ABSTRACT

Title of thesis: MAXIMIZING INFLUENCE OF SIMPLE AND COMPLEX CONTAGION ON REAL-WORLD NETWORKS

Geoffrey Moores
Master of Science, 2020

Thesis directed by: Professor Aravind Srinivasan
Department of Computer Science

Contagion spread over networks is used to model many important real-world processes from a wide variety of domains including epidemiology, marketing, and systems engineering. A large body of research provides strong theoretical guarantees on simple contagion models, but recent research identifies many real-world processes that feature complex contagions whose spread may depend on multiple exposures or other complex criteria.

We present a rigorous study of real-world and artificial networks across simple and complex contagion models. We identify domain-dependent features of real-world networks extracted from publicly-available networks as a guide to solving contagion-related decision problems. We then examine the performance of multiple influence-maximization algorithms across a space of networks and contagion models to develop an experimentally justified guide of best practices for related problems. In particular, genetic algorithms are an extremely viable candidate for these problems, especially with complex graphs and processes.

MAXIMIZING INFLUENCE OF SIMPLE AND COMPLEX CONTAGION ON REAL-WORLD NETWORKS

by

Geoffrey Moores

Thesis 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
Master of Science
2020

Advisory Committee:
Professor Aravind Srinivasan, Chair/Advisor
Professor Anil Vullikanti
Professor John Dickerson

Acknowledgments

My sincere thanks to the advisory committee, Professors Srinivasan, Vullikanti, and Dickerson, whose critical thought and attention greatly improved this work. In particular, my appreciation goes to Professor Srinivasan for his consistent guidance and mentorship throughout the past eighteen months. His focus and insights were instrumental to my growth as a scientist and professional.

I would also like to thank the National Science Foundation for their financial and organizational support. This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE 1840340. The opinions in this document are mine and do not necessarily reflect the opinions of the NSF, nor the US military, by whom I am employed.

Finally, my thanks to my family and friends. Most of all to my wife, Emma, whose patience and love are the foundation for my successes.

Table of Contents

Chapter 1: Introduction

Network science encompasses a broad domain of interdisciplinary sciences that leverage utility in the representation graph models offer. The explosion of network data and parallel growth in computing power in the past decades offers fertile ground for investigating the behavior of complex systems. One area where this is particularly evident is modeling diffusion processes over graphs. Originally brought into mainstream network science discussion by Kempe, Kleinberg, and Tardos [1] as well as Domingos and Richardson [2], many researchers have since developed various diffusion models and sophisticated algorithms to identify vertices or other structures of interest. Exactly what is "of interest" is application dependent. In some scenarios, we seek simply to predict the outcome of a process. In viral marketing, we select a set of initial nodes to maximize how many nodes the process eventually reaches (frequently referred to as influence maximization). In epidemic mitigation, we desire to limit the spread of a disease by providing some form of medical care or altering the connections of the network.

The works of Tang [3] and Sadeh [4] represent the cutting edge of the influence maximization problem under the classic assumption that the contagion investigated is *simple* (see 1.1). But Kempe et al. [1] proved that there is no $n^{1-\epsilon}$ approximation

factor, $\forall \epsilon > 0$, where n is the number of nodes, for general *complex* contagions on unrestricted graphs unless $\mathbf{P} = \mathbf{NP}$. Since then, various attempts to identify specific cases for complex processes, where a constant approximation *is* viable, have identified that this boundary is rather steep: even given restricted classes of graphs or limiting how many vertices may act in a complex fashion there are drastic costs on achievable approximation factors [5, 6].

Were these complex processes infrequent in the systems network scientists hope to study, we might hope or choose to avoid them where possible. However, in the past decade, researchers leveraged the ability to study massive, real-world contagions on social media to identify that many real-world processes of interest exhibit distinctly complex behaviors [7–9]. Furthermore, the complexity of the problem is not just dependent on the process itself. For example, epidemic mitigation does not admit a constant approximation guarantee even when the underlying process studied is simple. Consider the scope of many networks of interest today: upwards of hundreds of thousands of nodes and millions of edges. As the epidemic research centers of the world seek to predict and manage the rapid spread of potentially deadly viruses, such as COVID-19 in early 2020, they must contend with a massive, dynamic network and continually update their model in real time. Complex processes cannot be avoided.

We take a focused dive into this realm of massive and complex network problems. Introductions to the terminology, processes, and problem statements are in this chapter. A reader well acquainted with network processes may safely skip this chapter on a first pass, with the exception of 1.3, which introduces two novel complex processes. In Chapter 2, using a variety of methods for community detection

with emphasis on stochastic block models for their flexibility, we extract general characteristics particular to a variety of real-world networks across many domains. The community structures found are considered in relation to the performance of hybrid-approximation algorithms. Readers familiar with basic centrality measures and stochastic block models may wish to begin reading Chapter 2 with the discussion of the map equation (Section 2.3.2). Chapter 3 compares various genetic algorithm variants in the context of influence maximization. Finally, in Chapter 4 we analyze the performance of greedy algorithms, common heuristics, and alternative algorithms (in particular genetic algorithms) in simple and complex environments with artificial and real-world networks. It is our hope that these results will provide the reader with confidence in selecting appropriate algorithms and models to understand real-world complex contagion.

1.1 Simple Contagion

Given a graph $G = (V, E)$, where V represents the set of vertices and E the set of edges, a *discrete diffusion process* on the network is defined as a set of states $s_j \in \mathbf{S}$ such that each vertex v_i has a state $S(v_i) \in \mathbf{S}$ and transition rules that map a vertex and its neighborhood to a new state at the next time step: $F(S^t(v_i), N(v_i)) \rightarrow S^{t+1}$. In this thesis, we will use the terms graph and network interchangeably, and similarly node or vertex and edge or link. If an edge exists in the network between nodes v_i and v_j then $e_{ij} \in E$. Typically the neighborhood is the direct neighbors of a given node: $N(v_i) = \{v_j \text{ s.t. } e_{ij} \in E\}$. With the introduction

of edge weights, node weights, and/or directional edges, processes can be specified to a high degree of detail. However, unless otherwise specified, we will consider only undirected and unweighted graphs in this thesis. Given this broad definition of process, many theoretical and real-world processes of interest may be modeled and there is a robust literature on diffusion processes over networks that ranges from digital viruses to biological diseases [10].

A simple contagion is a subset of diffusion processes where the transition functions can be represented entirely by functions involving pairwise interactions and there are no stored intermediate states — that is, a node cannot accumulate evidence of multiple interactions prior to a transition. Less precisely but perhaps more meaningfully, a single infected neighbor is sufficient for a node to (perhaps probabilistically) become infected itself in any simple contagion. Many real-world processes are commonly understood to act as simple contagions, most notably the spread of disease between individuals. For example, one sick associate is all it takes to become sick yourself; while many interactions with sick people may increase your risk of sickness, there is no complex interaction between these people that produces the transmission – they all represent a distinct chance of catching the disease. We consider three common models of simple contagions used heavily in the literature: the Independent Cascade (IC) model, the Susceptible Infected Recovered (SIR) model, and the Random Weighted Linear Threshold (RW-LT) model.

1.1.1 Independent Cascade (IC) Model

In the IC model, nodes are in either an Active s_1 or Inactive s_0 state, with some non-empty set of initially active nodes A_0. At each timestep, each newly active node i has a single chance to infect each neighbor j with probability p_{ij}, which may be determined based on edge weights, or uniform across all edges, etc. The process terminates when no nodes are infected at a given step (either because all infection attempts failed, or because the (connected) graph is completely infected).

1.1.2 Susceptible Infected Recovered (SIR) Model

The SIR model has three states: Susceptible s_0, Infected s_1, and Recovered s_2. It generalizes the IC model, usually following the same mechanics except that an infected node becomes recovered with some likelihood each timestep or after a certain number of steps, and afterwards may not be infected again or infect its neighbors. A standard SIR model with recovery rate equal to zero is the IC model, but there are many studies where the SIR model is employed with alternative or more complex transition rules. For example, in the work of Chen et al. [11], an infected node infects exactly one neighbor chosen at random each timestep, provided that neighbor is in state s_0 (susceptible).

1.1.3 Random-Weighted Linear Threshold (RW-LT) Model

The RW-LT model has two states, Active s_1 or Inactive s_0, and at t_0 each node is randomly assigned a threshold $\theta_i \in \mathbb{U}[0, 1]$ where $\mathbb{U}$ is the uniform distribution

—- therefore, the thresholds are not known *a priori*. When the ratio of infected neighbors to all neighbors of a node i exceeds its θ_i, it becomes active in the next timestep. While this may appear to be a complex transition function, it can be shown that if all the weights are initialized uniformly at random the process is equivalent to the IC model [1]. However, the general Linear Threshold (LT) model is not simple (see Section 1.2.1).

1.2 Complex Contagion

Complex contagion is a subset of diffusion processes where some of the transition functions are dependent on the total set of infected neighbors or there are stored intermediate states that can be equated to such a transition function. Intuitively, in a complex process the presence of n infected neighbors significantly changes the transition likelihood or next state that cannot be correctly interpreted as n separate infection attempts. In some real-world settings, the spreading processes are well modeled by a mixture of simple and complex transition functions [12].

Complex contagions empirically manifest in most commonly in social settings such as the spread of hashtags across Twitter or the shared usage of voice chat and other social-media services [7–9]. They can also be found when modeling neural networks, as the tendency for a neuron to fire depends on a certain number of inputs exceeding that neuron's internal threshold. We consider the Linear Threshold (LT) model and introduce the Saturated Linear Threshold (S-LT), the Ugander Complex (UC), and Community Complex (CC) models in this thesis, which we define in the

following subsections.

We first take a moment to define submodular functions, which will be put into context in Section 1.3, but is relevant to our understanding of these complex processes. Given a function which maps a subset S of a universe $\mathcal{U}$ to a real-valued or integer number, $F(S) \rightarrow \mathbb{R}/\mathbb{I}$, we say F is submodular if:

$$\forall A, B \subseteq \mathcal{U} \text{ s.t. } A \subseteq B \text{ and } \forall i \in U \setminus B :$$

$$F(A \cup i) - F(A) \geq F(B \cup i) - F(B)$$

Many functions in combinatorial optimization are submodular which can provide great insight and guarantees on algorithm design and performance (see Vondrák's dissertation for a survey [13]). However, in general, the influence function of an initially-active set of nodes on any of these complex processes is non-submodular.

1.2.1 Linear Threshold (LT) Model

The LT model has two states, as in the RW-LT above, active and inactive. Each node v_i additionally has a *fixed* threshold θ_i and transitions to active when some function $f(N(v_i)) \geq \theta_i$. Given a set of initially active nodes A_0, the process runs until there are no new activations in a given time step. Expected influence of a seed set over the LT process is monotonic and non-submodular.

1.2.2 Saturated Linear Threshold (S-LT) Model

The S-LT model has three states: susceptible, active, and saturated. Each node v_i has fixed thresholds θ_i^A and θ_i^S, transitions to active when some function $\theta_i^A \leq f(N(v_i)) < \theta_i^S$, and transitions to saturated when $f(N(v_i)) \geq \theta_i^S$. Nodes remain saturated once they reach this state, regardless of whether $f(N(v_i))$ remains above the saturation threshold. Nodes may pass directly into the saturated state. Given a set of initially active nodes A_0, the process runs until there are no new activations in a given time step. In general, expected influence over the S-LT process is non-monotonic and non-submodular.

1.2.3 Ugander Complex (UC) Model

The UC model exhibits tunable behavior that was empirically found by Ugander et al. [8]: the chance of infection is strongly related to the number of connected components among a set of infected neighbors. Given a function $F(i) \to [0, 1]$, $i \in \mathbb{N}$, each node has a probability $F(k)$ of becoming active the *first* time that its infected neighbors, when considered as a subgraph, represent exactly k different connected components. For example, the first infected neighbor v_j of a node v_i is by definition one connected component (itself), and represents an $F(1)$ chance of infecting v_i. Should another neighbor node, v_k, become infected, there are two possibilities. If there is an edge between v_j and v_k, i.e. $e_{jk} \in E$, then there is still only one component among v_i's set of infected neighbors, and there is no additional chance of infection. If $e_{jk} \notin E$, then there are now two distinct connected components in v_i's

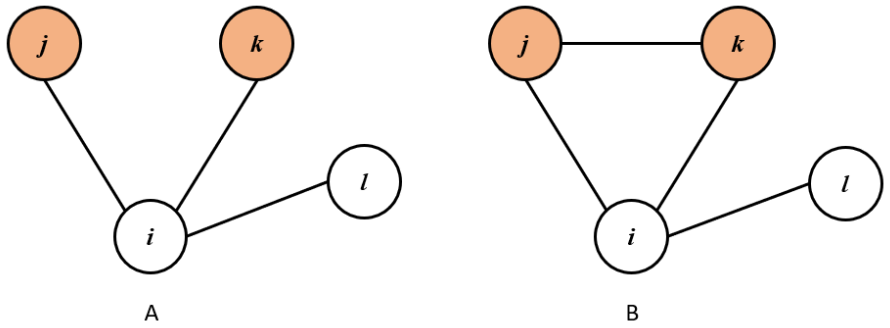

Figure 1.1: Two simple graphs, A and B, illustrate the difference in number of connected components in the subgraph of infected neighbors. In each the subgraph of infected neighbors for v_i includes nodes v_j, v_k. However, in A there is no edge e_{jk}, so the number of infected components is 2. In B, $e_{jk} \in E$, and therefore the number of infected components is simply 1.

infected neighborhood, and there is now an additional $F(2)$ chance to infect v_i. See Fig 1.1 for an illustration. Expected influence over the UC process is non-monotonic and non-submodular in general.

1.2.4 Community Contagion (CC) Model

In the **CC** model, nodes are in an active or inactive state, have a threshold θ_i, and there is a set of sets of communities $\mathbf{C} = \{C_1, ..., C_k\}$, where $C_i = \{c_{i,1}, ..., c_{i,n}\}$, such that each node belongs to a single community in each set C_k. See Fig 1.2 for an illustration. As such, we can construct a set of k partitions of V according to these communities. Given the sets of active and inactive nodes at a time step, in the next step a node becomes active if some function $F(Communities(N(i))) > \theta_i$. This model can represent the intuition that hearing something from multiple different sources is more convincing than hearing it many times from a single source. Expected influence over the CC process is monotonic and non-submodular in general.

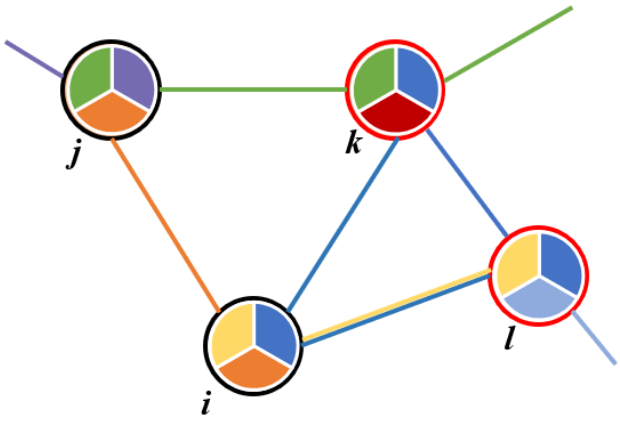

Figure 1.2: A simple illustration of a graph where each node belongs to a community in each of three sets of communities. Community membership is represented by the colors inside each node, and edges are shared between nodes which belong to the same community in any given set. Some nodes may share multiple edges by having a common community in more than one set. In this example, supposing v_k and v_l are infected, v_i would have two infected neighbors from two community sources (yellow and dark blue). If v_k was not infected, v_i would have only one infected neighbor but still two community sources.

As we will see in the following section, these models introduce some inherent complexity in their representation and additionally happen to be much harder to provide any approximation guarantees for under well-studied problems. However, it is increasingly apparent that these are the type of processes we must use to accurately model many interesting real-world events.

1.3 Influence Maximization and Related Problems

The study of spreading processes on graphs came into mainstream focus when Domingos and Richardson [2] posed the question of selecting a set of nodes that maximize the size of a process' total spread. Define $\sigma_P(G, A)$ as the total number of nodes infected on a graph G under process P and initially active nodes A. Then the Influence Maximization problem is defined as follows:

1.3.1 INF-MAX

Given a graph G, process P, and integer K, choose a set of nodes $A_0 \subset V$, $|A_0| = k$, $s.t.\ A_0 = argmax_A \mathbb{E}[\sigma_P(G, A)]$.

This problem has many interesting applications even beyond the viral marketing scenario Domingos and Richardson imagined. Unfortunately, even calculating the exact expected influence for a fixed set of nodes $\mathbb{E}[\sigma_P(G, A)]$ is #**P**-complete in general [14,15]. Typically some method of sampling the process is used to obtain an arbitrarily good estimation of $\mathbb{E}[\sigma_P(G, A)]$. For simple contagions, to determine an approximation of the optimal A_0, a greedy algorithm is usually employed that selects at each step the node that most improves the expected number of infections by the process, given the current set of selected nodes. Nemhauser proved that greedy maximization of submodular functions provides a $1 - 1/e$ bound on the approximation factor [16]. Kempe et al. [1] showed that the expected influence of given seed set for the IC and RW-LT processes is submodular and therefore a greedy algorithm for influence maximization achieves a $1 - 1/e - \epsilon$ approximation factor, when $\sigma_P(*)$ is estimated within a $1 \pm \delta$ factor and ϵ is a function of δ. The time complexity of their algorithm in total is $knm\frac{n^2}{\epsilon^2} \ln \frac{1}{\delta}$.

Nemhauser states that given a function $f(\cdot)$ that is non-negative, monotone and submodular, a greedy algorithm that adds an element with the largest marginal increase in $f(\cdot)$ each step produces a k-element set A such that $f(A) \geq (1 - 1/e) \cdot max_{B:|B|=k} f(B)$. This illuminates a distinction between complex and simple contagions. It can be shown that the simple contagions P as defined in 1.1 produce a

$\sigma_P(\cdot)$ that is submodular and monotone, whereas the expected influence of complex contagions is frequently non-submodular and perhaps not monotonic. In general, most simple processes that are strictly progressive (that is, there are no transition rules that can revert a node to an previous state) will be monotone and submodular.

Since the original result of Kempe et al. [1], much work has been done that fine tunes their greedy algorithm for the IC process. Notably, Leskovec et al. [17] recognized that since $\sigma_P(\cdot)$ is submodular, considerable computational effort can be spared by ignoring any candidate additions v_i at a given step that contributed less marginal increase at the *last* step than the current step's front runner v^*: that is, $\Delta\sigma_P(A \cup v_i) < \Delta\sigma_P(A' \cup v^*); |A'| = |A| + 1$. In a series of papers, Chen et al. [18,19] considered heuristics such as discounted degree counts and limited neighborhood process evaluations to improve computational cost. Borgs and Tang, with their colleagues, reduced the computational cost to a factor of $\epsilon^{-2}(k+l)(n+m)\log n$ with a clever use of "reverse reachable" sets computed prior to execution of the greedy algorithm [3,20] . More recently, Sadeh, Cohen, and Kaplan [4] presented an algorithm that runs on order of $\epsilon^{-2}k^3 \ln \frac{n}{\delta}(\tau\bar{m} + k(m^* + nk)\ln n)$, which they prove is a tight upper bound to achieve the $1 - 1/e - \epsilon$ guarantee, which leverages samples from a limited step process evaluation. Interestingly, and taking a different tack, Akbarpour and his team recently demonstrated that simply selecting a random set of $k+s$ nodes is often as influential as carefully selecting k nodes under any of these methods, even when s is small relative to k [21].

Unfortunately, all of these rigorous reductions in computational effort that maintain the strong approximation guarantee have no such guarantee under com-

plex processes. In fact, it can be shown that no $n^{1-\epsilon}$, $\forall \epsilon > 0$, can be achieved in the general case for complex processes unless $P = NP$ [1]. Some recent attempts to find specific cases with restrictions on graph form or the transition functions illuminate just how distinct the separation between submodular and non-submodular processes is: even with a small number n^{γ} of "almost submodular" transitioning nodes (the rest being completely defined by submodular functions), there is no $1/n^{\gamma/c}$ approximation factor unless $P = NP$ [6].

There are many other interesting problems besides **INF-MAX**, and each usually increases the complexity beyond constant approximation guarantees, even under simple contagions. We define two common alternate problems that we explore further in Chapter 4.

1.3.2 Epidemic Mitigation

In the Epidemic Mitigation (EM-P, where the P refers to the underlying process) problem, we seek a set I of size k *inoculated* nodes that cannot be infected that minimizes $\mathbf{E}_A[\sigma_P(G, A, I)]$, where the expectation is taken over the distribution of the epidemic seed set $A \sim \mathbf{A}$ and randomness in the process P. Inoculating a member of A has no effect. In Chapter 4, we will fix A for each graph to reduce the computational expense. We select A for all graphs by first infecting a random node, then conducting a small (10) node breadth-first-search from that "patient zero". For clarity and consistency with all other data presented, in Chapter 4 we present the number of nodes that were **not** infected (minus the number of inoculated nodes)

rather than the number that were infected. This casts the problem as a maximization problem.

Minimizing the spread of a contagion is non-submodular in the set of inoculated nodes. Consider a scenario where two populations are separate but for two members who travel between each population. If we inoculate both of these members, we could completely contain an outbreak that initiates in either of the populations; while inoculating just one may have little effect if the process is highly contagious.

1.3.3 Adversarial Inf Max

Adversarial Influence Maximization (ADV-P) models two (or more) competing contagions, A and B, which interact in some manner. A node that is infected at a given timestep by only one contagion transitions according to the process P rules normally. However, once infected by a contagion, it may not become infected by the alternate contagion. Nodes that are contested (satisfy both A and B influence functions at the same timestep) are declared *null*, and belong to neither contagion for the remainder of the process.

The (ADV-P) problem is as follows: given G, k, P, choose a set a^* of k nodes $a^* = argmax_{a \in V} \mathbf{E}[\sigma_{P_A}(G, a, b)]$ where expectation is taken over the distribution of $b \sim \mathbf{B}$ and the randomness in P. That is, we seek to select a set of nodes which will maximize the spread of process A in competition with the expected behavior of process B. This expected behavior is a function of which seed sets B may select

and simulations of the process. Instead of describing a distribution $\mathbf{B}$, we fix the set b in all Adversarial trials to be the top-10 nodes, ranked by degree (10 nodes was chosen to be competitive in relation to the k values in all experiments) . This poses a significant challenge but reduces the cost to effectively estimate process results. All dual-selected nodes $c \in a \cap b$ are immediately declared *null*. Although a strategy could simply choose $a = b$ to prevent B from spreading well, this would result in a score of 0, as the process a would not spread either.

Chapter 2: Modeling and Properties of Real-World Networks

Much of the network-science literature quantifies the form and structural characteristics of networks in order to gain a better understanding of what networks are similar, how they might behave in various environments, and how to generate artificial network structures that are useful for modeling purposes. At the most elemental end of the spectrum, one can measure the number of nodes or edges or degree of nodes (the number of edges connected to a given node). If instead we consider a node and its neighborhood we gain more interesting measures such as the clustering coefficient and a variety of other measures of a node's (or edge's) local structure [11]. Additional complexity introduces measures of groups of nodes, such as the size of a group's total neighborhood [22].

If we consider the whole graph as input, we can compute betweeness, eigenvector centrality (closely related to Google's PageRank), and many other powerful metrics. Introducing sets of nodes at this level, we can calculate group betweenness or the fragmentation or disruption of the group [23]. Most of these measures are so called *centrality* measures, which seek to gain some understanding of a node's importance (context dependent) to the network. An exhaustive review of centrality measures is beyond the scope of this thesis, but we define a few that we use consis-

tently below and refer the reader to Barabási's textbook [10] for a thorough coverage of the most frequent measures. In the next few sections, we cover in depth the measures we will use heavily in this thesis, including degree distribution and stochastic block models (SBM), in terms of measurement, generative models, and theoretical relevance to real-world networks. We then leverage these measurements to conduct a wide survey of real-world networks that identifies in which domains we might expect to find community structures, scale-free structures, or random structures.

2.1 Classic Centrality Measures

The degree of a vertex v_i:

$$k_i = |v_j \in V \ s.t. \ e_{ij} \in E|. \tag{2.1}$$

In the case of a directed network, we may distinguish:

$$k_i^{out} = |v_j \in V \ s.t. \ e_{ij} \in E|. \tag{2.2}$$

$$k_i^{in} = |v_j \in V \ s.t. \ e_{ji} \in E|. \tag{2.3}$$

If we define $\omega_{st}(v_i)$ and ω_{st} as the number of shortest paths from s to t through v_i and total, respectively, then the betweenness of a vertex v_i is:

$$B(v_i) = \sum_{s \neq v_i \neq t} \frac{\omega_{st}(v_i)}{\omega_{st}}. \tag{2.4}$$

Defining $N_i = v_j \ s.t. \ e_{ij} \in E$ and $edges(N_i) = \{e_{jk} \ s.t. \ e_{jk} \in E \wedge v_j, v_k \in N_i\}$, the clustering coefficient of v_i is:

$$C(v_i) = \frac{2 * |edges(N_i)|}{k_i(k_i - 1)}. \tag{2.5}$$

2.2 Degree Distribution and Erdős-Rényi, Barabási-Albert Graphs

As previously mentioned, analyzing the distribution of a graph's node-specific properties is a good way to gain perspective on the whole network. The degree distribution, for example, maps the integers to the number of vertices in the network that have that degree. This distribution is trivial to compute and can offer many insights into what type of network is under study. As described in Barabási's text [10], Erdős-Rényi random networks tend to have a degree distribution that is well described by a Poisson distribution: There are a high number of nodes with degree $= pN$, and the probability of having many fewer or many more is very low. Erdős-Rényi networks are parameterized by a number of nodes N and a probability p, and each e_{ij} is sampled independently from $\sim \mathbf{B}(p)$ where $\mathbf{B}$ is the Bernoulli distribution. Although Erdös-Rényi networks exhibit a highly homogeneous structure and are not often descriptive of real-world networks at large, they can serve as important test cases and are may be a good approximation of subgraphs of a real networks.

Real-world networks often have a "scale-free" property — they have many nodes of low degree, but with high probability there is also a small set of nodes with extremely high degree. Their degree distributions are well modeled by a power law

with exponent $\gamma \in (2, 3)$ [10]. The Barabási-Albert generative model is parameterized by a number of nodes N, a small integer m, and a (small) initial connected subgraph of m_0 nodes with random links. A single node is added each iteration with m links, with preferential attachment to nodes in the graph with high degree. This tends to build scale-free networks and can be a good approximation for real-world networks or their subgraphs.

2.3 Community Structure and Stochastic Block Models

The community structure of a network refers to a partition of vertices into groups that tend to associate with each other and/or associate similarly with other groups of the partition. In a social-network context, these can be directly interpreted as real-world communities. The partition can also be useful as a clustering tool that simplifies the model under study or as an input to modeling spreading processes. There are many techniques for recovering community partitions given a graph G with a range of complexities and guarantees. In general, one defines some metric of how "good" a partition is, for example modularity as defined in Newman and Girvan's paper [24], and then applies a search algorithm to seek a maximum among the space of partitions.

