

SPARSIFICATION OF SOCIAL NETWORKS USING RANDOM WALKS

by

BRYAN WILDER

A thesis submitted in partial fulfillment of the requirements
for the Honors in the Major Program in Computer Science
in the College of Engineering and Computer Science
and the Burnett Honors College
at the University of Central Florida
Orlando, FL

Spring Term 2015

Thesis Chair: Gita Sukthankar, PhD

© 2015 Bryan Wilder

ABSTRACT

Analysis of large network datasets has become increasingly important. Algorithms have been designed to find many kinds of structure, with numerous applications across the social and biological sciences. However, a tradeoff is always present between accuracy and scalability; otherwise promising techniques can be computationally infeasible when applied to networks with huge numbers of nodes and edges. One way of extending the reach of network analysis is to sparsify the graph by retaining only a subset of its edges. The reduced network could prove much more tractable.

For this thesis, I propose a new sparsification algorithm that preserves the properties of a random walk on the network. Specifically, the algorithm finds a subset of edges that best preserves the stationary distribution of a random walk by minimizing the Kullback-Leibler divergence between a walk on the original and sparsified graphs. A highly efficient greedy search strategy is developed to optimize this objective. Experimental results are presented that test the performance of the algorithm on the influence maximization task. These results demonstrate that sparsification allows near-optimal solutions to be found in a small fraction of the runtime that would be required using the full network. Two cases are shown where sparsification allows an influence maximization algorithm to be applied to a dataset that previous work had considered intractable.

DEDICATION

This thesis is dedicated my family. The last four years would not have been possible without their support, advice, and company.

ACKNOWLEDGMENTS

First, I extend my sincerest gratitude to my adviser, Dr. Gita Sukthankar, for her essential guidance throughout the development of this thesis. Next, I would like to thank the other mentors who have spent countless hours working with me over the last few years. In particular, Dr. Anne Kandler and Dr. Kenneth Stanley have provided invaluable advice and helped teach me what it means to do research. Lastly, I thank my parents, John Wilder and Taru Wilder, who taught me to follow my passion and supported me every step of the way.

TABLE OF CONTENTS

LIST OF FIGURES	viii
CHAPTER 1: INTRODUCTION	1
CHAPTER 2: PREVIOUS WORK	3
Sparsification	3
Application-based sparsifiers	3
Spectral and cut sparsifiers	4
Backbone detection	5
Influence maximization	6
Problem overview	6
Submodularity and the greedy algorithm	7
Heuristic approaches	8
CHAPTER 3: METHODOLOGY	10
Objective formulation	10
Problem statement	14

Proposed algorithm 15

Runtime analysis 15

Lazy Evaluation and Submodularity 16

CHAPTER 4: RESULTS 19

 Theoretical analysis 19

 Experimental design 24

 Datasets 24

 Influence maximization algorithms 25

 Sparsification algorithms 26

 Experimental results 26

 Influence spread 27

 Speedup 29

 Scaling to large networks 31

CHAPTER 5: CONCLUSION 34

LIST OF REFERENCES 36

LIST OF FIGURES

4.1	Influence spread for the net-HEP dataset	28
4.2	Influence spread for the email-Enron dataset	29
4.3	Runtime for the net-HEP dataset	30
4.4	Runtime for the email-Enron dataset	31
4.5	CELF applied to the email-Enron dataset	32
4.6	InfluMax applied to the net-HEP dataset	33

CHAPTER 1: INTRODUCTION

Analysis of large social networks offers many insights: detecting communities [8, 21, 6, 1], determining the most influential actors [15, 4], predicting the spread of influence [12, 17], and many others. However, algorithms are always limited by their scalability, often putting massive datasets out of reach. Real world networks of interest can contain millions of nodes and billions of edges, which prevents many effective methods from being used in important situations.

One way to help existing algorithms scale to larger datasets is to construct a more concise representation of the network with fewer nodes or edges. If this new graph matches the structural properties of the original, then analysis can be conducted with a smaller dataset and the results propagated back to the original network. *Sparsification* is one such technique, in which all of the nodes of the original graph are preserved but only a subset of edges are retained [25, 10, 19]. An effective sparsification method will preserve desirable features of the connectivity of the larger network with a small number of edges, bringing the network within reach of other algorithms. As a preprocessing step, sparsification must be highly computationally efficient in order to tackle large datasets that are infeasible for more elaborate algorithms.

For this thesis, I develop and evaluate a new sparsification algorithm that is based on preserving the properties of a random walk on the graph. The details of the algorithm are given in Chapter 3. This method promises to fill a gap in the existing literature on sparsification, explored in Chapter 2. Namely, it retains global properties of network organization while being simple, computationally efficient, and working only on the topology of the network with no need for additional information. This algorithm will be tested as a preprocessing step for network analysis on real data. The goal is to enable analysis of datasets that are out of reach for current algorithms.

Chapter 2 reviews the existing literature on sparsification, as well as the target application for this

work, influence maximization. Chapter 3 then gives the problem formulation and introduces the proposed algorithm. This section also includes runtime analysis of the algorithm and explores an optimization strategy based on the principle of submodularity. Chapter 4 presents results related to the performance of the algorithm. This includes theoretical analysis of the greedy optimization strategy as well as empirical results from applying the algorithm to the influence maximization task on two real world networks. The major conclusion from these experiments is that the proposed algorithm allows influence maximization to be performed in a fraction of the time without sacrificing the quality of the solution. This chapter concludes by giving two examples of influence maximization algorithms that can be applied to unprecedentedly large networks using sparsification. In both of these cases, previous work has concluded that running the given influence maximization algorithm on a dataset of this size is infeasible. Lastly, Chapter 5 summarizes the contributions made and offers suggestions for future work.

CHAPTER 2: PREVIOUS WORK

Relevant work includes both algorithms for sparsification and the algorithms that may benefit from a sparsified graph. The first section discusses previous sparsification algorithms, while the second explores the chosen application area, influence maximization.

Sparsification

Sparsification has been addressed by researchers working in a number of different disciplines and communities. Previous algorithms can be roughly broken down into three categories depending on their goals.

Application-based sparsifiers

Perhaps most relevant to the proposed work are sparsification algorithms designed specifically to aid with particular network analysis tasks. For example Satuluri et al. [23] developed a sparsification algorithm specifically for graph clustering. The underlying principle is to preferentially retain edges between nodes that are in the same cluster by selecting edges between nodes with similar neighbors. Thus, the structure that is retained is highly tuned to the application domain: clusters should be preserved, but other forms of organization in the graph will be lost by design.

Another algorithm by Mathioudakis et al. [19] is specifically directed at influence maximization. It uses a maximum likelihood method to find a set of edges that generate an observed set of influence cascades with the greatest probability. Notably, this method requires information beyond the topology of the network: it uses example sets of node activations to produce the estimate. This is

useful for tasks where such data is available but limits its applicability when only the structure is known. In contrast, the proposed algorithm operates only on the adjacency structure of the graph.

The proposed algorithm also departs from these sparsifiers in that it attempts to retain a more general kind of structure associated with the flow of information through a network. Hopefully, this will prove useful across multiple application domains.

Spectral and cut sparsifiers

Another class of sparsifiers developed in theoretical computer science preserves a given feature of the graph to within some degree of precision. One prominent example is cut sparsifiers. These algorithms produce a subgraph for which the value of every cut is preserved up to a multiplicative factor of $1 \pm \epsilon$. A result from Benczr and Karger [2] shows that such a sparsification can be constructed in $\mathcal{O}(m \log^2 n)$ time for an unweighted graph and will contain $\mathcal{O}(n \log n / \epsilon^2)$ edges. This algorithm has primarily been used to construct more efficient approximation algorithms for computing cuts and flows in graphs [10].

Spectral sparsifiers are a generalization of cut sparsifiers that preserve the Laplacian quadratic form of the graph to within a multiplicative factor. Spielman and Teng [25] showed that a sparsification with $\mathcal{O}(n \log n / \epsilon^2)$ edges can be produced in $\mathcal{O}(m \log^c n)$ time for some constant c . Since the eigenvalues of the Laplacian matrix are related many properties of dynamic processes on networks, this sparsifier may be of interest for some network analysis problems. However, the method has not been tested on real-world (or synthetic) graphs, so it unknown how useful the results are, or whether the constant c is small enough to support effective scaling.

