 38(4):399–419, 2004.

[5] Avijit Mukherjee and Mark Hansen. A dynamic stochastic model for the single airport ground holding problem. *Transportation Science*, 41(4):444–456, 2007.

[6] Shai Ben-David and Margareta Ackerman. Measures of clustering quality: A working set of axioms for clustering. In Daphne Koller, Dale Schuurmans, Yoshua Bengio, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 121–128. Curran Associates, Inc., 2009.

[7] Brenda L. Dietrich. Monge sequences, antimatroids, and the transportation problem with forbidden arcs. *Linear Algebra and its Applications*, 139(Supplement C):133 – 145, 1990.

[8] Ilan Adler, Alan J. Hoffman, and Ron Shamir. Monge and feasibility sequences in general flow problems. *Discrete Applied Mathematics*, 44(1):21 – 38, 1993.

[9] Gaspard Monge. Mémoire sur la théorie des déblais et des remblais. In *Histoire de l'Académie Royale des Sciences de Paris*. 1781.

[10] Haiqing Song, Raymond K. Cheung, and Haiyan Wang. An arc-exchange decomposition method for multistage dynamic networks with random arc capacities. *European Journal of Operational Research*, 233(3):474 – 487, 2014.

[11] Faramroze G. Engineer, George L. Nemhauser, and Martin W. P. Savelsbergh. Dynamic programming-based column generation on time-expanded networks: Application to the dial-a-flight problem. *INFORMS Journal on Computing*, 23(1):105–119, 2011.

[12] Gregory D. Glockner and George L. Nemhauser. A dynamic network flow problem with uncertain arc capacities: Formulation and problem structure. *Operations Research*, 48(2):233–242, 2000.

[13] Raymond K. Cheung and Warren B. Powell. An algorithm for multistage dynamic networks with random arc capacities, with an application to dynamic fleet management. *Operations Research*, 44(6):951–963, 1996.

[14] Warren B. Powell and Linos F. Frantzeskakis. Restricted recourse strategies for dynamic networks with random arc capacities. *Transportation Science*, 28(1):3–23, 1994.

[15] Linos F. Frantzeskakis and Warren B. Powell. A successive linear approximation procedure for stochastic, dynamic vehicle allocation problems. *Transportation Science*, 24(1):40–57, 1990.

[16] Ming Hu and Yun Zhou. Dynamic type matching. Technical report, Rotman School of Management Working Paper 2592622, 2016.

[17] Rainer E. Burkard. Monge properties, discrete convexity and applications. *European Journal of Operational Research*, 176(1):1 – 14, 2007.

[18] Rainer E. Burkard, Bettina Klinz, and Rdiger Rudolf. Perspectives of Monge properties in optimization. *Discrete Applied Mathematics*, 70(2):95 – 161, 1996.

[19] A. Vince. A framework for the greedy algorithm. *Discrete Applied Mathematics*, 121(1):247 – 260, 2002.

[20] Avi Dechter and Rina Dechter. On the greedy solution of ordering problems. *ORSA Journal on Computing*, 1(3):181–189, 1989.

[21] Bernhard Korte and Lszl Lovsz. Greedoids - a structural framework for the greedy algorithm. In WILLIAM R. PULLEYBLANK, editor, *Progress in Combinatorial Optimization*, pages 221 – 243. Academic Press, 1984.

[22] Jack Edmonds. Matroids and the greedy algorithm. *Mathematical Programming*, 1(1):127–136, Dec 1971.

[23] R. M. Karp, U. V. Vazirani, and V. V. Vazirani. An optimal algorithm for on-line bipartite matching. In *Proceedings of the Twenty-second Annual ACM Symposium on Theory of Computing*, pages 352–358. ACM, 1990.

[24] Nikhil R. Devanur and Kamal Jain. Online matching with concave returns. In *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing*, pages 137–144. ACM, 2012.

[25] Mohammad Mahdian and Qiqi Yan. Online bipartite matching with random arrivals: An approach based on strongly factor-revealing LPs. In *Proceedings of the Forty-third Annual ACM Symposium on Theory of Computing*, pages 597–606. ACM, 2011.