The definition of useful metrics is, in particular, an open question, as each metric may have applications of particularly high utility and an accompanying complexity advantages depending on what search algorithms lend themselves to it. For a survey of the current state of the community detection in networks, we refer the

reader to Fortunato's community detection survey paper [25]. In this thesis, we use two metrics for structure detection: Stochastic Block Model fit [26], and the length of the map equation [27].

2.3.1 Stochastic Block Models

Stochastic Block Models (SBM) are an extremely flexible generative model for representing community structure in networks.

Parameter	Specification	Meaning
k	$\in \mathbf{Z}$	number of communities
α	$\in \{O, 1\}^k \; s.t. \; \sum_i \alpha_i \; = \; 1$	probability node belongs to a community i
Π	$\in [0, 1]^{k \times k}$	a matrix that defines inter-community interaction (edge) likelihood

Classic SBM parameters.

To generate a graph $G \sim SBM(\mathbf{Z}, \Pi)$, each edge e_{ij} drawn from $Bernoulli(\pi_{kl})$ provided $z_i = k$ and $z_j = l$. The spectrum of associative community interactions enable the SBM to model many real-world structures, but it also encompasses other structures, such as dis-associative communities. Its generality and statistically principled approach make it a rigorous baseline for community detection and the SBM model enjoys significant attention for this reason.

Daudin et al. [26] leveraged the Integrated Complete-Data Log-Likelihood criterion of Biernacki [28] to build a variational expectation-maximization algorithm that takes a graph G as input and returns a high likelihood partition $\mathbf{Z}$. Because the standard SBM does not allow for heterogeneous-degree structures within a community, as is found in many real-world networks, Karrer and Newman developed a

heterogeneous-degree SBM model and greedy-partitioning algorithm to recover such a structure [29]. Côme and Latouche [30] developed a greedy algorithm for faster performance and simultaneous number of communities detection, while Peixoto [31] developed an agglomerative heuristic as a starting point for unbiased MCMC sampling methods. Further generalizing the classic model, Airoldi et al. introduced a mixed SBM model that allows nodes to belong to a distribution of communities [32]. In a later result [33], they developed an additional multi-layered SBM variant that allows for a community associations to vary across different layers while the community affiliations of nodes remains constant. Finally, in an alternative layered model, Peixoto leveraged a nested SBM model that resolves smaller scale structures than those that the classic model can identify [34].

2.3.2 The Map Equation

The map equation [27] approaches community detection from a dynamics perspective. If one imagined a random walker on a given graph and wished to communicate its path, a simple solution might be a list of unique node indices in the order of travel. For efficiency, one might compress this route information by giving each node a community, and then providing node indices unique to each community or a community transfer code. If the walker tends to stay within communities more often that traversing between them, then the code will be shorter due to the frequent, smaller intra-community node indices.

The map equation formalizes this concept by defining a lower bound on the

optimal encoding of a random walk on a given network, which is sensitive to the flow of a given network. The partition-detection algorithm [35] uses an agglomerative technique for reducing the code length of a partition, where nodes initially belong to their own partitions and get iteratively combined until the map equation cannot be further reduced.

2.4 A Variable-Community Multi-Layer Stochastic Block Model

As an alternative extension to the multi layer models of [33,34], we investigate a "variable-community" multi-layer SBM (VC-ML) that allows each node to belong to a distinct community in each layer. Although extension to a degree-corrected form could take the same form as in Karrer's DC-SBM [29], we leave this for future work and instead focus on homogeneous degree-distribution SBMs for each layer. From a generative viewpoint, a set of standard SBMs is sufficient to define this model, but from a recovery stance current algorithms require extension. We explore a set of algorithms that take as input a graph G that is the aggregate of a known number of "layers", H. Each layer h is itself a graph g_h generated from an SBM, with a consistent set of nodes across layers. $e_{ij} \in G$ if $e_{ij} \in g_h$ for any h, but the layer origin of edges is unknown (if it were known, one should simply use current recovery algorithms per layer). For a simple example, see figure 2.1. The output of the algorithm is a likely partition of edges into layers and nodes to communities for each layer.

The likelihood of a given node partition Z and edge partition A will look

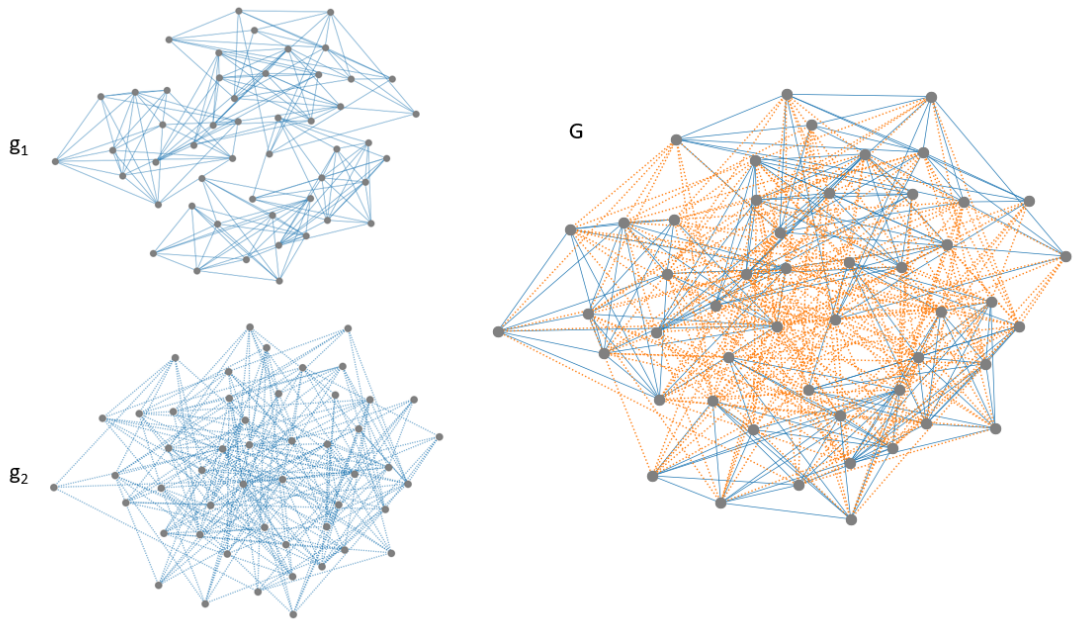

Figure 2.1: Two simple SBMs, g_1 and g_2, on fifty nodes, combine to form G. Edge origins are highlighted (blue for g_1 and orange for g_2). Although there are no overlaps in this case (g_1 and g_2 share communities but are strictly associative and disassociative, respectively) any edge that is shared by layers would appear only once in G.

familiar to the classic SBM, simply extended to consider each layer in turn (note the product over layers):

$$P(G|Z,A) = \prod_{h=0}^{H} \prod_{i}^{N} \prod_{q}^{k} \alpha_{q,h}^{z_{iq}^h} \prod_{i \neq j}^{N} \prod_{l,p}^{k} \left[\pi_{lp,h}^{A_{ij}^h} - (1 - \pi_{lp,h})^{(1-A_{ij}^h)} \right]^{z_{il}^h z_{jp}^h}$$

where $A_{ij}^h = 1$ if $A_{ij} \in G$ and that edge belongs to layer h, and $z_{iq}^h = 1$ if node i belongs to community q in layer h. $\pi_{lp,h}$ and $\alpha_{q,h}$ are the model probabilities for inter-community edges of communities l and p and the community q prior likelihood for layer h.

This is distinctly more difficult than single layer SBM recovery for two reasons.

The first is an enormous increase in search space: rather than k^N partitions, there are now $k^{HN}M^H$, where $M = |E|$ and k is the number of communities per layer. The second is that many previous algorithms made use of greedy or other heuristics to converge to likely partitions quickly, and it is not obvious how (if possible at all) to extend these techniques when it is not certain whether an edge should be considered in one layer or another. To illuminate this, consider an algorithm that searches the space by refining a candidate solution over time. This candidate solution will be a valid partition of edges into one of the layers and nodes to communities at any given time. At each refinement, it may either change the community assignment of a node or the layer assignment of an edge. Suppose it changes the layer assignment of an edge: this should in turn have an impact on the community-affiliation statistics of the layers from and to which it moved. To evaluate the true difference in quality of this swap there may be additional movements needed to optimize the community assignments in each layer. Over a sequence of many edge swaps, it may be difficult to migrate candidate solutions' node partitions away from local maxima even after edge migrations have taken away the support for those maximums.

The most promising approach was an extension of the variational approximation scheme originally proposed by Daudin [26] and used in the alternative multilayer model of Airoldi [33]. In the variational approximation, we represent edge membership and node memberships by a distribution over all possible assignments rather than a discrete choice. In each main iteration of the algorithm, the node distributions are updated iteratively using the conditional expectations of the node membership based on all other nodes' current memberships until numeric conver-

gence:

$$z_{iq}^{h} \propto \alpha_{q,h} \prod_{j \neq i} \prod_{l} \left[\pi_{lq,h}^{A_{ij}^{h}} - (1 - \pi_{lq,h})^{(1 - A_{ij}^{h})} \right]^{z_{jl}^{h}}.$$

Then the edge memberships are set proportional to how likely the edge is by layer according to the current node membership distributions. Finally, current maximum-likelihood estimates for community prior probabilities $\alpha_{q,h}$ and community affiliations $\pi_{lp,h}$ are calculated by simple average node membership and edge presence.

With this technique, we were able to resolve simple planted partitions, see Fig 2.2. However, we note that these results required some tuning of initial conditions, in particular the community affiliations should begin as associative to some degree to find associative structures (for example, set the main diagonal of the estimated matrix π to $> .5$, and the off diagonals $< .25$). This is important because it detracts from a fundamental strength of the SBM approach: there is no inherent bias to any particular type of block structure. We believe it may be necessary to narrow the massive search space in some way, but that this does not kill the usefulness of the algorithm; after all, there are many times we might rightfully assume associative structures (for example, friendship networks). However, if we are going to make such assumptions on structure, that may indicate that other techniques, which rely on such assumptions (such as modularity maximization), are better suited to this task. It may well be that these other techniques are a better starting point to the VC-ML recovery problem, and we leave that as a direction for future work.

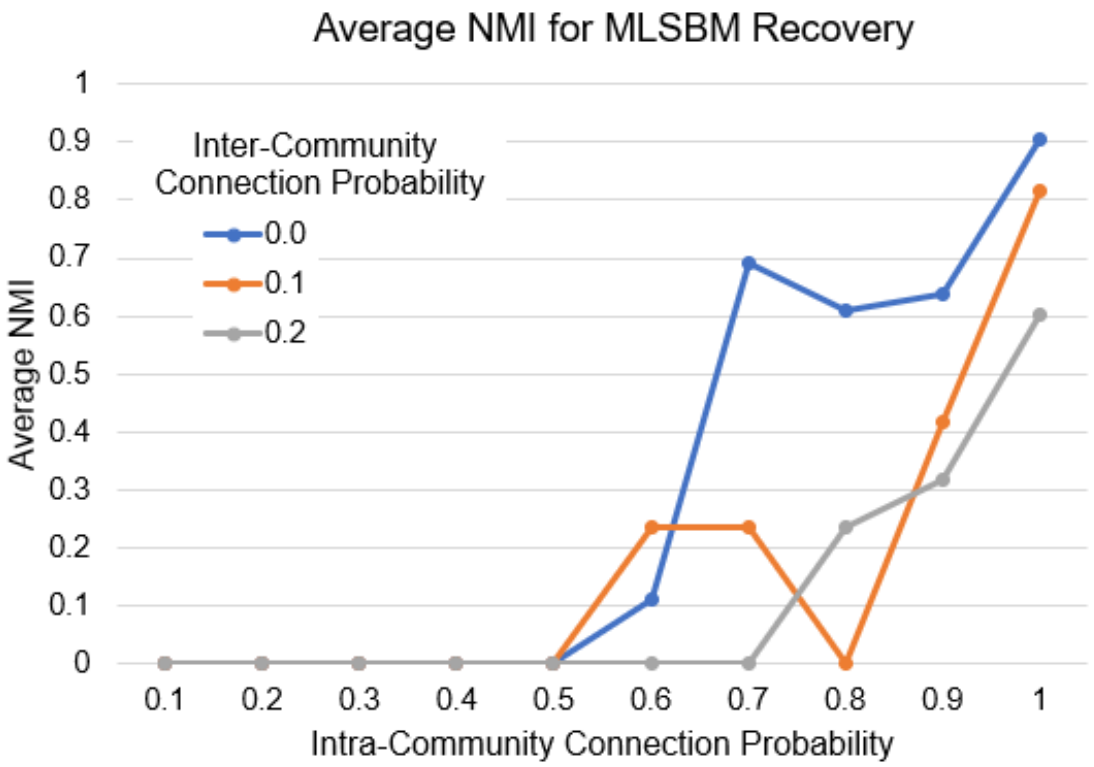

Figure 2.2: Normalized mutual information, averaged over layers, for two-layer VC-ML SBM models of 500 nodes. Each layer has 5 equal sized communities with intra-community connection probabilities varied over the x-axis above. Inter-community connection probabilities were in $\{0.0, 0.1, 0.2\}$ and are shown with separate color lines according to the legend. NMI scores are taken for the most likely partition found over 10 trials.

2.4.1 Less Promising Techniques

In this section we briefly cover a variety of other algorithms we applied to solve the VC-ML recovery problem and their results.

- Metropolis-Hastings Algorithm, an extension of Peixoto's algorithm [31]. To sidestep the lengthy Monte-Carlo Markov-Chain mixing time, an agglomerative heuristic was used in Peixoto's work. It is not clear if a similar technique could be applied to VC-ML, and without such a speed up the mixing times are too long to reach a stable state even for modest graphs (on the order of

hundreds of nodes).

- A greedy algorithm that sweeps through a random ordering of nodes and edges, moving each according to maximum increase in log-likelihood. We observed no positive effect.

- A greedy algorithm that, each iteration, first moves all nodes to maximize the model's log-likelihood, then moves edges according to maximum likelihood based on current model parameter estimates. This does appear to work with some consistency, but it appears to prefer separating the layers into an associative layer and a dissociative layer. In certain test cases we believe this was caused by the inability of the algorithm to apply an edge to two or more layers simultaneously (despite two or more being present among the true contributing layers). This forced the algorithm to choose where to "ration its evidence". Offering the algorithm a choice to simultaneously place any edge among 1 to k layers deteriorated performance. We are of the opinion that offering a multigraph as input (and therefore how many edges there are between any two nodes) is untrue to the problem formulation and the application to real-world networks we envisioned.

- Multiple genetic algorithms, the best of which used a greedy SBM-maximizer for each individual's node membership layers and conducted partition-aware crossover between the layers' node partitions that shared the most mutual information, as well as between the edge partitions. Partition-aware crossover attempts to find a mapping between two partitions such that community as-

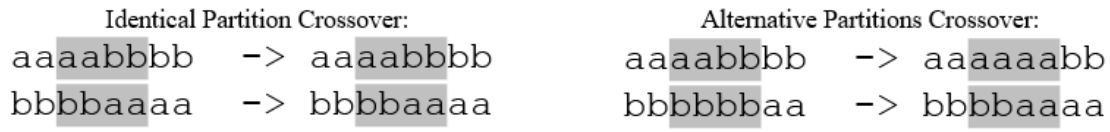

Figure 2.3: Two example crossover events are pictured, one between partitions that are identical but labeled alternatively, and one between partitions with different assignments. The gray highlighting represents the elements chosen for crossover. In the first we desire any crossover event to produce exactly the same children, because the partitions actually agree on every element's community. In the second, we would expect some change in most crossover events. Regardless, we must determine what element A in individual 1 should map to in individual 2 (B), etc. We then use these maps to convert the crossover section before swapping it into the new partitions.

signments that seem to describe similar sets of nodes are respected even though the absolute mapping among partitions may be different. This mapping is then used to convert the data that is crossed over so that the true information in each partition is respected. As an example, see Fig 2.3. We believed this would allow solutions to collaborate and move past local maxima / prevent known stable solutions with mixed-community assignments, but this performed similar to a simple-greedy algorithm while being much more expensive (both in memory and run time).

2.5 Domain Dependent Structures of Real-World Networks

In this section, we explore a set of networks made available by Network Repository [36] to identify predictable and informative trends in the structure of real-world networks. To survey a wide variety of different network types, our survey includes up to 30 networks (where available) across 20 domains, resulting in a total pool of 201 networks. To maintain a feasible computational burden all networks cho-

sen were restricted to less than 100K edges. For consistency (and because not all networks documented their properties completely) all networks were assumed to be undirected. See Appendix A for a complete listing of graphs used. Our approach explores the community structures, degree distributions, and clustering coefficients of these graphs as described in the following subsection.

2.5.1 Methodology: Real-World Structure Analysis

Individual graph analysis consists of the following measurements:

1. Community Partitions:
   (a) Stochastic Block Model (SBM)
   (b) Degree-Corrected SBM (DC-SBM)
   (c) Infomap
2. Partition Modularity
3. Partition Robustness
4. Clustering Coefficients
5. Degree Distribution
   (a) Power law Fit
   (b) Poisson fit
6. Community Degree Distributions
   (a) Power law Fit
   (b) Poisson fit

For community partition detection (metric 1 above), each graph was subjected to the partitioning algorithms listed, each being run 5 times and the highest scoring structure selected. We used the Infomap algorithm in igraph's Python library [37] and the SBM partitioning functions in graph-tool [38]. As discussed in Fortunato's survey [25], simply finding a partition that maximizes some structure detection metric (SBM likelihood, map equation) may be misleading. Even in random graphs,

certain community detection algorithms may report structures with high confidence that happen to enjoy much higher likelihoods or scores than the field of potential solutions. In their paper, [39] Karrer and others suggest investigating any alleged structure's *robustness*, which is a measure of any algorithm's suggested partition changes as the graph is perturbed. The intuition is that a "true" structure should be resilient to small changes in the graph structure, whereas random structures will be volatile and irrecoverable after small changes.

There are two components to the robustness measurement (metric 3): measuring the similarity between two suggested partitions (the outputs of the community-detection algorithm) and a tunable perturbation model. In Karrer's case [39], they use the null configuration model, which randomly samples all graphs that share the same degree distribution. In order to make this tunable (the default null configuration model rewires every edge) they first select a random proportion $\alpha \in [0, 1]$ of edges to remove from the graph. Then a random sample of αN edges, proportional to the product of degrees: $Pr(selecting\, e_{ij}) \propto k_i k_j$, is added to the graph. This ensures that the expected degree of each node $k_i^{'}$ for any perturbed graph remains equal to the degree in the original graph k_i. Palowitch et al. [40] extended the configuration model to weighted graphs, and we extend their model to incorporate an α as in Karrer's study [39]. In this model, we rewire a portion α of edges of a weighted graph and preserve, in expectation, the degree and *strength* of each vertex (strength is the sum of incident edge weights).

To measure the similarity between two partitions, we begin with the Mutual Information Score between two partitions length n, A and B. If n_a is the number of

elements equal to a in A, then we can define the probability that a randomly selected element is a in A: $P(a) = \frac{n_a}{n} \, \forall a \in A$. Similarly for B, we have $P(b) = \frac{n_b}{n} \, \forall b \in B$. Also let n_{ab} be the number of elements that are a in A and b in B, and $P(a,b) = \frac{n_{ab}}{n}$. Then we can define the mutual information I:

$$I(A,B) = \sum_{a \in A, b \in B} P(a,b) \log \frac{P(a,b)}{P(a)P(b)}.$$

The mutual information is zero for uncorrelated partitions, larger for similar partitions, and it's upper limit increases with the size of the partition. Karrer proposes using the Variation of Information to calculate robustness because it is additionally a metric among the space of all partitions [39]:

$$V(A,B) = H(A) + H(B) - 2I(A,B)$$

where $H(A) = -\sum_{a \in A} P(a) \log P(a)$ is the entropy function. $V(A,B) \in [0, \log n]$, and we normalize it in all of our results by dividing by $\log n$ to facilitate comparisons between graphs of different sizes. $V(A,B)$ will be 0 between partitions which are identical and $\log n$ (or 1 for the normalized case) for uncorrelated partitions.

To measure the robustness of a graph structure with respect to a community-detection algorithm, we evaluate the Variation of Information between the original structure and the structure resolved across a range of perturbation values $\alpha \in \{0.05, 0.10, 0.15, 0.20, 0.25, 0.30, 0.35\}$. At each value α the variation is taken between the original structure and the structure detected among 5 realizations of

the perturbed graph. The average of these values is stored for each *alpha*, then we average across all seven values of α for a total variation in information score. For intuitive convenience, we calculate robustness as $1 - average$(Variation across perturbations), so that 1 indicates a completely robust structure (in the explored range) and 0 a completely volatile structure. The α values were chosen to maximize exploration of a useful range; in Karrer's paper [39], a majority of experiments revealed distinct characteristics in this range and perturbing past $\alpha = 0.40$ provided little additional qualitative information. Due to computational expense, we were unable to evaluate robustness for SBM and DC-SBM structures on graphs with over 5K nodes or 20K edges, which reduced the pool to 141 graphs. Robustness results for infomap cover the entire original pool of 201 graphs.

For each graph as a whole (metric 5), and then for each community subgraph within (metric 6, communities defined for each partitioning algorithm according to the structures resolved), we analyzed the degree distribution and the goodness-of-fit of a Poisson distribution (which describes Erdös-Rényi random graphs) and a power-law distribution (which describes scale-free graphs, including Barabási-Albert graphs). The procedures to fit and test goodness-of-fit for a proposed power-law distribution are explained in detail in section 4.13 of Clauset's paper [41], and the goodness-of-fit of a Poisson distribution is accomplished with a Chi-Squared test. We considered power laws with a goodness-of-fit p-value ≥ 0.1 to be a good fit [41], and used 500 synthetic data sets giving our p-value an error $\epsilon \approx .022$. For the Poisson distribution we considered a p-value ≥ 0.05 to be a good fit.

Modularity (metric 2) and global clustering coefficient (metric 4) were calcu-

lated in the standard fashion.

After gathering the above characteristics for all graphs, we employ a k-means algorithm as a simple way to cluster the graphs based on their characteristics. Because many of the graphs provided to Network Repository [36] may share a single source and/or particular methodology, it would be natural for groups as labeled according to Network Repository to share characteristics and group together. Therefore, if some alternative labeling still correlated with the statistical grouping of the k-means algorithm that might indicate a general trend. We explore this in the following results section.

2.5.2 Experimental Results

For all results, we label the graphs according to one of two keys: the Network Repository source labels or a group of "meta" labels: Biological, Social, Infrastructure, Other. Fig 2.4 shows the number of networks in each category and the color mapping that is used in all subsequent figures. Note the secondary color column of the legend maps the Network Repository labels to the meta label to which it belongs. Most labels should be self explanatory except "Labeled", which is not a type of graph in particular but a collection of interesting graphs where users had also submitted some type of edge or node labels (not used in this thesis). A complete list of all graph titles is included in Appendix A, as well as additional source information for the graph data where available (Appendix A.

We first observe summaries of simple characteristics across our graph popu-

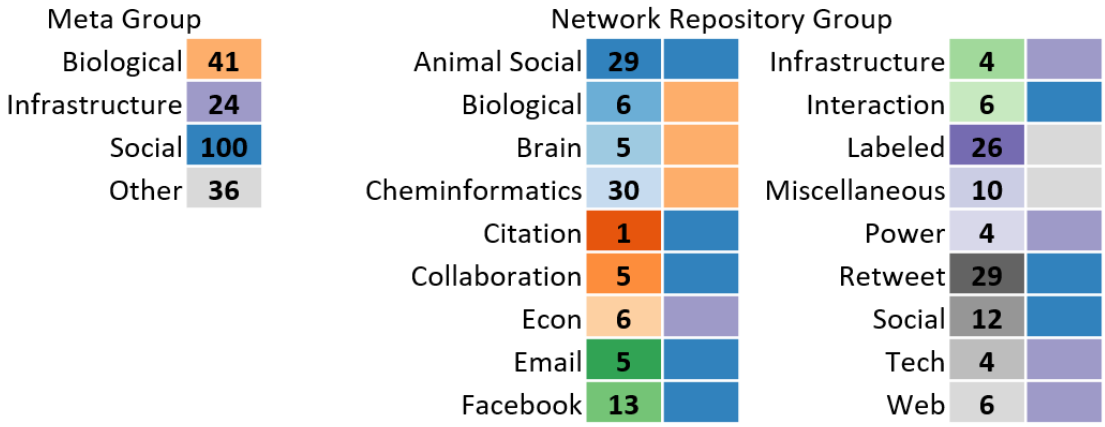

Figure 2.4: Legend for all experimental figures on real-world networks. Number of networks for each category is in bold to the left of the category. Note meta color for Network Repository labels is noted in secondary column.

lation. For most results, we will present a pair of charts that are color keyed by Network Repository label and meta type. Fig 2.5 shows the distribution of average degree and clustering coefficients across network sizes. There is considerable diversity in the population even among these basic characteristics. As mentioned earlier, the network repository labels have a tendency to identify self-similar graphs, but the clustering coefficient meta chart demonstrates that generalities may be drawn. For example, it appears that most social graphs tend to have a low clustering coefficient, whereas the infrastructure and biological graphs exercise considerable variance in this regard.