Backbone detection

Outside of computer science, methods for finding the “backbone” of a network have been developed by researchers working in other disciplines concerned with networks. Roughly, backbone detection attempts to find a core structure of links that are in some way the most significant in order to allow easier visualization and analysis. A traditional method for reducing edge density is simple thresholding: in a weighted graph, setting a global threshold and removing all edges with lower weights. However, this is clearly inappropriate for many real-world graphs, which display a heavy-tailed degree distribution, because no global threshold is appropriate for all parts of the network.

The first attempt at backbone detection was made by Serrano et al. [24]. They assume a null model where the normalized weights of a node with degree k are drawn from a random subdivision of the interval $[0, 1]$ into k segments. Edges are retained by the “disparity filter” when they are significant at level α with respect to this null distribution. This filter sets different thresholds depending on a node’s degree, which avoids the problems of global thresholds. It is worth noting that this method is only applicable to weighted networks.

Subsequent work has built on the disparity filter by proposing other null distributions by which to model edge weights. For example, Foti et al. [9] introduce a nonparametric method that uses the empirical distribution of edge weights at each node. Another recent algorithm developed by Radicchi et al. [22] uses a null model where the adjacency structure of the network is fixed but edge weights are randomly drawn from the empirical distribution of all weights in the network.

These methods are often useful for producing an illustrative summary of a network that contains the most important connections at each node. However, it is important to note that, contrary to the objective of this thesis, they do not attempt to preserve global properties of the network. Addition-

ally, they are only applicable to weighted networks.

Influence maximization

This thesis focuses on the application domain of influence maximization. The proposed algorithm is primarily designed to preserve the dynamic properties of the network. Thus, applications dealing with spreading processes and information flow are a natural fit. Influence maximization is one such domain, and has been the subject of a great deal of research. This allows the present algorithm to be benchmarked using a variety of existing methods.

Problem overview

The influence maximization problem, first formalized by Kempe, Kleinberger, and Tardos [15] as a discrete optimization problem, is to choose a seed set of k nodes which will maximize the spread of some information through a network. A model of influence propagation is assumed, where nodes are activated by their neighbors, and then have some chance of activating other nodes, and so on. The most popular models unfold in discrete time steps. In the Linear Threshold Model (LTM) [14], nodes become active at each step once a sum of activations from their neighbors passes a given threshold. In the Independent Cascade Model (ICM) [11], when a node becomes active it has one chance to infect its neighbors, and does so with some probability p_{ij} set by the underlying system. Kempe et al. [15] showed that the influence maximization problem is NP-hard under both models. This hardness result has motivated a wide range of work on approximation algorithms and heuristic optimization methods for the problem. Influence maximization appears to be a hard problem in general, not just the problem formulated by Kempe et al. For example, maximizing the influence spread under a continuous time transmission model was proven NP-hard

by Gomez-Rodriguez and Schölkopf [12]. Even good approximation ratios may be infeasible for the general case. For example, Chen [3] showed that a related problem of finding the smallest seed set which will influence the whole network (or a given fraction of it) cannot be approximated within a polylogarithmic factor. These results hold even for a number of special cases, and positive results were obtained only for extremely restricted classes of inputs such as trees.

Submodularity and the greedy algorithm

Approximation algorithms for influence maximization are commonly based on the property of submodularity. Roughly, submodularity formalizes the property of diminishing returns in the objective function of the influence maximization problem. Given a ground set of items X , a function $f : \mathcal{P}(X) \rightarrow \mathbb{R}$ is submodular if $f(\{x\} \cup A) - f(A) \leq f(\{x\} \cup B) - f(B)$ for all $x \in X \setminus (A \cup B)$ and $A, B \in \mathcal{P}(X)$ such that $A \subseteq B$. That is, the marginal value to adding an item x can only decrease as additional items are selected. Kempe et al. proved that the objective function for influence maximization (the expected spread of a given seed set) is submodular under particular models. Later, Mossel and Roch [20] extended this guarantee to a much more general setting. These results allow researchers on influence maximization to draw on the well-developed body of work on maximizing submodular functions. In particular, it is known that the greedy algorithm (defined as the one which successively chooses the item with highest marginal gain) achieves an approximation ratio of $1 - \frac{1}{e}$ when the objective function is submodular. Submodularity is critical; Kempe et al. showed that for models where the influence spread is not submodular, it is hard to approximate within a factor $n^{1-\varepsilon}$ for any $\varepsilon > 0$. Accordingly, Kempe et al. considered models where submodularity holds and introduced the use of the greedy algorithm for performing influence maximization. Leskovec et al. [17] improved on this algorithm by noting that submodularity allows lazy evaluation of the marginal gains, which saves considerable computational cost. The resulting algorithm, called CELF, has been widely used for influence maximization. It is important to note that even computing the influ-

ence spread of a set of nodes in common propagation models is #P-hard, as shown by Chen, Wang, and Wang [4]. Accordingly, the greedy algorithms of Kempe et al. and Leskovec et al. approximate the influence spread of the nodes under consideration via Monte Carlo simulations of the model. This simulation is by far the most computationally intensive part of the method, and later work has explored several ways to shortcut a full simulation of propagation through the network.

Heuristic approaches

Subsequent work on influence maximization has explored both heuristic optimization strategies and alternative models. Chen, Wang, and Wang [4] conclude from the hardness of computing influence spreads that strategies like CELF, which require this quantity to be explicitly evaluated, are unlikely to scale well. Accordingly, they design a heuristic which considers only a local region around each node and can be applied to graphs with millions of vertices. This algorithm, called PMIA, represents the state of the art for the general influence maximization problem. It works by considering a variation of the ICM called the maximum influence arborescence model (MIA). The algorithm finds the shortest path between each pair of nodes, and then discards paths whose total propagation probability falls below some threshold. Influence spreads in these arborescences can be computed much more efficiently than the traditional ICM, allowing a greedy selection strategy to run orders of magnitude faster. Another prominent method is the SPIM algorithm by Kimura and Saito [16], which simplifies the ICM by only allowing nodes to be influenced along the shortest path from the seed set, or a path that is at most one edge longer.

A different set of algorithms target the Linear Threshold Model. The LDAG algorithm [5] approximates the propagation of influence by considering a directed acyclic graph (DAG) around each node. Since influence in the Linear Threshold Model can be found in linear time when the graph is a DAG (compared to the #P-Hard problem of calculating it for the general case), this algorithm

provides a significant speedup over the standard greedy method. Another approach is the SimPath algorithm [13], which incorporates several optimizations. First, the authors note that influence in the LTM can be computed through the simple paths outwards from the seed set alone. Although enumerating simple paths is itself hard, since influence probabilities decay quickly, only a small neighborhood is considered. Second, it also suffices in the LTM to find the spread of each node's out-neighbors, so the algorithm uses a heuristic to approximate a vertex cover of the graph, and only computes the spread of nodes contained in the cover. Lastly, only the few most promising nodes are evaluated at each iteration. Together, these improvements make SimPath significantly more efficient than previous algorithms for the LTM.

Gomez-Rodriguez and Schölkopf [12] investigate an alternative model of influence propagation which uses a continuous-time model, motivated by the observation that information travels along different links in a network at varying speeds. They show that the influence spread in this model can be analytically evaluated, and employ a greedy algorithm to maximize it. Better results are achieved than PMIA, SP1M, and CELF, demonstrating the value of a continuous time approach. However, their algorithm is considerably less scalable and becomes infeasible for graphs with more than a thousand nodes.

Influence maximization is a topic that has received much attention due to its wide potential applications. Because state of the art methods often confront challenges to scalability, it is a promising application for sparsification algorithms.

CHAPTER 3: METHODOLOGY

Many questions of interest, such as influence maximization, deal in some way with the behavior of processes taking place on a network. A good proxy for such behavior is a random walk, which gives a generic model of dynamic processes. Subject to certain connectivity conditions, a random walk on a graph is an irreducible and aperiodic Markov chain which converges to a stationary distribution over the vertices independently of the initial condition. The proposed algorithm finds a sparsification which approximately preserves this limiting behavior. It does so by minimizing the Kullback-Leibler divergence between the stationary distribution of a walk on the original and sparsified networks.