[26] Aranyak Mehta, Amin Saberi, Umesh Vazirani, and Vijay Vazirani. Adwords and generalized online matching. *J. ACM*, 54(5), October 2007.

[27] D. Gale and L. S. Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.

[28] Alvin E. Roth and Marilda Sotomayor. *Two-sided matching: a study in game-theoretic modeling and analysis*. Cambridge University Press, 1990.

[29] Pavlos Eirinakis, Dimitrios Magos, Ioannis Mourtos, and Panayiotis Miliotis. Finding all stable pairs and solutions to the many-to-many stable matching problem. *INFORMS Journal on Computing*, 24(2):245–259, 2012.

[30] Hideo Konishi and M. Utku Ünver. Credible group stability in many-to-many matching problems. *Journal of Economic Theory*, 129(1):57 – 80, 2006.

[31] Marilda Sotomayor. Implementation in the many-to-many matching market. *Games and Economic Behavior*, 46(1):199 – 212, 2004.

[32] Mourad Baïou and Michel Balinski. Many-to-many matching: stable polyandrous polygamy (or polygamous polyandry). *Discrete Applied Mathematics*, 101(1):1 – 12, 2000.

[33] Juan Sebastin Pereyra. A dynamic school choice model. *Games and Economic Behavior*, 80(Supplement C):100 – 114, 2013.

[34] Ettore Damiano and Ricky Lam. Stability in dynamic matching markets. *Games and Economic Behavior*, 52(1):34 – 53, 2005.

[35] Ron Shamir. A fast algorithm for constructing Monge sequences in transportation problems with forbidden arcs. *Discrete Mathematics*, 114(1):435 – 444, 1993.

[36] Noga Alon, Steven Cosares, Dorit S. Hochbaum, and Ron Shamir. An algorithm for the detection and construction of Monge sequences. *Linear Algebra and its Applications*, 114(Supplement C):669 – 680, 1989. Special Issue Dedicated to Alan J. Hoffman.

[37] Balzs Kotnyek and Octavio Richetta. Equitable models for the stochastic ground-holding problem under collaborative decision making. *Transportation Science*, 40(2):133–146, 2006.

[38] Michael O. Ball, Robert Hoffman, Amedeo R. Odoni, and Ryan Rifkin. A stochastic integer program with dual network structure and its application to the ground-holding problem. *Operations Research*, 51(1):167–171, 2003.

[39] Octavio Richetta and Amedeo R. Odoni. Solving optimally the static ground-holding policy problem in air traffic control. *Transportation Science*, 27(3):228–238, 1993.

[40] Juan Camilo Castillo, Dan Knoepfle, and Glen Weyl. Surge pricing solves the wild goose chase. In *Proceedings of the 2017 ACM Conference on Economics and Computation*, pages 241–242. ACM, 2017.

[41] Masabumi Furuhata, Maged Dessouky, Fernando Ordez, Marc-Etienne Brunet, Xiaoqing Wang, and Sven Koenig. Ridesharing: The state-of-the-art and future directions. *Transportation Research Part B: Methodological*, 57(Supplement C):28 – 46, 2013.

[42] Niels Agatz, Alan Erera, Martin Savelsbergh, and Xing Wang. Optimization for dynamic ride-sharing: A review. *European Journal of Operational Research*, 223(2):295 – 303, 2012.

[43] Gerardo Berbeglia, Jean-Franois Cordeau, and Gilbert Laporte. Dynamic pickup and delivery problems. *European Journal of Operational Research*, 202(1):8 – 15, 2010.

[44] Jean-Françcois Cordeau, Gilbert Laporte, and Stefan Ropke. Recent models and algorithms for one-to-one pickup and delivery problems. In Bruce Golden, S. Raghavan, and Edward Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 327–357. Springer US, Boston, MA, 2008.

[45] Laurence A. Wolsey. *Integer Programming*. Wiley, 1998.

[46] Dimitris Berstimas and John N. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, Charlestown,MA, 1997.