Only 8 graphs' degree distributions are well described by by a Poisson distribution, 7 of which are animal social graphs (the other an enzyme relationship graph), and all below 40 nodes. 29 networks are well described by a power-law degree distribution, with sizes from 34 to 40K nodes 2.6. In particular, the retweet networks happen to fit with high probability (14/29). This may reflect a fairly strict

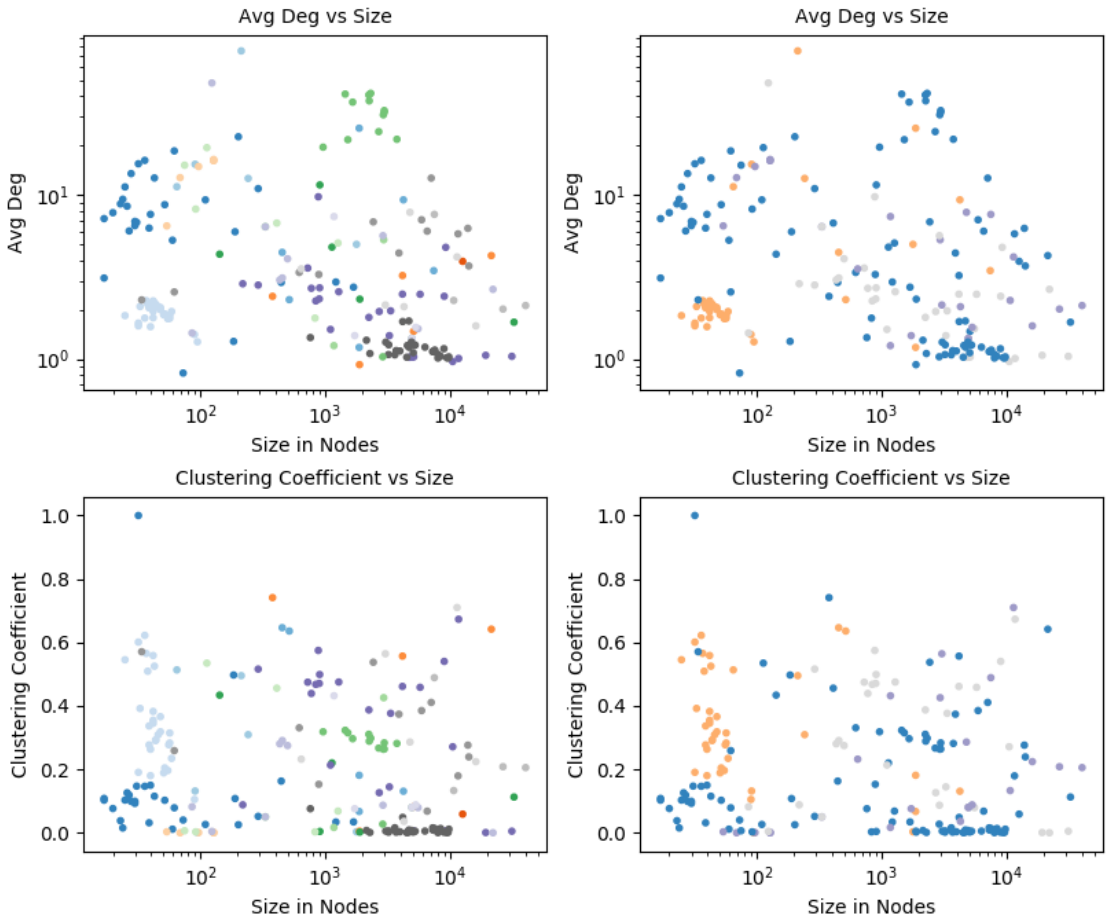

Figure 2.5: Average degree and clustering coefficients for all graphs in the set.

and predictable process of that network, where highly shared tweets tend to pass through the network quickly and people who have many followers and have already been retweeted many times are much more likely to be seen and retweeted again within the lifetime of the process. This "first-mover advantage" is a structural principle that guides the construction of Barabási-Albert graphs and statistically yields power-law degree distributions with high probability. This still leaves a vast majority of networks (164/201) that are not well described by a simple degree distribution.

Next we show the average community size per graph and modularity of partitions, for each graph and each partitioning algorithm. Stochastic block model

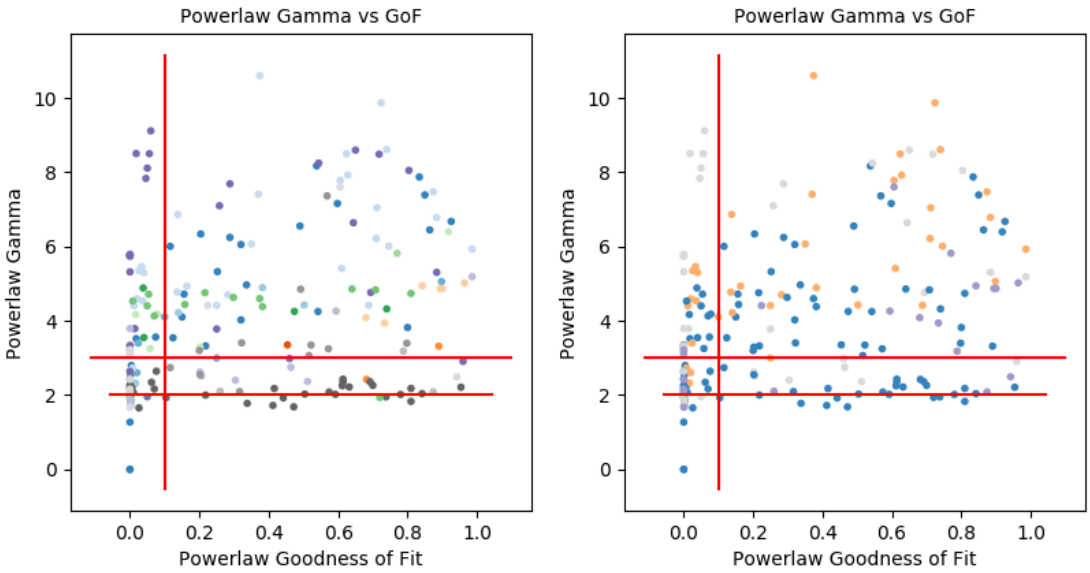

Figure 2.6: Power-law fits plotted for each graph, with the goodness-of-fit on the x-axis and fitted γ on the y-axis. We consider only $\gamma \in [2, 3]$, as below 2 is not valid in the limit as the network grows to infinity, and above 3 the network is essentially in the random regime.

results are represented by the circular markers, degree-corrected SBM results by stars, and infomap results by the down triangular markers in Fig 2.7. We note no obvious trends for any type of graph appear, but by inspection there is a positive correlation between infomap and block model partition modularities. Additionally, the average community size grows very slowly as a function of graph size (note log scale on x-axis). This relative indifference to total graph size may be of use in certain heuristics that leverage partitioning algorithms, and is discussed further in Chapter 4.

We gauge the value of the partitions found with a view of the robustness over modularity and graph size (in nodes) in Figs 2.10, 2.11. Although there is no objective cut off for a significant partition, we may subjectively determine communities with a high modularity and robustness and significant. Interestingly, infomap

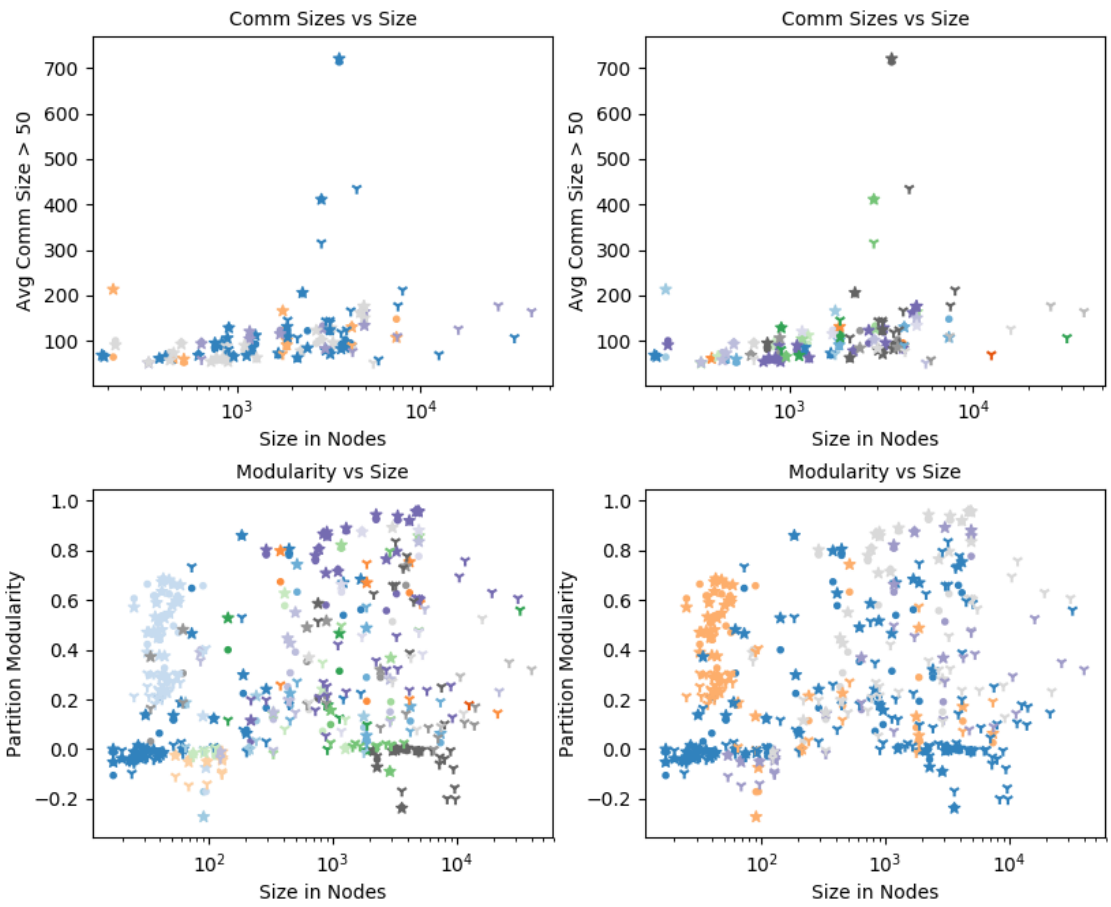

Figure 2.7: Average community size and modularity of community partitions for all graphs in the set.

appears to produce more robust partitions the larger the structure is, while SBM structures are agnostic or slightly disfavor large graphs. Social networks are frequently non-modular and variably robust, which suggests they have the least obvious structures or are least likely to have significant community structures. This result is surprising because typically we develop "community detection" algorithms deliberately for the social context.

Another way to measure the significance of a partition is to evaluate its utility in the context of dynamic processes on the graph. We go into the detail of hybrid algorithms in Section 4.2, but for now it is sufficient to understand that the partition

provided is used to simplify the search space of the algorithm. For all real-world networks with greater than five hundred nodes, we measured the maximize influence of seed sets selected using hybrid algorithms under the Linear Threshold and Saturated Linear Threshold processes. As comparison, for all graphs we evaluated the spread of the top-K degree nodes, and if the graph was small enough (less than 5000 nodes) a set selected by the simple greedy algorithm. K, the seed set size, was set to 1% of the nodes, or 30, whichever was less.

The relative performance of the hybrid algorithms are plotted against the robustness and modularity of the partitions, for SBM and Infomap partitions respectively. Relative performance is measured as:

$$Relative\_Performance(Hybrid, Alternate) = \frac{[Spread(Hybrid) - Spread(Alternate)]}{N},$$

where N is the size of the graph in nodes. In Figs 2.8, 2.9, relative performance versus Top-K is blue and versus Greedy is orange. Simple linear regressions show low R^2 values, but some positive correlation in robustness and hybrid algorithm performance, particularly for Infomap hybrid results. This indicates that robustness is capturing some useful information about the quality of structures found in the network. It is less clear that a relationship exists between modularity and hybrid algorithm performance. This may be for many reasons, but at a minimum we know that a highly modular partition does not necessarily yield a useful simplification of graph structure in relation to dynamic processes.

Figs 2.12, 2.13 shows a histogram of graphs binned by the fraction of their

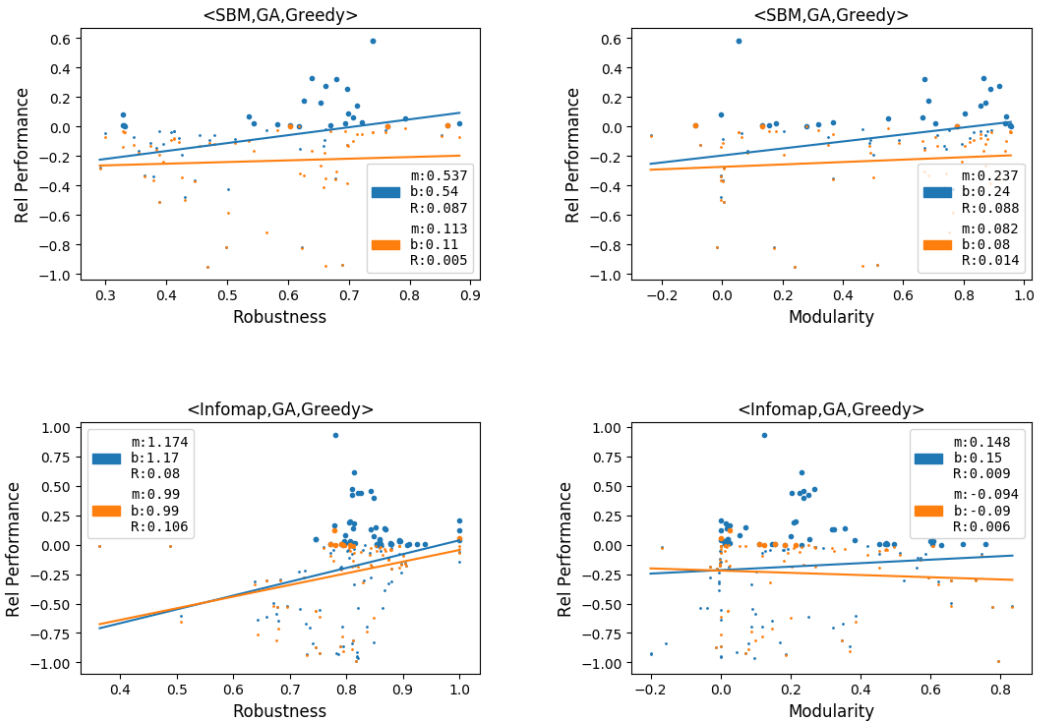

Figure 2.8: Linear Threshold: Relative performance of hybrid algorithms, using SBM and Infomap partitions, versus Top-K Degree (blue) and Greedy (orange) selections. Results normalized by graph size, and larger data points correspond to results where the hybrid algorithm outperformed the given alternate algorithm.

communities that fit a Poisson or power-law distribution, for each partitioning algorithm. For example, the green bar at 0.0 on the x-axis for the power-law distribution indicates 22 graphs, when partitioned by infomap, had $< .02$ subgraph communities that were well fit by a power-law distribution. Fig 2.14 offers a histogram of the γ fits to all of the intragraph communities. Notably many of the fits are outside of the acceptable range, and a majority of those that are in $[2, 3]$ are produced by a degree-corrected SBM recovery. Fig 2.13 indicates the preferences of the partitioning algorithms: infomap is much less likely to produce partitions with power-law distribution than the SBM models, but interestingly if it does produce such a subgraph it

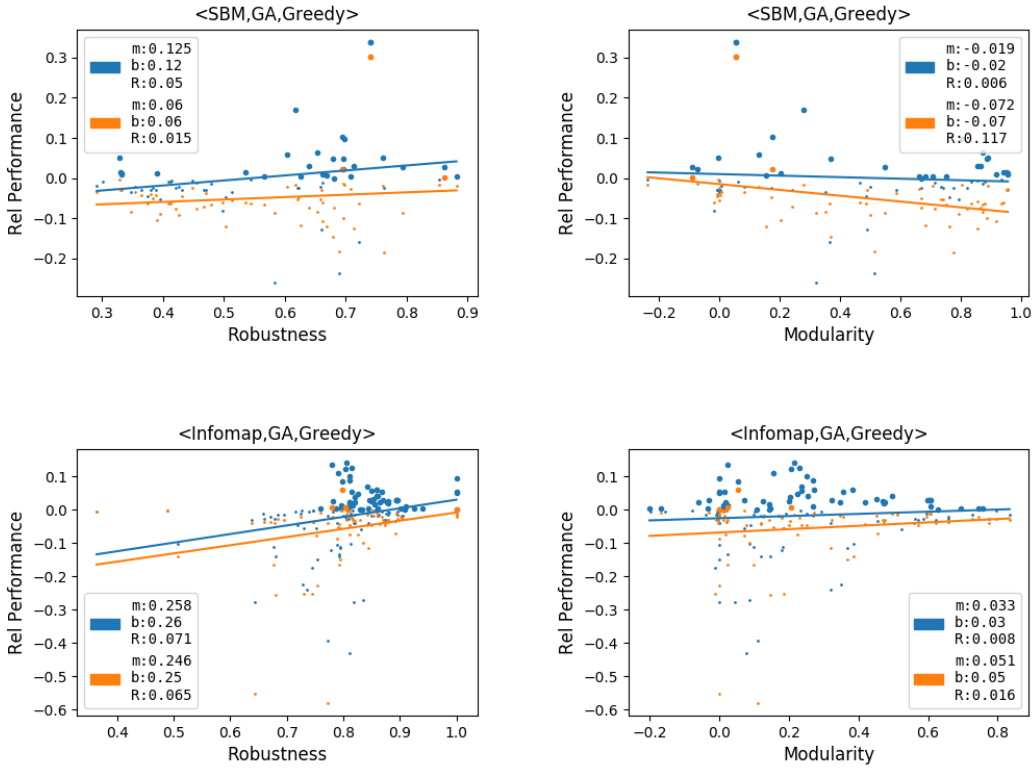

Figure 2.9: Saturated Linear Threshold: Relative performance of hybrid algorithms, using SBM and Infomap partitions, versus Top-K Degree (blue) and Greedy (orange) selections. Results normalized by graph size, and larger data points correspond to results where the hybrid algorithm outperformed the given alternate algorithm.

is much more likely to be a valid fit. None of the partitioning algorithms were likely to find subgraphs with Poisson degree distributions, which informs us that even the structural and dynamic *components* of real-world graphs are not well-approximated by the Erdős-Rényi random graph. These results may be relevant in the context of designing alternative hybrid approximation algorithms, which may perform best on certain graph substructures. If one has a subroutine that performs well on scale-free graphs, one would do well to choose a degree-corrected SBM to partition your graph first.

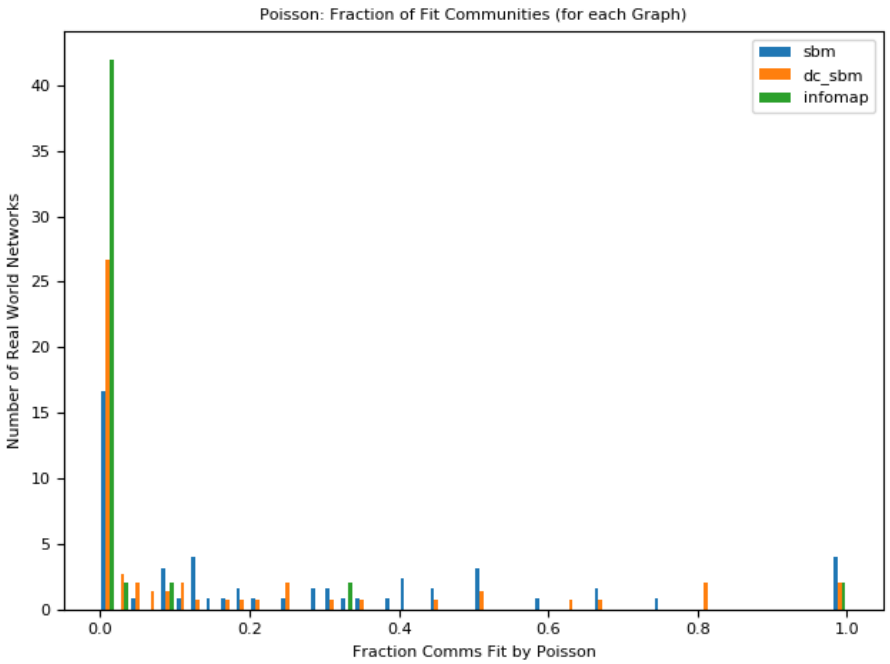

Figure 2.12: Histograms of all graphs, binned by the fraction of their subgraph communities that were fit by Poisson degree distributions.

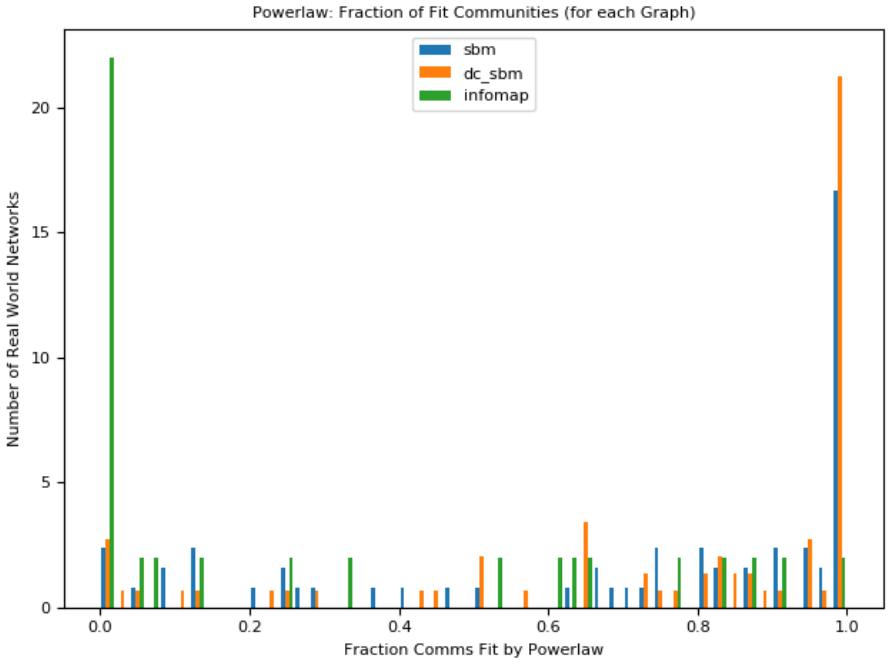

Figure 2.13: Histograms of all graphs, binned by the fraction of their subgraph communities that were fit by power-law degree distributions.

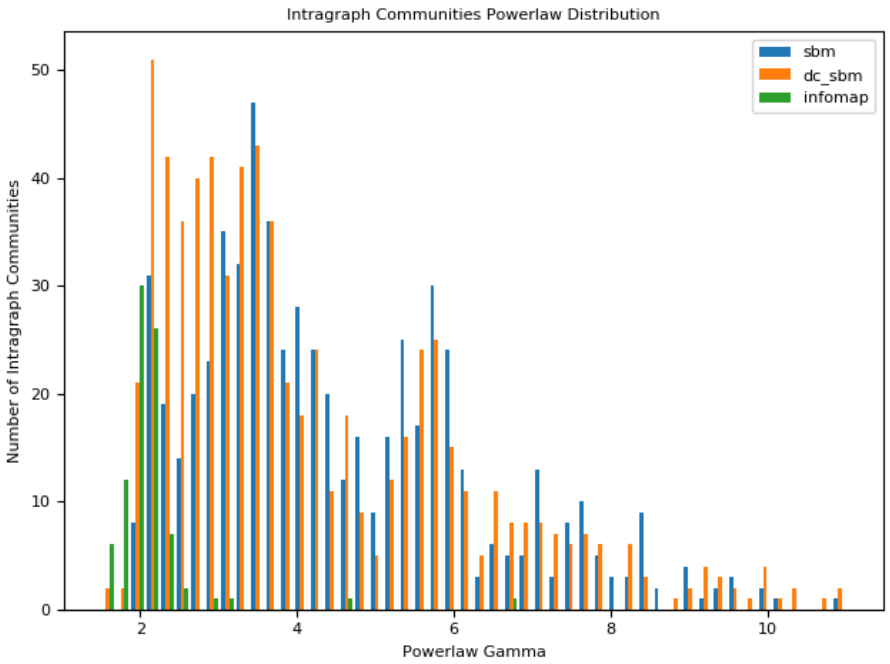

Figure 2.14: A histogram of the γ fits for all of the subgraph communities, for each partitioning algorithm.

Finally, we clustered all graphs using K-means clustering, $k \in [2, 14]$, using as attributes the graph's: {infomap modularity, infomap robustness, poisson fit (Boolean), power-law fit (gamma or 0 if no fit), global clustering coefficient, average degree, average comm size}. We normalized all attributes to a mean of zero and standard deviation of one before clustering. To identify a good candidate k value, we observed the total distance of all graphs from their assigned cluster centroids and the mutual information score between the assigned K-means partition and the Network Repository groups partition. We hypothesized that a meaningful clustering would exhibit low total mean distance and some agreement with the Network Repository labels.

Fig 2.15 shows total mean distance as well as cluster's normalized mutual information as a function of k. The normalized mutual information of a random

k-group partition and the Network Repository labeling is also shown in green as a control. After a handful of runs, it appears that a 6-8 group clustering or 10-12 group clustering is most valid. Fig 2.16 shows a visualization of the clusters by their Network Repository groups and meta labels. Edge lengths were set proportional to distance between the centroids, so the visualization offers some relative information on how different clusters are (but being a 2-D representation of 7-dimensional space it is by necessity a distorted projection).

Fig 2.16 shows that many of the clusters are determined by Network Repository labels with a strong identity, for example the 15 group cluster that is almost entirely Facebook graphs. It is also interesting to note that although there are no strict guidelines for a majority of the graphs, there are regions of these network visualizations that correspond to the meta labels (e.g. most of the biological networks are in the top left quadrant).

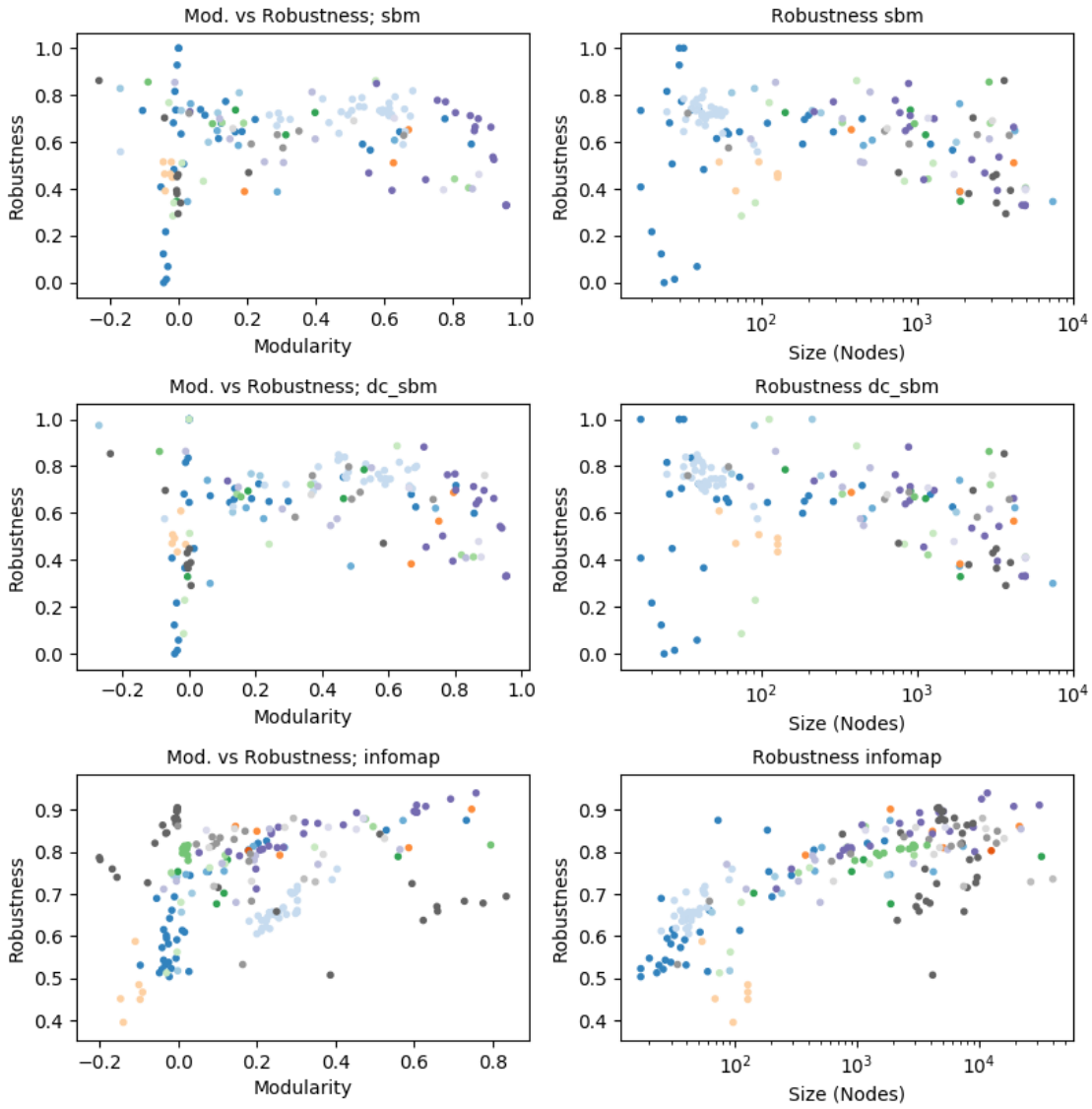

Figure 2.10: Modularity and Robustness of all graphs, for each partitioning algorithm.