Objective formulation

The stationary distribution of a walk on a random graph is given by

$$\pi(i) = \frac{d(i)}{2|E|}.$$

That is, the walk spends time at each node i proportional to its degree $d(i)$. Now, let $E' \subset E$ be a sparsification of the network. Let $d_{E'}(i)$ be the degree of node i , considering only edges contained in E' . The Kullback-Leibler divergence between the distribution of a walk on the original and

sparsified graphs is

$$\begin{aligned}
D_{KL}(\pi||\pi_{E'}) &= \sum_i \pi_i \log \frac{\pi(i)}{\pi_{E'}(i)} \\
&= \sum_i \pi(i) \left[\log \frac{d(i)}{2|E|} - \log \frac{d_{E'}(i)}{2|E'|} \right] \\
&= \sum_i \pi(i) \left[\log \frac{d(i)}{d_{E'}(i)} + \log \frac{|E'|}{|E|} \right] \\
&= \sum_i \pi(i) \left[\log \frac{d(i)}{d_{E'}(i)} \right] + \log \frac{|E'|}{|E|}
\end{aligned}$$

The final expression has two noteworthy properties. First, the final term depends only on the cardinality of E' . Therefore, it is invariant with respect to particular the choice of edges. Second, the summation is over terms which depend only on the degree of each node. That is, the objective function is local in the sense that the value of any given edge depends only on the terms in the summation corresponding to its endpoints, and no others. This will prove important for efficient optimization.

Unfortunately, it will not be possible to use this formulation in general because a sparsification could assign some node a degree of zero, resulting in an infinite KL divergence. However, it is possible to add a small modification to the input graph so that the terms are always finite. Specifically, we can consider a random surfer (as in the PageRank algorithm) instead of a random walker. A random surfer takes a step in the random walk with probability $1 - \tau$ for some $\tau > 0$, and with probability τ teleports to a random node in the graph. This teleportation probability is equivalent to adding directed edges from every vertex to every other vertex with sufficient weight so that one of these supplementary edges is followed with probability τ no matter what vertex the walker is at. In effect, the graph is fully connected by a set of weak edges, guaranteeing the existence of a

stationary distribution and a finite KL divergence.

As the algorithm progresses, maintaining a constant probability τ requires adjusting these weights. Let E_k be a subset of k edges: the set of edges obtained after the algorithm has been run for k steps. Let $d(i)$ be the degree of vertex i not counting these supplementary edges and $d_{E_k}(i)$ be the degree of i considering only edges which are included in E_k . Now, we will calculate the total weight which must be added outgoing from each vertex to create a teleportation probability of τ . Call this total weight w . For each vertex we have

$$\frac{w}{w + d_{E_k}(i)} = \tau$$

which implies

$$w = \frac{\tau}{1 - \tau} d_{E_k}(i).$$

Summing over all vertices, the weight directed towards each vertex is

$$\begin{aligned} c_{E_k} &\triangleq \sum_j \frac{1}{n} \left(\frac{\tau}{1 - \tau} \right) d_{E_k}(j) \\ &= \frac{1}{n} \left(\frac{\tau}{1 - \tau} \right) |E_k| \end{aligned}$$

which gives each vertex a final weight of $d_{E_k}(i) + c_{E_k}$. The corresponding total weight in the graph is $\frac{1}{1-\tau}|E_k|$. Having obtained these totals, we can derive the objective to be minimized. Call the stationary distribution of a random walk on the original graph π and the distribution on the

subgraph π_{E_k} . The Kullback-Leibler divergence given a sparsification E_k is

$$\begin{aligned}
D_{KL}(\pi||\pi_{E_k}) &= \sum_i \pi_i \log \frac{\pi(i)}{\pi_{E_k}(i)} \\
&= \sum_i \pi(i) \left[\log \frac{d(i) + c_E}{2 \left(\frac{1}{1-\tau}\right) |E|} - \log \frac{d_{E_k}(i) + c_{E_k}}{2 \left(\frac{1}{1-\tau}\right) |E_k|} \right] \\
&= \sum_i \pi(i) \left[\log \frac{d(i) + c_E}{d_{E_k}(i) + c_{E_k}} + \log \frac{|E_k|}{|E|} \right] \\
&= \sum_i \pi(i) \log \frac{d(i) + c_E}{d_{E_k}(i) + c_{E_k}} + \log \frac{|E_k|}{|E|}.
\end{aligned}$$

The reduction in the divergence after adding an edge (p, q) is

$$\begin{aligned}
D_{KL}(\pi||\pi_{E_k}) - D_{KL}(\pi||\pi_{E_{k+1}}) &= \left[\sum_i \pi(i) \log \frac{d(i) + c_E}{d_{E_k}(i) + c_{E_k}} + \log \frac{|E_k|}{|E|} \right] - \\
&\quad \left[\sum_i \pi(i) \log \frac{d(i) + c_E}{d_{E_{k+1}}(i) + c_{E_{k+1}}} + \log \frac{|E_{k+1}|}{|E|} \right] \\
&= \sum_i \pi(i) \log \frac{d_{E_{k+1}}(i) + c_{E_{k+1}}}{d_{E_k}(i) + c_{E_k}} + \log \frac{|E_k|}{|E_{k+1}|} \\
&= \sum_{i \notin \{p,q\}} \pi(i) \log \frac{d_{E_k}(i) + c_{E_{k+1}}}{d_{E_k}(i) + c_{E_k}} + \sum_{i \in \{p,q\}} \pi(i) \log \frac{d_{E_k}(i) + 1 + c_{E_{k+1}}}{d_{E_k}(i) + c_{E_k}} + \\
&\quad \log \frac{|E_k|}{|E_{k+1}|}.
\end{aligned}$$

The first term in this expression sums over all vertices that are not endpoints of the newly added edge. This term is small because the only change to the degree of these vertices comes from manipulations of the "teleport" edges in order to keep the total probability of teleportation constant. In fact, this change is small enough to be ignored in the optimization. Assuming that vertices have

degree $O(1)$, as is typical for real world networks:

$$\begin{aligned}
\sum_{i \notin \{p,q\}} \pi(i) \log \frac{d_{E_k}(i) + c_{E_{k+1}}}{d_{E_k}(i) + c_{E_k}} &= \sum_{i \notin \{p,q\}} \pi(i) \log \frac{d_{E_k}(i) + \frac{1}{n} \left(\frac{\tau}{1-\tau} \right) |E_{k+1}|}{d_{E_k}(i) + \frac{1}{n} \left(\frac{\tau}{1-\tau} \right) |E_k|} \\
&= \sum_{i \notin \{p,q\}} \pi(i) \log \frac{d_{E_k}(i) + \frac{1}{n} \left(\frac{\tau}{1-\tau} \right) (|E_k| + 1)}{d_{E_k}(i) + \frac{1}{n} \left(\frac{\tau}{1-\tau} \right) |E_k|} \\
&= \sum_{i \notin \{p,q\}} \pi(i) \log \left[1 + \frac{\frac{1}{n} \left(\frac{\tau}{1-\tau} \right)}{d_{E_k}(i) + \frac{1}{n} \left(\frac{\tau}{1-\tau} \right) |E_k|} \right] \\
&= \sum_{i \notin \{p,q\}} \pi(i) \log \left[1 + \frac{\frac{1}{n} \left(\frac{\tau}{1-\tau} \right)}{O(1) + \frac{1}{n} \left(\frac{\tau}{1-\tau} \right) O(n)} \right] \\
&= \sum_{i \notin \{p,q\}} \pi(i) \log \left[1 + O\left(\frac{1}{n}\right) \right] \\
&< \sum_i \pi(i) \log \left[1 + O\left(\frac{1}{n}\right) \right] \\
&= \log \left[1 + O\left(\frac{1}{n}\right) \right].
\end{aligned}$$

This term quickly goes to 0 as n increases. Accordingly, we will not consider this term in the subsequent analysis, and will simply augment the degree of each node by c_{E_k} to ensure that the stationary distribution exists at each step. Where this term does not impact the results, it will be absorbed into the degree of each node to keep notation compact.

Problem statement

Given this formulation of the objective, we wish to solve the problem

$$E_k = \arg \min_{E' \subset E, |E'| \leq k} D_{KL}(\pi || \pi_{E'})$$

Here, k is the number of edges that the sparsification will contain. In practice, k could be fixed ahead of time, or it could be chosen during the execution of the algorithm (e.g. stopping execution when the marginal gain of adding an edge drops below some threshold).