[47] Maria Fonoberova. *Algorithms for Finding Optimal Flows in Dynamic Networks*, pages 31–54. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[48] Martin Skutella. *An Introduction to Network Flows over Time*, pages 451–482. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[49] Balázs Kotnyek. An annotated overview of dynamic network flows. Technical Report RR-4936, INRIA, September 2003.

[50] Jay E. Aronson. A survey of dynamic network flows. *Annals of Operations Research*, 20(1):1–66, Dec 1989.

[51] John M. Mulvey and Hercules Vladimirou. Solving multistage stochastic networks: An application of scenario aggregation. *Networks*, 21(6):619–643, 1991.

[52] Giorgio Gallo, Giustino Longo, Stefano Pallottino, and Sang Nguyen. Directed hypergraphs and applications. *Discrete Applied Mathematics*, 42(2):177 – 201, 1993.

[53] Nan Kong, Andrew J. Schaefer, and Shabbir Ahmed. Totally unimodular stochastic programs. *Mathematical Programming*, 138(1):1–13, Apr 2013.

[54] Arthur F. Veinott. Extreme points of leontief substitution systems. *Linear Algebra and its Applications*, 1(2):181 – 194, 1968.

[55] Robert G. Jeroslow, Kipp Martin, Ronald L. Rardin, and Jinchang Wang. Gainfree leontief substitution flow problems. *Mathematical Programming*, 57(1):375–414, May 1992.

[56] Riccardo Cambini, Giorgio Gallo, and Maria Grazia Scutellà. Flows on hypergraphs. *Mathematical Programming*, 78(2):195–217, Aug 1997.

[57] John R. Birge and François Louveaux. *Introduction to Stochastic Programming*. Springer, 1997.

[58] Jack Edmonds. Optimum branchings. *Journal of Research of the National Bureau of Standards - B. Mathematics and Mathematical Physics*, 71B(4), 1967.

[59] D. R. Fulkerson. Packing rooted directed cuts in a weighted directed graph. *Mathematical Programming*, 6(1):1–13, Dec 1974.

[60] Amedeo R. Odoni. *The Flow Management Problem in Air Traffic Control*, pages 269–288. Springer Berlin Heidelberg, Berlin, Heidelberg, 1987.

[61] Octavio Richetta and Amedeo R. Odoni. Dynamic solution to the ground-holding problem in air traffic control. *Transportation Research Part A: Policy and Practice*, 28(3):167 – 185, 1994.

[62] Lara S Cook and Bryan Wood. A model for determining ground delay program parameters using a probabilistic forecast of stratus clearing. *Air traffic control quarterly*, 18(1):85–108, 2010.

[63] Jun Chen and Dengfeng Sun. Stochastic ground-delay-program planning in a metroplex. *Journal of Guidance, Control, and Dynamics*, 2017.

[64] Jonathan Cox and Mykel J Kochenderfer. Ground delay program planning using markov decision processes. *Journal of Aerospace Information Systems*, pages 134–142, 2016.

[65] P Liu and Mark Hansen. Scenario-free sequential decision model for the single airport ground holding problem. *7th USA/Europe Air Traffic Management Research and Development Seminar, Barcelona, Spain*, 2007.

[66] Jonathan Cox and Mykel J Kochenderfer. Optimization approaches to the single airport ground-holding problem. *Journal of Guidance, Control, and Dynamics*, 38(12):2399–2406, 2015.

[67] Jennifer Gentry, Kent Duffy, and William J. Swedish. Airport capacity profiles. Technical Report F055-L11-014, MITRE Corporation, prepared for the Federal Aviation Administration, 2014.

[68] Parikshit Ram and Alexander G. Gray. Density estimation trees. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 627–635, New York, NY, USA, 2011. ACM.

[69] Stijn Van Dongen. Graph clustering via a discrete uncoupling process. *SIAM Journal on Matrix Analysis and Applications*, 30(1):121–141, 2008.

[70] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, pages 226–231. AAAI Press, 1996.

[71] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining, (First Edition).* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

[72] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Comput. Surv.*, 31(3):264–323, September 1999.

[73] Twan van Laarhoven and Elena Marchiori. Axioms for graph clustering quality functions. *Journal of Machine Learning Research*, 15:193–215, 2014.