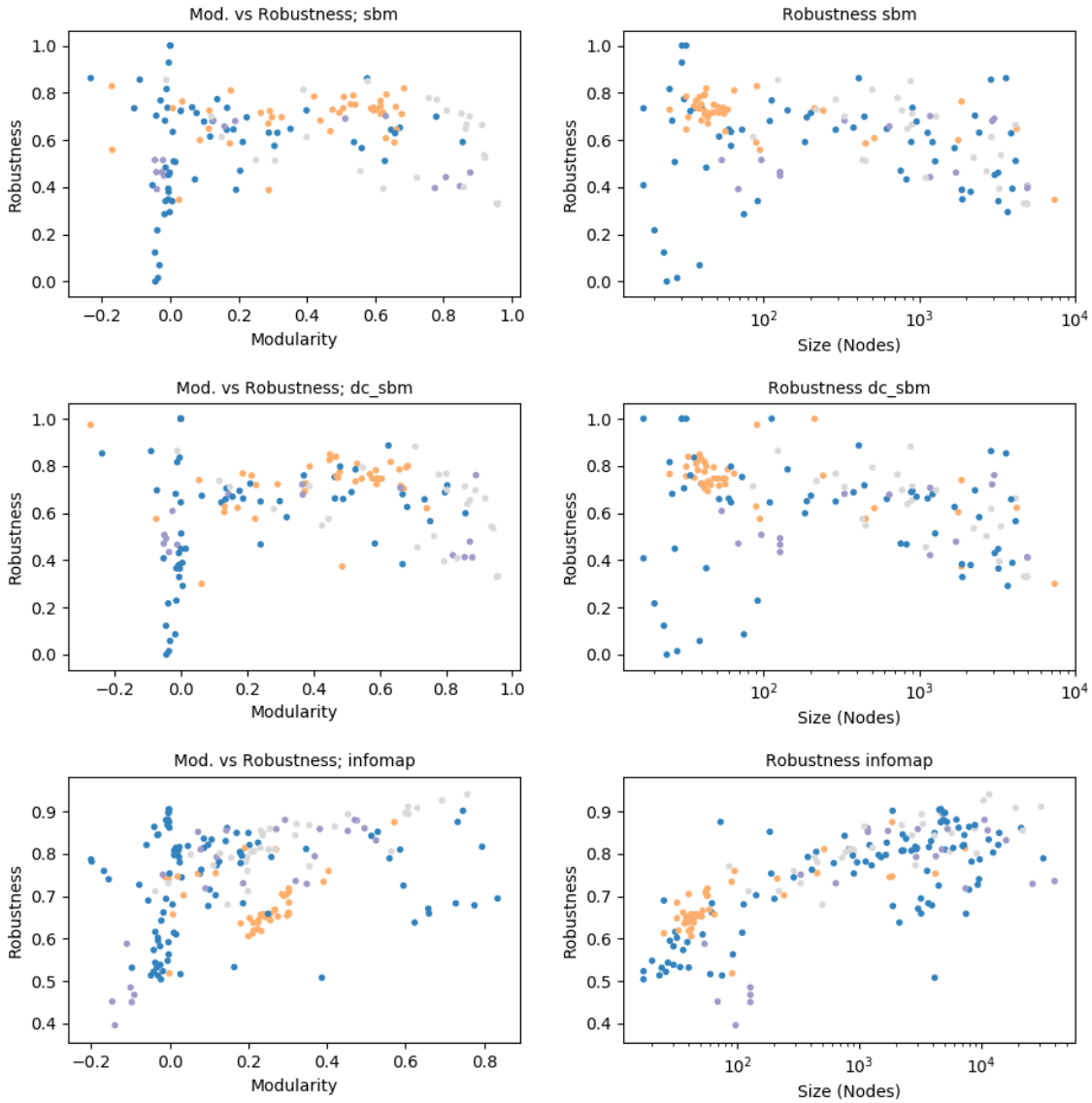

Figure 2.11: Modularity and Robustness of all graphs, for each partitioning algorithm.

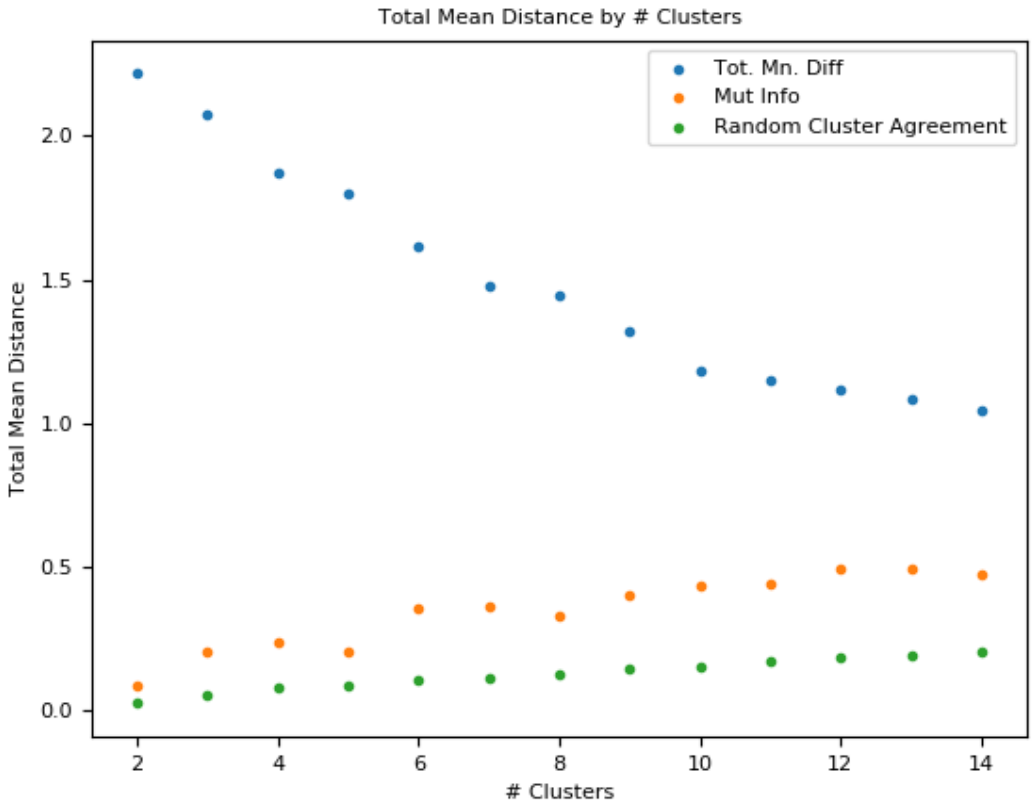

Figure 2.15: Total mean distance and normalized mutual information of a k-clustering, by value of k in $[2, 14]$. In this run, it appears that 7 clusters is a good candidate.

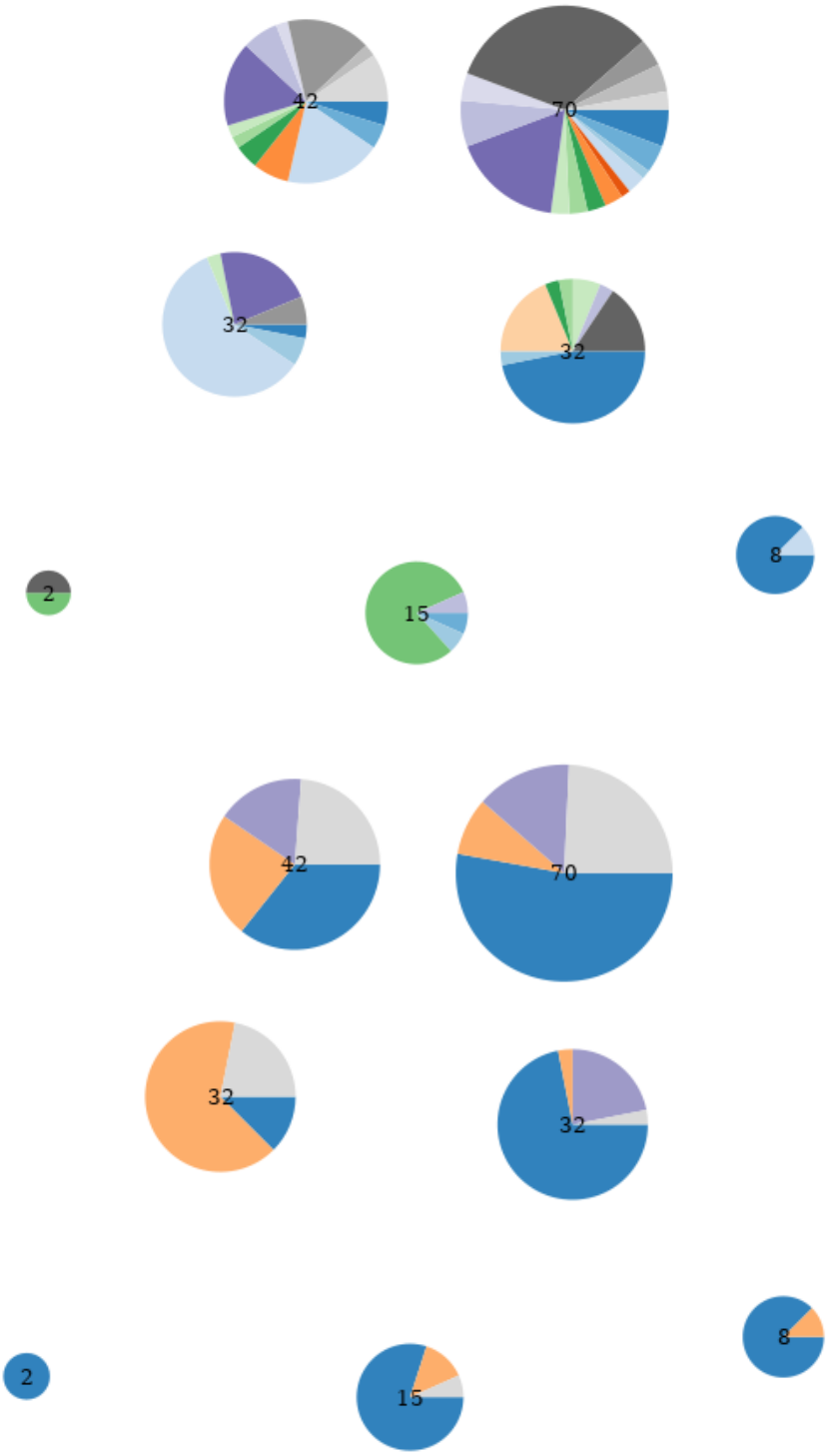

Figure 2.16: Visualizations of the 7-Cluster assignments, where each pie chart is a k-cluster and the pie chart reveals the Network Repository labels and meta labels, respectively. Clusters that are visualized farther apart have relatively more distant centroids, and the number of graphs belonging to the cluster are noted in the pie chart centers.

Chapter 3: Genetic Algorithms for Maximizing Influence

In preparation for use of genetic algorithms in Chapter 4, we first analyze the sensitivity of performance to a variety of settings and techniques. Genetic algorithms are a combinatorial optimization method that leverage a simulated evolution of a population of candidate solutions, using random mutations and fitness dependent reproduction, to search the solution space. For a primer on genetic algorithms, we offer Holland's introduction [42] and for an in depth exploration of genetic algorithms used for network optimization problems Gen, Cheng, and Lin's book [43]. Lim provides a more recent survey which covers structured-population variants and additional refinements to improve performance [44].

The classic genetic algorithm's summary is presented in Algorithm 1. Even with this simple pseudocode, we've already made some design decisions (only children are mutated, for example), while trying to be as general as possible. Exactly how we implement functionality for **parent_selection**, **crossover**, **mutation**, and **selection** is a source of considerable research in the evolutionary algorithms community. Additionally, we must choose a way to represent our solutions to the problem at hand (that is, how do we encode a solution as an individual in our population) and then decide how we score its fitness. Suppose we get all these things tuned well

— we must then still decide how often to mutate individuals, how often to conduct crossover, how big a population to simulate, etc. This chapter explores a wide breadth of these possibilities in the context of influence-maximization problems on graphs. We find that genetic algorithms are relatively insensitive to a handful of domain specific optimizations, and that mutation rate and population size are the most critical parameters for solution quality.

Algorithm 1: Top-level Genetic Algorithm

```
INPUT: pop_size, generations, num_elites ∈ N; PR_M, PR_C ∈ [0, 1]
population ← [pop_size * individuals]
for t < generations do
  for i in population do
    fitness[i] ← fitness(i)
  end for
  next_generation ← [ ]
  elites ← top_fitness(population, num_elites)
  next_generation.append(elites)
  while length(next_generation) < pop_size do
    parents ← parent_selection(population)
    children ← crossover(parents)
    children ← mutation(children)
    next_generation.append(selection(children + parents))
  end while
  population ← next_generation
end for
OUTPUT best_individual(population)
```

3.1 Representation and Genetic Operator Evaluation

3.1.1 Representation

First we cover a range of likely candidates for solution representation, mutation, and crossover for the influence-maximization problem. We consider two ways

to encode solutions to the IM problem, which we label **K-List** and **N-List**. Given an **IM** problem of seed set size k, a **K-List** is a list of exactly k elements in $[0, N-1]$, where N is the number of nodes in the graph. Fitness is simply the average *additional* nodes infected if the nodes indicated by the **K-List** are used as a seed set for the process of the IM problem. Mutation and crossover operators on **K-Lists** are always swap operations, resulting in **K-Lists** themselves.

A **N-List** is a list of $[0, N]$ elements in $[0, N-1]$ — any size solution is allowed during the evolutionary process. Of course, we eventually want only solutions of size k, so the fitness evaluation of **N-List** individuals penalizes oversize solutions. Critically, this penalty increases over time but starts very light, which allows the algorithm to explore a broad range of nodes and then leverage this knowledge as it pares its solutions down to size k. Mutation and crossover operations on **N-Lists** result in potentially different length children than the original solutions.

3.1.2 Mutation

Genetic algorithms are interesting partly because of how well they work on a variety of applications without application specific knowledge. Randomly mutating a bit or gene in an individual is typically sufficient variation to yield useful solutions. However, in certain applications it may be advantageous to leverage problem knowledge to focus the mutation operator. We compare **Random (R)** mutation with **Neighbor (N)** and **Distance (D)** mutation. **N** mutation will select a new node from the neighbors of the nodes in the set, while **D** mutation, inspired by the

work of Zhang et al. [45], selects from the furthest half of nodes, in terms of hops. One or the other may be preferred based on the type of process for the IM problem.

Each method above is implemented with unbiased selection of nodes (given they meet the initial selection criteria, i.e. are a neighbor for **N**), and a weighted selection chance based on node degree. For example, in **D-W**, first a pool of nodes that are distant from a selected node are chosen. Then one node is selected from this pool, where the chance of selecting a given node from this pool is proportionate to its degree.

3.1.3 Crossover

Crossover is any operator that takes two individual solutions and results in two modified solutions, which each contain components from both original solutions. It is desirable that the crossover operation exhibits *respect*: the information passed on should represent the same information in solution space for the child as it did for the parent. Because our solutions are encoded using a consistent integer labeling of nodes, this is almost guaranteed, as any solution will interpret a set of integers the same way: use these nodes in the seed set. We should additionally ensure crossover is conducted only on portions of the individuals that are different. Otherwise, we may have awkward situations where one child gets a component solution from parent A that is very similar or identical to the component solution from parent B, and its solution would be redundant in total (or no longer a **K-List**).

We explore three crossover variants, **Single/Double Pt (SP/DP)**, **Network**

Mask (NM), and **Degree Cooperation (DC)**. **SP** and **DP** point are for **K-List** and **N-List**, respectively, and are the most standard crossover operations: choose a location in the individual solutions, and swap everything after. **DP** selects a swap after point for each individual, since the **N-List** parents may have different lengths. **NM** constructs a set of X nodes starting from a random node in one of the parent individuals, using a breadth first search (BFS). X is a random integer in $[0, N]$ and is analogous to the crossover point in **SP/DP**. All nodes covered by the BFS are swapped to one child, and the rest to the other (for each parent). For **K-Lists**, **NM** must conduct two BFS and select the same amount of material to crossover from each parent's BFS to maintain solution size. **NM** is designed to respect solution components that may be interacting locally in complex ways: it is more likely to take a regionally coherent sub-solution to incorporate in a new solution this way. **DC** is a simplified version of Zhang's Elite Cooperation Heuristic [45], and conducts an ordering of the parents according to node degree before swapping. One parent is ordered in descending degree, the other ascending, and then the crossover occurs as in **SP/DP**. This creates one child with a larger proportion of high degree nodes, ideally resulting in a more favorable new solution.

3.1.4 Representation and Genetic Operators Experiment

To study the impact of these operator variants, we conducted IM trials on each combination of the above operators (e.g. **K-List, R-W, DC**) for a total of 36 unique genetic algorithm combinations. Each GA variant was applied to solve a set

of IM problems as follows:

- Independent Cascade (probability of infection uniformly $c = .1, .25, .4$)
- Linear Threshold (threshold uniformly $T = .2, .3, .4$)

Seed sets were size $k = 2, 3, 5, 10, 15$ on a set of five small (500 nodes with one exception) graphs as follows (see Fig. 3.1). This results in 150 trials per GA variant, for a total of 5400 trials. The genetic-algorithm parameters were kept constant for all trials with a population of 100 individuals evolved over 100 generations, probability of mutation equal to 1, probability of crossover equal to .75, 2-parent tournament selection, and one simulated process evaluation to determine fitness. For mutation, each **K-List** swapped one gene / node each iteration, and **N-Lists** had a slightly more complicated mutation that would preferentially mutate towards a list of length k, potentially losing more than one node in length if it was greater than k. Tournament selection randomly samples the population for n (2, in this case) individuals and selects the one with higher fitness. Typically we would conduct more simulations to evaluate fitness for IC problems, but since these were all relative results we were able to increase the breadth of trials at the cost of higher variance in results. Additionally, strong solutions and their descendants will eventually undergo many more than one process evaluation as they begin to dominate the population. Since the fixed threshold LT is deterministic, one trial is all that is needed for fitness evaluation.

Our results indicate that the specific genetic operators do not have a significant impact on the fitness of the proposed solution, but the representation heavily favors

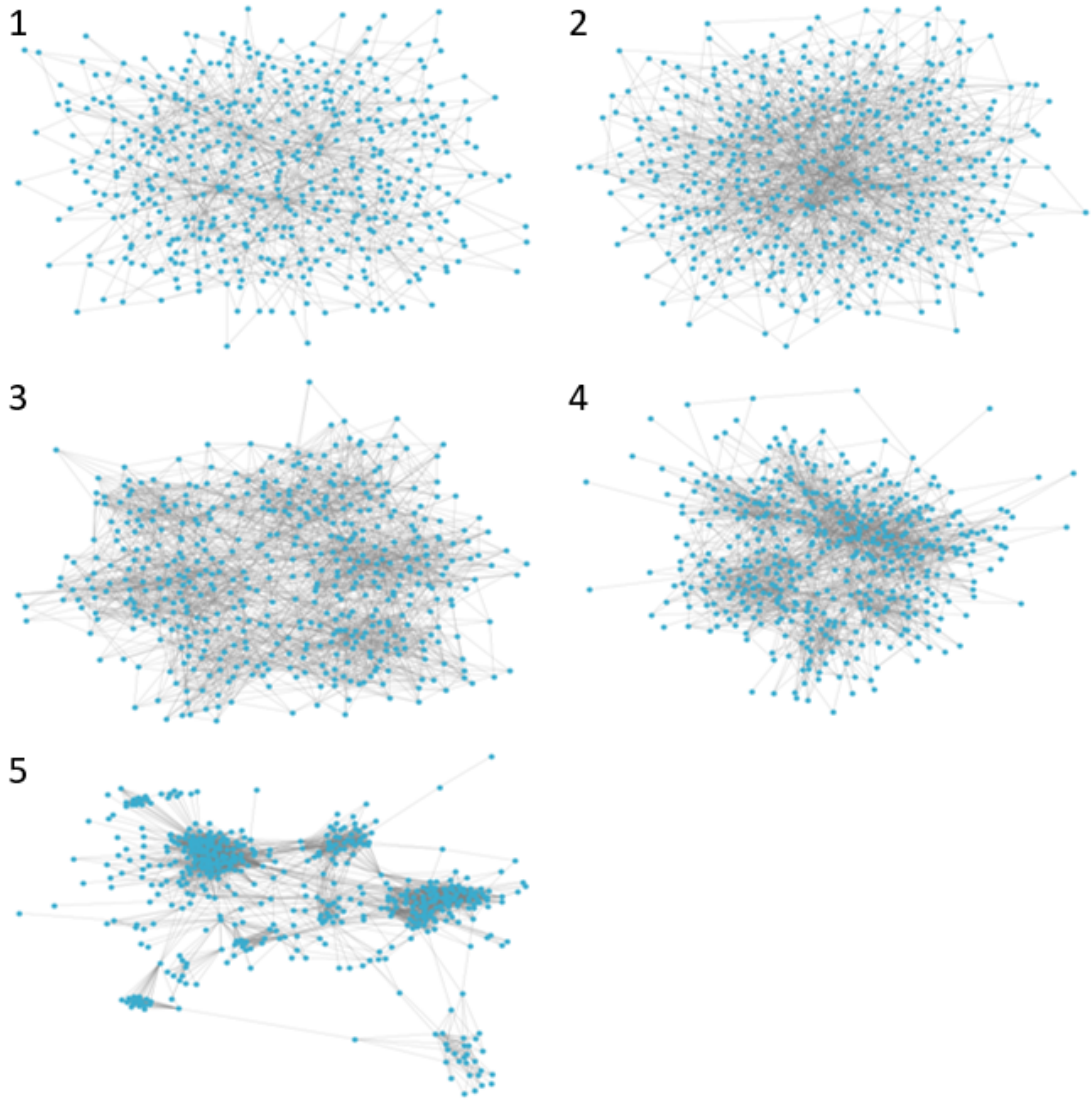

Figure 3.1: 1,2: Barabási-Albert Graphs, m={2,3}. 3: Stochastic Block Model with 10 communities and expected intra-community degree 8 and expected inter-community degree 2 (total across all other communities). 4: Degree-Corrected SBM with the same parameters as the SBM and degree sequence drawn from a power-law with $\gamma = 2$. 5: 532 node subgraph of the Ego-Facebook graph available from SNAP [46].

K-List. In most trials the **N-List** representation seems to do a poor job paring down large solutions to return a size k solution. It is possible that given a much larger search space (that is, a larger graph), the proposed advantage of **N-List** (sampling more of the nodes early on) would produce some benefit. However, building a

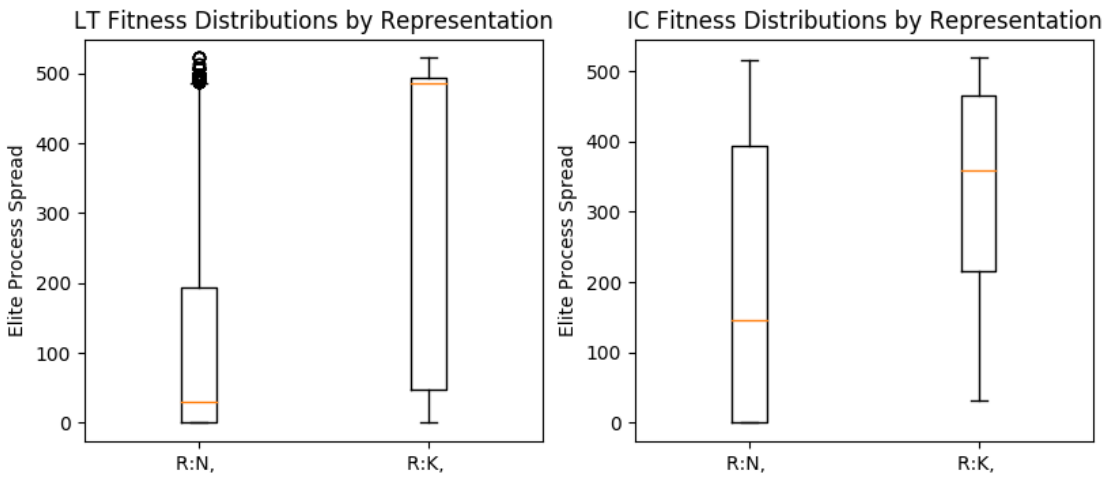

Figure 3.2: Box and whisker plots for all solutions' spread for **K-List** and **N-List** representations.

fitness function that would pressure the population into size k lists efficiently and predictably was time consuming and frequently produced poor solutions. We would only suggest using **N-List** as a second attempt or if the mutation and crossover operators were worked to intelligently pare solutions down over time. While the results presented in Fig.s 3.2, 3.3, 3.4, and 3.5 are averaged over all trials, these observations were consistent even when we sliced the results into finer subsections of all trials.

Although solution quality was not dependent on the operators, it did appear that neighbor mutation had a positive effect on time to find a high quality solution, see Fig. 3.6.

3.2 Parameter Settings and Parent Selection

For the remainder of the experiments, all genetic algorithms will use K-List representation, random + unweighted mutation and single point crossover. We next

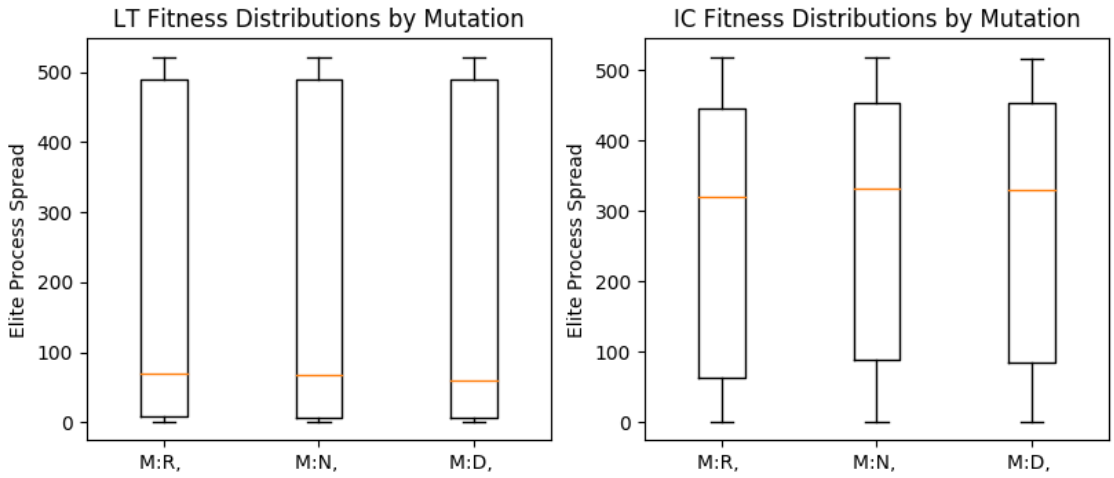

Figure 3.3: Box and whisker plots for all solutions' spread for each mutation type.

