

Fast algorithms to improve fair information access in networks

Dennis Robert Windham¹, Caroline J. Wendt¹, Alex Crane², Madelyn J Warr², Freda Shi², Sorelle A. Friedler³, Blair D. Sullivan², Aaron Clauset^{1,4,5}

¹Department of Computer Science, University of Colorado

²University of Utah

³Haverford College

⁴BioFrontiers Institute, University of Colorado

⁵Santa Fe Institute

ABSTRACT

We consider the problem of selecting k seed nodes in a network to maximize the minimum probability of activation under an independent cascade beginning at these seeds. The motivation is to promote fairness by ensuring that even the least advantaged members of the network have good access to information. Our problem can be viewed as a variant of the classic influence maximization objective, but it appears somewhat more difficult to solve: only heuristics are known. Moreover, the scalability of these methods is sharply constrained by the need to repeatedly estimate access probabilities.

We design and evaluate a suite of 10 new scalable algorithms which crucially do not require probability estimation. To facilitate comparison with the state-of-the-art, we make three more contributions which may be of broader interest. We introduce a principled method of selecting a pairwise information transmission parameter used in experimental evaluations, as well as a new performance metric which allows for comparison of algorithms across a range of values for the parameter k . Finally, we provide a new benchmark corpus of 174 networks drawn from 6 domains. Our algorithms retain most of the performance of the state-of-the-art while reducing running time by orders of magnitude. Specifically, a meta-learner approach is on average only 20% less effective than the state-of-the-art on held-out data, but about 75 – 130 times faster. Further, the meta-learner’s performance exceeds the state-of-the-art on about 20% of networks, and the magnitude of its running time advantage is maintained on much larger networks.

1 INTRODUCTION

Influence maximization [11, 30] is one of the most intensively studied problems in data mining, machine learning, and social network analysis. Given some model of information diffusion, commonly the *independent cascade* of Kempe, Kleinberg, and Tardos [30], the task is to determine where we should seed information such that it spreads as widely as possible. Formally, given a graph G and $S \subseteq V(G)$, for each $i \in V(G)$ let $\pi_i(S)$ denote the probability with which i is activated by an independent cascade seeded at S ; we will just write π_i when the set S is clear. Then for a given budget k , influence maximization is the problem of identifying a set $S \subseteq V(G)$ of cardinality k which maximizes $\sum_{i \in V(G)} \pi_i$.

Influence maximization is commonly motivated by commercial advertising, e.g., [11, 15, 40] but is also relevant in various other applications, including public health [31, 51], the distribution of economic opportunities [2], and the spread of scientific knowledge [41]. In these applications, the desiderata include not only widespread

but also *fair* dissemination of information. The difficulty of the latter goal is a natural consequence of structural heterogeneity in networks. Highly-connected or centrally located individuals have more opportunities to receive and spread information [12, 38, 49], while peripheral individuals with few connections participate less often in information exchanges [18, 38]. For example, in a pandemic, access to crucial resources—such as money, food and healthcare—is more difficult for socially disadvantaged groups, in part due to their more limited connectedness in social networks [20]. Similarly, connectedness shapes employment opportunities in professional social networks such as LinkedIn: well-connected job seekers are likely to fill lucrative openings sooner than others [50]. This situation has motivated the study of several variations of influence maximization, for example to ensure equitable information access with respect to demographic groups [45]. In this paper, we focus on the formulation of Fish *et al.* [18], which adopts a *Rawlsian* [39] notion of fairness in which the goal is to improve the information access of the worst-off individuals. Formally, the problem is as follows:

MAXIMIN INFLUENCE MAXIMIZATION

Input: A graph $G = (V, E)$ and an integer k .

Task: Select k vertices $S \subseteq V$ maximizing $\min_{i \in V} \pi_i$.

This maximin variant is NP-hard, and even a constant-factor approximation would imply $P = NP$ [18], in stark contrast to the greedy $(1 - 1/e)$ -approximation for the standard objective [30]. However, Fish *et al.* show that a heuristic approach (Myopic; see Section 3.1) optimizes the objective well in practice. Indeed, this approach even performs well when evaluated via the classic influence maximization objective, which it does not directly optimize. Unfortunately, this heuristic relies upon repeated Monte Carlo simulations of the activation probabilities π_i , which are a severe computational bottleneck and thereby constrain the scalability of the method; see Figure 1. Additionally, the method has only been evaluated on a small corpus of six networks.

We are thus motivated to (a) introduce new algorithms which alleviate or completely avoid the bottleneck imposed by probability estimation; (b) compare the performance of our methods against that of Myopic in a principled manner; and (c) expand the number and diversity of networks on which the MAXIMIN INFLUENCE MAXIMIZATION problem has been studied.

1.1 Related Work and Preliminaries

Fairness in ML and Social Networks. Fairness in machine learning is a well-studied area, and includes notions of fairness based both on individual characteristics and on group demographics [16, 17, 26]

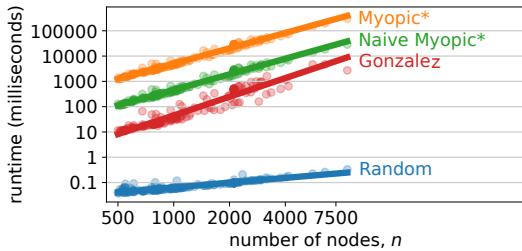


Figure 1: Algorithm runtime to select 10 new seeds vs. network size for algorithms in [18], averaged over 10 runs on an introduced large set of networks (see Section 2.1). Algorithms requiring a Monte Carlo simulation (ProbEst) to select seeds are denoted by a *.

(for surveys, see [10, 37]). Recently, questions of fairness of information access of individuals and demographic groups in a network have come to the fore [1, 18, 32, 43–45, 48]. A significant body of work focuses on the probability that an individual receives information that spreads through a network. This includes studies of seeding information at nodes to improve individual [18] or group [44, 48] access; as well as interventions to add edges [4–6] under varying notions of fairness. Other work considers notions of group fairness based on the network structure [3, 29, 32]. Saxena, Fletcher, and Pechenizkiy provide a recent survey of such work [42]. Fair clustering of individuals has also received significant attention, e.g., [13].

Independent Cascade. Considerations about the access of individuals to resources in a network build on structural concerns about social networks pioneered by Granovetter [24]. Necessary in any such study is a clear model for the dynamics of information propagation. Numerous models exist, notably including the independent cascade, generalized independent cascade, and linear threshold models [22, 23, 30]. A standard choice, also adopted in this paper, is the independent cascade, which can be defined via an iterative spreading process. At each step i , some subset S_i of nodes is activated, beginning with an initial seed set $S = S_0$. In round $i + 1$, each edge uv with $u \in S_i, v \notin \bigcup_{j \leq i} S_j$ activates v independently with probability α . The process stops after the first round i in which no nodes are activated, i.e., $S_i = \emptyset$, and a node v is said to be activated by the cascade if it is activated in any round, i.e., $v \in \bigcup S_i$.

Probability Estimation. Important in any analysis of independent cascades is the ability to measure the probability π_i with which node i is activated. Unfortunately, exact computation of π_i is #P-hard [11]. The standard approach is to estimate these values via *reverse influence sampling* (RIS) [8, 47], which may also be thought of as performing a series of Monte Carlo simulations of the cascade process. Theoretical bounds on the number of simulations needed to satisfy a given error tolerance ε have a quadratic dependence on ε in addition to quasilinear dependence on network size [47], and so in practice it is common to fix a reasonable number R of simulations, e.g., $R = 1000$. Algorithms for influence maximization and related problems then invoke a linear-time (regarding R as a constant) subroutine, referred to here as ProbEst, to provide estimated π_i values. In practice, however, algorithms requiring

such a subroutine can be orders of magnitude slower than those avoiding probability estimation altogether; we again refer to Figure 1. In the influence maximization literature, much effort has been expended to lessen the complexity of ProbEst while retaining quality guarantees, e.g., [27, 34, 46], as well as to empirically compare various strategies [35]. In this work, our approach is to develop methods which avoid RIS entirely and evaluate the resulting solution quality via empirical comparison against the state-of-the-art. A purely heuristic approach, while difficult to accept for the classic influence maximization problem, is palatable in our setting because strong inapproximability bounds are known for MAXIMIN INFLUENCE MAXIMIZATION even in the presence of an oracle which perfectly computes the π_i values [18].

1.2 Summary of Contributions

Network Corpus. In Section 2.1, we introduce a large and structurally diverse corpus of 174 networks on which to benchmark both our algorithms and the state-of-the-art. These networks are drawn from six domains: biological, social, economic, technological, transportation, and informational. Though our primary interest is in social networks, building a corpus from multiple domains enhances the structural diversity of our benchmark set; see the discussion in Section 2.1. Where relevant, we report results by domain.

Spreadability. When evaluating algorithms in the presence of an independent cascade, it is necessary to choose a value for the parameter α which governs the probability of information transmission along edges. It is common in the literature to choose several values, under the assumption that small values fundamentally represent “low” information spread while large values represent “high” spread. However, this qualitative assessment of a particular α value is not consistent across networks, due to the varying underlying combinatorial structure. We introduce *spreadability*, which enforces mathematical rigor in the assessment of a particular α value as “low”- or “high”-spreading for a given network. We use this framework to select appropriate α values for each network in our corpus, ensuring that we can measure algorithm performance under multiple distinct regimes of information spread.

Performance Metric. Even with a fixed α value and a single network, it is non-trivial to compare the performance of algorithms. Among other factors, this is because we must account for varying choices of budget k , random number generator seeding, and the impracticality of precisely measuring the problem objective. We introduce a performance metric which incorporates all of these factors, and allows us to evaluate the performance of an algorithm via a single number β . Intuitively, β indicates the marginal improvement in the problem objective that we can expect if we use a given algorithm to add an additional information seed. We defer a more formal description to Section 2.3.

Algorithms. In Section 3, we introduce 10 new algorithms for the MAXIMIN INFLUENCE MAXIMIZATION problem, which we partition into three categories. The first two replace the Monte Carlo simulations used by the state-of-the-art to estimate activation probabilities with heuristics based on breadth-first search and personalized page rank. The complexity of our subroutines is similar to the Monte Carlo approach, but the hidden constants are vastly improved, and

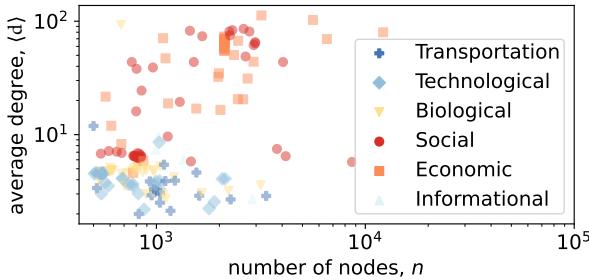


Figure 2: Average degree of a network as a function of network size (number of nodes) for the corpus of 174 networks from 6 distinct domains used in our study.

this is reflected in improved running times. The third category of algorithms eliminates probability estimation entirely, instead using topological features of the network to inform seed selection.