Proposed algorithm

To introduce the proposed algorithm define for all $E' \subset E, u \in E/E'$

$$\Delta D_{KL}(E', u) = D_{KL}(\pi || \pi_{E'}) - D_{KL}(\pi || \pi_{E' \cup \{u\}})$$

which is the marginal benefit of adding an edge u . The greedy algorithm maximizes this marginal benefit at each stage of the algorithm:

$$E_0 = \emptyset$$

$$E_{k+1} = E_k \cup \arg \max_{u \in E/E'} \Delta D_{KL}(E_k, u) \quad \forall k < k_{\max}$$

That is, each stage of the algorithm produces a sparsification with $k + 1$ edges from one with k edges by maximizing the marginal gain of the edge that is added.

Runtime analysis

The computational cost of the algorithm comes from computing the marginal benefits and ordering them to choose the greatest. In order to minimize these costs, the marginal benefits can be stored in a priority queue (implemented, e.g., using a binary heap) and updated only when necessary. Because of the locality property of the objective, when an edge is added, the only marginal benefits which change are those of edges which share a vertex with the most recent addition. For

each of these modifications, recomputation of the marginal benefit takes time $\mathcal{O}(1)$, and updating the priority queue takes time $\mathcal{O}(\log m)$. Assuming that each node has degree $\mathcal{O}(1)$, realistic for real-world networks, the runtime per iteration of the algorithm is $\mathcal{O}(\log m)$ because only $\mathcal{O}(1)$ adjustments to the heap are needed. Thus, final runtime is $\mathcal{O}(k \log m)$. In this formulation, the runtime depends only logarithmically on the size of the network. However, it will remain to be seen experimentally whether larger networks require a proportionally greater number of edges, which would imply a computational cost of $\mathcal{O}(m \log m)$. A potentially useful fact is that if sparsifications of varying densities are needed for the same network, then the runtime depends only on the largest number of edges desired. The greedy algorithm performs an online construction which finds all sparsifications of lesser cardinality en route to the final solution.

Lazy Evaluation and Submodularity

The runtime of the proposed algorithm can be greatly decreased in practice by noting that the marginal gain of adding each edge can be evaluated in a lazy manner. Leskovec et al. [17] showed that when the objective function is submodular, the number of evaluations of marginal gains can be greatly reduced. Specifically, each item can be marked as “valid” or “invalid”. In the first round, each item is evaluated and the one with the greatest value is chosen. In subsequent rounds, every item starts out marked as invalid. Then, the item with the greatest value is examined. If the item is marked valid, it is chosen. If it is marked invalid, then its marginal gain is evaluated. The item is then marked as valid and reinserted into the priority queue. The algorithm then draws the next element from the queue, proceeding in this manner until the item with the greatest value is marked valid. Importantly, this item can be chosen even if other elements are invalid. Submodularity guarantees that an item’s value from a previous round serves as an upper bound on its value after more items have been selected. Oftentimes, the best item at the start of the round will remain the

best after it has been reevaluated. In this case, that item can be selected immediately, saving the evaluation of all of the other items.

Although the objective function considered here is not submodular, this is due only to a term that is constant for all edges. Therefore, edges can be ranked according to a function that *is* submodular, allowing the CELF optimization to be applied.

Theorem 1. *Let $f(E')$ be the negative Kullback-Leibler divergence between the stationary distribution of a random walk on graph G and on a sparsified graph with edges E' . Then, f is submodular up to an additive term independent of the edges contained in E'*

Proof. Given a sparsification E' , the marginal gain to adding an edge x connecting nodes u and v is

$$\Delta f(x, E') = \sum_{i \notin \{u, v\}} \pi(i) \log \frac{d_{E_k}(i) + c_{E_{k+1}}}{d_{E_k}(i) + c_{E_k}} + \sum_{i \in \{u, v\}} \pi(i) \log \frac{d_{E_k}(i) + 1 + c_{E_{k+1}}}{d_{E_k}(i) + c_{E_k}} + \log \frac{|E_k|}{|E_{k+1}|}.$$

The last term is independent of the choice of edges, so we will restrict consideration to the first two terms. The first two terms are both submodular, so their nonnegative linear combination is submodular as well. For both cases, we will use the property that for any constant c , the function $g(x) = \log \frac{x+c}{x}$ is monotonically decreasing in x because $\frac{\partial g}{\partial x} = -\frac{c}{cx+x^2} < 0$ for $x > 0$. Consider the first term. Note that the definition of c_{E_k} allows us to rewrite it as

$$\sum_{i \notin \{u, v\}} \pi(i) \log \frac{d_{E_k}(i) + c_{E_k} + \frac{1}{n} \left(\frac{\tau}{1-\tau} \right)}{d_{E_k}(i) + c_{E_k}}.$$

Since $|A| \leq |B|$, $c_A \leq c_B$. Additionally, $d_A(i) \leq d_B(i) \forall i \in V$. Therefore, each term in the summation is at least as great for A as for B . Now consider the second term. It may be rewritten

in the same manner as

$$\sum_{i \in \{u,v\}} \pi(i) \log \frac{d_{E_k}(i) + c_{E_k} + \frac{1}{n} \left(\frac{\tau}{1-\tau} \right) + 1}{d_{E_k}(i) + c_{E_k}}$$

and the same argument applies. □

This result allows the CELF optimization to be incorporated into the proposed algorithm, greatly reducing the number of evaluations which are needed in practice.

CHAPTER 4: RESULTS

This chapter presents an evaluation of the sparsification algorithm proposed for this thesis on both a theoretical and empirical level. The first section is devoted to theoretical analysis. The following two sections introduce the experimental design and present empirical results on the algorithm's performance.

Theoretical analysis

This section will analyze the greedy optimization strategy, and show that it obtains the optimal sparsification with respect to the KL divergence. As explained in Chapter 3, the objective function for the sparsification problem can be expressed as

$$f(E') = \sum_{i \in V} \frac{d(i)}{2|E|} \left[\log \frac{d(i)}{d_{E'}(i)} + \log \frac{|E'|}{|E|} \right],$$

in which the summation decomposes into two terms. The first is specific to the edges which are chosen for the sparsification, while the second depends only on the number of edges which are chosen. Therefore, we can analyze the simpler function

$$g(E') = \sum_{i \in V} \frac{d(i)}{2|E|} \log \frac{d(i)}{d_{E'}(i)}, \tag{4.1}$$

which omits the unnecessary term. It will be helpful to define the marginal gain of adding a single edge u connecting vertices i and j as

$$\begin{aligned}
\Delta g(E', u) &= g(E') - g(E' \cup \{u\}) \\
&= \sum_{i \in V} \frac{d(i)}{2|E|} \log \frac{d_{E' \cup \{u\}}(i)}{d_{E'}(i)} \\
&= \frac{d(i)}{2|E|} \log \frac{d_{E'}(i) + 1}{d_{E'}(i)} + \frac{d(j)}{2|E|} \log \frac{d_{E'}(j) + 1}{d_{E'}(j)}. \tag{4.2}
\end{aligned}$$

The main result of this section is the following:

Theorem 2. *Let $E_k = \arg \min_{|E'|=k} g(E')$ be the optimal sparsification with respect to g which contains k edges. Then, $E_{k+1} = E_k \cup \{\arg \max_{u \in E \setminus E_k} \Delta g(E_k, u)\}$, $0 \leq k < m$.*

From this, it easily follows that the greedy algorithm outlined in Chapter 3 achieves the optimal value. In order to prove Theorem 2, we will need the following lemma:

Lemma 1. *If for some sparsification E' there exist subsets of edges $A \subseteq E'$, $B \subseteq E \setminus E'$ such that $|A| = |B|$ and $\Delta g(E' \setminus A, B) > \Delta g(E' \setminus A, A)$ then there exist edges $e_A \in A$ and $e_B \in B$ such that $\Delta g(E' \setminus \{e_A\}, \{e_B\}) > \Delta g(E' \setminus \{e_A\}, \{e_A\})$*

Proof. Suppose that there exist no such edges. Then, we can find an upper bound on $g(E' \setminus A, B)$ and a lower bound on $\Delta g(E' \setminus A, A)$ which together imply $\Delta g(E' \setminus A, A) \geq \Delta g(E' \setminus A, B)$, a contradiction of the hypothesis. Start with the upper bound on $\Delta g(E' \setminus A, B)$. Let $\Delta d(i) =$