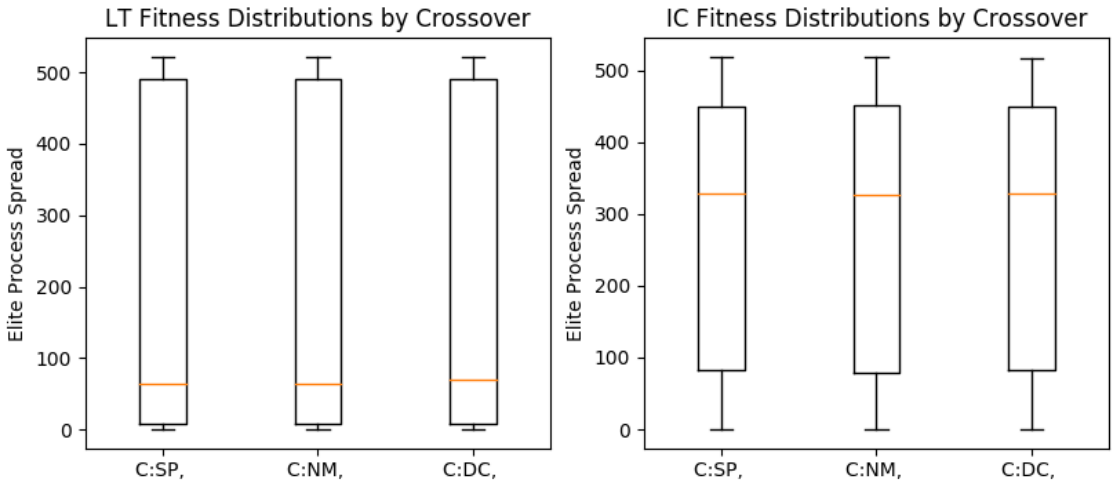

Figure 3.4: Box and whisker plots for all solutions' spread for each crossover type.

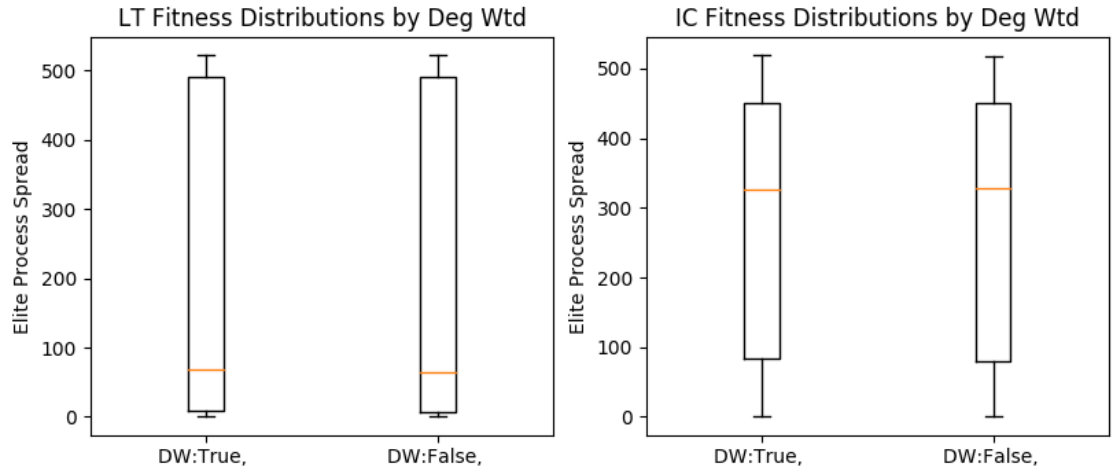

Figure 3.5: Box and whisker plots for all solutions' spread for degree weighted or unbiased mutation.

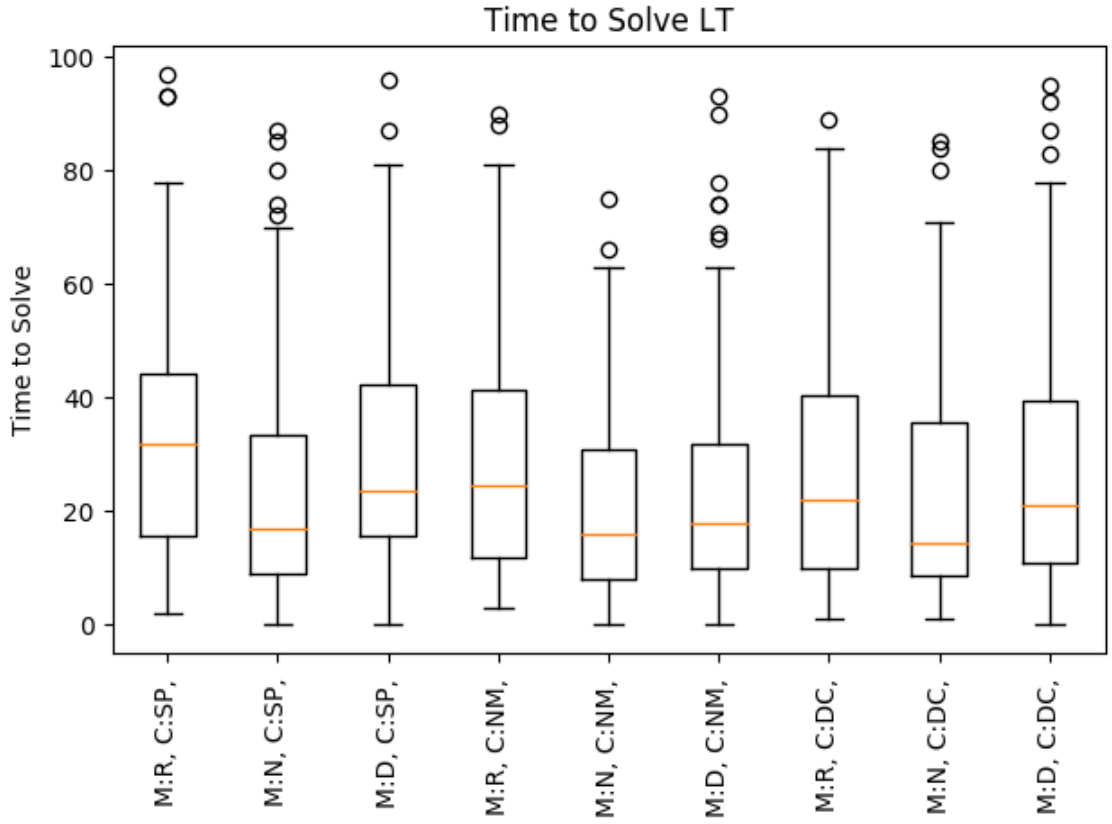

Figure 3.6: Box and whisker plots for all LT solutions' time (in generations) to find a solution 90% as strong as it's final individual over mutation and crossover pairings.

examine changing the parameters to the GA, starting with probability of mutation and crossover. We chose IC with $c = .1$ and LT with $t = .4$ because these seemed to most differentiate good solutions from bad in the first experiment. In the IC trials, the GA estimated individual fitness with 10 IC process simulations. We created a static test pool by generating 5 DC-SBMs (same parameters as before), then averaged the best solution's spread over all test pool graphs at $k \in 5, 10$ for IC, $k \in 10, 15$ for LT, and $k \in 15, 25$ for a large graph LT trial. For the LT large graph trials, we generated the DC-SBM with 2000 nodes and 40 communities. The DC-SBM was chosen as other graphs seemed not to distinguish good solutions and bad as well in the previous experiment. As Fig. 3.7 indicates, best solution value typically increases with mutation rate for LT trials, while crossover has a minor effect if any. These results are consistent with a similar experiment highlighted in Fig. 5 of Zhang's paper [45], which finds that as long as the mutation rate is not very low, the GA should find reasonable solutions.

The IC trials have less clear interpretations: it seems there are many permissible pairings and no outstanding trend. In all of the IC results the average fitness of the population quickly converged (within 10 generations the average fitness was .90 of the elite fitness), perhaps revealing that IC problems at this scale are too small to illustrate impact of parameter decisions.

Next we compare fitness proportionate selection versus tournament selection (default in all other experiments). Fitness proportionate selection is an alternative parent selection method that selects from the population with a probability weighted by fitness of individuals. Fig. 3.8 shows the average fitness over time of the GA

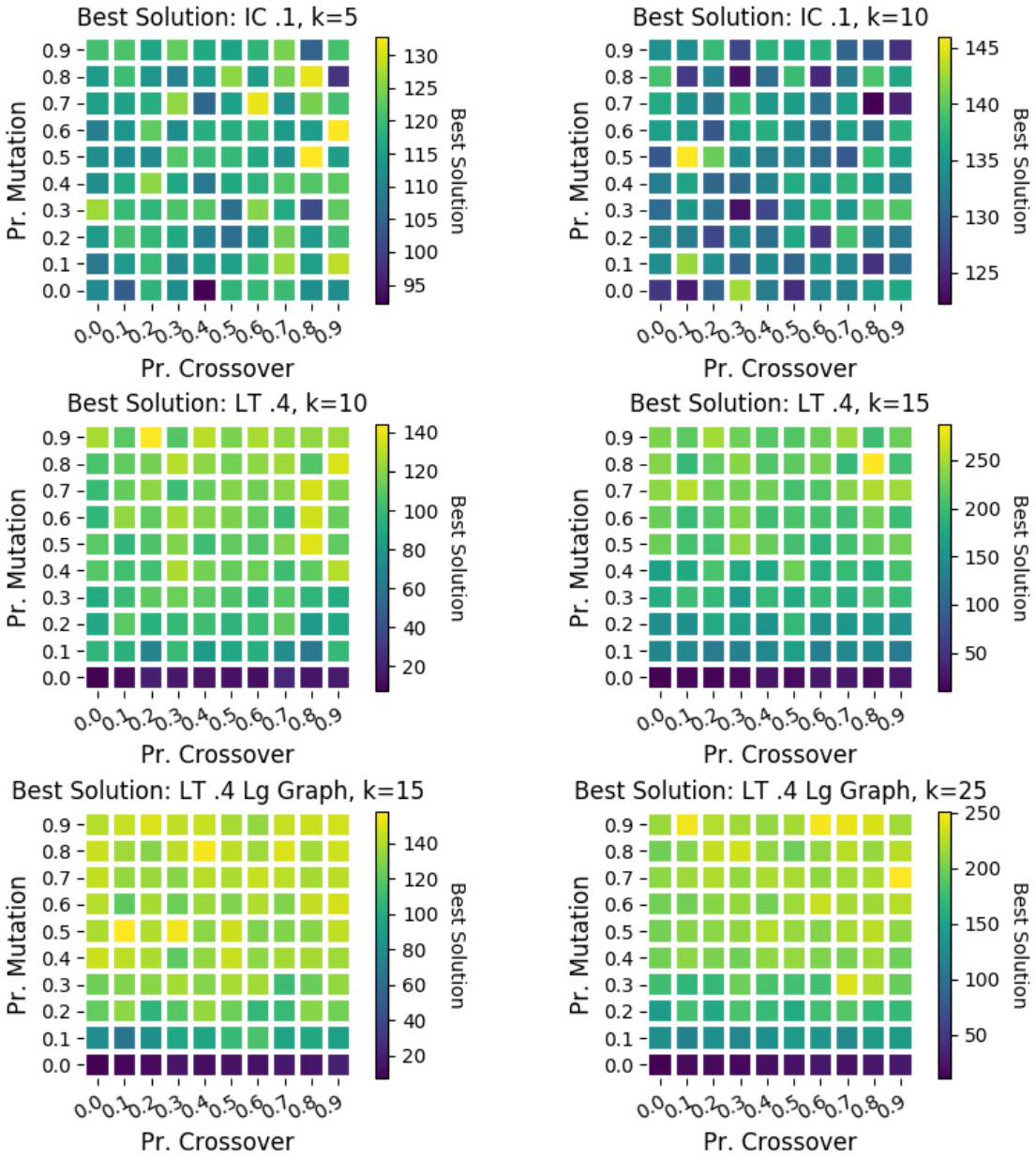

Figure 3.7: Heatmaps of the average value of solutions in total spread over a set of graphs using GA's with various probabilities of mutation and crossover.

populations under LT, $t = .4$, on five SBMs and five DCSBMs, at $k = 10, 15$. Tournament selection produces almost strictly dominating fitness trends in each case.

Utility of population size seems to be a logarithmic function, as seen in Fig.

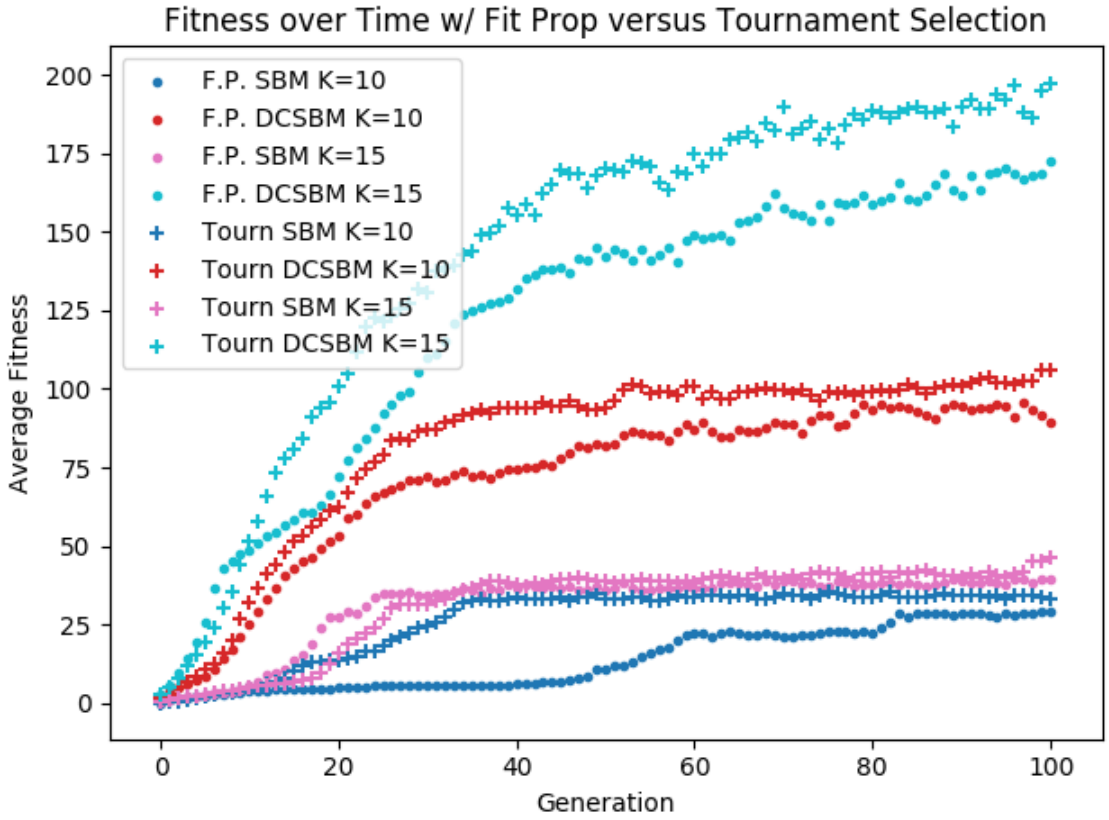

Figure 3.8: Average population fitness over time for tournament selection (crosses) and fitness proportionate selection (circles) over four test trials. Results averaged over 5 graphs.

3.9, which compares the fitness of solutions on LT, $t = .4$, for a two graphs at $k = 10, 15$ and displays a wider study of solution quality versus population size as graph size (in nodes) increases. In support of some modest trends visible in Fig. 3.9, in Chapter 4 we find that selecting a population size as a constant fraction of the number of nodes works well. This result is interesting and conflicts with the results of Cui et al. [47], who found that population size was not a big determinant of solution value and that it could be quite small without losing value (down to 5 members of the population on graphs up to 15k nodes!).

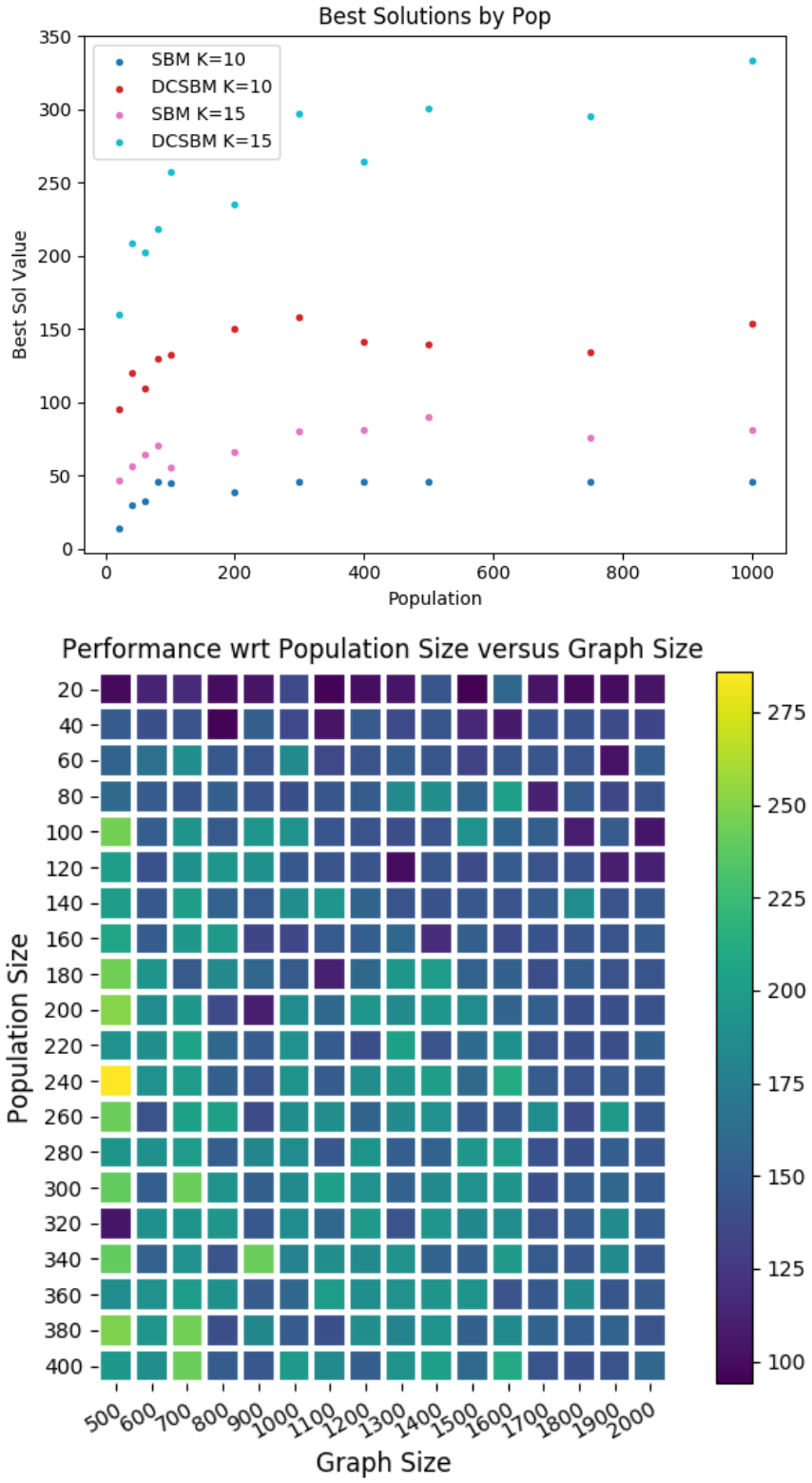

Figure 3.9: Top: Average solution spread for various population sizes. Results averaged over 5 graphs. Bottom: Solution quality for DCSBM of various sizes in nodes versus population size (1 trial per data point).

Chapter 4: Maximizing Influence and other Problems

In this chapter we take a look at the performance of multiple set-selection algorithms in a variety of influence-maximization problem variants. Besides offering quantitative results on relative performance and suggesting what algorithms work best for different problem statements, we also wanted to offer some *a priori* intuition on whether or not a problem warrants potentially expensive computation in the first place. To this end we explore relative performance of seed selection algorithms on a range of process parameters and simple graphs with varying edge densities in Section 4.1. Then we cover the general methodology for all follow-on experiments in Section 4.3, as it is consistent with few exceptions across all experiments, then cover results by problem type in sequence.

4.1 Simple Graphs and Classic Diffusion Models

In this experiment we explore the performance of four seed-selection strategies on the Independent Cascade and Linear Threshold processes over simple graph structures. Infection probability and infection threshold are both uniform across all nodes for each trial, and are varied from .1 to .7 in .1 increments between trials. For graph types, we explore Erdős-Rényi graphs with average degrees in

$\{1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0\}$, Barabási-Albert graphs with $m = \{1.0, 1.25, 1.5, 1.75, 2.0, 2.25, 2.5\}$, and Stochastic Block Models with 10 equal sized communities with expected intra-community degree $\{0.5, 1.0, 2.0\}$ and inter-community degree $\{0.5, 1.0, 2.0\}$ (all pairs were tested except where intra-community and inter-community degrees were equal, resulting in 6 parameter settings). For each graph type and parameter, 5 graphs were generated with 500 nodes.

We compared four seed-selection algorithms: Greedy, Genetic Algorithm (GA), Top-Deg, and Random. Seed sets were size $k = \{5, 10\}$. These are modest seed set sizes, which we expect to provide the starkest indicator of relative performance (small seed sets that terminate quickly can only infect a few additional nodes, while those that initiate a cascade will have a large influence). As seed-set size grows (with proportionally higher thresholds or probabilities of infection) we would expect the relative performance of algorithms to converge. This is because the graph becomes more saturated at t_0 for all algorithms, and even processes that fall off will activate a number of new nodes proportional to seed-set size. Greedy adds a single node to the set at a time, choosing the one with the highest delta spread determined by simulation (in this experiment, 5 process evaluations were executed per candidate node per k). The GA works as in Chapter 3 and executed one simulation to evaluate fitness (this may seem very low, but in effect it naturally scales to many simulations for good solutions, because we expect a good solution to get repopulated, mutated, and explored over many generations). Top-Deg executes no costly simulations, it simply takes the top-k nodes ranked by degree. Random selects k nodes at random. We are interested in regions where the spread between algorithm performance (that

is, greatest seed-set performance minus worst seed-set performance) is very high or very low.

Figs 4.1,4.2 demonstrate a clear relationship between the graph and process parameters and the differential impact between algorithms. In general, if the graph is sparse and the process not very contagious, it won't matter much how we select our seed set, and the similarly for very contagious and highly connected graphs. The comparison between Erdős-Rényi and Barabási-Albert graphs reveals that scale-free structures should be considered as an accelerant - note the sharper slope of the sensitive region in the BA heatmap. There also appears to be a periodic pattern on the Stochastic Block Model results, which may be a result of how many communities are easily reachable / possibly reachable given a certain contagion probability or threshold. The dominant algorithm is different for each process, it appears that Top-Deg is sufficient for the simple IC process while Greedy is a good starting point for the non-submodular LT process. Had we allowed the Greedy algorithm or GA more simulations they certainly would have performed better, so this is not to say that very simple heuristics like Top-Deg are better than Greedy or GA on IC processes. It was simply to show there are sensitive and indifferent regions in the process / graph space that can inform our dedication of computing resources in certain algorithms. As a final note, the heatmaps in Figs 4.1,4.2 do not significantly change in character when the comparison is just between Top-Deg, GA, and Greedy, although the magnitude of the quality spread does change considerably. We mention this to argue these result are more general than simply showing where Random selection fails.

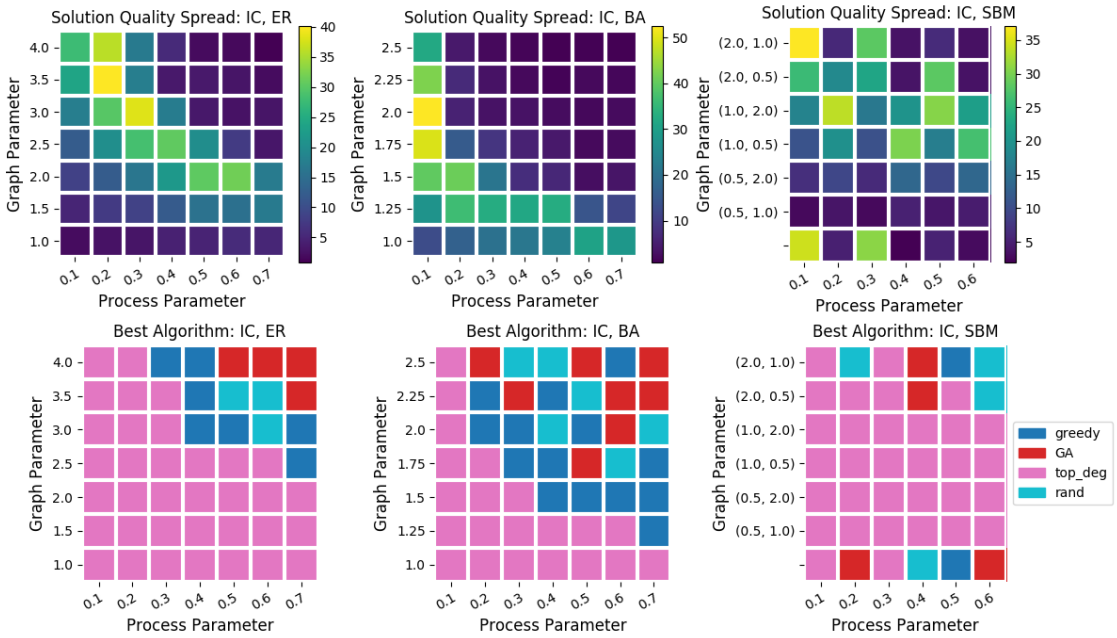

Figure 4.1: Heatmaps of the average spread of best solution and worst solution for all algorithms at each IC process parameter and graph parameter combination, followed by the best algorithm at each. All results averaged over 10 trials, 5 graphs each at $k = \{5, 10\}$.