Evaluation, Meta-Learning, and Scalability. In Section 4, we conduct a comprehensive evaluation of our methods, together with a comparison against the state-of-the-art (Myopic [18]). In Section 4.1, we show that on most networks at least one of our algorithms is nearly as effective as or even more effective than Myopic. Meanwhile, our algorithms are much faster, improving on the running time of Myopic by 1-4 orders of magnitude; see Section 4.2. We build on these insights by introducing a meta-learner (see Section 4.3), which uses structural features of a network to predict which of our algorithms will be most effective on that network. This approach allows us to recover roughly 80% of the performance of Myopic while reducing running time by factors of about 75 – 130. Finally, in Section 4.4 we show that these improvements in running time persist even on several much larger networks.

2 NETWORK CORPUS AND EVALUATION METHODS

2.1 Network Corpus

In order to investigate the effects of network structure on algorithm performance, we construct a corpus of 174 networks from six domains: biological (34), social (44), economic (43), technological (32), transportation (17), and informational (4). We include non-social networks in our study in order to more fully characterize the behavior of algorithms on structurally diverse real-world networks. When relevant, we report our results by domain, so that social and non-social networks can be contrasted. An overview of the corpus is presented in Figure 2, with summary statistics in Table S1.

Networks were curated from the Index of Complex Networks [14], a large-scale index of research-quality networks spanning all domains of science, as well as the Netzschleuder network catalogue and repository [36] and the corpus of Ghasemian, Hosseini Mardi, and Clauset [21]. All networks included in our corpus are simple graphs, meaning their edges are undirected, unweighted, and there are no self-loops. Further, they are unipartite, i.e., they only have one type of node.

Past work on such corpora indicates that domains (and even sub-domains, e.g., online social networks vs. offline social networks) are

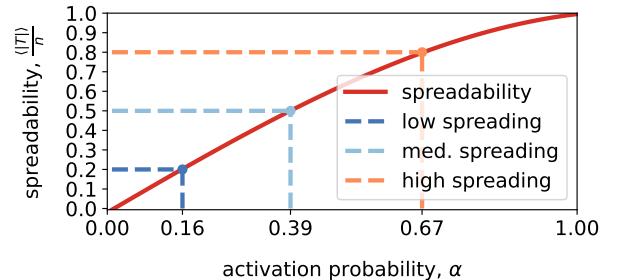


Figure 3: Spreadability on a network is quantified by the average fraction of a network’s nodes $\langle |T| \rangle / n$ in a tree T grown through an independent cascade from a random initial seed for a given α . We define ‘low’, ‘medium’, and ‘high’ spreadability as the α that activates, on average, 20%, 50%, and 80% of the network, respectively.

highly distinguishable based on their structure alone [28]. Hence, a specific effort was made to (i) balance the classes, so that no domain was more than 25% of the corpus, (ii) avoid over-representing networks from particular sources (e.g., Twitter follower networks), and (iii) ensure that the minimum network size was large enough to provide good results for information spreading tasks (minimum number of nodes $n_{\min} = 500$). These choices improve the breadth and variety of network structure represented in the corpus, and its utility in the analyses of this study.

2.2 Spreadability

Background. In our work, we leverage the independent cascade model [30] to study the spread of information in networks and estimate π_i . Given a network G and a set of activated nodes $Q \subseteq V$, we grow a forest on G under the independent cascade model by flipping a coin for each edge e_{ij} , where $i \in V \setminus Q$, $j \in Q$, exactly once, so that i is added to Q on a successful flip. Each edge e_{ij} is considered as a transmission path at most once, and we define the probability of successful transmission to be α . We note that mathematically, it is equivalent to conceptualize the independent cascade as follows: delete every edge in the graph independently with probability $(1 - \alpha)$; the set Q of activated nodes is exactly the set of nodes that remain reachable from the seed set S . As noted earlier, computing the exact probability of activation for a node i , denoted π_i , is #P-hard [11]. As such, we adopt the standard Monte Carlo simulation approach, but return later to consider practical consequences of this choice.

Briefly, using ProbEst, given transmission probability α , number of simulation rounds R , and a seed set S , we estimate π_i for every $i \in V$ using R independent cascades originating from S [18]. The worst-case time complexity of ProbEst is $O(R(|S| + 2m))$ [18], where m is the number of edges in G . In practice, the computational cost of ProbEst increases both as the seed set grows and as α increases, because both changes tend to increase the size of the induced information cascades. Past work used $R = 1000$ as a balance between statistical accuracy for π_i and computational cost, and we follow this precedent [18]. Here, ProbEst is used as a subroutine

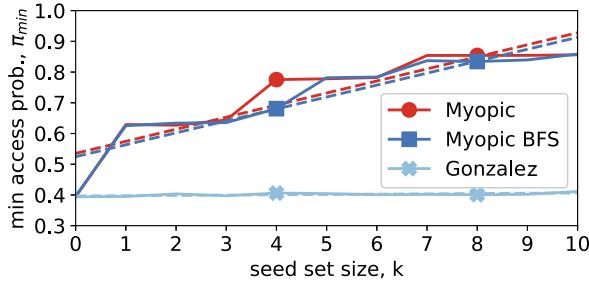


Figure 4: Minimum access probability π_{\min} vs. seed set size k , with a best fit line (Myopic $\hat{\beta} = 0.039$), averaged over 20 runs, evaluated on a large economic network ($n = 2113$ nodes, $m = 57927$ edges), with $\alpha = 0.4$ and a budget of $k = 10$ seeds, plus one random initial seed.

in some algorithms, as well as to evaluate the performance of algorithms by estimating the achieved minimum activation probability of a computed seed set.

Selecting α values. Prior work evaluated algorithm performance using transmission probabilities $\alpha \in \{0.3, 0.4, 0.5\}$ [18]. However, the resulting information cascades, and hence the associated access probabilities, are not a simple function of α ; they instead also depend on the network’s structure. For example, denser, more connected networks contain many more paths by which information can spread than do sparse networks. Hence the same α will tend to produce larger cascades on the former, and smaller cascades on the latter. To control for these structure-induced differences in information cascades, we introduce the concept of *spreadability*, which jointly accounts for the impact of network structure and transmission rate α on the sizes of information cascades.

We can construct a fine-grained spreadability function that relates a particular choice of α to the fraction of a network activated under the independent cascade model from a uniformly random initial seed. For a given α , the spreadability $f(\alpha)$ on a particular network the fraction of nodes that are activated from a uniformly random initial seed, averaged over R trials. This calculation produces a monotonically increasing curve, as seen in Figure 3. We find that computing spreadability for each $\alpha \in \{0.01, 0.02, \dots, 0.99\}$ provides ample resolution to choose α close to target spreadabilities of 0.2, 0.5, 0.8 (which we refer to as “low,” “medium,” and “high” spreadabilities respectively). We set $R = 1000$, as the spreadability curve tends to stabilize near this value and we get diminishing returns for larger R .

2.3 New Metric for Algorithm Evaluation

As discussed in Section 1.2, it is not straightforward to compare the performance of algorithms, even with a fixed network and α value. There are several reasons for this. First, we do not want to make strong assumptions about the “most useful” value for the parameter k (the number of seeds to be added), and it is possible that algorithm \mathcal{A} obtains a better objective score (the minimum activation probability π_{\min}) than algorithm \mathcal{A}' for one value of k , but not for another. Moreover, many algorithms (in particular those

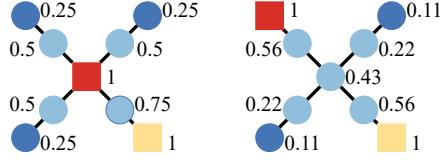


Figure 5: Illustrations of two runs of Myopic for different initial seeds (red), with new selected seeds (yellow), and fixed $\alpha = 0.5$. Numbers indicate π for each node after the new seed is selected. Initialization significantly affects the performance of Myopic.

using ProbEst) have some randomness. Finally, even our analysis of the quality of a solution is inexact; recall that exact computation of the π_{\min} achieved by a seed set S is #P-hard [11]. Thus, our ability to compare two solutions is limited by the precision of ProbEst.

We overcome these challenges by introducing a metric β , which intuitively measures the marginal gain in the problem objective (the minimum activation probability π_{\min}) which we can expect when asking an algorithm to add one additional seed. Formally, for a given algorithm \mathcal{A} , network G , independent cascade parameter α , and budget k , we run \mathcal{A} on (G, α, k) and record a value $\pi_{\min}(k)$ indicating the achieved objective score. We repeat for each $k \in \{1, 2, \dots, 10\}$, computing the points $(k, \pi_{\min}(k)) \in \mathbb{N} \times [0, 1]$. We then record the slope β_1 of the line of best fit for these points. We repeat the entire process multiple (generally 20) times, producing slopes β_1, β_2, \dots . The metric β is the mean of these values, i.e., the average slope of the line of best fit; see Figure 4. Henceforth, when comparing the performance of algorithms, we do so via the metric β .

3 ALGORITHMS FOR FAIR INFORMATION ACCESS

3.1 Algorithms from Prior Work

Four previously introduced heuristics for choosing seed nodes to maximize π_{\min} are Greedy, Myopic, Naive Myopic and Gonzalez, along with Random as a baseline for comparison [18]. Random selects k seeds by uniformly sampling nodes from the network. Given a partial seed set, Gonzalez selects as the next seed the node that is the furthest (in the shortest-path metric) from all nodes in the current set [18]. The Greedy algorithm iteratively selects a new seed by choosing the node with the highest marginal gain relative to the current seed set according to ProbEst. Due to its immensely high computational cost, Greedy is not practical for most networks [18], and is not used in our study.

In contrast, Myopic uses ProbEst with the current seed set, and selects as the next seed the node with the lowest π_i . Naive Myopic is similar to Myopic, but only runs ProbEst once, at initialization, and then selects as the seed set the k nodes with the lowest π_i values. In past work, on a small set of networks, Myopic was found to perform best. However, because Myopic depends on ProbEst, it is computationally expensive, which limits its applicability to large networks. In practice, Myopic is always the second slowest algorithm, after Greedy [18].