[74] Pankaj K. Agarwal, Shariel Har-Peled, and Kasturi R. Varadarajan. Geometric approximation via coresets. In Jacob E. Goodman, János Pach, and Emo Welzl, editors, *Combinatorial and Computational Geometry*, volume 52, pages 1–30. MSRI Publications, 2005.

[75] Sariel Har-Peled and Soham Mazumdar. On coresets for k-means and k-median clustering. In *Proceedings of the Thirty-sixth Annual ACM Symposium on Theory of Computing*, STOC '04, pages 291–300, New York, NY, USA, 2004. ACM.

[76] Mihai Bādoiu, Sariel Har-Peled, and Piotr Indyk. Approximate clustering via core-sets. In *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*, STOC '02, pages 250–257, New York, NY, USA, 2002. ACM.

[77] Salvador Garcia, Joaquin Derrac, Jose Cano, and Francisco Herrera. Prototype selection for nearest neighbor classification: Taxonomy and empirical study. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(3):417–435, March 2012.

[78] Isaac Triguero, Joaquín Derrac, Salvador Garcia, and Francisco Herrera. A taxonomy and experimental study on prototype generation for nearest neighbor classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(1):86–100, Jan 2012.

[79] M. Daszykowski, B. Walczak, and D.L. Massart. Representative subset selection. *Analytica Chimica Acta*, 468(1):91 – 103, 2002.

[80] Robert D. Clark. OptiSim: An extended dissimilarity selection method for finding diverse representative subsets. *Journal of Chemical Information and Computer Sciences*, 37(6):1181–1188, 1997.

[81] R. W. Kennard and L. A. Stone. Computer aided design of experiments. *Technometrics*, 11(1):137–148, 1969.

[82] Elena Lloret and Manuel Palomar. Text summarisation in progress: A literature review. *Artif. Intell. Rev.*, 37(1):1–41, January 2012.

[83] Ani Nenkova and Kathleen McKeown. Automatic summarization. *Foundations and Trends in Information Retrieval*, 5(23):103–233, 2011.

[84] Karen Spärck Jones. Automatic summarising: The state of the art. *Information Processing & Management*, 43(6):1449 – 1481, 2007.

[85] Michael Mampaey and Jilles Vreeken. Summarizing categorical data by clustering attributes. *Data Mining and Knowledge Discovery*, 26(1):130–173, Jan 2013.

[86] Michael Mampaey, Jilles Vreeken, and Nikolaj Tatti. Summarizing data succinctly with the most informative itemsets. *ACM Trans. Knowl. Discov. Data*, 6(4):16:1–16:42, December 2012.

[87] Christian Borgelt. Frequent item set mining. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(6):437–456, 2012.

[88] Xifeng Yan, Hong Cheng, Jiawei Han, and Dong Xin. Summarizing itemset patterns: A profile-based approach. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, KDD '05, pages 314–323, New York, NY, USA, 2005. ACM.

[89] David W. Scott. *Multivariate Density Estimation: Theory, Practice, and Visualization*. John Wiley & Sons, 2015.

[90] Simon J. Sheather. Density estimation. *Statist. Sci.*, 19(4):588–597, 11 2004.

[91] Christopher J. C. Burges. Dimension reduction: A guided tour. *Foundations and Trends in Machine Learning*, 2(4):275–365, 2010.

[92] Xiaoming Huo, Xuelei Sherry Ni, and Andrew K. Smith. A survey of manifold-based learning methods. In T. Warren Liao and Evangelos Triantaphyllou, editors, *Recent Advances in Data Mining of Enterprise Data: Algorithms and Applications*. World Scientific Publishing Co., Inc., 2008.

[93] Imola K. Fodor. A survey of dimension reduction techniques. Technical Report UCRL-ID-148494, Lawrence Livermore National Laboratory, 2002.

[94] Prabir Burman and Wolfgang Polonik. Multivariate mode hunting: Data analytic tools with measures of significance. *Journal of Multivariate Analysis*, 100(6):1198 – 1218, 2009.