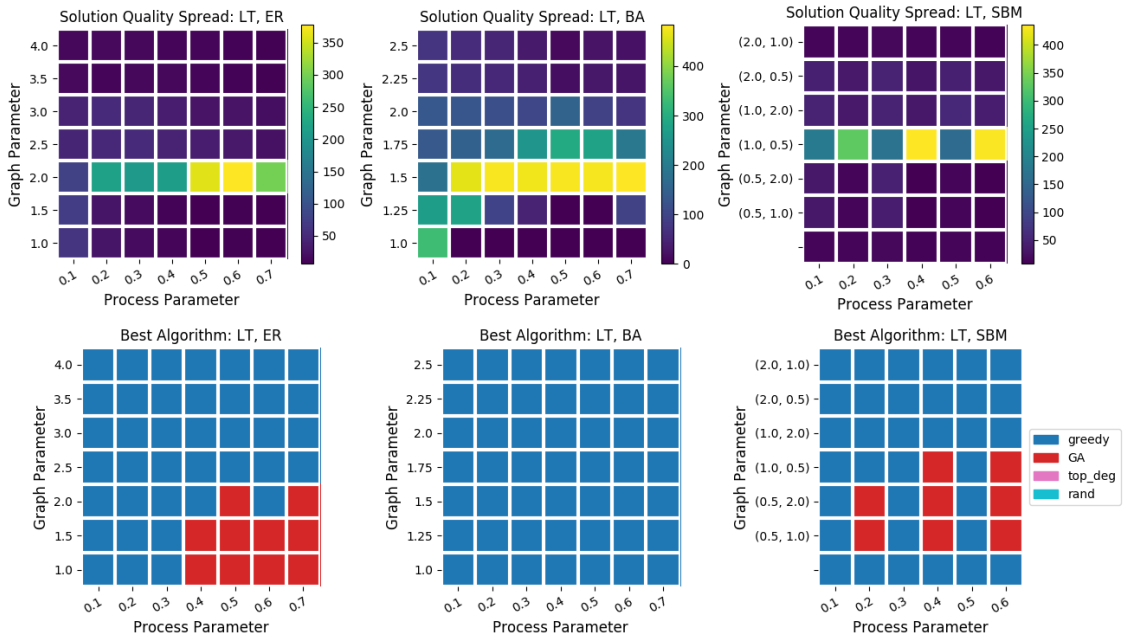

Figure 4.2: Heatmaps of the average spread of best solution and worst solution for all algorithms at each LT process parameter and graph parameter combination, followed by the best algorithm at each. All results averaged over 10 trials, 5 graphs each at $k = \{5, 10\}$.

4.2 Hybrid Partitioning Algorithm

Before moving on to Methodology in 4.3, we introduce one final algorithm we term a **Hybrid Partitioning Algorithm**. In a hybrid partitioning algorithm we seek to reduce the search space considerably by leveraging a partition of the graph's structure. A hybrid algorithm will place k nodes among the m subgraphs defined by the partition, using any appropriate algorithm, resulting in a "high-level" solutions of the form $\{x_1, x_2, ..., x_m\}$ where x_i is the number of nodes the solution will put in partition i, and $\sum_{i=1}^{m} x_i = k$. A (potentially different) algorithm will determine which nodes actually get chosen for each partition i, given x_i. The hybrid algorithm will then evaluate the total set built by the high and low level algorithms in its search for a seed set. If the structure of the graph is meaningful and in particular associative, we hypothesize that focusing attention on the structures to target and ignoring complex (but possibly small) inter-community interactions will yield strong solutions more efficiently.

A similar technique was recently used by Angell and Schoenebeck [48], where, instead of a single partition, a hierarchical-structure detection algorithm based on distance between nodes was used to process the input graph into a full binary tree. Then they employed a dynamic programming technique using the binary tree to assemble max influence seed sets. In our results going forward, we refer to each hybrid algorithm by a tuple ⟨partition_type, high-level alg, low-level alg⟩. For example, ⟨SBM, GA, Greedy⟩ would refer to a hybrid algorithm that partitions the graph with a SBM recovery algorithm, searches the high-level structures using a

GA, and given x_i determines which nodes for each community in the partition using a Greedy algorithm.

4.3 Methodology

In the experiments that follow, we take a look at the relative performance of various algorithms on a variety of IM problem variants. In particular, for each we compare the solution quality of a Greedy algorithm, a Genetic Algorithm, Top-K Degree, and Random. For Greedy and GA solutions we also record the number of process evaluations required and real time to calculate. All experiments were run on Intel i5 processors between 2.6 and 3.5 GHz with serial computation (no parallelization across cores was implemented). Due to the variance in processor speeds and use of the computers for additional tasks as the experiments ran in the background, the real time run data has influences beyond just computational difficulty of the problem at hand. However, all results for any given problem variant were executed on the same processor, and the trials were executed for each algorithm in sequence for any given graph / process pair, such that any temporary resource scarcity would likely impact all algorithms equally (but perhaps not equally across all process parameters / graph types). Therefore we expect relative performance with respect to run time within each problem variant to provide informative insights. Run times shown are for the GA to calculate its $k = 15$ node seed set and for the Greedy algorithm to calculate all of its seed sets (as it is an incremental algorithm, this is the same as if we only asked it to calculate a $k = 15$ node set). We cover

problem specific parameters and exceptions in their associated sections, and cover here the constants across all experiments.

4.3.1 Algorithm Variants and Parameters

All greedy algorithms featured no optimization with the exception of using the reverse-reachable set variant of Tang [3] for the IC trials. The basic genetic algorithm (labeled "GA") used 100 max generations, a population of 100 individuals, 2-member tournament selection, $P_m = .8$, $P_c = .3$, 2 elites, random (unweighted) mutations, and single-point crossover. Additionally, in all trials we explore a GA that scales its population based off the graph size (labeled "Scaled GA"), where $pop\_size = N/5$. We also introduced two heuristics to improve run time. The first is an early-stopping criterion: if the best elite each generation remains unchanged for X generations, the GA is terminated and that individual returned ($X = 10$ in all basic results, and $X = 20$ in all scaled results). The second heuristic is a cache of fitness evaluation results that stores the fitness of $3 \times pop\_size$ recent evaluations.

On deterministic processes, each algorithm evaluated fitness only once per candidate individual. The total number of trials per evaluation, for each algorithm, is noted. Final estimated spread results for stochastic processes are estimated with 100 process simulations per solution.

4.3.2 Graphs

We conduct all experiments on the same set of 60 graphs. Each type of graph is represented at three different scales: $N = \{500, 1000, 2000\}$. For each scale, we generate five graphs as follows:

- Five Barabasi-Albert Graphs with average degrees $\langle k \rangle = \{2, 4, 6, 8, 10\}$
- Six Stochastic Block Models, 10 equal size communities, with $\langle k_{intra\cdot} \rangle = \{1, 3, 5\}$ and $\langle k_{inter} \rangle = \{1, 3, 5\}$. To keep the sets of each type similar in size, graphs with equal average intra-community and inter-community degrees were omitted.
- Six Degree-Corrected SBMs, 10 equal size communities, $\langle k_{intra} \rangle = \{1, 3, 5\}$, $\langle k_{inter} \rangle = \{1, 3, 5\}$, $\gamma = 2$. Again, if $\langle k_{intra} \rangle == \langle k_{inter} \rangle$, the graph was omitted.
- Real-World Networks (see Table 4.3)

4.4 Results: Influence-Maximization and Variants

Due to the breadth of the study, we start first with some general trends and insights before presenting process specific parameter settings and their results. Presented in Fig. 4.4 is a comparison of algorithm performance and run time across all process types. Relative spread for a single trial is calculated as the algorithm's seed selection's performance divided by the best performing selection for that trial

	Social	Biological	Infrastructure
N ≈ 500	ia-infect-Dublin (410, 2765) aves-weaver-social (445, 1304) fb-pages-food (620, 2091)	bio-diseasome (516, 1188) bio-celegans (453, 2025)	
N ≈ 1000	email-univ (1133, 5451) rt-twitter-copen (761, 1029) soc-wiki-Vote (889, 2914)		inf-euroroad (1174, 1417) Power-eris1176 (1176, 8688)
N ≈ 2000	soc-hamsterster (2426, 16630) Ca-CSphd (1882, 1740) Socfb-Simmons81 (1518, 32988)	bio-yeast-protein-inter (1870, 2394)	Power-bcspwr09 (1723, 2394)

Figure 4.3: Network names (as taken from Network Repository [36]) followed by number of nodes and edges, by category and size. We attempted to sample proportionately across the graph meta domains for each size. There was not a suitable 500 node infrastructure graph nor 1000 node biological graph, so instead there are two biological graphs at the first scale and two infrastructure at the second. All graphs were unweighted and undirected.

(whichever algorithm chose it). The results in Fig. 4.4 are averaged over every single trial for each given process. Relative time is calculated similarly, where this time the fastest algorithm's time is the divisor.

Immediately we see that the scaled genetic algorithm and greedy solutions are dominant over the heuristic and random choices. Further they are typically very close in solution quality with each other, however the scaled GA is better on average for most processes. It is interesting to note the greedy algorithm is most competitive on all of the Linear Threshold type processes, only being outperformed by the scaled GA on ADV-LT. Top-K also performs relatively well for the LT processes, when compared to how poorly it does on the others. The only exception is that Top-K is almost the dominant algorithm for EM-IC, suggesting that the very simple heuristic is a strong candidate for inoculation strategies. If we compare the effects of introducing an adversary, the margin between the Greedy algorithm and scaled GA greatly increases in favor of the GA; we see this by comparing LT versus ADV-LT

and UC versus ADV-UC. Despite generally stronger performance on most processes, the Scaled GA is a bit faster on average and when it does take longer it appears to find much better solutions.

The comparison between GA and Scaled GA results suggests that the GA performance is resilient to a poorly selected population size. With results from Chapter 3 and the data that a low population GA runs significantly faster than a Greedy algorithm, this makes a small population GA an ideal algorithm for probing an unknown problem domain. A quick run time and relatively competitive results — as long as the population is within a wide, reasonable range — should grant some idea of whether the scenario warrants employment of more expensive algorithms. As can be seen in the process specific results, the Scaled GA typically had many more process simulations even when total runtime was lower than the greedy approach. This is likely because the greedy approach, when adding to the "tail end" of the set, has already selected a strong subset of nodes. Then it must simulate the process with this subset plus each next node as a candidate, potentially spending a lot of time on multi-step process simulations. While we might expect a similar situation in the GAs, where stronger populations begin taking longer to evaluate, the early stopping mechanism prevents that in the case where value is no longer being added by variation.

As a final general highlight, in the IC and LT trials we explored hybrid algorithms and found them to be unpredictable and frequently perform worse than Top-K, even when given similar processing time as the Scaled GA and Greedy algorithms. As these are supposed to be much more efficient and still produce reasonable

Process	Submodular	Monotonic
IC	Y	Y
LT	N	Y
CC	N	Y
EM-IC	N	Y
ADV-LT	N	Y
UC	N	N
ADV-UC	N	N
S-LT	N	N

Relative Spread				
Greedy	Sc-GA	GA	Top-K	Rand
0.99	0.85	0.84	0.95	0.70
0.98	0.93	0.85	0.76	0.24
0.84	**0.98**	0.84	0.44	0.42
0.95	**0.97**	0.93	0.96	0.85
0.91	**0.92**	0.81	0.10	0.09
0.86	**0.89**	0.84	0.42	0.27
0.80	**0.90**	0.79	0.27	0.09
0.96	0.93	0.81	0.63	0.16

Relative Time		
Greedy	Sc-GA	GA
1.67	8.90	3.11
7.03	3.03	1.76
10.85	10.03	1.00
6.92	4.84	1.00
2.93	4.94	1.06
6.22	4.91	1.01
4.30	7.63	1.01
4.87	4.15	1.04

Figure 4.4: A summary of results between Greedy, Scaled GA (Sc-GA), fixed population GA, Top-K Degree, and Random K seed sets on all problem variants explored. Relative spread is calculated as the average of the selected algorithm's spread divided by maximum spread for each trial, across all trials (graphs, process parameters, seed set size), for every process. Similarly, relative time is calculated as the average of the algorithm's time divided by the fastest algorithm for each trial (excluding Top-K and Random). For example, a relative spread of 1.00 would indicate that algorithm was the maximum performer on every single trial, and a relative time of 1.00 would indicate it was the fastest on each trial.

seed selections, we caution their use to cases with well-defined structures, as suggested by Figs 2.8, 2.9.

4.4.1 Independent Cascade

Fig. 4.5 shows the Independent Cascade (monotonic, submodular) process with uniform infection probabilities scaled by the average degree $\langle k \rangle$ of the graph: $c = \{\frac{.2}{\langle k \rangle}, \frac{.4}{\langle k \rangle}, \frac{.6}{\langle k \rangle}, \frac{.8}{\langle k \rangle}\}$. The greedy algorithm in this experiment comes from Tang [3] and makes use of reverse-reachable set sampling to improve runtime. Because of the large number of trials to run, we allowed a maximum expected error $\epsilon = .3$ such that the total approximation guarantee on the seed sets returned is $\alpha \approx .33$. In practice, the actual performance of the seed sets returned is probably well within that bound. The genetic algorithm conducts 10 process simulations per fitness evaluation.

As the most studied and tractable model, here we expected the fine-tuned

greedy algorithm from Tang to perform well. As seen in Fig. 4.5, it is strictly dominant in all cases. Top-K is also a strong contender for this simple process.

4.4.2 Linear Threshold

Fig. 4.6 shows the Linear Threshold (monotonic, non-submodular) process with uniform thresholds scaled by the median degree k^* of the graph: $t_0 = \frac{1}{2k^*}, t_3 = \frac{1}{2}$, and t_1, t_2 uniformly between t_0 and t_3. Greedy algorithms from here on are classic greedy algorithms: to determine which node to add, every node is considered as an addition to the current set and simulations are run to determine the maximal addition.

The results show a drastic reduction in the capacity of Top-K and Random. Scaled GA appears to be the dominant algorithm on more complex and larger graphs, while Greedy is best on simpler graphs.

4.4.3 Ugander Complex

Fig. 4.7 shows the Ugander Complex (non-monotonic, non-submodular) process with infection probabilities defined by $pr\_infection(x_i) = infection\_factor(x_i) \times c$, where $infection\_factor(x)$ is 0 if $x <= 1$, 1 if $x == 2$, 1.5 if $x == 3$, and .1 otherwise. x_i is the number of distinct, connected components that are infected in the neighborhood of v_i and c ranges in $c = \{.2, .4, .6, .8\}$. Both the Greedy and GA algorithms evaluated solutions with 5 simulations during their search process.

In these results we see a dramatic reduction in the simple heuristic's value,

in particular on the DC-SBM graphs. Top-K likely spreads its seed sets too far across the many communities, choosing the highest degree node in each, but failing to saturate many nodes past their thresholds for activation. With this complex process, only our more expensive algorithms offer reliable performance.

4.4.4 Community Contagion

Fig. 4.8 shows for Community Contagion (monotonic, non-submodular) process with uniform thresholds scaled by the median number of communities known $k^{'}$ of the graph: $t_0 = \frac{1}{2k^{'}}$, $t_3 = \frac{1}{2}$, and t_1, t_2 uniformly between t_0 and t_3. Since we need a special type of graph with labeled edges and communities to run this process, we generated our own with Algorithm 2. Given a set of graphs with known community partitions, algorithm 2 generates a new graph with all edges from the union of the set. The merged graph's nodes and edges aggregate all original graph community affiliations of nodes and edges. We generate the CC merged graphs from the same set of SBM, DCSBM, and RWN graphs used in all other trials by merging each set of $(X-1)$ graphs from the original set of X. For example, one merged SBM is the merge of the first four SBMs, leaving out the fifth. In the special case of the RWN graphs, we used the best fit communities from SBM recovery to define the original communities for each RWN graph.

In Fig. 4.8 we find extremely diverse behaviors across algorithms and non-linear growth as seed-set size increases. More than any other process, CC has much of its dynamic information hidden from simple analysis, resulting in a sharp

Algorithm 2: Merge Algorithm for Community Contagion Graph Generation from Experiment Graphs

```
for type in ["SBM","DCSBM","RWN"] do
  for size in [500,1000,2000] do
    graph_subset ← all_graphs[type,size]
    for i == 0, i < 4 do
      cc_graph = merge(graph_subset - graph_subset[i])
      cc_graphs.append(cc_graph)
    end for
  end for
end for
```

divide between our expensive algorithms and heuristics. This is not to say some fast heuristics might not work - perhaps Top-K sorted by number of communities to which each node belongs would perform well.

4.4.5 Epidemic Mitigation

Fig. 4.9 shows results for epidemic mitigation (monotonic, non-submodular) with Independent Cascade process with uniform infection probabilities scaled by the average degree $\langle k \rangle$ of the graph: $c = \{\frac{.2}{\langle k \rangle}, \frac{.4}{\langle k \rangle}, \frac{.6}{\langle k \rangle}, \frac{.8}{\langle k \rangle}\}$. The epidemic begins with a random node (sampled and fixed before all trials began, for each graph) and up to 10 neighbors infected. The algorithm cannot inoculate an infected node. GA and Greedy algorithms evaluated solutions with 10 simulations during their search process.

EM-IC shows an unexpectedly strong performance from Top-K, with only Scaled GA outperforming on average and only specifically on the SBM and RWN graphs. Here we note a stark contrast from the UC case where spreading the seed selection with Top-K was detrimental. In an epidemic mitigation scenario, reducing

the connectivity of many distributed communities appears to work very well (see DC-SBM results).

4.4.6 Adversarial LT

Fig. 4.11 shows results for Adversarial (monotonic, non-submodular) Linear Threshold process with uniform thresholds scaled by the median degree k^* of the graph: $t_0 = \frac{1}{2k^*}, t_3 = \frac{1}{2}$, and t_1, t_2 uniformly between t_0 and t_3. The adversary seed set selection is always the top 10 nodes ranked by degree.

4.4.7 Adversarial UC

Fig. 4.10 shows results for Adversarial (non-monotonic, non-submodular) Ugander Complex process with parameters as in uncontested UC above. The adversary seed set selection is always the top 10 nodes ranked by degree. GA and Greedy algorithms evaluated solutions with 5 simulations during their search process.

For both ADV-UC and ADV-LT, we see the Scaled GA took much longer to compute solutions but that these solutions were on average better than greedy selections.

4.4.8 Saturated Linear Threshold

Fig. 4.12 shows results for the Saturated Linear Threshold (non-monotonic, non-submodular) process with activation threshold as in Linear Threshold trials but also a saturation threshold equal to $t_S = 1.5t$. This model spreads as in the

Linear Threshold model, except when a node's infected neighbors cross its saturation threshold t_S it becomes saturated and no longer can be infected or exerts influence on other nodes.

We introduced S-LT specifically to test for a breaking point in the greedy algorithm. We imagined a deliberately non-monotonic process would prove difficult for a greedy strategy, however this was not the case. Greedy still performed quite well, outperforming Scaled GA results on average. The only significant exception seems to be some trials on the larger RWN graphs.

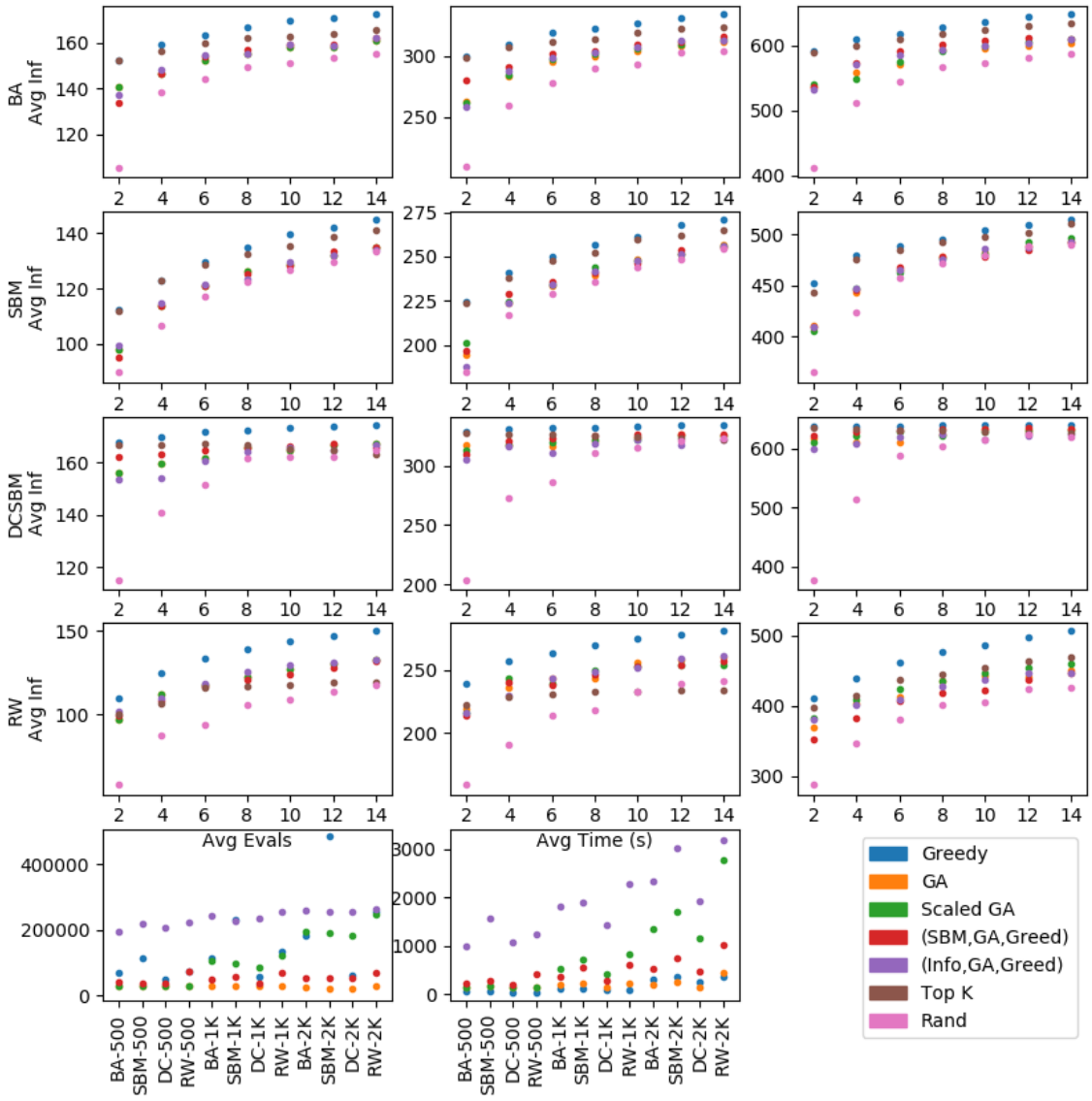

Figure 4.5: Performance of seed sets in expected spread across various graph types, averaged over the IC process with uniform infection probabilities scaled by the average degree $\langle k \rangle$ of the graph: $c = \{\frac{.2}{\langle k \rangle}, \frac{.4}{\langle k \rangle}, \frac{.6}{\langle k \rangle}, \frac{.8}{\langle k \rangle}\}$ and 5 graphs per point. Each row represents a different graph type, and the columns correspond to graph size, from left to right: 500, 1000, and 2000 nodes.

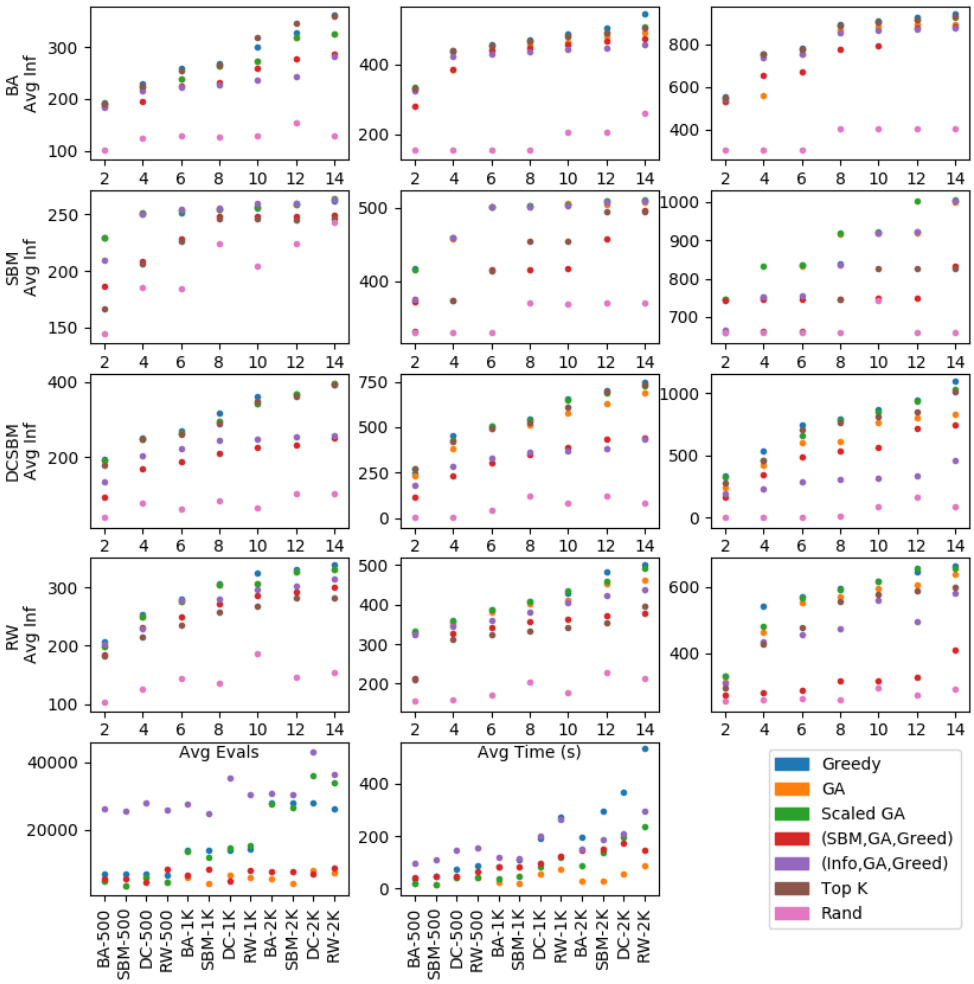

Figure 4.6: Linear Threshold: Performance of seed sets in expected spread across various graph types, averaged over the LT process with uniform thresholds scaled by the median degree k^* of the graph. Each row represents a different graph type, and the columns correspond to graph size, from left to right: 500, 1000, and 2000 nodes. The final row includes timing results (excluding Rand and Top-K, which are $O(K)$ and $O(M)$ respectively).