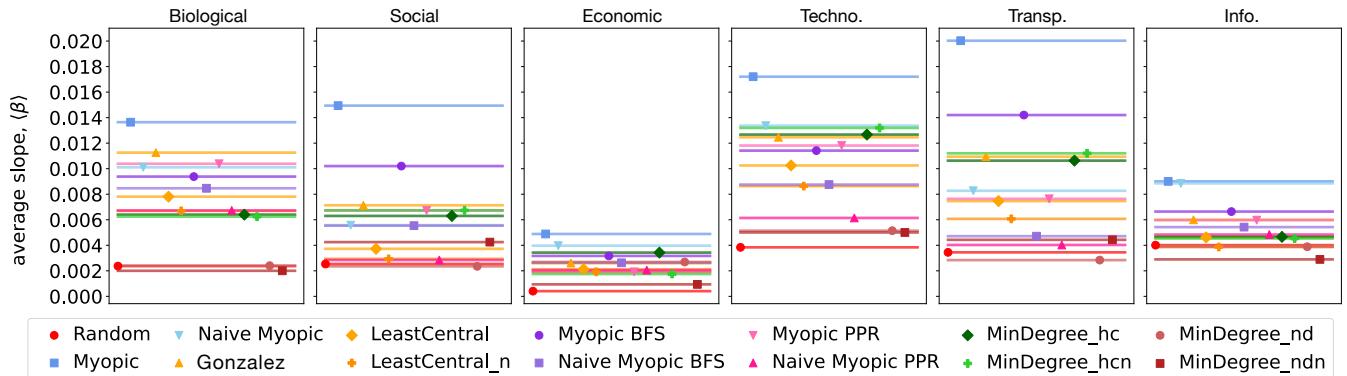


Figure 6: Mean performance of the intervention algorithms on each domain in the corpus under medium spreadability. Each algorithm’s performance is averaged over a given domain, 20 runs per network.

Algorithm Initialization. Myopic, Naive Myopic, and Gonzalez all start with an initial seed. Past work chose this initial seed to be the highest degree node. Our initial experiments indicate that this choice confers a substantial advantage to these methods (Figure 5), and that some of the previous positive results are thus attributable to this initial seed choice rather than to the algorithms’ subsequent choices. To mitigate this bias we instead initialize all heuristics with a seed set composed of a single uniformly randomly selected initial seed. Moreover, for each evaluation round on a particular network, we initialize all algorithms to use the same random seed, which further controls for the effects of different initial seed choices. This initial seed choice is not counted against the budget k (Figure 4).

3.2 New Fast Algorithms

In addition to the four heuristics from prior work, here we introduce 10 new heuristics to maximize the minimum π_i on a budget. These heuristics are designed to be computationally lightweight, scaling to far larger networks, while also matching or exceeding the performance of Myopic on the corpus. We group the new algorithms into three families: BFS-based, PPR-based and Topology-based.

BFS-based: Myopic BFS and Naive Myopic BFS. In the two BFS-based heuristics, Myopic BFS and Naive Myopic BFS, we swap the ProbEst component of Myopic and Naive Myopic with a simple breadth-first search to estimate π_i . The breadth-first search component is initialized with a random seed k_0 , and transmission probability α . It proceeds to “peel” the network, starting at k_0 , in breadth-first fashion, estimating π_i for each node as the probability that i receives a transmission from k_0 through any nodes it connects to in the previous BFS layer, as well as through nodes it is connected to in its own layer. All subsequent iterations update existing π_i estimates during the breadth-first traversal from new candidate seeds.

While this approach does not exactly measure π_i , it is much faster than ProbEst, taking $O(n \cdot \langle k \rangle)$ per iteration, because for most networks, including those in our corpus, the mean degree $\langle k \rangle \ll 1000$ (Figure 2). The key design principle of this algorithm is to capture complex network structures that Gonzalez could not account for. In Figure 4, we find that Myopic BFS almost matches Myopic’s performance in situations where Gonzalez lags behind.

PPR-based: Myopic PPR and Naive Myopic PPR. In the PPR-based heuristics, Myopic PPR and Naive Myopic PPR, we use Personalized PageRank (PPR) to estimate π_i instead of ProbEst or BFS. Personalized PageRank (implemented with networkx [25]) performs a random walk that probabilistically restarts from nodes in the seed set [9]. The ranking produced by PPR is not a direct estimate of π_i ; we treat the PPR values as being correlated, such that a lower PPR score is a proxy for a lower π_i value.

Topology-based: LeastCentral and MinDegree. These heuristics are based on the intuition that nodes with low π_i values have distinctive structural positions or patterns of connectivity. LeastCentral selects the non-seed node i with the lowest closeness centrality c_i as the next seed. Similarly, LeastCentral_n selects the lowest centrality node i ’s highest degree neighbor as the next seed. Here, the closeness centrality of a node i is the inverse of the average shortest path length from i to all other reachable nodes in the same connected component [19, 33]. Lower closeness centrality implies the node is less reachable by the rest of the network, and therefore is expected to have lower π_i .

The four remaining topology-based heuristics exploit a network’s degree structure to make decisions, based on the observation that in practice Myopic tends to select nodes with low degree as seeds. The first two heuristics take a non-seed node with the lowest degree, breaking ties by choosing the node with the lowest harmonic centrality [7, 33] among same-degree nodes. In the case of MinDegree_hc, it chooses that node itself as the next seed, while MinDegree_hcn chooses the highest-degree neighbor of that node. The two remaining variations replace harmonic centrality in the aforementioned logic with the neighbor degree, i.e. breaking ties by choosing the node with the highest neighbor degree (sum of degrees across neighbors). These heuristics are called MinDegree_nd and MinDegree_ndn, respectively.

4 EXPERIMENTAL RESULTS

4.1 Performance of Algorithms on the Corpus

We evaluate and compare the performance of 14 algorithms in total (10 new algorithms and 4 from prior work), applied to all 174 networks in the corpus. Our code and data are available at

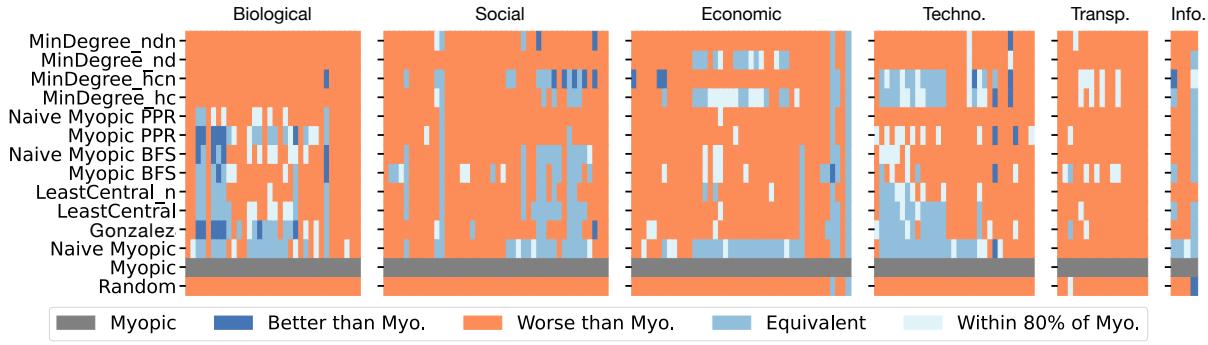


Figure 7: Performance of intervention algorithms on the network corpus, relative to Myopic and sorted in ascending order by network size within each domain. Under medium spreadability, 24% of networks have no algorithm better than or within 80% of Myopic’s performance. “Equivalent” is defined as within one standard error of β for Myopic; typically about 0.001.

<https://github.com/TheoryInPractice/FairInfoAccessHeuristics>. We are interested in algorithms that operate well on a tight budget and so let $k \in \{1, 2, \dots, 10\}$ seed nodes. For each spreadability level (low, medium, high), we produce a $10 \times 14 \times 174$ matrix, where each entry is the β performance of an algorithm after adding k seeds in a network, averaged over 20 runs. We focus on the medium spreadability regime here, and include results for low and high spreadability in Appendix C. Figure 6 displays the mean performance of each algorithm on networks by domain under medium spreadability (see Figure S1 for low and high spreadability results). Across domains, Myopic produces the best average performance.

To further examine the results, we sort within each domain by network size in ascending order, and then score algorithms as better than, “equivalent” to (within one std. err.), within 80% of, or worse than Myopic (see Figure 7; low and high spreadability results in Figure S2). In this way, we can assess whether the better average performance of Myopic applies to individual networks compared to other algorithms. This experiment reveals that Myopic is not universally the best algorithm for all networks in any spreadability setting; on many individual networks other algorithms perform equivalently or better. In the medium spreadability setting, for only 24% of networks is there no algorithm that performs at least 80% as well as Myopic (Figure 7).

This variability in performance across networks suggests that network structure plays a critical role in governing the relative performance of different algorithms. Figure 8 plots the instances for which each algorithm was the best performing algorithm on a network against that network’s mean degree $\langle k \rangle$. Myopic tends to perform better on networks with lower average degree, although it also performs well on many networks with larger mean degree. In practice, we find that the final seed set produced by Myopic is often composed primarily of low-degree nodes located in a network’s periphery, and these nodes may be too far removed from other disadvantaged nodes to meaningfully improve their π_i values.

A second takeaway is that a few specific algorithms tend to perform better than Myopic in certain settings (Figure 8), specifically MinDegree_hcn and Gonzalez. The performance of MinDegree_hcn in particular tends to improve over Myopic with increasing average degree, while Gonzalez does best in networks with very low

mean degrees. Furthermore, we note that in Figure S1a, on average MinDegree_hcn outperforms Myopic in the economic domain, and from Table S1, we see that the economic domain has the highest mean degrees. The MinDegree_hcn algorithm selects as seeds the highest-degree neighbors of low-degree nodes. As result, new information cascades seeded at these nodes will tend to spread quickly to a number of disadvantaged nodes in the network.

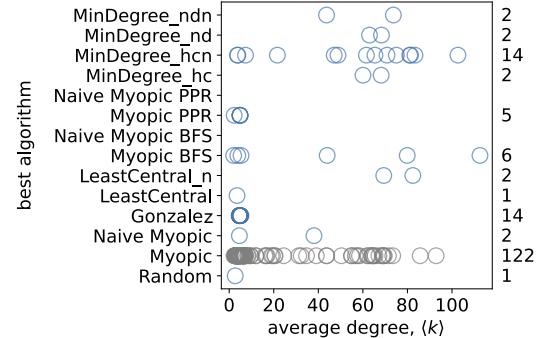


Figure 8: Best-performing algorithm vs. mean degree of the network (medium spreadability), for all networks. Counts on the right show total circles per line, i.e., the number of times an algorithm was the best over the whole corpus.

4.2 Algorithm Runtime

We evaluate all algorithm runtimes on the corpus, selecting $k=10$ new seeds, averaged over 10 runs. For fair comparison, we run all algorithms on a single core of an AMD Ryzen 5900X, overclocked to 5.00Ghz, with 32GB RAM, and measure the runtime in milliseconds. Two major performance bottlenecks are ProbEst, used by Myopic and Naive Myopic, and an All-Pairs-Shortest-Paths (APSP) computation, used by Gonzalez, LeastCentral, LeastCentral_n, MinDegree_hc, and MinDegree_hcn. Both ProbEst and APSP have efficient parallel implementations, but we restrict them to a single core to ensure fair comparison with other algorithms.