[95] Jerome H. Friedman and Nicholas I. Fisher. Bump hunting in high-dimensional data. *Statistics and Computing*, 9(2):123–143, Apr 1999.

[96] Jon Kleinberg. An impossibility theorem for clustering. In *Proceedings of the 15th International Conference on Neural Information Processing Systems*, NIPS'02, pages 463–470, Cambridge, MA, USA, 2002. MIT Press.

[97] 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, 1990.

[98] Teresa W. Haynes, Stephen Hedetniemi, and Peter Slater. *Fundamentals of domination in graphs*. CRC Press, New York, New York, 1998.

[99] S.T. Hedetniemi and R.C. Laskar. Bibliography on domination in graphs and some basic definitions of domination parameters. In Stephen T. Hedetniemi, editor, *Topics on Domination*, volume 48 of *Annals of Discrete Mathematics*, pages 257 – 277. Elsevier, 1991.

[100] Chin Kuan Ho, Yashwant Prasad Singh, and Hong Tat Ewe. An enhanced ant colony optimization metaheuristic for the minimum dominating set problem. *Applied Artificial Intelligence*, 20(10):881–903, 2006.

[101] L. A. Sanchis. Experimental analysis of heuristic algorithms for the dominating set problem. *Algorithmica*, 33(1):3–18, May 2002.

[102] Abhay K. Parekh. Analysis of a greedy heuristic for finding small dominating sets in graphs. *Information Processing Letters*, 39(5):237 – 240, 1991.

[103] Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38(Supplement C):293 – 306, 1985.

[104] Doron Chen and Reuven Chen. New relaxation-based algorithms for the optimal solution of the continuous and discrete p-center problems. *Computers & Operations Research*, 36(5):1646 – 1655, 2009.

[105] Sourour Elloumi, Martine Labbé, and Yves Pochet. A new formulation and resolution method for the p-center problem. *INFORMS Journal on Computing*, 16(1):84–94, 2004.

[106] C. Caruso, A. Colorni, and L. Aloi. Dominant, an algorithm for the p-center problem. *European Journal of Operational Research*, 149(1):53 – 64, 2003.

[107] Taylan Ilhan, F. Aykut Özsoy, and Mustafa Pinar. An efficient exact algorithm for the vertex p-center problem and computational experiments for different set covering subproblems. Technical Report 588, Department of Industrial Engineering, Bilkent University, 2002.

[108] Edward Minieka. The m-center problem. *SIAM Review*, 12(1):138–139, 1970.

[109] Tatjana Davidović, Dušan Ramljak, Milica Šelmić, and Dušan Teodorović. Bee colony optimization for the p-center problem. *Computers & Operations Research*, 38(10):1367 – 1376, 2011.

[110] Borut Robič and Jurij Mihelič. Solving the k-center problem efficiently with a dominating set algorithm. *Journal of Computing and Information Technology*, 13(3):225–234, 2005.

[111] Refael Hassin, Asaf Levin, and Dana Morad. Lexicographic local search and the p-center problem. *European Journal of Operational Research*, 151(2):265 – 279, 2003.

[112] Nenad Mladenović, Martine Labbé, and Pierre Hansen. Solving the p-center problem with tabu search and variable neighborhood search. *Networks*, 42(1):48–64, 2003.

[113] Jurij Mihelič and Borut Robič. Approximation algorithms for the k-center problem: An experimental evaluation. In Ulrike Leopold-Wildburger, Franz Rendl, and Gerhard Wäscher, editors, *Operations Research Proceedings 2002: Selected Papers of the International Conference on Operations Research (SOR 2002), Klagenfurt, September 2–5, 2002*, pages 371–376. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[114] Dorit S. Hochbaum and David B. Shmoys. A best possible heuristic for the k-center problem. *Mathematics of Operations Research*, 10(2):180–184, 1985.

[115] Erwan Scornet, Gérard Biau, and Jean-Philippe Vert. Consistency of random forests. *Ann. Statist.*, 43(4):1716–1741, 08 2015.

[116] Gérard Biau, Luc Devroye, and Gábor Lugosi. Consistency of random forests and other averaging classifiers. *Journal of Machine Learning Research*, 9:2015–2033, 2008.