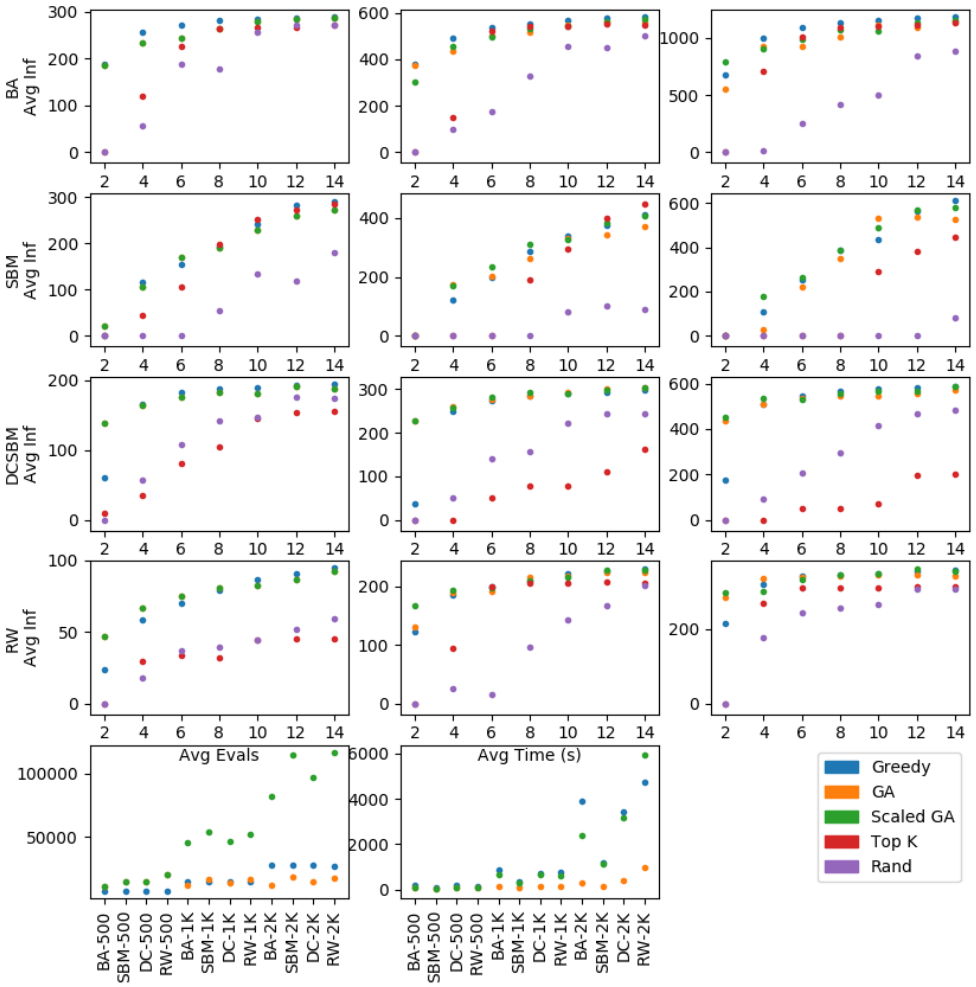

Figure 4.7: Ugander Complex: Performance of seed sets in expected spread across various graph types, averaged over the UC process. Each row represents a different graph type, and the columns correspond to graph size, from left to right: 500, 1000, and 2000 nodes.

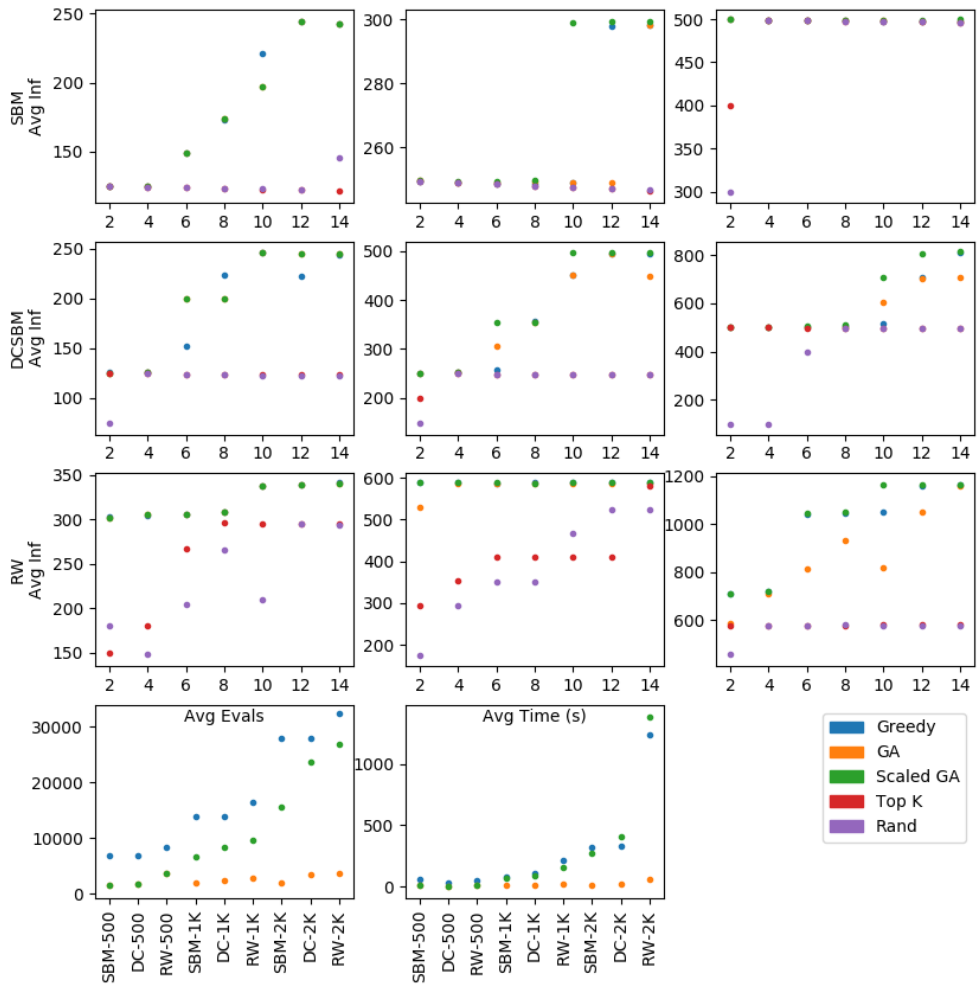

Figure 4.8: Community Contagion: Performance of seed sets in expected spread across various graph types, averaged over the CC process. Each row represents a different graph type, and the columns correspond to graph size, from left to right: 500, 1000, and 2000 nodes.

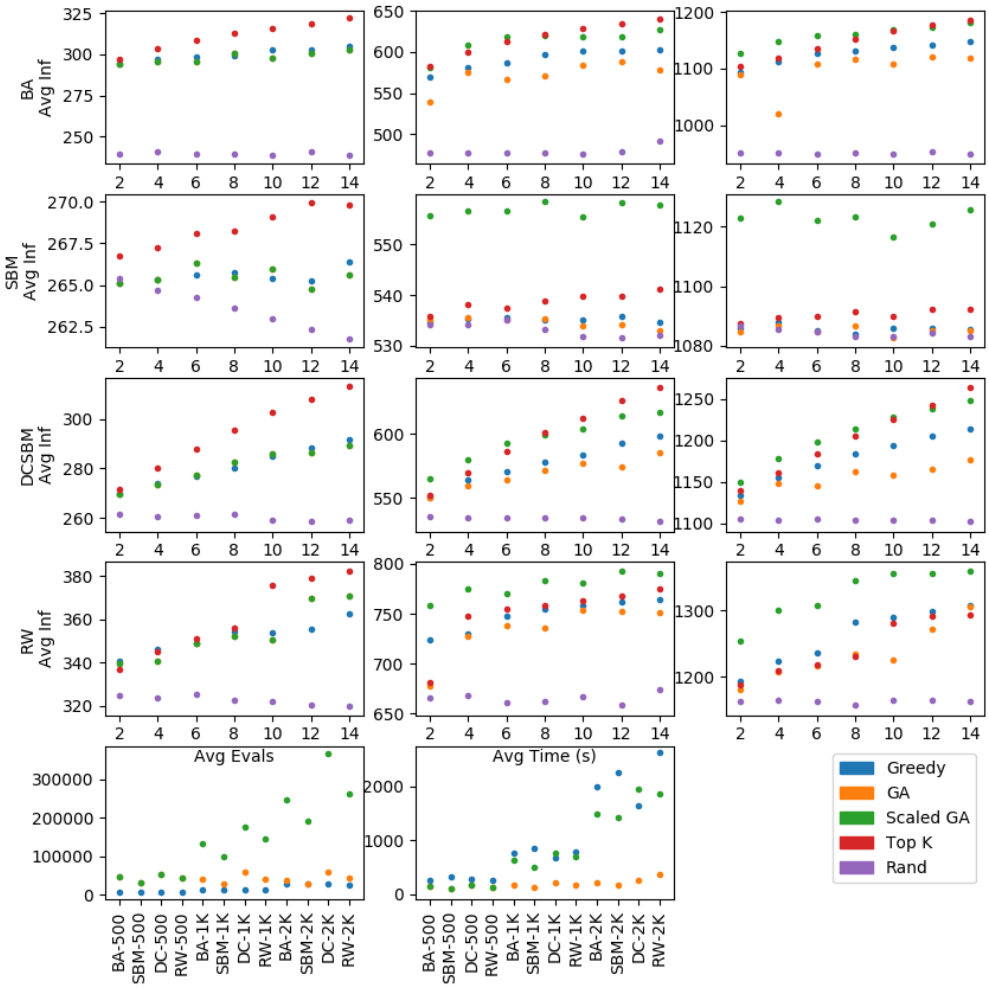

Figure 4.9: Epidemic Mitigation: Performance of seed sets in expected number of nodes *never* infected across various graph types, averaged over the IC process with uniform infection probabilities scaled by the average degree $\langle k \rangle$ of the graph: $c = \{\frac{.2}{\langle k \rangle}, \frac{.4}{\langle k \rangle}, \frac{.6}{\langle k \rangle}, \frac{.8}{\langle k \rangle}\}$ and 5 graphs per point. Each row represents a different graph type, and the columns correspond to graph size, from left to right: 500, 1000, and 2000 nodes.

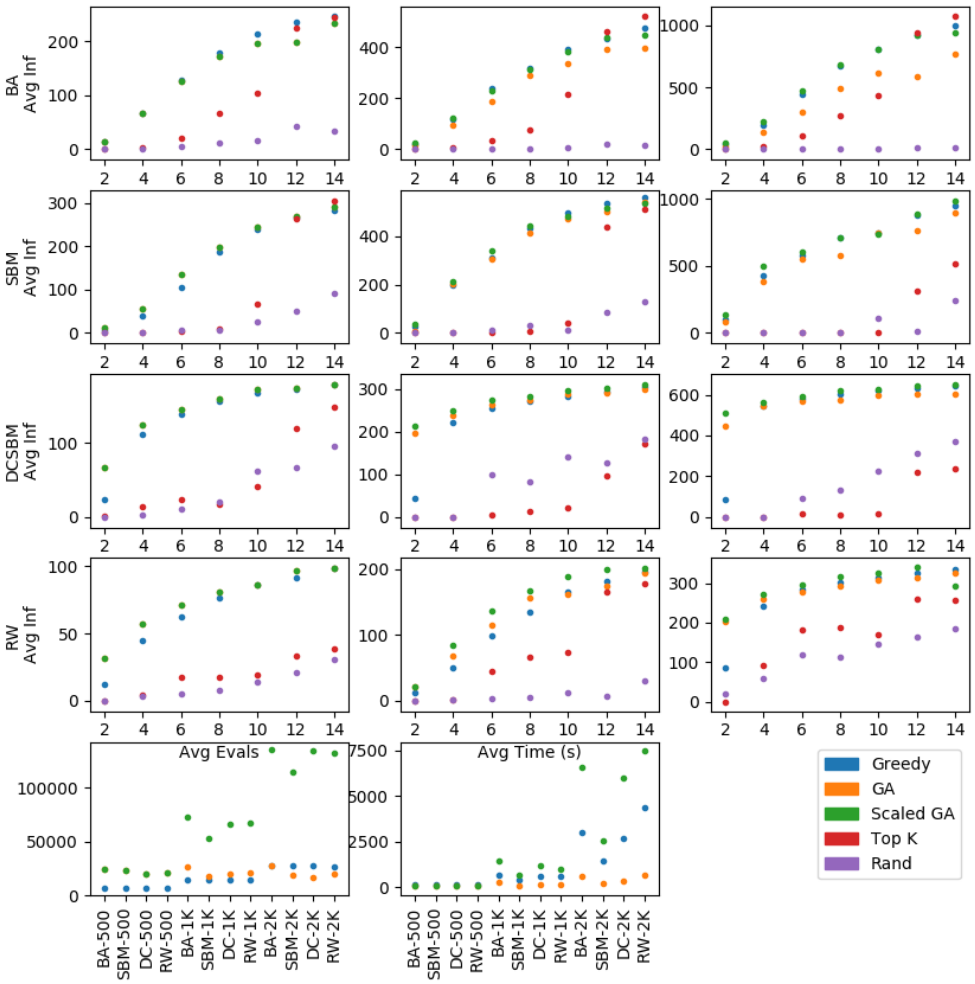

Figure 4.10: Adversarial UC: Performance of seed sets in expected spread across various graph types, averaged over the Adversarial UC process. Each row represents a different graph type, and the columns correspond to graph size, from left to right: 500, 1000, and 2000 nodes.

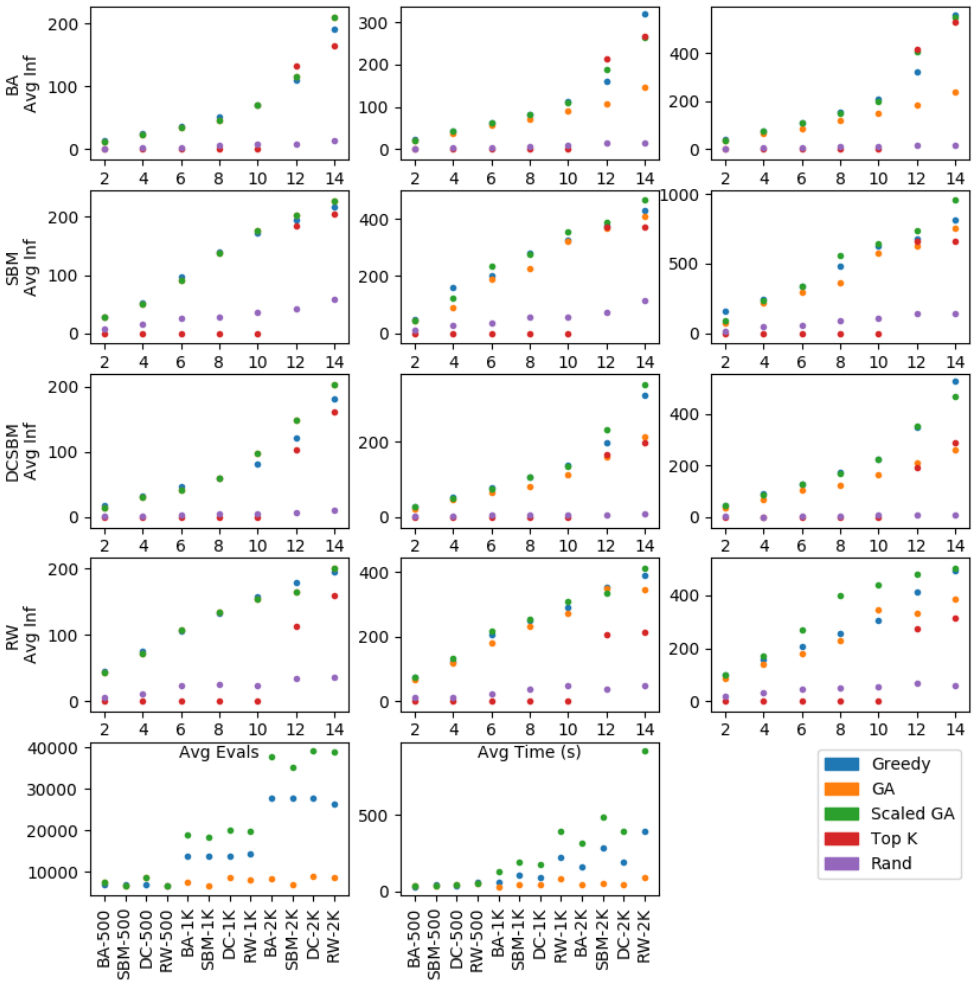

Figure 4.11: Adversarial LT: Performance of seed sets in expected spread across various graph types, averaged over the Adversarial LT process. Each row represents a different graph type, and the columns correspond to graph size, from left to right: 500, 1000, and 2000 nodes.

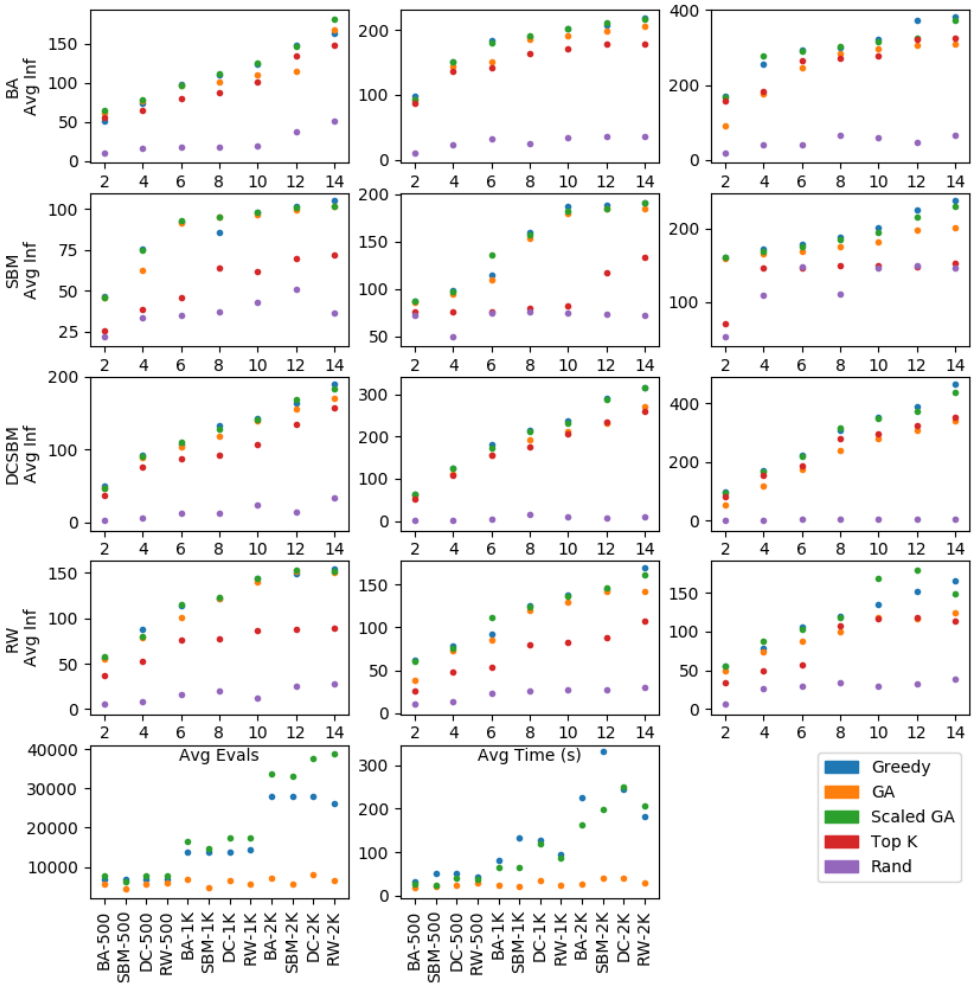

Figure 4.12: Saturated LT: Performance of seed sets in expected spread across various graph types, averaged over the Saturated Linear Threshold process. Each row represents a different graph type, and the columns correspond to graph size, from left to right: 500, 1000, and 2000 nodes.

Chapter 5: Conclusion

Through the course of this thesis, we have examined the complexity of real-world graphs and processes which are (mostly) intractable to approximation guarantees. We began with studies of real-world graphs, up to a hundred-thousand edges, and attempted to find patterns in their community and flow structures. We found few strong generalizations and what correlations did exist might be attributed to specific data collection and graph design decisions. We also found that very few component structures within real-world graphs matched commonly used random-graph structures. Robustness, our measure of community structure "truth", revealed some correlation with the ability to leverage found structures to understand process dynamics.

Then, in Chapter 3, we took a deep dive into various design choices for genetic variation and other parameter settings in genetic algorithms. We found little motivation to pursue domain or problem specific genetic operators, noting competitive results for genetic algorithms with classic, network agnostic operators. Furthermore, we noted general resilience to the settings of mutation and crossover probability: so long as the mutation probability is above .3, typically the evolutionary algorithm performed well. Motivated by the performance invariance to crossover probabil-

ity, we considered a Probabilistic Model Building Genetic Algorithm which better shares information among the population. Initial trials indicate this as a promising direction for future work, but that it is sensitive to some parameter settings. In particular, PMBGA's may be a very strong candidate for selecting large seed sets, for while most algorithms scale poorly with seed set size (Greedy is linear in seed set), PMBGA can actually run *faster* with larger seed sets. We concluded the chapter with a study of the strong positive relationship between the size of the population and the size of the network, noting in particular a sharp drop-off for increasing population size based on size of the graph.

Using this knowledge, we build a set of synthetic and real-world graphs across small and moderate graph sizes. This single test bed was exposed to each process introduced in Chapter 1, each under a specialized and dynamic range of process parameters, for the set of algorithms we discussed throughout. We note that performance for linear heuristics is unpredictable but frequently competitive. Greedy algorithms typically perform well but scale poorly for large graphs and high-variance processes. Without specific modifications, they also are relatively rigid in their performance. We leave these additional adaptations for a greedy approach for future work, specifically the following: an optimization similar to CELF but without any approximation guarantee [17], which keeps track of which nodes had a high delta spread on the last round and rations the total process simulations for each iteration across the most promising nodes from the prior step. This may greatly reduce runtime spent evaluating poor node candidates which add negligible influence to an already high-performing subset. Introducing simulated annealing is also a strong

candidate for further study, as this allows the greedy strategy to explore some with some variation (see Zhang's paper [45] for some comparison work with simulated annealing and evolutionary algorithms).

Genetic algorithms seemed quite robust, offering strong performance even without problem-specific parameter settings. Given a similar amount of processing time, genetic algorithms frequently outperformed all other algorithms, particularly on more complex processes. Finally we note that the hybrid algorithms are unpredictable but very fast (although if we have measured the robustness of the partitioning algorithm we have some insights). An additional direction for future work is a hierarchical, rather than fixed two-tier, hybrid system, which may yield more consistent results.

As might be expected, the depth of graph and process complexity did not offer us any dominant algorithm choice or perfect predictor of which algorithm will work best for a given scenario. Instead we have learned that it is best to have a variety of techniques, at least some of which accept tuning to the scale and complexity of the problem. As a rule of thumb, running a linear time heuristic and greedy or genetic algorithm at a small scale should be done to probe an unfamiliar problem territory. Based on the relative performance an investigator can then move forward more confidently, adjusting and improving a more complex algorithm like the GA, or accepting the heuristic as good enough.

Appendix A: Real-World Networks

Tables A.1-A.6 lists all graphs used in the real-world networks by name, type, size, and edge type.

Network Name	Type	Meta Type	Nodes	Edges	Weighted	Discrete
aves-barn-swallow-contact-network	animal_social	social	17	53	TRUE	TRUE
aves-barn-swallow-non-physical	animal_social	social	17	122	TRUE	TRUE
aves-geese-female-foraging	animal_social	social	20	156	TRUE	TRUE
aves-geese-male-foraging	animal_social	social	23	203	TRUE	TRUE
aves-hens-pecking-order	animal_social	social	32	496	FALSE	TRUE
aves-songbird-social	animal_social	social	110	1027	TRUE	FALSE
aves-sparrow-social-2009	animal_social	social	31	211	TRUE	FALSE
aves-sparrow-social-2010	animal_social	social	40	305	TRUE	FALSE
aves-sparrow-social	animal_social	social	52	454	TRUE	FALSE
aves-sparrowlyon-flock-season2	animal_social	social	46	348	TRUE	TRUE
aves-sparrowlyon-flock-season3	animal_social	social	27	163	TRUE	TRUE
aves-thornbill-farine	animal_social	social	62	1151	TRUE	TRUE
aves-weaver-social	animal_social	social	445	1304	TRUE	TRUE
aves-wildbird-network	animal_social	social	202	4574	TRUE	FALSE
insecta-ant-trophallaxis-colony2	animal_social	social	39	245	TRUE	TRUE
insecta-beetle-group-c1-period-2	animal_social	social	30	207	TRUE	FALSE
insecta-beetle-group-c2-period-2	animal_social	social	30	195	TRUE	FALSE
mammalia-asianelephant	animal_social	social	73	60	FALSE	TRUE
mammalia-bat-roosting-indiana	animal_social	social	43	546	TRUE	TRUE
mammalia-bison-dominance	animal_social	social	26	222	TRUE	TRUE
mammalia-dolphin-florida-forage	animal_social	social	190	1134	TRUE	TRUE
mammalia-dolphin-florida-overall	animal_social	social	291	3182	TRUE	TRUE
mammalia-hyena-networkb	animal_social	social	36	585	TRUE	FALSE
mammalia-voles-bhp-trapping	animal_social	social	1686	4623	TRUE	TRUE
mammalia-voles-kcs-trapping	animal_social	social	1218	3592	TRUE	TRUE

Table A.1: Real world networks used from Network Repository [36].

Network Name	Type	Meta Type	Nodes	Edges	Weighted	Discrete
bio-yeast-protein-inter	biological	biological	1870	2203	FALSE	TRUE
bio-dmela	biological	biological	7393	25569	FALSE	TRUE
bio-diseasome	biological	biological	516	1188	FALSE	TRUE
bio-CE-PG	biological	biological	1871	47754	TRUE	FALSE
bio-celegans	biological	biological	453	2025	FALSE	TRUE
bio-HS-LC	biological	biological	4227	39484	TRUE	FALSE
bn-cat-mixed-species_brain_1	brain	biological	65	730	TRUE	TRUE
bn-fly-drosophila_medulla_1	brain	biological	1781	8911	TRUE	TRUE
bn-macaque-rhesus_brain_1	brain	biological	242	3054	TRUE	TRUE
bn-macaque-rhesus_cerebral-cortex_1	brain	biological	91	1401	TRUE	TRUE
bn-mouse_brain_1	brain	biological	213	16089	TRUE	TRUE
ca-CondMat	collaboration	social	21363	91286	FALSE	TRUE
ca-CSphd	collaboration	social	1882	1740	FALSE	TRUE
ca-Erdos992	collaboration	social	5094	7515	FALSE	TRUE
ca-GrQc	collaboration	social	4158	13422	FALSE	TRUE
ca-netscience	collaboration	social	379	914	FALSE	TRUE
ENZYMES_g1	cheminformatics	biological	37	84	FALSE	TRUE
ENZYMES_g10	cheminformatics	biological	32	53	FALSE	TRUE
ENZYMES_g13	cheminformatics	biological	42	75	FALSE	TRUE
ENZYMES_g15	cheminformatics	biological	36	64	FALSE	TRUE
ENZYMES_g16	cheminformatics	biological	55	97	FALSE	TRUE
ENZYMES_g103	cheminformatics	biological	59	115	FALSE	TRUE
ENZYMES_g105	cheminformatics	biological	33	69	FALSE	TRUE
ENZYMES_g108	cheminformatics	biological	38	82	FALSE	TRUE
ENZYMES_g112	cheminformatics	biological	51	95	FALSE	TRUE
ENZYMES_g113	cheminformatics	biological	52	98	FALSE	TRUE
ENZYMES_g118	cheminformatics	biological	95	121	FALSE	TRUE
ENZYMES_g121	cheminformatics	biological	42	82	FALSE	TRUE
ENZYMES_g123	cheminformatics	biological	90	127	FALSE	TRUE
ENZYMES_g134	cheminformatics	biological	32	51	FALSE	TRUE
ENZYMES_g147	cheminformatics	biological	40	84	FALSE	TRUE
ENZYMES_g148	cheminformatics	biological	39	80	FALSE	TRUE
ENZYMES_g149	cheminformatics	biological	39	82	FALSE	TRUE
ENZYMES_g158	cheminformatics	biological	40	63	FALSE	TRUE
ENZYMES_g167	cheminformatics	biological	43	94	FALSE	TRUE
ENZYMES_g171	cheminformatics	biological	48	99	FALSE	TRUE
ENZYMES_g172	cheminformatics	biological	25	46	FALSE	TRUE

Table A.2: Real world networks used from Network Repository [36].