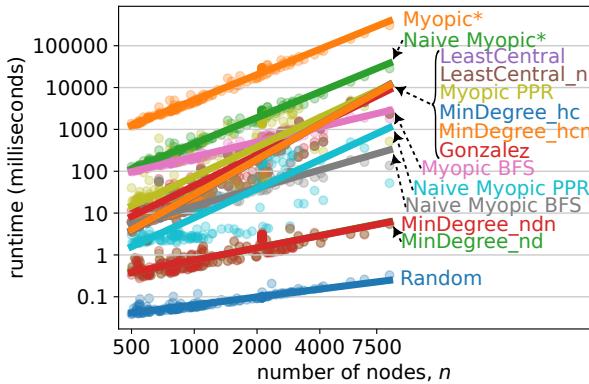


Figure 9: Runtime of old and new algorithms across the network corpus with $k = 10$ seeds averaged over 10 runs, showing a substantial advantage in running time for the new algorithms. Note: algorithms with * require ProbEst.

All of the new algorithms are substantially faster than Myopic and Naive Myopic (Figure 9). Because they only require sorting two lists, MinDegree_ndn and MinDegree_nd are the most efficient, improving running times over ProbEst-based algorithms by a factor of 1000-10000x, depending on the size of the network (Figure 9). BFS-based algorithms are marginally slower than these fastest algorithms, and algorithms that use an All-Pairs-Shortest-Paths (APSP) calculation fall between the ProbEst and BFS algorithm groups. As expected from the asymptotics, BFS algorithms tend to scale more slowly (in terms of runtime) with network size than do ProbEst algorithms, while APSP algorithms scale more quickly (Figure 9).

The low upfront cost of APSP-based algorithms makes them far faster than ProbEst-based algorithms in many practical settings, being 10-100x faster on networks with less than $n = 10^6$. However, the asymptotic cost of APSP is cubic in the network size, implying that for sufficiently large networks, ProbEst will be faster. For our corpus, we estimate the crossover point when Myopic becomes more efficient than MinDegree_hc to occur between 1,262,000 and 1,515,000 nodes (95% CI, from 1000 bootstraps). However, approximation algorithms for APSP could potentially extend their practical efficiency much further.

4.3 A Fast Meta-Learning Algorithm

We can exploit the variability in algorithm performance, and the fact that even among non-Myopic algorithms no alternative is superior on all networks, by introducing a meta-learner algorithm that combines multiple scalable heuristics to approximate the state-of-the-art performance of the Myopic algorithm. We compare this algorithm to a fast ensemble algorithm that uses an oracle to make perfect predictions about which scalable algorithm is best to apply on a particular test network, thus providing an upper bound on the meta-learner’s possible performance.

The meta-learner algorithm leverages the scalability of non-ProbEst algorithms while retaining good overall performance under all spreadability regimes. The task is as follows: given a particular network G and knowledge of the information’s spreadability

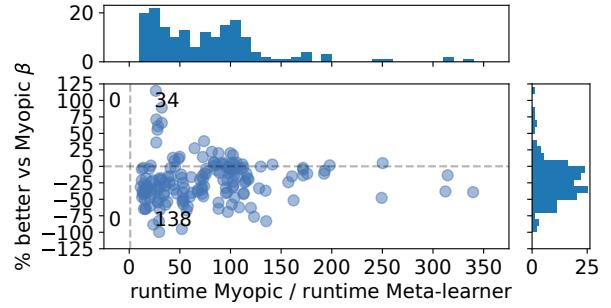


Figure 10: Performance difference vs. speedup for the meta-learner algorithm under medium spreadability, with marginal histograms, averaged over 1000 runs. Extreme outliers have been removed for visualization purposes. Average performance difference relative to Myopic is $-20.11\% \pm 29.34$ (mean \pm stddev), for an average speedup factor of 76.28 ± 64.07 . For 34 of the networks (19.8%) the meta-learner strictly outperforms Myopic.

(low, medium, or high), select the scalable algorithm with the best marginal benefit β for improving information access.

As shown previously, many of the heuristics do not perform well, and so we begin by narrowing the set of available algorithms. For each of the three spreadability settings, we select the set \mathcal{X} of five algorithms (excluding Myopic and Naive Myopic) that maximize the number of networks for which at least one among the set performs at least 80% as well as Myopic. This produces sets

- $\mathcal{X}_{\text{high}} = \{\text{Gonzales}, \text{LeastCentral}, \text{Myopic BFS}, \text{Naive Myopic BFS}, \text{MinDegree}_\text{hc}\}$
- $\mathcal{X}_{\text{medium}} = \{\text{Gonzales}, \text{Myopic BFS}, \text{Myopic PPR}, \text{MinDegree}_\text{hc}, \text{MinDegree}_\text{hcn}\}$
- $\mathcal{X}_{\text{low}} = \{\text{Gonzalez}, \text{Myopic BFS}, \text{Myopic PPR}, \text{MinDegree}_\text{hcn}, \text{MinDegree}_\text{ndn}\}$

For the meta-learner algorithm, we then learn a 5-way random forest classifier to predict the best algorithm in \mathcal{X} to apply to a given network, using nine of the network’s topological features as the feature set (Figure S5). We train and evaluate the meta-learner approach using an 80-20 train-test split among networks in the corpus, with meta-learner algorithm selection and model training both performed on the training set, and we report the mean performance over the test set. The meta-learner’s runtime is the runtime of the trained model and the single algorithm it selects.

In the fast ensemble algorithm, for a given network in the corpus, the oracle runs all algorithms in \mathcal{X} and evaluates the performance of each using ProbEst to calculate their respective β s, and then returns the single algorithm with the highest β for that network. In this way, the oracle acts like an optimal classifier over \mathcal{X} (cf. the meta-learner algorithm). The runtime of the fast ensemble algorithm is simply that of the single algorithm it selects.

Compared to Myopic, the meta-learner is dramatically more efficient, with an average runtime that is 76.26 ± 64.07 times faster under medium spreadability (Figure 10) and 133.35 ± 79.32 times faster under high spreadability (Figure S7). Improvement in scalability comes

with a modest cost to performance, such that the meta-learner produces β values that are, on average, $20.11\% \pm 29.34$ lower than those of Myopic under medium spreadability (with similar results for high); we note that the wide variance in these numbers reflects the broad range of difficulty across networks in the corpus. In contrast, the fast ensemble algorithm's performance is only $9.34\% \pm 28.34$ lower for medium spreadability (similar results for high), indicating both room for improvement by the meta-learner with a better feature set as well as an upper limit to that improvement with the current scalable algorithms. We note, however, that lower performance is not universal: for 34 and 22 networks (20% and 12.8%), the meta-learner outperforms Myopic on medium and high spreadability, respectively (Figs. 10 and S7). Under low spreadability, the fast meta-learner's average performance generally exceeds Myopic because of the inherent precision limitation of ProbEst in this setting.

4.4 Scaling to Larger Networks

In this section, we verify that our methods remain much faster than Myopic even on much larger networks. We consider two networks, Email Network (EU Research Institution) (which we shorten to Email (EU)) and Google+ (2013). These have ~ 34 thousand and ~ 87 thousand nodes, respectively; other summary statistics are reported in Table S2. On these networks, we evaluate Myopic against the five algorithms in X_{high} , as defined in Section 4.3: Gonzales, Least Central, Myopic BFS, Naive Myopic, BFS, and MinDegree_hc. We use high-spreadability α values. We ran these experiments on identical hardware equipped with 40 physical cores (Intel(R) Xeon(R) Gold 6230 CPU @ 2.10 GHz) and 19100 MB of RAM. As with the smaller networks, the budgets tested are $k \in \{1, 2, \dots, 10\}$.

We change several parameters from our previous runs so that our experiments are better suited to the size of these networks. After initial experiments with the number of Monte Carlo simulations in ProbEst set to $R = 1000$, it became clear that 1000 simulation rounds was not enough for networks with tens of thousands of nodes, as some of our evaluations produced a negative β . With exact probability computations, π_{\min} would monotonically increase with each seed added, so $\beta < 0$ indicates highly inaccurate estimations. Thus, we increase the number of Monte Carlo simulations to $R = 10000$ in order to better estimate π_i for large networks. However, due to the increase in simulation rounds and network size, running ProbEst to evaluate the algorithms' performances is extremely expensive. To balance between computation time and a thorough analysis, we run only 3 trials of our algorithms with high-spreadability α values. We also use only 1000 simulations when executing the spreadability computations (recall Section 2.2) to select the appropriate α . We note that calculating the target α would have required approximately 30 days of compute time for Google+ (2013) with $R = 10000$. After using 1000 simulations to complete the spreadability computations, we tested the selected α values with $R = 10000$, verifying that they activate on average 78.9% and 79.4% of Email (EU) and Google+ (2013), respectively, which is quite close to the target activation percentage of 80%.

As with the broader corpus, when evaluated on the two larger networks, the five representative algorithms from X_{high} are orders of magnitude faster than Myopic, with some of the algorithms maintaining seed choice quality comparable to Myopic. Indeed, the

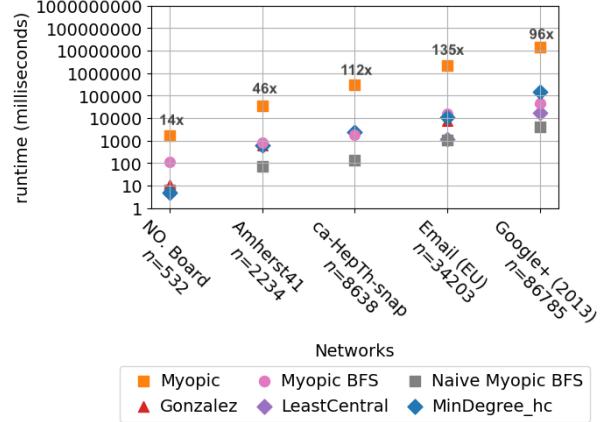


Figure 11: Runtime of Myopic and the algorithms of X_{high} on five social networks of various sizes under high spreadability: Norwegian Board of Directors (2006), Amherst41, ca-HepTh-snap, Email (EU), and Google+ (2013). Numbers in the plot indicate how much slower Myopic is than the second-slowest algorithm for that network.

slowest of the algorithms from X_{high} is 135 times faster than Myopic on Email (EU), and 96 times faster on Google+ (2013). These results (see Figure 11) indicate that the running time advantage of our methods over Myopic is not diminished on networks much larger than those in the corpus of Section 2.1. Moreover, the performance loss of our methods (relative to Myopic) is similar (though slightly greater) to that observed for the larger corpus; see Figure S10 in Appendix D.