$d_{(E' \setminus A) \cup B}(i) - d_{E' \setminus A}(i)$. Then,

$$\begin{aligned}
\Delta g(E' \setminus A, B) &= \sum_{i \in V} \log \frac{d(i) + \Delta d(i)}{d(i)} \\
&= \sum_{i \in V} \log \frac{d(i) + 1}{d(i)} + \log \frac{d(i) + 2}{d(i) + 1} + \dots + \log \frac{d(i) + \Delta d(i)}{d(i) + \Delta d(i) - 1} \\
&\leq \sum_{i \in V} \Delta d(i) \log \frac{d(i) + 1}{d(i)} \\
&= \sum_{u \in A} \Delta g(E' \setminus A, u)
\end{aligned}$$

where the final equality follows because B and E' are disjoint. A similar argument lets us find a lower bound on $\Delta g(E' \setminus A, A)$. Let $\Delta d(i) = d_{E'}(i) - d_{E' \setminus A}(i)$. We have

$$\begin{aligned}
\Delta g(E' \setminus A, A) &= \sum_{i \in V} \log \frac{d(i) + \Delta d(i)}{d(i)} \\
&= \sum_{i \in V} \log \frac{d(i) + 1}{d(i)} + \log \frac{d(i) + 2}{d(i) + 1} + \dots + \log \frac{d(i) + \Delta d(i)}{d(i) + \Delta d(i) - 1} \\
&\geq \sum_{i \in V} \Delta d(i) \log \frac{d(i) + \Delta d(i)}{d(i) + \Delta d(i) - 1} \\
&= \sum_{u \in A} \Delta g(E' \setminus A, u)
\end{aligned}$$

where the last equality follows because $A \subset E'$. Supposing that there are no edges $e_A \in A$ and $e_B \in B$ with $\Delta g(E' \setminus A, e_b) > \Delta g(E' \setminus A, e_A)$, then $\sum_{u \in A} \Delta g(E' \setminus A, u) \geq \sum_{u \in B} \Delta g(E' \setminus A, u)$. Combining these bounds we have

$$\Delta g(E' \setminus A, B) \leq \sum_{u \in B} \Delta g(E' \setminus A, u) \leq \sum_{u \in A} \Delta g(E' \setminus A, u) \leq \Delta g(E' \setminus A, A)$$

which contradicts the hypothesis that $\Delta g(E' \setminus A, B) > \Delta g(E' \setminus A, A)$. Therefore, we must conclude that the desired edges exist. \square

This lemma gives us the necessary tools to prove the main result.

Proof of Theorem 1. The statement follows by induction. The base case holds trivially. $E_0 = \emptyset$ so

$$\begin{aligned} E_1 &= \{\arg \max_{u \in E} \Delta g(\emptyset, u)\} \\ &= E_0 \cup \{\arg \max_{u \in E \setminus E_0} \Delta g(E_0, u)\} \end{aligned}$$

Now assume that the $E_{i+1} = E_i \cup \{\arg \max_{u \in E \setminus E_i} \Delta g(E_i, u)\} \forall i < k$ and we will show that the statement holds for k . This is because after adding the next edge, it will never become preferable to exchange edges included in E_k for edges that were previously excluded. Let $e^* = \arg \max_{u \in E \setminus E_k} \Delta g(E_k, u)$. For now, define $E_{k+1} \triangleq E_k \cup \{e^*\}$, and the proof will show that E_{k+1} is in fact the optimal set of $k + 1$ edges.

Suppose for contradiction that there exist subsets $A \subseteq E_{k+1}$ and $B \subseteq E \setminus E_{k+1}$ with $|A| = |B|$ and $\Delta g(E_{k+1} \setminus A, B) > \Delta g(E_{k+1} \setminus A, A)$. Then, by Lemma 1 there exist edges $e_A \in A$ and $e_B \in B$ such that $\Delta g(E_{k+1} \setminus \{e_A\}, \{e_B\}) > \Delta g(E_{k+1} \setminus \{e_A\}, \{e_A\})$. That is, there is at least one unincluded edge that it would be preferable to include in E_{k+1} over an existing edge.

Case 1: e_A does not share a vertex with e^* . Because E_k is optimal, $\Delta g(E_k \setminus \{e_A\}, \{e_A\}) \geq \Delta g(E_k \setminus \{e_A\}, \{e_B\})$. However, because neither e_A or e_B share a vertex with e^* , no term in the summation in Equation 4.2 changes for either edge. Therefore, $\Delta g(E_{k+1} \setminus \{e_A\}, \{e_A\}) \geq \Delta g(E_{k+1} \setminus \{e_A\}, \{e_B\})$, which contradicts the assertion that there exist such an e_A and e_B .

Case 2: e_A shares a vertex with e^* , and this vertex is also shared with e_B . Call the common vertex i . Call the vertex of e_A that is not common with e_B vertex j , and the corresponding vertex for e_B vertex k . Because i is common to e_A and e_B , the summations in $\Delta g(E_{k+1} \setminus \{e_A\}, \{e_A\})$ and $\Delta g(E_{k+1} \setminus \{e_A\}, \{e_B\})$ differ only at the terms related to the vertices that are not shared. Therefore,

$\Delta g(E_{k+1} \setminus \{e_A\}, \{e_B\}) > \Delta g(E_{k+1} \setminus \{e_A\}, \{e_A\})$ can only hold if the term for k is greater than the term for j . But, if this were true then we would have $\Delta g(E_k \setminus \{e_A\}, \{e_B\}) \geq \Delta g(E_k \setminus \{e_A\}, \{e_A\})$ because all terms except the one related to vertex i are unchanged by the addition of e^* . This contradicts that E_k is optimal because it would have been preferable to include e_B at some earlier stage.

Case 3: e_A shares a vertex with e , and this vertex is not shared with e_B . Note that e^* 's other vertex may be shared with e_B without substantially altering the argument since this only decreases the marginal value of e_B . Call the shared vertex i , the other vertex belonging to e_A vertex k , the other vertex belonging to e^* vertex j , and the vertices of e_B m and n . Because $e^* \notin E_k$, we must have

$$\frac{d(k)}{2|E|} \frac{d_{E_k}(k)}{d_{E_k}(k) - 1} \geq \frac{d(j)}{2|E|} \frac{d_{E_k}(j) + 1}{d_{E_k}(j)}$$

since all other terms of the respective Δg functions are equal. Additionally, because e^* is the optimal edge to add, it must hold that

$$\frac{d(i)}{2|E|} \frac{d_{E_k}(i) + 1}{d_{E_k}(i)} + \frac{d(j)}{2|E|} \frac{d_{E_k}(j) + 1}{d_{E_k}(j)} \geq \frac{d(m)}{2|E|} \frac{d_{E_k}(m) + 1}{d_{E_k}(m)} + \frac{d(n)}{2|E|} \frac{d_{E_k}(n) + 1}{d_{E_k}(n)},$$

again because all other terms in the summation are equal between e^* and e_B . Note also that the degrees of k , m , and n do not change from E_k to E_{k+1} , while the degree of i increases by one.

Together, these facts imply

$$\frac{d(i)}{2|E|} \frac{d_{E_{k+1}}(i)}{d_{E_{k+1}}(i) - 1} + \frac{d(k)}{2|E|} \frac{d_{E_{k+1}}(k)}{d_{E_{k+1}}(k) - 1} \geq \frac{d(m)}{2|E|} \frac{d_{E_{k+1}}(m) + 1}{d_{E_{k+1}}(m)} + \frac{d(n)}{2|E|} \frac{d_{E_{k+1}}(n) + 1}{d_{E_{k+1}}(n)},$$

which in turn proves that $\Delta g(E_{k+1} \setminus \{e_A\}, \{e_A\}) \geq \Delta g(E_{k+1} \setminus \{e_A\}, \{e_B\})$. This contradicts the requirements that there exists such an e_A and e_B .

Because all cases result in a contradiction, we must reject the assumption that there exist subsets $A \subseteq E_{k+1}$ and $B \subseteq E \setminus E_{k+1}$ such that $\Delta g(E_{k+1} \setminus A, B) > \Delta g(E_{k+1} \setminus A, A)$. If no such subsets exist, then $E_{k+1} = E_k \cup \{e^*\}$ is indeed the optimal set of $k + 1$ edges, as desired. \square

The conclusion from this section is that the chosen algorithmic approach is necessarily optimal given the problem formulation. The following experimental results evaluate this formulation in practice by testing the proposed algorithm as a preprocessing step for various influence maximization problems.