[117] Nicolai Meinshausen. Quantile regression forests. *Journal of Machine Learning Research*, 7:983–999, 2006.

[118] Galen R. Shorack and Jon A. Wellner. *Empirical Processes with Applications to Statistics*. Society for Industrial and Applied Mathematics, 2009.

[119] Michael R Kosorok. *Introduction to empirical processes and semiparametric inference*. Springer Science+Business Media, LLC, 2007.

[120] Vladimir N. Vapnik and Alexey Ya. Chervonenkis. On the uniform convergence of the frequencies of occurrence of events to their probabilities. In Bernhard Schölkopf, Zhiyuan Luo, and Vladimir Vovk, editors, *Empirical Inference: Festschrift in Honor of Vladimir N. Vapnik*, pages 7–12. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[121] Ingrid K. Glad, Nils Lid Hjort, and Nikolai G. Ushakov. Correction of density estimators that are not densities. *Scandinavian Journal of Statistics*, 30(2):415–427, 2003.

[122] M. Kaluszka. On the devroye-gyrfi methods of correcting density estimators. *Statistics & Probability Letters*, 37(3):249 – 257, 1998.

[123] Leslaw Gajek. On improving density estimators which are not bona fide functions. *Ann. Statist.*, 14(4):1612–1618, 12 1986.

[124] Dimitris Bertsimas, Guglielmo Lulli, and Amedeo Odoni. An integer optimization approach to large-scale air traffic flow management. *Operations Research*, 59(1):211–227, 2011.

[125] Kenneth D Kuhn. A methodology for identifying similar days in air traffic flow management initiative planning. *Transportation Research Part C: Emerging Technologies*, 69:1–15, 2016.

[126] Bill Flathers, Matt Fronzak, Mark Huberdeau, Claudia McKnight, Ming Wang, and Gene Wilhelm. A framework for the development of the ATM-weather integration concept. Technical report, The MITRE Corporation, 2013.

[127] Alexander Estes, Michael O. Ball, and David Lovell. Predicting performance of ground delay programs. In *12th USA/Europe Air Traffic Management R&D Seminar, Seattle, WA*, 2017.

[128] J Arturo Olvera-López, J Ariel Carrasco-Ochoa, J Francisco Martínez-Trinidad, and Josef Kittler. A review of instance selection methods. *Artificial Intelligence Review*, 34(2):133–143, 2010.

[129] Wo-Ruo Chen, Yong-Huan Yun, Ming Wen, Hong-Mei Lu, Zhi-Min Zhang, and Yi-Zeng Liang. Representative subset selection and outlier detection via isolation forest. *Anal. Methods*, 8:7225–7231, 2016.

[130] Pei-chen Barry Liu, Mark Hansen, and Avijit Mukherjee. Scenario-based air traffic flow management: From theory to practice. *Transportation Research Part B: Methodological*, 42(7):685–702, 2008.

[131] Luis Delgado, Xavier Prats, and Banavar Sridhar. Cruise speed reduction for ground delay programs: A case study for San Francisco International Airport arrivals. *Transportation Research Part C: Emerging Technologies*, 36:83–96, 2013.

[132] S Penny, T Lewis, B Hoffman, T White, and J Krozel. Cluster analysis for the annualization of ACES simulated NAS metrics. Technical report, Metron Aviation, 2009.

[133] Shon Grabbe, Banavar Sridhar, and Avijit Mukherjee. Similar days in the NAS: an airport perspective. In *AIAA Aviation Technology, Integration, and Operations Conference*, 2013.

[134] Feng Cheng, John Gulding, Bryan Baszczewski, and Ruth Galaviz. An optimization model for sample day selection in NAS-wide modeling studies. In *Integrated Communications, Navigation and Surveilance Conference (ICNS), 2011*, pages L6–1. IEEE, 2011.

[135] Kris Poncelet, Hanspeter Höschle, Erik Delarue, Ana Virag, and William D'haeseleer. Selecting representative days for capturing the implications of integrating intermittent renewables in generation expansion planning problems. *IEEE Transactions on Power Systems*, 32:1936 – 1948, 2017.