5 DISCUSSION AND CONCLUSION

We evaluate new and existing algorithms, and introduce a meta-learning method for choosing k seed nodes to maximize the minimum access probability π_i of a node in a network. The meta-learner achieves a large (75x) speedup over the existing state-of-the-art, with a modest decrease in performance over a large corpus of networks. The previous Monte Carlo method for calculating a node's information access has high computational costs that limit the applicability of algorithms that rely on it, while the algorithms we introduced here scale-up to much larger networks.

To evaluate the algorithms, we introduce a large cross-domain network corpus, and a new performance metric β . We also introduce the concept of spreadability on a network to choose the independent cascade activation probability α , accounting for the influence of a network's structural features on information spread.

Our findings have implications for the design of intervention strategies that aim to improve information access of the disadvantaged individuals in a network. This work suggests that the structure of the network plays a significant role in the performance of the algorithms, with average degree being a particularly important factor. The introduction of an ensemble method based on fast algorithms that do not rely on Monte Carlo simulation also leaves open the incorporation of future lightweight algorithms.

ACKNOWLEDGEMENTS

The authors thank Daniel B. Larremore and Zachary Kilpatrick for helpful feedback, and they acknowledge the BioFrontiers IT group at the University of Colorado Boulder for their support with data storage infrastructure, data management services, and High Performance Computing resources.

FUNDING

This work was supported in part by National Science Foundation Awards IIS 1956183 (DRW, CJW, AC), IIS 1956286 (ACr, MW, FS, BS), and IIS 1955321 (SF).

REFERENCES

- [1] Junaid Ali, Mahmoudreza Babaei, Abhijnan Chakraborty, Baharan Mirsoleiman, Krishna P Gummadi, and Adish Singla. On the fairness of time-critical influence maximization in social networks. *IEEE Transactions on Knowledge and Data Engineering*, 35(3):2875–2886, 2021.
- [2] Abhijit Banerjee, Arun G Chandrasekhar, Esther Duflo, and Matthew O Jackson. The diffusion of microfinance. *Science*, 341(6144):1236498, 2013.
- [3] Ashkan Bashardoust, Hannah C Beilinson, Sorelle A Friedler, Jiajie Ma, Jade Rousseau, Carlos E Scheidegger, Blair D Sullivan, Nasanbayar Ulzii-Orshikh, and Suresh Venkatasubramanian. Information access representations and social capital in networks. *arXiv preprint arXiv:2010.12611*, 2020.
- [4] Ashkan Bashardoust, Sorelle Friedler, Carlos Scheidegger, Blair D Sullivan, and Suresh Venkatasubramanian. Reducing access disparities in networks using edge augmentation. In *Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency*, pages 1635–1651, 2023.
- [5] Ruben Becker, Gianlorenzo D’Angelo, and Sajid Ghobadi. Improving fairness in information exposure by adding links. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 14119–14126, 2023.
- [6] Aditya Bhaskara, Alex Crane, Md Mumtahin Habib Ullah Mazumder, Blair D Sullivan, and Prasanth Yalamanchili. Optimizing information access in networks via edge augmentation. *arXiv preprint arXiv:2407.02624*, 2024.
- [7] Paolo Boldi and Sebastiano Vigna. Axioms for centrality, 2013.
- [8] 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.
- [9] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Comput. Netw.*, 30:107–117, 1998.
- [10] Simon Caton and Christian Haas. Fairness in machine learning: A survey. *ACM Computing Surveys*, 56(7):1–38, 2024.
- [11] 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, 2010.
- [12] Cecilia Cheng, Hsin-Yi Wang, Leif Sigerson, and Chor-Lam Chau. Do the socially rich get richer? a nuanced perspective on social network site use and online social capital accrual. *Psychological Bulletin*, 145(7):734–764, 2019.
- [13] Flavio Chierichetti, Ravi Kumar, Silvio Lattanzi, and Sergei Vassilvitskii. Fair clustering through fairlets. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4–9, 2017, Long Beach, CA, USA*, pages 5029–5037, 2017.
- [14] Aaron Clauset, Ellen Tucker, and Matthias Sainz. The colorado index of complex networks. <https://icon.colorado.edu/>, 2016.
- [15] 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, 2001.
- [16] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *Proceedings of the 3rd innovations in theoretical computer science conference*, pages 214–226, 2012.
- [17] Michael Feldman, Sorelle A Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. Certifying and removing disparate impact. In *proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 259–268, 2015.
- [18] Benjamin Fish, Ashkan Bashardoust, Danah Boyd, Sorelle Friedler, Carlos Scheidegger, and Suresh Venkatasubramanian. Gaps in information access in social networks? In *The World Wide Web Conference, WWW ’19*. ACM, May 2019.
- [19] Linton C. Freeman. Centrality in social networks conceptual clarification. *Social Networks*, 1(3):215–239, 1978.
- [20] Joshua P. Garoon and Patrick S. Duggan. Discourses of disease, discourses of disadvantage: A critical analysis of national pandemic influenza preparedness plans. *Social Science & Medicine*, 67(7):1133–1142, 2008.
- [21] Amir Ghasemian, Homa Hosseini Mardi, and Aaron Clauset. Evaluating overfit and underfit in models of network community structure. *IEEE Transactions on Knowledge and Data Engineering*, 32:1722–1735, 2019.
- [22] Jacob Goldenberg, Barak Libai, and Eitan Muller. Talk of the network: A complex systems look at the underlying process of word-of-mouth. *Marketing Letters*, 2001.
- [23] M. Granovetter. Threshold models of collective behavior. *The American Journal of Sociology*, 83(6):1420–1443, 1978.
- [24] Mark S. Granovetter. The strength of weak ties. *American Journal of Sociology*, 78(6):1360–1380, 1973.
- [25] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using networkx. In Gaël Varoquaux, Travis Vaught, and Jarrod Millman, editors, *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA, 2008.
- [26] Moritz Hardt, Eric Price, and Nati Srebro. Equality of opportunity in supervised learning. *Advances in Neural Information Processing Systems*, 30:3323 – 3331, 2016.
- [27] Keke Huang, Sibo Wang, Glenn Bevilacqua, Xiaokui Xiao, and Laks VS Lakshmanan. Revisiting the stop-and-stare algorithms for influence maximization. *Proceedings of the VLDB Endowment*, 10(9):913–924, 2017.
- [28] K. Ikebara and A. Clauset. Characterizing the structural diversity of complex networks across domains, 2017.
- [29] Zeinab S Jalali, Qilan Chen, Shwetha M Srikanta, Weixiang Wang, Myunghwan Kim, Hema Raghaban, and Sucheta Soundarajan. Fairness of information flow in social networks. *ACM Transactions on Knowledge Discovery from Data*, 17(6):1–26, 2023.
- [30] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD ’03*, page 137–146. Association for Computing Machinery, 2003.
- [31] Jure Leskovec, Andreas Krause, Carlos Guestrin, 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, 2007.
- [32] Anay Mehrotra, Jeff Sachs, and L Elisa Celis. Revisiting group fairness metrics: The effect of networks. *Proceedings of the ACM on Human-Computer Interaction*, 6(CSCW2):1–29, 2022.
- [33] M. E. J. Newman. *Networks*. Oxford University Press, Oxford, UK, 2nd edition, 2018.
- [34] Hung T Nguyen, My T Thai, and Thang N Dinh. Stop-and-stare: Optimal sampling algorithms for viral marketing in billion-scale networks. In *Proceedings of the 2016 international conference on management of data*, pages 695–710, 2016.
- [35] Naoto Ohsaka. The solution distribution of influence maximization: A high-level experimental study on three algorithmic approaches. In *Proceedings of the 2020 ACM SIGMOD international conference on management of data*, pages 2151–2166, 2020.
- [36] Tiago P. Peixoto. The netschleuder network catalogue and repository. <https://doi.org/10.5281/zenodo.7839981>, 2020.
- [37] Dana Pessach and Erez Shmueli. A review on fairness in machine learning. *ACM Computing Surveys (CSUR)*, 55(3):1–44, 2022.
- [38] Pierre M. Picard and Yves Zenou. Urban spatial structure, employment and social ties. *Journal of Urban Economics*, 104:77–93, 2018.
- [39] John Rawls. *A Theory of Justice*. Harvard University Press, 2009.
- [40] Matthew Richardson and Pedro Domingos. Mining knowledge-sharing sites for viral marketing. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 61–70, 2002.
- [41] Everett M Rogers, Arvind Singhal, and Margaret M Quinlan. Diffusion of innovations. In *An integrated approach to communication theory and research*, pages 432–448. Routledge, 2014.
- [42] Akrati Saxena, George Fletcher, and Mykola Pechenizkiy. Fairnsa: Algorithmic fairness in social network analysis. *ACM Computing Surveys*, 56(8):1–45, 2024.
- [43] Ana-Andreea Stoica and Augustin Chaintreau. Fairness in social influence maximization. In *Companion Proceedings of The 2019 World Wide Web Conference*, pages 569–574, 2019.
- [44] Ana-Andreea Stoica, Jessy Xinyi Han, and Augustin Chaintreau. Seeding network influence in biased networks and the benefits of diversity. In *Proceedings of The Web Conference 2020*, pages 2089–2098, 2020.
- [45] Ana-Andreea Stoica, Christopher Riederer, and Augustin Chaintreau. Algorithmic glass ceiling in social networks: The effects of social recommendations on network diversity. In *Proceedings of the 2018 World Wide Web Conference*, pages 923–932, 2018.
- [46] Youze Tang, Yanchen Shi, and Xiaokui Xiao. Influence maximization in near-linear time: A martingale approach. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data*, pages 1539–1554, 2015.
- [47] 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*, pages 75–86,

- 2014.
- [48] Alan Tsang, Bryan Wilder, Eric Rice, Milind Tambe, and Yair Zick. Group-fairness in influence maximization. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 5997–6005, 2019.
 - [49] Gergely Tóth, Johannes Wachs, Riccardo Di Clemente, and Jari Saramäki. Inequality is rising where social network segregation interacts with urban topology. *Nature Communications*, 12(1):1143, 2021.
 - [50] Dashun Wang and Brian Uzzi. Weak ties, failed tries, and success. *Science*, 377(6612):1256–1258, 2022.
 - [51] Amulya Yadav, Hau Chan, Albert Xin Jiang, Haifeng Xu, Eric Rice, and Milind Tambe. Using social networks to aid homeless shelters: Dynamic influence maximization under uncertainty. In *AAMAS*, volume 16, pages 740–748, 2016.

A NETWORK CORPUS

Summary statistics for the network corpus can be found in Table S1.