Experimental design

In order to evaluate the practical benefits of applying sparsification, this section presents results obtained from several real-world network datasets. The experiments focus on the use of sparsification as a preprocessing step for influence maximization. Accordingly, the impact of sparsification is examined for several different influence maximization algorithms, demonstrating that sparsification allows high-quality results to be obtained in a fraction of computational time that would be required when using the entire network.

Datasets

There are two principle datasets used in this analysis:

- **net-HEP**: This is a collaboration network taken from the high-energy physics section of the arXiv. The nodes correspond to authors, who are connected by an edge if they have coauthored at least one paper together. This network contains 15,233 nodes and 31,398

edges. The data set has been frequently used as a test case for influence maximization [15, 4, 13] and is available from <http://research.microsoft.com/en-us/people/weic/projects.aspx>.

- email-Enron: This is the dataset of email correspondence from the Enron corporation. Nodes correspond to employees, and edges link those who have exchanged at least one email. This dataset is larger, containing 36,692 nodes and 183,831 edges. It is available from the SNAP network repository [18].

Influence maximization algorithms

Several influence maximization algorithms are used for the experiments in this paper:

- CELF: The greedy algorithm of Leskovec et al. [17], which incorporates lazy evaluation of marginal gains.
- SP1M: An algorithm by Kimura and Saito [16] which approximates the influence spread in the ICM by assuming that each node can be influenced along the shortest path from the seed set, or a path that is at most one edge longer.
- PMIA: A state of the art influence maximization algorithm by Chen et al. [4]. PMIA restricts influence spread evaluations to a local area around each node.
- InfluMax: A recent algorithm developed by Gomez-Rodriguez and Schölkopf [12]. It performs influence maximization in a continuous-time version of the ICM. InfluMax has been shown to achieve higher influence spreads than previous approaches, but is not scalable to large networks.

Sparsification algorithms

Several sparsification algorithms are evaluated based on how well they serve as a preprocessing step for influence maximization. As explained in Chapter 2, previous sparsification methods are not directly comparable to the algorithm developed here. The closest is that of Mathioudakis et al. [19], which also deals with sparsification of influence networks. However, the authors assume that the algorithm has access to observed traces of information diffusions in the network. The algorithm proposed here uses only the adjacency structure of the network, so it would not be useful to directly compare the two approaches. Accordingly, the proposed algorithm is compared to two baseline approaches which represent common heuristics for such tasks. The algorithms considered are:

- RW: The random-walk based sparsifier proposed here.
- Degree: Rank edges according to the sum of the degrees of their endpoints and choose the top k .
- Centrality: Rank edges according to the sum of the eigenvector centrality of their endpoints and choose the top k .

Experimental results

This section presents the performance of each of the three sparsification algorithms on the chosen datasets. The first two subsections deal with the tradeoff between the influence spread that is achieved vs the computational time required for each influence maximization algorithm, while the last subsection presents results in which the CELF and InfluMax algorithms are applied to unprecedentedly large datasets using the proposed sparsification algorithm.

Influence spread

The objective of influence maximization is to compute the optimal seed set with respect to the expected influence spread on the graph. Therefore, the natural benchmark for a chosen sparsification is how closely the influence spread achieved on the sparsified graph approaches the spread on the full graph. The results presented here characterize the gain in influence spread as the sparsifier is allowed to select an increasing fraction of edges in the graph. For each sparsifier and influence maximization algorithm, the influence spread achieved using a given fraction of edges is compared to that achieved on the full graph. Note that the objective function is always evaluated using the complete graph: seed nodes are selected using the sparsified graph, and then the propagation model is simulated on the full graph in order to calculate the influence spread. Figure 4.1 shows results for the net-HEP dataset, and Figure 4.2 shows results for the email-Enron dataset. Due to its high computational cost, CELF is not applied to the email-Enron dataset. InFLUmax is not sufficiently scalable to be directly applied to either full dataset. The colored lines show the influence spread achieved by each sparsifier, while the black line provides the results obtained on the full graph for comparison. Each influence maximization algorithm is given a budget of 50 nodes.

An immediate conclusion from these results is that sparsification is extremely effective for the influence maximization task: only a small fraction of the original edges are needed to obtain the same quality of results. This holds across both datasets. However, the sparsifiers under consideration do not converge equally quickly to the optimal value. On the larger dataset, email-Enron, the RW algorithm converges extremely quickly, while Degree and Centrality sometimes require as much as 50% of the edges in order to reach the same value. On the net-HEP dataset, both Degree and RW converge relatively quickly, with Degree converging somewhat faster.

A particularly interesting set of results concerns the high-sparsity regime, where each sparsifier is allowed to select only 5-500 edges. For convenience, the right column of both plots magnifies this

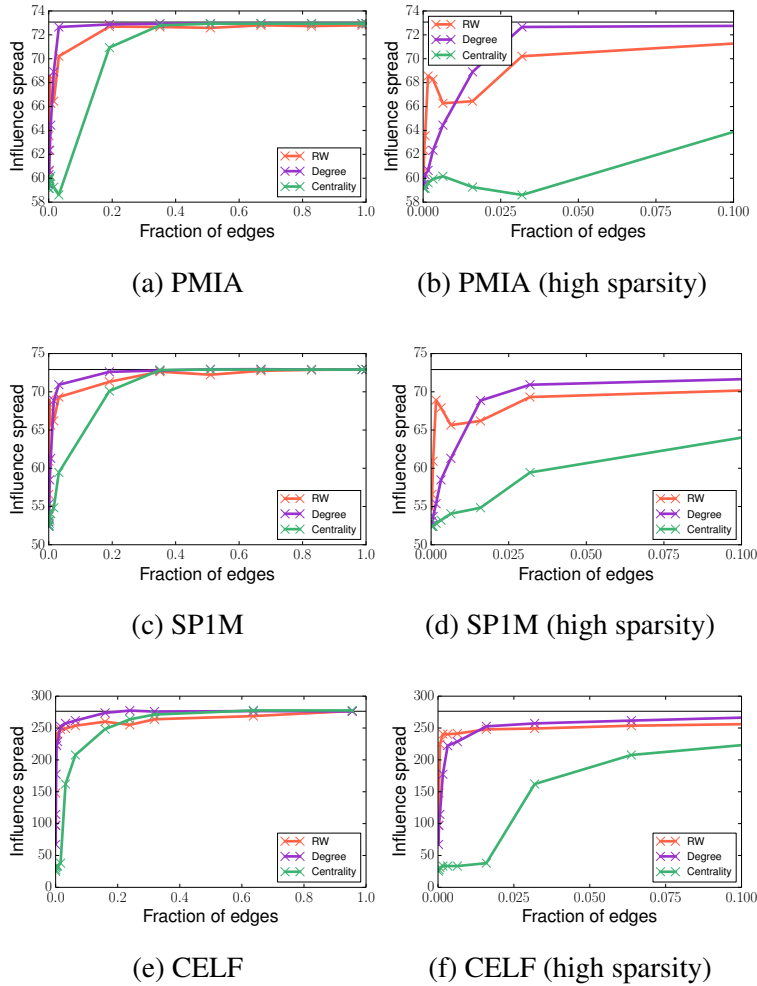


Figure 4.1: Influence spread for the net-HEP dataset

range of the graph. On the email-Enron network, only 500 edges (out of over 118,000) are required for the RW algorithm to reach the optimal value for both the PMIA and SP1M algorithms. On the net-HEP dataset, the performance of RW dips slightly at intermediate values. However, the results are already close to optimal at only 20 edges (out of over 31,000), and reach optimality by 500 edges. It is striking that the number of edges required for RW to produce optimal results is approximately the same for both datasets, one of which contains nearly 4 times the number of edges. This property, which is not shared by the baseline algorithms, is extremely useful for sparsification