[136] Fernando De Sisternes Jimenez and Mort D. Webster. Optimal selection of sample weeks for approximating the net load in generation planning problems. Technical Report ESD-WP-2013-03, Massachusetts Institute of Technology. Engineering Systems Division, 2013.

[137] Paul Nahmmacher, Eva Schmid, Lion Hirth, and Brigitte Knopf. Carpe diem: A novel approach to select representative days for long-term power system modeling. *Energy*, 112:430 – 442, 2016.

[138] Samira Fazlollahi, Stephane Laurent Bungener, Pierre Mandel, Gwenaelle Becker, and Franois Marchal. Multi-objectives, multi-period optimization of district energy systems: I. selection of typical operating periods. *Computers & Chemical Engineering*, 65:54 – 66, 2014.

[139] M. S. ElNozahy, M. M. A. Salama, and R. Seethapathy. A probabilistic load modelling approach using clustering algorithms. In *2013 IEEE Power Energy Society General Meeting*, pages 1–5, July 2013.

[140] Daniel Kahneman and Amos Tversky. On the reality of cognitive illusions. *Psychological Review*, 103(3), 1996.

[141] Roger Buehler, Dale Griffin, and Michael Ross. Exploring the "planning fallacy": Why people underestimate their task completion times. *Journal of Personality and Social Psychology*, 67(3):366, 1994.

[142] Amos Tversky and Daniel Kahneman. Judgment under uncertainty: Heuristics and biases. *Science*, 185(4157):1124–1131, 1974.

[143] Wing Yee Lee, Paul Goodwin, Robert Fildes, Konstantinos Nikolopoulos, and Michael Lawrence. Providing support for the use of analogies in demand forecasting tasks. *International Journal of Forecasting*, 23(3):377–390, 2007.

[144] Dan Lovallo, Carmina Clarke, and Colin Camerer. Robust analogizing and the outside view: two empirical tests of case-based decision making. *Strategic Management Journal*, 33(5):496–512, 2012.

[145] P.N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Pearson Education Limited, Harlow, 2014.

[146] Murray Rosenblatt. Remarks on some nonparametric estimates of a density function. *The Annals of Mathematical Statistics*, 27(3):832–837, 1956.

[147] Emanuel Parzen. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33(3):1065–1076, 1962.

[148] Alfred Inselberg. The plane with parallel coordinates. *The Visual Computer*, 1(2):69–91, 1985.

[149] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *Eighth IEEE International Conference on Data Mining, 2008*, pages 413–422. IEEE, 2008.

[150] Hae-Sang Park and Chi-Hyuck Jun. A simple and fast algorithm for k-medoids clustering. *Expert Systems with Applications*, 36(2, Part 2):3336 – 3341, 2009.

[151] Damien Francois, Vincent Wertz, and Michel Verleysen. The concentration of fractional distances. *IEEE Transactions on Knowledge and Data Engineering*, 19(7):873–886, 2007.

[152] Michael O Ball and Guglielmo Lulli. Ground delay programs: Optimizing over the included flight set based on distance. *Air Traffic Control Quarterly*, 12(1):1–25, 2004.

[153] Yi Liu and Mark Hansen. Evaluation of the performance of ground delay programs. *Transportation Research Record: Journal of the Transportation Research Board*, (2400):54–64, 2013.

[154] M. Ball, P. Swaroop, C. Barnhart, C. Yan, M. Hansen, L. Kang, Y. Liu, and V. Vaze. Service level expectation setting for air traffic flow management: Practical challenges and benefits assessment. In *Proceedings of the 12th USA/Europe Air Traffic Management R&D Seminar. 2015*, 2015.

[155] Prem Swaroop and Michael Ball. Consensus-building mechanism for setting service expectations in air traffic flow management. *Transportation Research Record: Journal of the Transportation Research Board*, (2325):87–96, 2013.

[156] Sreeta Gorripaty, Mark Hansen, and Alexey Pozdnukhov. Decision support framework to assist air traffic management. In *IEEE/AIAA 35th Digital Avionics Systems Conference (DASC), 2016*, pages 1–6. IEEE, 2016.