B PSEUDOCODE FOR ALGORITHMS

For completeness, here we provide pseudocode for all 10 new algorithms, along with high-level descriptions. We also describe the ensemble and meta-learner approaches.

Algorithm 1: Myopic BFS

Approximates π_i values by performing BFS traversal from the most recent seed and computing π_i activation probabilities for every node in the network, taking into account only its “parents” and “neighbors.” For a node x at distance t from the seed, a “parent” is any node at distance $t - 1$ from the seed that shares an edge with x , and a “neighbor” is any node at distance t that shares an edge with x . When adding a new seed, update the old probabilities with a new BFS traversal.

Algorithm 1: Myopic BFS

```

Input : $G = (V, E), s, \alpha$ .
Output: $S \subseteq V$  with  $|S| = s + 1$ .
// Initialize  $S$  with a random node.
1  $S = \{s_0 \in_R V\}$ 
2 while  $|S| \leq s$  do
3    $Q \leftarrow S[-1]$ 
   // initialize queue with latest seed
4   while  $|Q| < 0$  do
5     Perform BFS traversal and record distance  $t$  to  $S[-1]$ 
     for every node
6   for  $v \in V$  do
7     Approximate access probability  $\pi_i \forall v \in V$  // see
     the writeup earlier for more info
     // Choose the node with min. activation prob.
8    $x = \operatorname{argmin}_{v \in V \setminus S} \operatorname{Prob}(v)$ 
9    $S = S \cup \{x\}$ 
10 return  $S$ 

```

Algorithm 2: Naive Myopic BFS

Approximates π_i values in the same way as Myopic BFS. Reuses the approximations from the first traversal when adding each subsequent seed.

Algorithm 3: Myopic PPR

Choose an initial seed node uniformly at random. Then, for each new seed, perform a pass of Personalized Page Rank, with personalization parameter set to bias random walk restarts from the current seed set. Sort the nodes by PPR in ascending order, choose s lowest-scoring nodes as new seeds.

Algorithm 4: Naive Myopic PPR

Choose an initial seed node uniformly at random. Then, perform a pass of Personalized Page Rank, with personalization parameter set

Algorithm 2: Naive Myopic BFS

```

Input : $G = (V, E), s$ .
Output: $S \subseteq V$  with  $|S| = s + 1$ .
// Initialize  $S$  with a random node.
1  $S = \{s_0 \in_R V\}$ 
2  $Q \leftarrow S[-1]$ 
   // initialize queue with first random seed
3 while  $|Q| < 0$  do
4   Perform BFS traversal and record distance  $t$  to  $S[-1]$  for
   every node
5 for  $v \in V$  do
6   Approximate access probability  $\pi_i \forall v \in V$  // see the
   writeup earlier for more info
   // Choose  $s$  nodes with min. activation prob.
7 while  $|S| \leq s$  do
8    $x = \operatorname{argmin}_{v \in V \setminus S} \operatorname{Prob}(v)$ 
9    $S = S \cup \{x\}$ 
10 return  $S$ 

```

Algorithm 3: Myopic PPR

```

Input : $G = (V, E), s$ .
Output: $S \subseteq V$  with  $|S| = s + 1$ .
// Initialize  $S$  with a random node.
1  $S = \{s_0 \in_R V\}$ 
2 while  $|S| \leq s$  do
3    $x = \operatorname{argmin}_{v \in V \setminus S} \operatorname{PPR}(v, S)$ 
4    $S = S \cup \{x\}$ 
5 return  $S$ 

```

to bias random walk restarts from the initial random seed. Sort the nodes by PPR in ascending order, choose s lowest-scoring nodes as new seeds.

Algorithm 4: Naive Myopic PPR

```

Input : $G = (V, E), s$ .
Output: $S \subseteq V$  with  $|S| = s + 1$ .
// Initialize  $S$  with a random node.
1  $S = \{s_0 \in_R V\}$ 
   // Sort the nodes by ascending PPR
2  $v_1, v_2, \dots, v_n = \operatorname{PPR}(G, S)$ 
3  $S = S \cup \{v_1, v_2, \dots, v_s\}$ 
4 return  $S$ 

```

Algorithm 5: LeastCentral

Choose an initial seed node uniformly at random. Then choose the node with the lowest closeness centrality as the new seed.

	net. count	$\langle n \rangle$	$\langle m \rangle$	$\langle k \rangle$	$\langle C \rangle$	$\langle \ell_{\max} \rangle$	$\langle \sigma^2 \rangle$
full corpus	174	1495.42	26288.68	22.60	0.24	13.79	1997.12
biological	34	927.50	2830.24	7.00	0.08	11.59	285.87
social	44	1647.75	29055.57	28.11	0.42	11.02	1099.06
economic	43	2367.40	71946.23	51.97	0.38	7.19	6647.82
technological	32	843.13	1651.50	4.07	0.07	17.84	69.55
transportation	17	1243.65	2148.53	3.80	0.10	35.76	37.79
informational	4	1561.75	4124.25	6.41	0.11	8.00	174.06

Table S1: Summary statistics of the network corpus used to evaluate algorithms for the Information Access Gap Minimization problem, showing the number of networks by scientific domain, along with the average number of nodes $\langle n \rangle$, average number of edges $\langle m \rangle$, average degree $\langle k \rangle$, average clustering coefficient (transitivity) $\langle C \rangle$, average diameter $\langle \ell_{\max} \rangle$, and average variance of the degree distribution $\langle \sigma^2 \rangle$ for networks in that domain.

Algorithm 5: LeastCentral

```

Input :  $G = (V, E), s$ .
Output:  $S \subseteq V$  with  $|S| = s + 1$ .
// Initialize  $S$  with a random node.
1  $S = \{s_0 \in_R V\}$ 
2 while  $|S| \leq s$  do
    // Choose node with min. Close. Centrality
    3  $x = \operatorname{argmin}_{v \in V \setminus S} CC(v)$ 
    4  $S = S \cup \{x\}$ 
5 return  $S$ 

```

Algorithm 6: LeastCentral_n

Choose an initial seed node uniformly at random. Then choose a node with the smallest closeness centrality and select that node's highest-degree neighbor to be the next seed.

Algorithm 6: LeastCentral_n

```

Input :  $G = (V, E), s$ .
Output:  $S \subseteq V$  with  $|S| = s + 1$ .
// Initialize  $S$  with a random node.
1  $S = \{s_0 \in_R V\}$ 
2 while  $|S| \leq s$  do
    // Choose node with min. Close. Centrality
    3  $x = \operatorname{argmin}_{v \in V \setminus S} CC(v)$ 
    // Choose the highest degree neighbor.
    4  $y = \operatorname{argmax}_{v \in N(x)} d(v)$ 
    5  $S = S \cup \{y\}$ 
6 return  $S$ 

```

Algorithm 7: MinDegree_hc

Choose an initial seed node uniformly at random. Then, identify all minimum degree nodes, and sort by harmonic centrality. Finally, choose the node with the lowest harmonic centrality as the new seed.

Algorithm 7: MinDegree_hc

```

Input :  $G = (V, E), s$ .
Output:  $S \subseteq V$  with  $|S| = s + 1$ .
// Initialize  $S$  with a random node.
1  $S = \{s_0 \in_R V\}$ 
2 while  $|S| \leq s$  do
    // Choose nodes with min. degree.
    3  $V' = \{v \in V \setminus S : d(w) \geq d(v) \forall w \in V \setminus S\}$ 
    // Choose node with min. Harm. Centrality.
    4  $x = \operatorname{argmin}_{v \in V'} HC(v)$ 
    5  $S = S \cup \{x\}$ 
6 return  $S$ 

```

Algorithm 8: MinDegree_hcn

Choose an initial seed node uniformly at random. Then, identify all minimum degree nodes, and sort by harmonic centrality. Finally, choose the highest-degree neighbor of the node with the lowest harmonic centrality as the new seed.

Algorithm 8: MinDegree_hcn

```

Input :  $G = (V, E), s$ .
Output:  $S \subseteq V$  with  $|S| = s + 1$ .
// Initialize  $S$  with a random node.
1  $S = \{s_0 \in_R V\}$ 
2 while  $|S| \leq s$  do
    // Choose nodes with min. degree.
    3  $V' = \{v \in V \setminus S : d(w) \geq d(v) \forall w \in V \setminus S\}$ 
    // Choose node with min. Harm. Centrality.
    4  $x = \operatorname{argmin}_{v \in V'} HC(v)$ 
    // Choose highest degree neighbor.
    5  $y = \operatorname{argmax}_{v \in N(x)} d(v)$ 
    6  $S = S \cup \{y\}$ 
7 return  $S$ 

```

Algorithm 9: MinDegree_nd

Choose an initial seed node uniformly at random. Then, identify all minimum degree nodes, and sort by neighbor degree. Choose the node with the highest-degree neighbor as the new seed.

Algorithm 9: MinDegree_nd

Input : $G = (V, E), s$.
Output: $S \subseteq V$ with $|S| = s + 1$.

```

// Initialize S with a random node.
1  $S = \{s_0 \in_R V\}$ 
2 while  $|S| \leq s$  do
    // Choose nodes with min. degree.
    3  $V' = \{v \in V \setminus S : d(w) \geq d(v) \forall w \in V \setminus S\}$ 
    // Choose node with highest neigh. degree.
    4  $x = \operatorname{argmax}_{v \in V'} \sum_{w \in N(v)} d(w)$ 
    5  $S = S \cup \{x\}$ 
6 return  $S$ 
```

Algorithm 10: MinDegree_ndn

Choose an initial seed node uniformly at random. Then, identify all minimum degree nodes, and sort by neighbor degree. For the node with the highest-degree neighbor, choose the highest-degree neighbor itself as the new seed.

Algorithm 10: MinDegree_ndn

Input : $G = (V, E), s$.
Output: $S \subseteq V$ with $|S| = s + 1$.

```

// Initialize S with a random node.
1  $S = \{s_0 \in_R V\}$ 
2 while  $|S| \leq s$  do
    // Choose nodes with min. degree.
    3  $V' = \{v \in V \setminus S : d(w) \geq d(v) \forall w \in V \setminus S\}$ 
    // Choose node with highest neigh. degree.
    4  $x = \operatorname{argmax}_{v \in V'} \sum_{w \in N(v)} d(w)$ 
    // Choose highest degree neighbor.
    5  $y = \operatorname{argmax}_{v \in N(x)} d(v)$ 
    6  $S = S \cup \{y\}$ 
7 return  $S$ 
```

Algorithm 11: Fast Ensemble (Oracle)

Select five members of D , an ensemble of algorithms. To do so, perform greedy search for a set of five algorithms that attain $\geq 80\%$ of Myopic's performance on the network corpus, as evaluated by ProbEst. Then, for each new network, select the algorithm that performs best as suggested by the oracle, and apply it.