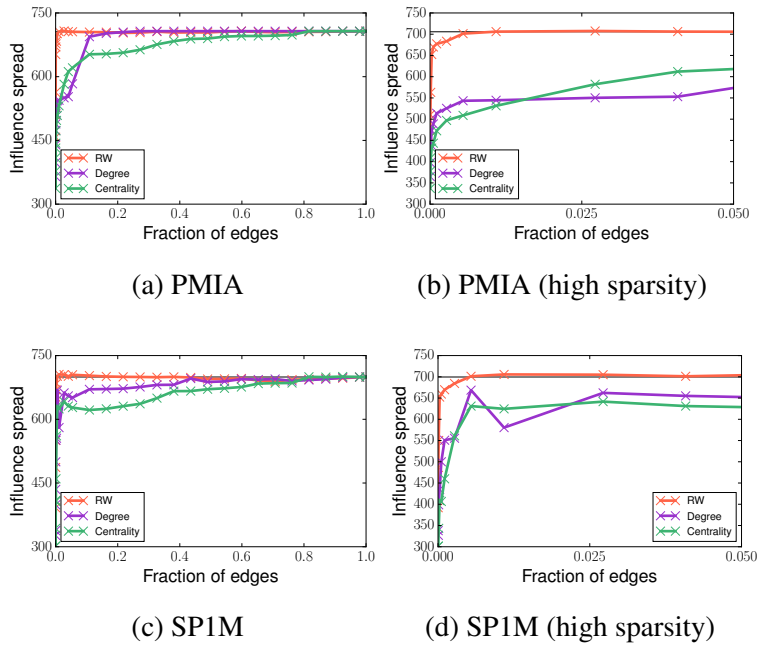


Figure 4.2: Influence spread for the email-Enron dataset

because it means that the number of edges which must be selected grows only slowly as a function of the graph size. So, graphs which are orders of magnitude too large for a given influence maximization algorithm could potentially be analyzed using sparsification without a significant loss in quality. This possibility is further explored later in the chapter.

Speedup

This section characterizes the tradeoff between the fraction of edges selected and the amount of time required to run the influence maximization algorithm. While both PMIA and SP1M are sufficiently scalable that the speedups are not necessary for the two sample datasets, these results help show the utility of sparsification for addressing datasets that are currently out of reach for these algorithms (unfortunately, analyzing such graphs is too memory-intensive for the computational

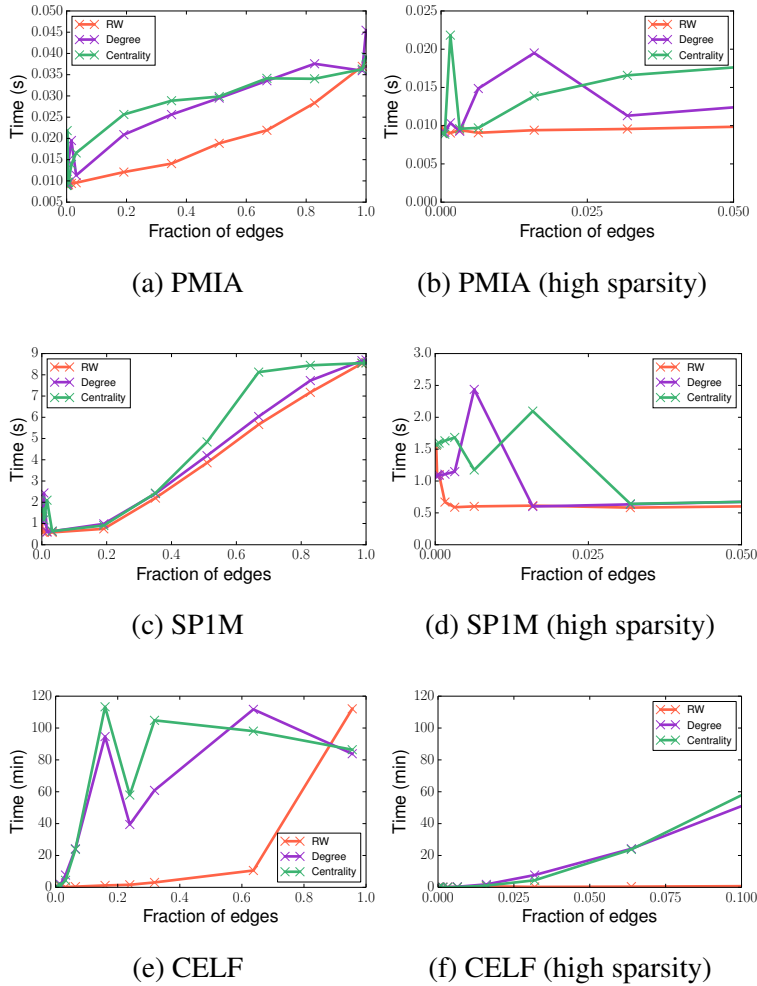


Figure 4.3: Runtime for the net-HEP dataset

resources available at present). Figure 4.3 presents results for the net-HEP dataset and Figure 4.4 presents results for the email-Enron dataset.

In all but one case, the RW algorithm provides the slowest growth in runtime as edges are added. It is important to note that the growth in the runtime for RW is always sublinear, though this is sometimes not the case for Degree and Centrality. Since the results of the previous section suggest that it will typically be sufficient to keep only a small fraction of edges, these results are magnified

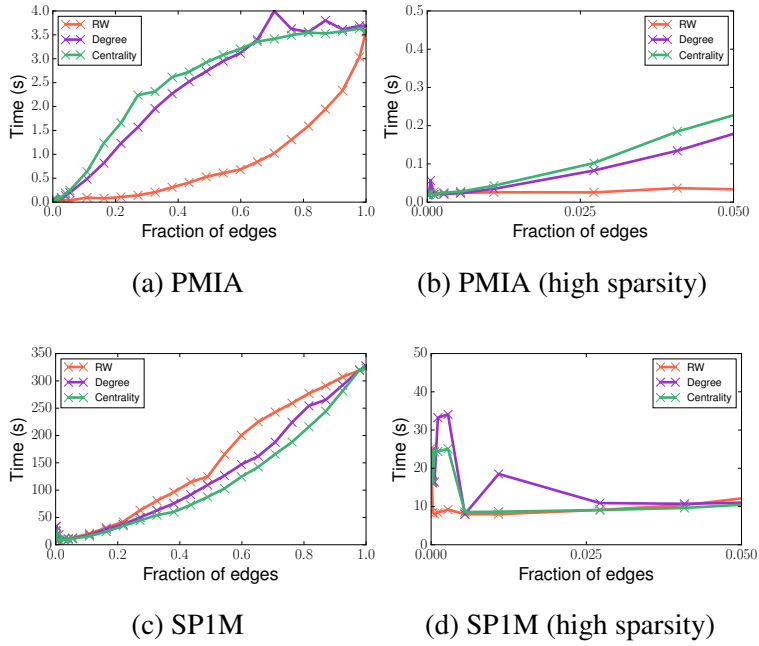


Figure 4.4: Runtime for the email-Enron dataset

in the right column of both figures. Here, it is apparent that optimal results can be delivered in only a small fraction of the time required by the original algorithm. On the larger email-Enron dataset, applying sparsification with 500 edges yields a speedup of anywhere from 13-35x, compared to using the full network. This suggests that sparsification can achieve dramatic speedups without sacrificing significant quality in the resulting influence spread.

Scaling to large networks

One of the most promising uses of sparsification is to apply influence maximization algorithms to networks which would otherwise be too large for computational tractability. Two of the algorithms considered here, CELF and InfluMax, are not scalable to large graphs. However, they can be used on graphs which are an order of magnitude larger than was previously possible using sparsification.

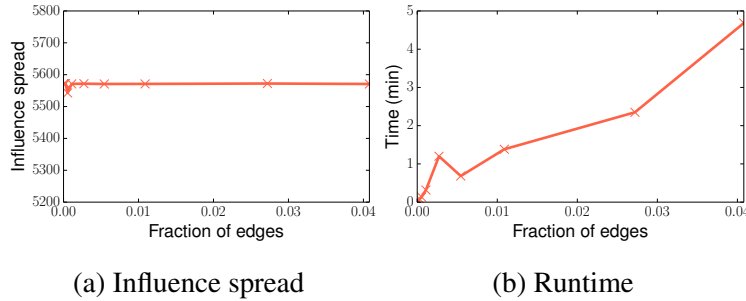


Figure 4.5: CELF applied to the email-Enron dataset

CELF has previously been applied to the net-HEP dataset [4]. However, larger networks have either been dismissed as impractical [4], or else running CELF on them required amounts of time that are usually infeasible (7 days in [13]). Here, CELF is applied to the email-Enron network in a matter of minutes. Figure 4.5 shows the runtime and influence spread of CELF applied to this dataset. The influence spread shown in Figure 4.5a increases only slightly as the algorithm runs, suggesting that a near-optimal seed set has been found. As can be seen in 4.5b, the runtime is uniformly less than five minutes.