Network Name	Type	Meta Type	Nodes	Edges	Weighted	Discrete
ENZYMES_g173	cheminformatics	biological	46	96	FALSE	TRUE
ENZYMES_g176	cheminformatics	biological	48	96	FALSE	TRUE
ENZYMES_g183	cheminformatics	biological	42	94	FALSE	TRUE
ENZYMES_g185	cheminformatics	biological	44	95	FALSE	TRUE
ENZYMES_g187	cheminformatics	biological	44	94	FALSE	TRUE
ENZYMES_g189	cheminformatics	biological	42	95	FALSE	TRUE
ENZYMES_g203	cheminformatics	biological	56	100	FALSE	TRUE
ENZYMES_g204	cheminformatics	biological	57	105	FALSE	TRUE
ENZYMES_g209	cheminformatics	biological	57	101	FALSE	TRUE
cit-DBLP	citation	social	12591	49620	TRUE	TRUE
eco-everglades	eco	infrastructure	69	880	TRUE	FALSE
eco-florida	eco	infrastructure	128	2075	TRUE	FALSE
eco-foodweb-baydry	eco	infrastructure	128	2106	TRUE	FALSE
eco-foodweb-baywet	eco	infrastructure	128	2075	TRUE	FALSE
eco-mangwet	eco	infrastructure	97	1446	TRUE	FALSE
eco-stmarks	eco	infrastructure	54	350	TRUE	FALSE
econ-beaflw	econ	infrastructure	502	44899	TRUE	FALSE
econ-beause	econ	infrastructure	507	39427	TRUE	FALSE
econ-mahindas	econ	infrastructure	1258	7461	TRUE	FALSE
econ-mbeacxc	econ	infrastructure	487	41686	TRUE	FALSE
econ-orani678	econ	infrastructure	2529	86767	TRUE	FALSE
econ-poli	econ	infrastructure	4008	4119	TRUE	FALSE
econ-poli-large	econ	infrastructure	15575	17427	TRUE	FALSE
econ-wm1	econ	infrastructure	260	2389	TRUE	FALSE
email-dnc	email	social	1892	4385	TRUE	TRUE
email-dnc-corecipient	email	social	906	10429	TRUE	TRUE
email-enron-only	email	social	143	623	FALSE	TRUE
email-EU	email	social	32430	54397	FALSE	TRUE
email-univ	email	social	1133	5451	FALSE	TRUE
ia-crime-moreno	interaction	social	829	1473	TRUE	TRUE
ia-escorts-dynamic	interaction	social	10106	35032	TRUE	TRUE
ia-fb-messages	interaction	social	1266	6451	FALSE	TRUE
ia-hospital-ward-proximity	interaction	social	75	1139	TRUE	TRUE
ia-infect-dublin	interaction	social	410	2765	FALSE	TRUE
ia-infect-hyper	interaction	social	113	2196	FALSE	TRUE
ia-workplace-contacts	interaction	social	92	755	TRUE	TRUE
inf-euroroad	infrastructure	infrastructure	1174	1417	FALSE	TRUE
inf-openflights	infrastructure	infrastructure	2939	15677	TRUE	TRUE
inf-power	infrastructure	infrastructure	4941	6594	FALSE	TRUE
inf-USAir97	infrastructure	infrastructure	332	2126	TRUE	FALSE

Table A.3: Real world networks used from Network Repository [36].

Network Name	Type	Meta Type	Nodes	Edges	Weighted	Discrete
AIDS	labeled	other	31175	32390	FALSE	TRUE
citeseer	labeled	other	3264	4536	FALSE	TRUE
CL-10K-1d8-L5	labeled	other	9221	44228	TRUE	TRUE
COIL-RAG	labeled	other	11687	11794	FALSE	TRUE
cora	labeled	other	2708	5278	TRUE	TRUE
COX2	labeled	other	19252	20289	FALSE	TRUE
DD6	labeled	other	4152	10320	FALSE	TRUE
DD21	labeled	other	5748	14267	FALSE	TRUE
DD68	labeled	other	775	2093	FALSE	TRUE
DD199	labeled	other	841	1902	FALSE	TRUE
DD242	labeled	other	1284	3303	FALSE	TRUE
DD244	labeled	other	291	822	FALSE	TRUE
DD349	labeled	other	897	2087	FALSE	TRUE
DD497	labeled	other	903	2453	FALSE	TRUE
DD687	labeled	other	725	2600	FALSE	TRUE
escorts	labeled	other	10106	35032	TRUE	TRUE
gene	labeled	other	1103	1672	FALSE	TRUE
internet-industry-partnerships	labeled	other	219	630	TRUE	TRUE
KKI	labeled	other	2238	4019	FALSE	TRUE
Letter-high	labeled	other	10482	10125	FALSE	TRUE
MSRC-9	labeled	other	8968	21644	FALSE	TRUE
Peking-1	labeled	other	3341	6575	FALSE	TRUE
PTC-FM	labeled	other	4925	5055	FALSE	TRUE
PTC-FR	labeled	other	5110	5266	FALSE	TRUE
PTC-MM	labeled	other	4695	4812	FALSE	TRUE
PTC-MR	labeled	other	4915	5054	FALSE	TRUE
TerroristRel	labeled	other	881	8592	FALSE	TRUE
ca-opsahl-collaboration	miscellaneous	other	22015	58585	TRUE	TRUE
Erdos02	miscellaneous	other	5534	8472	FALSE	TRUE
Erdos971	miscellaneous	other	433	1314	FALSE	TRUE
Erdos991	miscellaneous	other	454	1417	FALSE	TRUE
Harvard500	miscellaneous	other	500	2043	TRUE	TRUE
Journals	miscellaneous	other	124	5972	TRUE	TRUE
ODLIS	miscellaneous	other	2900	16377	TRUE	TRUE
Sandi_authors	miscellaneous	other	86	124	TRUE	TRUE
USAir97	miscellaneous	other	332	2126	TRUE	FALSE
USpowerGrid	miscellaneous	other	4941	6594	FALSE	TRUE
power-494-bus	power	infrastructure	494	586	TRUE	FALSE
power-662-bus	power	infrastructure	662	906	TRUE	FALSE
power-685-bus	power	infrastructure	685	1282	TRUE	FALSE
power-1138-bus	power	infrastructure	1138	1458	TRUE	FALSE

Table A.4: Real world networks used from Network Repository [36].

Network Name	Type	Meta Type	Nodes	Edges	Weighted	Discrete
power-bcspwr09	power	infrastructure	1723	2394	FALSE	TRUE
power-bcspwr10	power	infrastructure	5300	8271	FALSE	TRUE
power-eris1176	power	infrastructure	1176	8688	FALSE	TRUE
power-US-Grid	power	infrastructure	4941	6594	FALSE	TRUE
rt_alwefaq	retweet	social	4171	7059	TRUE	TRUE
rt_assad	retweet	social	2139	2786	TRUE	TRUE
rt_bahrain	retweet	social	4676	7977	TRUE	FALSE
rt_barackobama	retweet	social	9631	9772	TRUE	FALSE
rt_damascus	retweet	social	3052	3869	TRUE	TRUE
rt_dash	retweet	social	6288	7434	TRUE	FALSE
rt_gmanews	retweet	social	8373	8717	TRUE	FALSE
rt_gop	retweet	social	4687	5529	TRUE	FALSE
rt_http	retweet	social	8917	10314	TRUE	FALSE
rt_islam	retweet	social	4497	4616	TRUE	FALSE
rt_israel	retweet	social	3698	4164	TRUE	TRUE
rt_justinbieber	retweet	social	9405	9583	TRUE	FALSE
rt_ksa	retweet	social	7302	8107	TRUE	FALSE
rt_lebanon	retweet	social	3961	4435	TRUE	TRUE
rt_libya	retweet	social	5067	5540	TRUE	FALSE
rt_lolgop	retweet	social	9765	10075	TRUE	FALSE
rt_mittromney	retweet	social	7974	8534	TRUE	FALSE
rt_obama	retweet	social	3212	3422	TRUE	TRUE
rt_occupy	retweet	social	3225	3939	TRUE	TRUE
rt_occupywallstnyc	retweet	social	3609	3830	TRUE	FALSE
rt_oman	retweet	social	4904	6226	TRUE	FALSE
rt_onedirection	retweet	social	7987	8100	TRUE	FALSE
rt_p2	retweet	social	4902	6016	TRUE	FALSE
rt_qatif	retweet	social	7537	8559	TRUE	FALSE
rt_saudi	retweet	social	7252	8060	TRUE	FALSE
rt_tcot	retweet	social	4547	5501	TRUE	FALSE
rt_uae	retweet	social	5248	6385	TRUE	FALSE
rt_voteonedirection	retweet	social	2280	2464	TRUE	FALSE
rt-twitter-copen	retweet	social	761	1029	FALSE	TRUE
fb-pages-company	social	social	14115	52127	FALSE	TRUE

Table A.5: Real world networks used from Network Repository [36].

Network Name	Type	Meta Type	Nodes	Edges	Weighted	Discrete
fb-pages-food	social	social	620	2091	FALSE	TRUE
fb-pages-government	social	social	7057	89429	FALSE	TRUE
fb-pages-politician	social	social	5908	41706	FALSE	TRUE
fb-pages-public-figure	social	social	11565	67038	FALSE	TRUE
fb-pages-sport	social	social	13866	86811	FALSE	TRUE
fb-pages-tvshow	social	social	3892	17239	FALSE	TRUE
soc-advogato	social	social	6551	39432	TRUE	FALSE
soc-dolphins	social	social	62	159	FALSE	TRUE
soc-hamsterster	social	social	2426	16630	FALSE	TRUE
soc-karate	social	social	34	78	FALSE	TRUE
soc-tribes	social	social	16	58	TRUE	TRUE
soc-wiki-Vote	social	social	889	2914	FALSE	TRUE
socfb-Amherst41	facebook	social	2235	90954	FALSE	TRUE
socfb-Bowdoin47	facebook	social	2252	84387	FALSE	TRUE
socfb-Hamilton46	facebook	social	2314	96394	FALSE	TRUE
socfb-Haverford76	facebook	social	1446	59589	FALSE	TRUE
socfb-Mich67	facebook	social	3748	81903	FALSE	TRUE
socfb-nips-ego	facebook	social	2888	2981	FALSE	TRUE
socfb-Oberlin44	facebook	social	2920	89912	FALSE	TRUE
socfb-Reed98	facebook	social	962	18812	FALSE	TRUE
socfb-Simmons81	facebook	social	1518	32988	FALSE	TRUE
socfb-Smith60	facebook	social	2970	97133	FALSE	TRUE
socfb-Swarthmore42	facebook	social	1659	61050	FALSE	TRUE
socfb-USFCA72	facebook	social	2682	65252	FALSE	TRUE
socfb-Wellesley22	facebook	social	2970	94899	FALSE	TRUE
tech-as-caida2007	tech	infrastructure	26475	53381	FALSE	TRUE
tech-internet-as	tech	infrastructure	40164	85123	FALSE	TRUE
tech-pgp	tech	infrastructure	10680	24316	TRUE	TRUE
tech-WHOIS	tech	infrastructure	7476	56943	FALSE	TRUE
web-edu	web	infrastructure	3031	6474	FALSE	TRUE
web-EPA	web	infrastructure	4271	8909	TRUE	TRUE
web-indochina-2004	web	infrastructure	11358	47606	FALSE	TRUE
web-polblogs	web	infrastructure	643	2280	FALSE	TRUE
web-spam	web	infrastructure	4767	37375	FALSE	TRUE
web-webbase-2001	web	infrastructure	16062	25593	FALSE	TRUE
mammalia-macaque-contact-sits	animal_social	social	28	378	TRUE	TRUE
mammalia-primate-association	animal_social	social	25	280	TRUE	FALSE
mammalia-raccoon-proximity	animal_social	social	24	226	TRUE	TRUE
reptilia-lizard-network-social	animal_social	social	60	318	TRUE	FALSE
reptilia-tortoise-network-fi-2009	animal_social	social	185	237	FALSE	TRUE

Table A.6: Real world networks used from Network Repository [36].

Appendix B: Real-World Network Sources

[B1] H. Jeong, S.P. Mason, A.L. Barabasi, and Z.N. Oltvai. Lethality and centrality in protein networks. *arXiv preprint cond-mat/0105306*, 2001.

[B2] Rohit Singh, Jinbo Xu, and Bonnie Berger. Global alignment of multiple protein interaction networks with application to functional orthology detection. *PNAS*, 105(35):12763–12768, 2008.

[B3] Kwang-Il Goh, Michael E Cusick, David Valle, Barton Childs, Marc Vidal, and Albert-László Barabási. The human disease network. *Proceedings of the National Academy of Sciences*, 104(21):8685–8690, 2007.

[B4] J. Duch and A. Arenas. Community identification using extremal optimization phys. *Rev. E*, 72:027104, 2005.

[B5] Ara Cho, Junha Shin, Sohyun Hwang, Chanyoung Kim, Hongseok Shim, Hyojin Kim, Hanhae Kim, and Insuk Lee. Wormnet v3: a network-assisted hypothesis-generating server for caenorhabditis elegans. *Nucleic acids research*, 42(W1):W76–W82, 2014.

[B6] Katrin Amunts, Claude Lepage, Louis Borgeat, Hartmut Mohlberg, Timo Dickscheid, Marc-Étienne Rousseau, Sebastian Bludau, Pierre-Louis Bazin, Lindsay B. Lewis, Ana-Maria Oros-Peusquens, Nadim J. Shah, Thomas Lippert, Karl Zilles, and Alan C. Evans. Bigbrain: An ultrahigh-resolution 3d human brain model. *Science*, 340(6139):1472–1475, 2013.

[B7] Wouter De Nooy, Andrej Mrvar, and Vladimir Batagelj. *Exploratory social network analysis with Pajek*, volume 27. Cambridge University Press, 2011.

[B8] Vladimir Batagelj and Andrej Mrvar. Some analyses of Erdős collaboration graph. *Social Networks*, 22(2):173–186, 2000.

[B9] Robert E Ulanowicz and Donald L DeAngelis. Network analysis of trophic dynamics in south florida ecosystems. *FY97: The Florida Bay Ecosystem*, pages 20688–20038, 1998.

[B10] Carlos J Melián and Jordi Bascompte. Food web cohesion. *Ecology*, 85(2):352–358, 2004.

[B11] Luis E. C. Rocha, Fredrik Liljeros, and Petter Holme. Information dynamics shape the sexual networks of Internet-mediated prostitution. *Proc. of the National Academy of Sciences*, 107(13):5706–5711, 2010.

[B12] SocioPatterns. Infectious contact networks. `http://www.sociopatterns.org/datasets/`. Accessed 09/12/12.

[B13] T. Opsahl and P. Panzarasa. Clustering in Weighted Networks. *Social networks*, 31(2):155–163, 2009.

[B14] Ryan A. Rossi, David F. Gleich, Assefaw H. Gebremedhin, and Mostofa A. Patwary. What if clique were fast? maximum cliques in information networks and strong components in temporal networks. *arXiv preprint arXiv:1210.5802*, pages 1–11, 2012.

[B15] Ryan A. Rossi, David F. Gleich, Assefaw H. Gebremedhin, and Mostofa A. Patwary. Fast maximum clique algorithms for large graphs. In *Proceedings of the 23rd International Conference on World Wide Web (WWW)*, 2014.

[B16] Truthy. Information diffusion research at indiana university. `http://truthy.indiana.edu/`. Accessed 10/20/12.

[B17] Benedek Rozemberczki, Ryan Davies, Rik Sarkar, and Charles Sutton. Gemsec: Graph embedding with self clustering. In *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2019*, pages 65–72. ACM, 2019.

[B18] M. Fire, R. Puzis, and Y. Elovici. Link prediction in highly fractional data sets. *Handbook of Computational Approaches to Counterterrorism*, 2012.

[B19] Kenneth E Read. Cultures of the central highlands, new guinea. *Southwestern Journal of Anthropology*, pages 1–43, 1954.

[B20] Wayne W Zachary. An information flow model for conflict and fission in small groups. *Journal of anthropological research*, pages 452–473, 1977.

[B21] David Lusseau, Karsten Schneider, Oliver J Boisseau, Patti Haase, Elisabeth Slooten, and Steve M Dawson. The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations. *Behavioral Ecology and Sociobiology*, 54(4):396–405, 2003.

[B22] Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. Signed networks in social media. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1361–1370. ACM, 2010.

[B23] Hamsterster. Hamsterster social network. `http://www.hamsterster.com`.

[B24] Paolo Massa, Martino Salvetti, and Danilo Tomasoni. Bowling alone and trust decline in social network sites. In *Dependable, Autonomic and Secure Computing, 2009. DASC'09. Eighth IEEE International Conference on*, pages 658–663. IEEE, 2009.

[B25] Amanda L Traud, Peter J Mucha, and Mason A Porter. Social structure of Facebook networks. *Phys. A*, 391(16):4165–4180, Aug 2012.

[B26] Amanda L Traud, Eric D Kelsic, Peter J Mucha, and Mason A Porter. Comparing community structure to characteristics in online collegiate social networks. *SIAM Rev.*, 53(3):526–543, 2011.

[B27] P. Mahadevan, D. Krioukov, M. Fomenkov, X. Dimitropoulos, A. Vahdat, et al. The internet as-level topology: three data sources and one definitive metric. *SIGCOMM*, 36(1):17–26, 2006.

[B28] WHOIS. Internet routing registries. `http://www.irr.net/`.

[B29] Internet-AS. Oregon RouteViews project. `http://www.routeviews.org/`.

[B30] Ryan A. Rossi, Sonia Fahmy, and Nilothpal Talukder. A multi-level approach for evaluating internet topology generators. In *Networking*, pages 1–9, 2013.

[B31] Marián Boguñá, Romualdo Pastor-Satorras, Albert Díaz-Guilera, and Alex Arenas. Models of social networks based on social distance attachment. *Physical Review E*, 70(5):056122, 2004.

[B32] N. Spring, R. Mahajan, and D. Wetherall. Measuring ISP topologies with rocketfuel. In *SIGCOMM*, volume 32, pages 133–145, 2002.

[B33] Lada A Adamic and Natalie Glance. The political blogosphere and the 2004 us election: divided they blog. In *Proceedings of the 3rd international workshop on Link discovery*, pages 36–43. ACM, 2005.

[B34] David Gleich, Leonid Zhukov, and Pavel Berkhin. Fast parallel pagerank: A linear system approach. *Yahoo! Research Technical Report YRL-2004-038*, 13:22, 2004.

[B35] Paolo Boldi, Bruno Codenotti, Massimo Santini, and Sebastiano Vigna. UbiCrawler: A scalable fully distributed web crawler. *Software: Practice & Experience*, 34(8):711–726, 2004.

[B36] Paolo Boldi, Marco Rosa, Massimo Santini, and Sebastiano Vigna. Layered label propagation: A multiresolution coordinate-free ordering for compressing social networks. In *WWW*, pages 587–596, 2011.

[B37] Carlos Castillo, Kumar Chellapilla, and Ludovic Denoyer. Web spam challenge 2008. In *Proceedings of the 4th International Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, 2008.

Bibliography

[1] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. *Theory of Computing*, 11(4):105–147, 2015.

[2] Pedro Domingos and Matt Richardson. Mining the network value of customers. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 57–66. ACM, 2001.

[3] Youze Tang, Xiaokui Xiao, and Yanchen Shi. Influence maximization: Near-optimal time complexity meets practical efficiency. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, pages 75–86, New York, NY, USA, 2014. ACM.

[4] Gal Sadeh, Edith Cohen, and Haim Kaplan. Influence maximization with few simulations. *arXiv preprint arXiv:1907.13301*, 2019.

[5] Grant Schoenebeck and Biaoshuai Tao. Beyond worst-case (in) approximability of nonsubmodular influence maximization. *ACM Transactions on Computation Theory (TOCT)*, 11(3):12, 2019.

[6] Qiang Li, Wei Chen, et al. Influence maximization with ϵ-almost submodular threshold functions. In *Advances in Neural Information Processing Systems*, pages 3801–3811, 2017.

[7] Daniel M Romero, Brendan Meeder, and Jon Kleinberg. Differences in the mechanics of information diffusion across topics: idioms, political hashtags, and complex contagion on twitter. In *Proceedings of the 20th international conference on World wide web*, pages 695–704. ACM, 2011.

[8] Johan Ugander, Lars Backstrom, Cameron Marlow, and Jon Kleinberg. Structural diversity in social contagion. *Proceedings of the National Academy of Sciences*, 109(16):5962–5966, 2012.

[9] Márton Karsai, Gerardo Iniguez, Kimmo Kaski, and János Kertész. Complex contagion process in spreading of online innovation. *Journal of The Royal Society Interface*, 11(101):20140694, 2014.

[10] Albert-László Barabási et al. *Network science*. Cambridge university press, 2016.

[11] Duanbing Chen, Linyuan Lü, Ming-Sheng Shang, Yi-Cheng Zhang, and Tao Zhou. Identifying influential nodes in complex networks. *Physica a: Statistical mechanics and its applications*, 391(4):1777–1787, 2012.

[12] Byungjoon Min and Maxi San Miguel. Competing contagion processes: Complex contagion triggered by simple contagion. *Scientific reports*, 8(1):10422, 2018.

[13] Jan Vondrák. Submodularity in combinatorial optimization. 2007.

[14] Wei Chen, Yifei Yuan, and Li Zhang. Scalable influence maximization in social networks under the linear threshold model. In *2010 IEEE international conference on data mining*, pages 88–97. IEEE, 2010.

[15] Chi Wang, Wei Chen, and Yajun Wang. Scalable influence maximization for independent cascade model in large-scale social networks. *Data Mining and Knowledge Discovery*, 25(3):545–576, 2012.

[16] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical programming*, 14(1):265–294, 1978.

[17] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 420–429. ACM, 2007.

[18] Wei Chen, Yajun Wang, and Siyu Yang. Efficient influence maximization in social networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 199–208. ACM, 2009.

[19] Wei Chen, Chi Wang, and Yajun Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1029–1038. ACM, 2010.

[20] Christian Borgs, Michael Brautbar, Jennifer Chayes, and Brendan Lucier. Maximizing social influence in nearly optimal time. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 946–957. SIAM, 2014.

[21] Mohammad Akbarpour, Suraj Malladi, and Amin Saberi. Just a few seeds more: value of network information for diffusion. *Available at SSRN 3062830*, 2018.

[22] G Moores, P Shakarian, B Macdonald, and N Howard. Finding near-optimal groups of epidemic spreaders in a complex network. *PLoS ONE*, 2014.

[23] Stephen P Borgatti. Identifying sets of key players in a social network. *Computational & Mathematical Organization Theory*, 12(1):21–34, 2006.

[24] Mark EJ Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004.

[25] Santo Fortunato and Darko Hric. Community detection in networks: A user guide. *Physics reports*, 659:1–44, 2016.

[26] J-J Daudin, Franck Picard, and Stéphane Robin. A mixture model for random graphs. *Statistics and computing*, 18(2):173–183, 2008.

[27] Martin Rosvall, Daniel Axelsson, and Carl T Bergstrom. The map equation. *The European Physical Journal Special Topics*, 178(1):13–23, 2009.

[28] Christophe Biernacki, Gilles Celeux, and Gérard Govaert. Assessing a mixture model for clustering with the integrated completed likelihood. *IEEE transactions on pattern analysis and machine intelligence*, 22(7):719–725, 2000.

[29] Brian Karrer and Mark EJ Newman. Stochastic blockmodels and community structure in networks. *Physical review E*, 83(1):016107, 2011.

[30] Etienne Côme and Pierre Latouche. Model selection and clustering in stochastic block models based on the exact integrated complete data likelihood. *Statistical Modelling*, 15(6):564–589, 2015.

[31] Tiago P Peixoto. Efficient monte carlo and greedy heuristic for the inference of stochastic block models. *Physical Review E*, 89(1):012804, 2014.

[32] Edoardo M Airoldi, David M Blei, Stephen E Fienberg, and Eric P Xing. Mixed membership stochastic blockmodels. *Journal of machine learning research*, 9(Sep):1981–2014, 2008.

[33] Qiuyi Han, Kevin Xu, and Edoardo Airoldi. Consistent estimation of dynamic and multi-layer block models. In *International Conference on Machine Learning*, pages 1511–1520, 2015.

[34] Tiago P Peixoto. Hierarchical block structures and high-resolution model selection in large networks. *Physical Review X*, 4(1):011047, 2014.

[35] A. Eriksson D. Edler and M. Rosvall. Mapequation.

[36] Ryan A. Rossi and Nesreen K. Ahmed. The network data repository with interactive graph analytics and visualization. In *AAAI*, 2015.

[37] Gabor Csardi, Tamas Nepusz, et al. The igraph software package for complex network research. *InterJournal, complex systems*, 1695(5):1–9, 2006.

[38] Tiago P. Peixoto. The graph-tool python library. *figshare*, 2014.

[39] Brian Karrer, Elizaveta Levina, and Mark EJ Newman. Robustness of community structure in networks. *Physical review E*, 77(4):046119, 2008.

[40] John Palowitch, Shankar Bhamidi, and Andrew B Nobel. The continuous configuration model: A null for community detection on weighted networks. *arXiv preprint arXiv:1601.05630*, 2016.

[41] Aaron Clauset, Cosma Rohilla Shalizi, and Mark EJ Newman. Power-law distributions in empirical data. *SIAM review*, 51(4):661–703, 2009.

[42] John H Holland. Genetic algorithms. *Scientific american*, 267(1):66–73, 1992.

[43] Mitsuo Gen, Runwei Cheng, and Lin Lin. *Network models and optimization: Multiobjective genetic algorithm approach*. Springer Science & Business Media, 2008.

[44] Ting Yee Lim. Structured population genetic algorithms: a literature survey. *Artificial Intelligence Review*, 41(3):385–399, 2014.

[45] Kaiqi Zhang, Haifeng Du, and Marcus W Feldman. Maximizing influence in a social network: Improved results using a genetic algorithm. *Physica A: Statistical Mechanics and its Applications*, 478:20–30, 2017.

[46] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[47] Laizhong Cui, Huaixiong Hu, Shui Yu, Qiao Yan, Zhong Ming, Zhenkun Wen, and Nan Lu. Ddse: A novel evolutionary algorithm based on degree-descending search strategy for influence maximization in social networks. *Journal of Network and Computer Applications*, 103:119–130, 2018.

[48] Rico Angell and Grant Schoenebeck. Don't be greedy: Leveraging community structure to find high quality seed sets for influence maximization. In *International Conference on Web and Internet Economics*, pages 16–29. Springer, 2017.