Algorithm 11: Fast Ensemble (Oracle)

Input : $G = (V, E), s$.
Output: $S \subseteq V$ with $|S| = s + 1$.

```

1 Initialize S with a random node.
2  $S = \{s_0 \in_R V\}$ 
3 Select  $a_1, a_2, a_3, a_4, a_5$  to maximize number of networks in
    the corpus for which performance of some  $a_i$  is  $\geq 80\%$  of
    Myopic
4  $D \leftarrow \{a_1, a_2, a_3, a_4, a_5\}$ 
    // Ask the oracle  $O$  which algorithm to use
5  $a_{best} \leftarrow O(G, D)$ 
    // Run the algorithm to get  $S$ 
6  $S = S \cup a_{best}(G, s)$ 
7 return  $S$ 
```

Algorithm 12: Meta-learner

Select five members of D , an ensemble of algorithms. To do so, perform greedy search for a set of five algorithms that attain $\geq 80\%$ of Myopic's performance on the network corpus, as evaluated by ProbEst. Using prior network corpus data, train a model M that, given a new network G , selects the best algorithm out of the ensemble based on topological features. Apply the selected algorithm to G .

Algorithm 12: Meta-learner

Input : $G = (V, E), s$.
Output: $S \subseteq V$ with $|S| = s + 1$.

```

1 Initialize S with a random node.
2  $S = \{s_0 \in_R V\}$ 
3 Select  $a_1, a_2, a_3, a_4, a_5$  to maximize number of networks in
    the corpus for which performance of some  $a_i$  is  $\geq 80\%$  of
    Myopic
4  $D \leftarrow \{a_1, a_2, a_3, a_4, a_5\}$ 
5 Train Random Forest Classifier  $M$  to select the best
    algorithm from  $D$  based on network topology, using data
    from the network corpus
    // Ask  $M$  which algorithm to use
6  $a_{best} \leftarrow M(G, D)$ 
    // Run the algorithm to get  $S$ 
7  $S = S \cup a_{best}(G, s)$ 
8 return  $S$ 
```

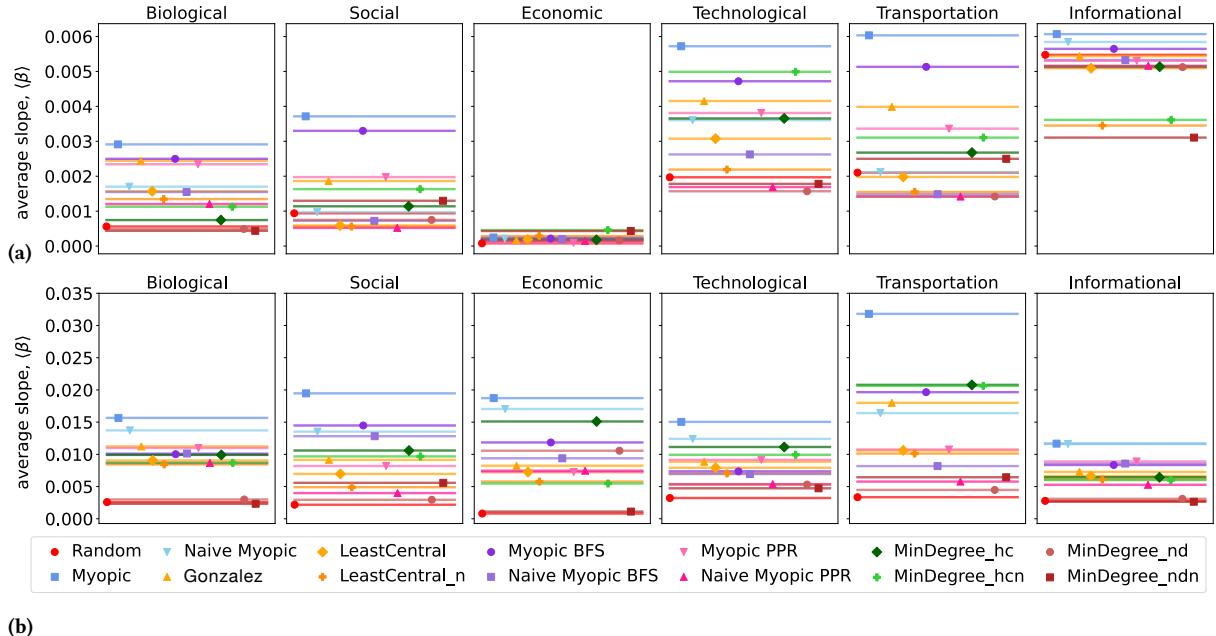


Figure S1: Mean performance of the intervention algorithms on each domain in the corpus, under (a) low spreadability and (b) high spreadability settings. Medium spreadability results are given in Figure 6.

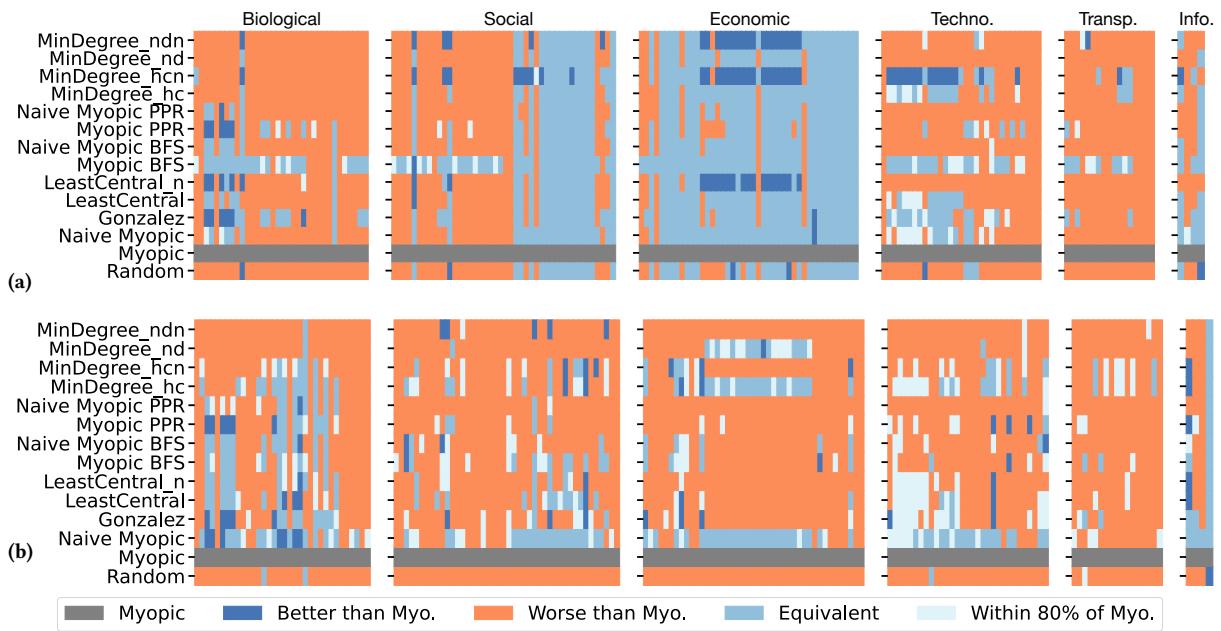


Figure S2: Performance of intervention algorithms on the network corpus, relative to Myopic and sorted in ascending order by network size within each domain. (a): low spreadability, 12% of networks have no algorithm better than or within 80% of Myopic's performance; (b): high spreadability, showing very similar results to medium spreadability (Figure 7). “Equivalent” defined as within one standard error of β for Myopic; typically about 0.001.

C SUPPLEMENTARY RESULTS

Algorithm performance for low and high spreadability. Figure S1 shows the mean performance β of all the algorithms applied to all

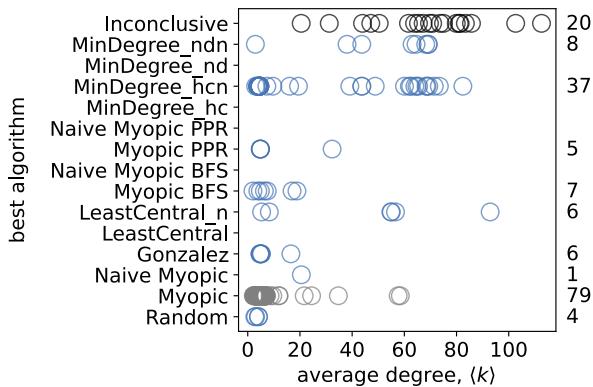


Figure S3: Best-performing algorithm on a network vs average degree of the network under low spreadability. Each network’s corresponding single most-performant algorithm is plotted with a circle. Total number of circles per line is given on the right-hand side, indicating how many times an algorithm was the best-performing one across the entire corpus.

the networks in the corpus, grouped into the six network domains: biological, social, economic, technological, transportation, and informational, for low and high spreadability settings. We observe qualitatively similar results across spreadability settings, in which Myopic is often on average the best performing algorithm. However, the relative ordering of other algorithms varies substantially across domains and spreadability settings.

Figure S2 expands on these results (following the experimental steps in the main text), showing for the low and high spreadability settings the performance of each algorithm relative to the performance of Myopic. Notably, in the low spreadability setting, many more algorithms fall into the “equivalent” category, in which their performance is statistically indistinguishable from Myopic (β within one standard error). This behavior is due to the low precision of ProbEst in this setting. We find many fewer cases of equivalence in the high spreadability setting, which is qualitatively similar to the results in the medium setting, in which across networks, many alternatives perform similarly as Myopic, and in some cases outperform Myopic. These results reinforce the finding that generally no algorithm is superior to others across networks and spreadability settings.

Figures S3 and S4 tabulate the counts of how often a particular algorithm was the best performing (highest β) and plot them as a function of network mean degree $\langle k \rangle$, for low and high spreadability, respectively. Here we see clearly that Myopic is by far the best algorithm in high spreadability (similar to results for medium, shown in Figure 8), but is much less so under low spreadability. We believe this difference is attributable to the imprecision of ProbEst in the low spreadability setting.

Meta-learner. To construct the meta-learner, we trained a model to predict which algorithm would perform best on a given network, given only the network’s features and the spreadability of the information in the Independent Cascade model. For this task, we held out

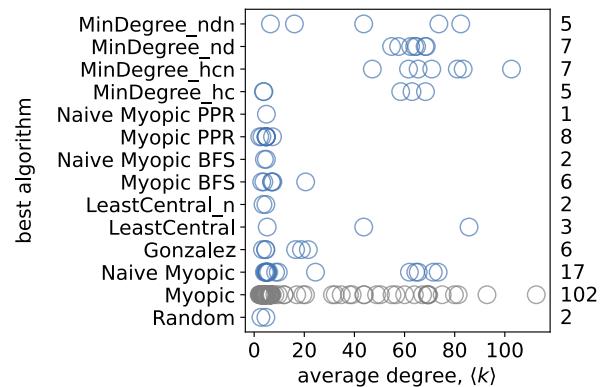


Figure S4: Best-performing algorithm on a network vs average degree of the network under high spreadability. Each network’s corresponding single most-performant algorithm is plotted with a circle. Total number of circles per line is given on the right-hand side, indicating how many times an algorithm was the best-performing one across the entire corpus.