InfluMax, while achieving influence spreads greater than that of PMIA or SPIM, has never been applied to a large network. Indeed, Du et al. [7] showed that it required more than 24 hours to select 3 seed nodes on a synthetic network with 128 nodes and 320 edges. Here, InfluMax is applied to net-HEP, which contains over 15,000 nodes and 31,000 edges, again with a budget of 3 nodes. Figure 4.6c shows the runtime of InfluMax on net-HEP. With sparsification, InfluMax can be run on a 31,000-edge network in less than an hour. Unfortunately, the true influence cannot be calculated because evaluating the influence spread of a seed set in the continuous time model used by InfluMax is intractable for large networks. However, Figure 4.6a shows the influence spread that is achieved on the sparsified network. This quantity increases as more edges are added. However, an increase should be expected regardless of whether the true influence spread is different

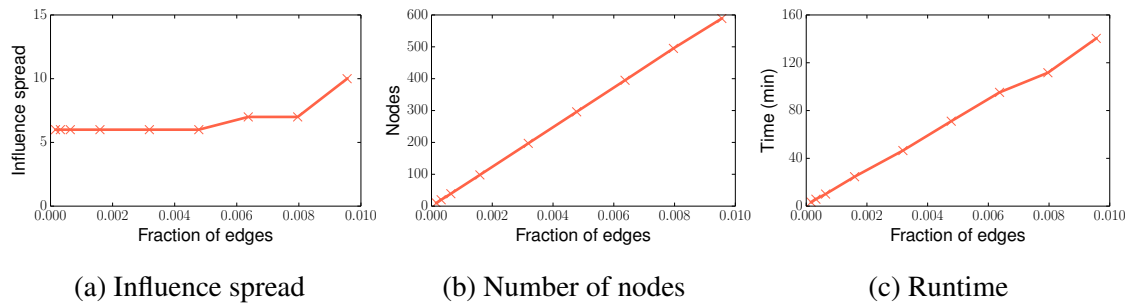


Figure 4.6: InfluMax applied to the net-HEP dataset

since additional edges increase the number of nodes that are reachable. Figure 4.6b shows that the number of nodes with non-zero degree after sparsification increases much faster than the influence spread as more edges are added. While not conclusive, this is reason to think that sparsification results in influence spreads that are close to the true value. Thus, the examples of both CELF and InfluMax demonstrate that sparsification can help algorithms scale to datasets which were previously out of reach.

CHAPTER 5: CONCLUSION

Dynamic processes are a crucial part of networked systems, and analysis of their properties underpins many tasks in network analysis. Because such processes pose hard computational problems, it is necessary to develop scalable algorithms which provide good approximations to the optimal solution. This thesis proposed a new sparsification algorithm specifically targeted at preserving the properties of dynamic processes while using a dramatically smaller set of edges. A formulation based on the stationary properties of random walks on the graph gives rise to an algorithm which is both optimal and computationally efficient.

This algorithm was evaluated on two real-world network datasets taken from the arXiv and the Enron emails. The experiments focused on the influence maximization task, as this provides an example of a computationally hard problem dealing with dynamic behavior which has been the subject of a great deal of recent work. The results show that the proposed sparsification method allows near-optimal sets of seed nodes to be found at a small fraction of the computational cost. Strikingly, the number of edges that must be retained for near-optimal results appears to grow only very slowly with the size of the network. This property could prove extremely useful across a variety of other influence maximization algorithms and perhaps other domains entirely.

Because only a small number of edges need be retained, two influence maximization algorithms were applied to datasets which previous work had considered out of reach. In both cases (CELF and InfluMax), sparsification allowed influence maximization to be performed on a network that was at least an order of magnitude larger than was previously possible. These experiments demonstrate that sparsification is a vital method for scaling algorithms to the extremely large datasets made available by social media.

Future work can improve the formulation of the objective by incorporating additional information.

For example, nodes and edges in a network are typically endowed with attributes such as preferences or group identities. Because of tendencies such as homophily, such attributes will impact the behavior of dynamic processes. Incorporating knowledge of attributes, when such information is available, could greatly improve the accuracy of the resulting model. Additionally, this work has focused on the stationary properties of a random walk for the sake of computational and analytic tractability. However, real-world processes are often far from stationarity, and modeling nonstationary behavior is another promising way of extending the algorithm

A final direction for future work is to test the sparsification algorithm on additional domains related to dynamic processes. While influence maximization is a well-studied example of such problems, examining additional tasks could help to better characterize the scope of problems that sparsification is useful for.

LIST OF REFERENCES

- [1] Edoardo M Airoldi, David M Blei, Stephen E Fienberg, and Eric P Xing. Mixed membership stochastic blockmodels. In *Advances in Neural Information Processing Systems*, pages 33–40, 2009.
- [2] Andras Benczur and David R Karger. Randomized approximation schemes for cuts and flows in capacitated graphs. *arXiv preprint cs/0207078*, 2002.
- [3] Ning Chen. On the approximability of influence in social networks. *SIAM Journal on Discrete Mathematics*, 23(3):1400–1415, 2009.
- [4] 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.
- [5] Wei Chen, Yifei Yuan, and Li Zhang. Scalable influence maximization in social networks under the linear threshold model. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 88–97. IEEE, 2010.
- [6] Aaron Clauset, Mark EJ Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical review E*, 70(6):066111, 2004.
- [7] Nan Du, Le Song, Manuel Gomez-Rodriguez, and Hongyuan Zha. Scalable influence estimation in continuous-time diffusion networks. In *Advances in Neural Information Processing Systems*, pages 3147–3155, 2013.
- [8] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010.

- [9] Nicholas J Foti, James M Hughes, and Daniel N Rockmore. Nonparametric sparsification of complex multiscale networks. *PloS One*, 6(2):e16431, 2011.
- [10] Wai Shing Fung, Ramesh Hariharan, Nicholas JA Harvey, and Debmalya Panigrahi. A general framework for graph sparsification. In *Proceedings of the forty-third annual ACM Symposium on Theory of Computing*, pages 71–80. ACM, 2011.
- [11] 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*, 12(3):211–223, 2001.
- [12] Manuel Gomez-Rodriguez and Bernhard Schölkopf. Influence maximization in continuous time diffusion networks. *arXiv preprint arXiv:1205.1682*, 2012.
- [13] Amit Goyal, Wei Lu, and Laks VS Lakshmanan. Simpath: An efficient algorithm for influence maximization under the linear threshold model. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 211–220. IEEE, 2011.
- [14] Mark Granovetter. Threshold models of collective behavior. *American Journal of Sociology*, pages 1420–1443, 1978.
- [15] 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*, pages 137–146. ACM, 2003.
- [16] Masahiro Kimura and Kazumi Saito. Tractable models for information diffusion in social networks. In *Knowledge Discovery in Databases: PKDD 2006*, pages 259–271. Springer, 2006.
- [17] 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. ACM, 2007.
- [18] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. <http://snap.stanford.edu/data>, June 2014.
- [19] Michael Mathioudakis, Francesco Bonchi, Carlos Castillo, Aristides Gionis, and Antti Ukkonen. Sparsification of influence networks. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 529–537. ACM, 2011.
- [20] Elchanan Mossel and Sebastien Roch. On the submodularity of influence in social networks. In *Proceedings of the thirty-ninth annual ACM Symposium on Theory of Computing*, pages 128–134. ACM, 2007.
- [21] Mark EJ Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004.
- [22] Filippo Radicchi, José J Ramasco, and Santo Fortunato. Information filtering in complex weighted networks. *Physical Review E*, 83(4):046101, 2011.
- [23] Venu Satuluri, Srinivasan Parthasarathy, and Yiye Ruan. Local graph sparsification for scalable clustering. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, pages 721–732. ACM, 2011.
- [24] M Ángeles Serrano, Marián Boguñá, and Alessandro Vespignani. Extracting the multiscale backbone of complex weighted networks. *Proceedings of the National Academy of Sciences*, 106(16):6483–6488, 2009.
- [25] Daniel A Spielman and Shang-Hua Teng. Spectral sparsification of graphs. *SIAM Journal on Computing*, 40(4):981–1025, 2011.