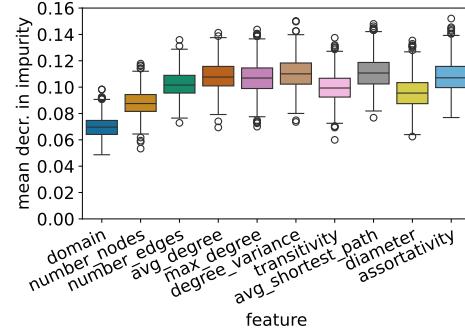


Figure S5: Meta-learner random forest feature importance for predicting which of five algorithms is the best choice for a given network. Higher values mean a feature contributes more to the predicted performance value.

Myopic and Naive Myopic, as the goal is to approximate their state-of-the-art performance using more scalable algorithms. Network features used are: the network domain, number of nodes n , average degree $\langle k \rangle$, maximum degree k_{\max} , degree variance σ^2 , clustering coefficient C , average shortest path $\langle \ell \rangle$, diameter ℓ_{\max} , and degree assortativity r . We use this approach rather than a representation learning approach to ensure better interpretability of what aspects of network structure influence performance predictability.

This multi-class classification task is challenging. Fine-tuned models come out with an accuracy of ≈ 0.51 . Figs. S5 and S6 show the learned feature importances and confusion matrix for the medium spreadability setting (similar results for low and high spreadability models). However, generally, incorrect algorithm choices by the model tend to select an alternative that is almost as good

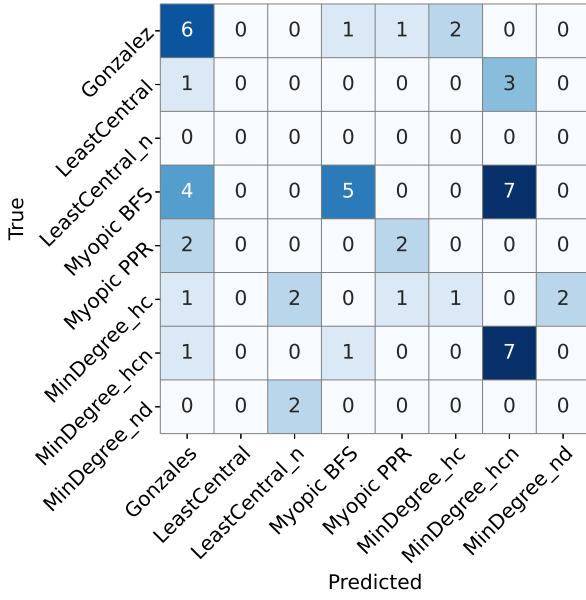


Figure S6: Confusion matrix of the best algorithm prediction model for the meta-learner.

as the true best, in terms of performance β , and so classification errors often do not translate into large losses in performance.

From the classification task, we find that the domain of a network contributes the least to the prediction accuracy, followed by number of nodes n . Other network features are roughly equally important (Figure S5). This result implies that network domain does not contribute much marginal information beyond a network's features, as has been found previously by Ikehara and Clauset [28]. That is, network features encode much the same information as network domain.

The accuracy of the meta-learning model under medium and high spreadability (Figs. 10 and S7) are similar, but with slightly larger performance loss and substantially larger runtime speedup in the case of high, than in medium.

Unsurprisingly, the fast ensemble algorithms (which use an oracle to choose the best heuristic for a given network) in both medium and high spreadability settings outperform the meta-learner. However, the gap between fast ensemble and meta-learning is only about 10-20%, indicating that while there is room for improvement in the meta-learner's classifier, the optimal performance is not far off. Moreover, the fast ensemble does outperform Myopic on more cases than does the meta-learning, but again, the gap is not enormous. We note that timing results for some algorithms (both those using ProbEst and the BFS-based class of new algorithms; see Section 3) may be impacted by our use of an adjacency matrix (rather than adjacency list) representation for all networks.

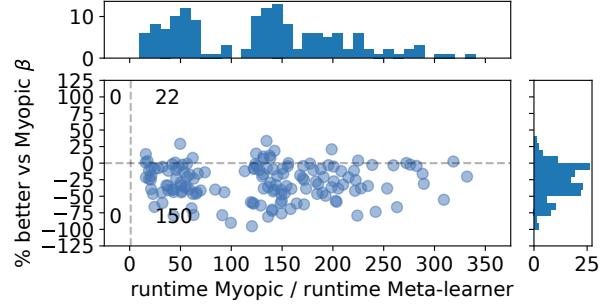


Figure S7: Performance difference vs. speedup for the meta-learning algorithm relative to Myopic under high spreadability, with marginal histograms. Results averaged over 1000 meta-learner runs, and extreme outliers have been removed for visualization purposes only. Average performance loss relative to Myopic is 28.20 ± 25.80 , with an average speedup factor of 133.35 ± 79.32 , but for 22 networks (12.9%), the meta-learner outperforms Myopic.

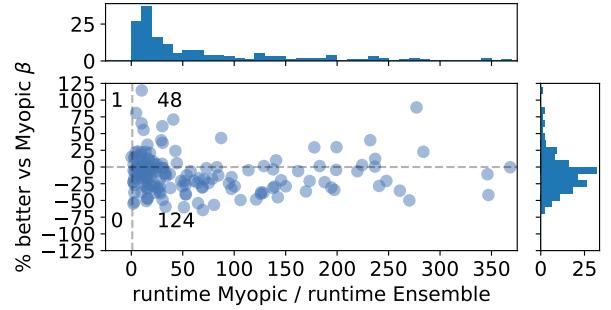


Figure S8: Performance difference vs. speedup for the fast ensemble algorithm relative to Myopic under medium spreadability, with marginal histograms. Extreme outliers have been removed for visualization purposes only. Average performance loss relative to Myopic is $9.34 \pm 28.34\%$, with an average speedup factor of 70.66 ± 85.65 , but for 48 networks (28.1%), the fast ensemble outperforms Myopic.

D RESULTS ON LARGER NETWORKS

	n	m	k	C	l_{max}	σ^2
Email (EU)	34,203	96,669	15.87	0.04	7	770
Google+ (2013)	86,785	688,602	5.65	0.27	47	1,809

Table S2: Statistics of Email (EU) and Google+ (2013) showing number of nodes n , number of edges m , average node degree k , clustering coefficient C , diameter l_{max} , and the variance σ^2 of the degree distribution.

In Table S2, we present summary statistics for the two larger networks used in Section 4.4. In Figure S10, we evaluate the performance of the five algorithms from X_{high} as well as that of Myopic.

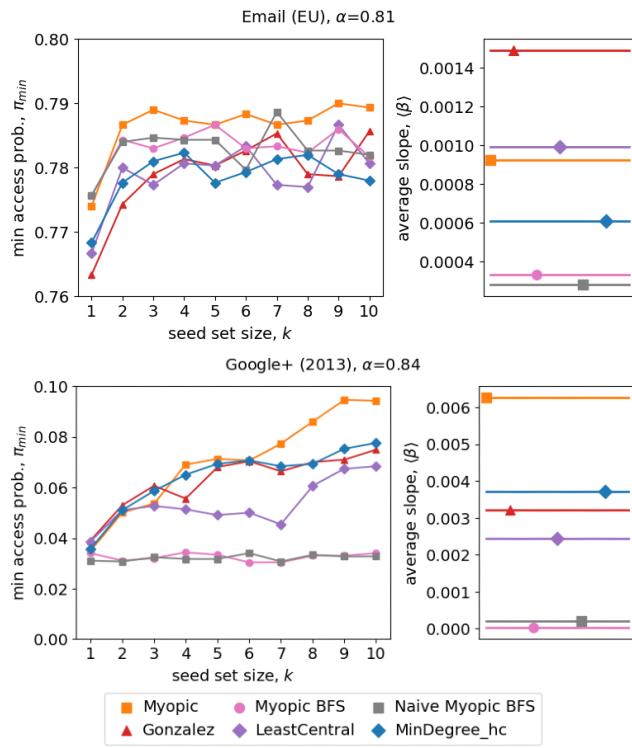


Figure S10: Minimum access probability vs. seed set size under the six algorithms in X_{high} , evaluated on the two larger networks, Email (EU) and Google+ (2013) under high spreadability. Small decreases in π_{\min} are discussed in Appendix D. The panel on the right shows β , the slope of the line of best fit, for each algorithm.

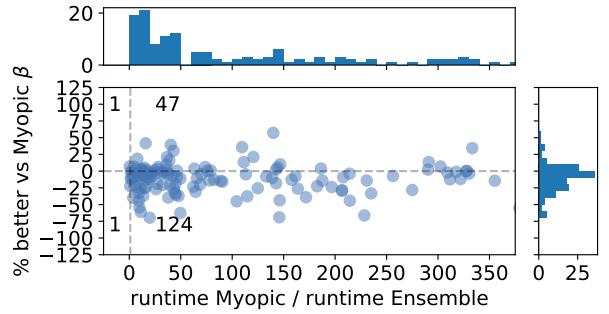


Figure S9: Performance difference vs. speedup for the fast ensemble algorithm relative to Myopic under high spreadability, with marginal histograms. Extreme outliers have been removed for visualization purposes only. Average performance loss relative to Myopic is $11.11 \pm 21.8\%$, with an average speedup of 101.88 ± 101.27 , but for 47 networks (27.6%), the fast ensemble algorithm outperforms Myopic.

When evaluating the performance of these algorithms, it is important to note that due to the size of the networks, a large increase in π_{\min} is not expected with any algorithm when choosing only 10 seeds. Additionally, when looking at the minimum access probability as seeds are added, there are small decreases in π_{\min} resulting from the imprecise nature of ProbEst. This is a limitation on our ability to evaluate these algorithms, but it is not a restriction on the algorithms themselves. That being noted, three out of the five algorithms (MinDegree_hc, Gonzalez and LeastCentral) produce β values close to, and in some cases even greater than, Myopic (Figure S10). Although the β values are predictably low for all algorithms, including Myopic, there is a general positive trend with most algorithms. From these results, we hypothesize that when scaled to large networks, the speedup of the algorithms is sustained, while seed choice quality remains comparable to Myopic for at least some algorithms.