Sparsification of Social Networks Using Random Walks

Bryan Wilder and Gita Sukthankar
Department of EECS (CS), University of Central Florida
bryan.wilder@knights.ucf.edu, gitars@eecs.ucf.edu

Abstract

There is often a tradeoff between accuracy and scalability when analyzing large network datasets; 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 in order to make the reduced graph more tractable. This paper proposes 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. This objective can be optimized using a highly efficient greedy search strategy. 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 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.

1 Introduction

Analyzing large social networks can reveal many insights including community structure [6, 18, 4, 1], influential actors [13, 3], and the spread of influence [10, 15]. 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 [22, 8, 17]. 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.

The aim of this paper is to develop and evaluate a new sparsification algorithm that is based on preserving the properties of a random walk on the graph. The objective is to specifically target applications that deal with dynamic processes taking place on networks such as influence spread. A random walk serves as a generic model of these behaviors and proves to be computationally amenable to sparsification. The proposed algorithm finds a sparsification that preserves, as closely as possible, the limiting behavior of a random walk on the network, as exhibited in its stationary distribution.

This method promises to fill a gap in the existing literature on sparsification. 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 is tested as a preprocessing step for network analysis on real data. Specifically, the influence maximization problem is used as a benchmark due to the wide variety of algorithms which are available for it. Results across a variety of datasets show that the proposed sparsification algorithm allows virtually the same quality of influence maximization to be performed while using

only a fraction of the time. Accordingly, two cases are shown where sparsification allows influence maximization algorithms to scale to datasets that previous work has considered out of reach.

2 Related Work

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.

2.1 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. [20] 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. [17] is specifically directed at influence maximization. It uses a maximum likelihood method to find a set of edges that generates 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.

2.2 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 [8].

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 [22] 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.

2.3 Backbone detection

An alternative approach to reducing the density of a network is backbone detection. Roughly, these algorithms attempt 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. [21]. 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. [7] introduced a nonparametric method that uses the empirical distribution of edge weights at each node. Another recent algorithm developed by Radicchi et al. [19] 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 paper, they do not attempt to preserve global properties of the network. Additionally, they are only applicable to weighted networks.

3 Algorithm Description

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.

3.1 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

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

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. Then, the additional 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

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

The reduction in the divergence after adding an

edge (p, q) is

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

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:

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

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.

3.2 Proposed algorithm

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

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

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).

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:

$$E_0 = \emptyset$$
$$E_{k+1} = E_k \cup \underset{u \in E/E'}{\arg\max}\ \Delta D_{KL}(E_k, u)\ \ \forall k < k_{\max}$$

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.

It can be shown that the greedy algorithm finds the optimal value of the objective function. Roughly, this is because the stationary distribution is the sum of terms which are local to each node. Therefore, the choice made at each iteration only influences the value of a few neighboring edges; a local choice never causes global changes in the value of edges. The consequence of this property is that choices which are optimal at each step individually remain optimal in hindsight, so a greedy algorithm succeeds in finding the best set of edges.

3.3 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 $\mathcal{O}(m \log m)$.

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. [15] showed that when the objective function is submodular, reevaluating the marginal gains at each step is unnecessary. Submodularity formalizes a property of diminishing returns in an objective function. Given a ground set of items X, a function $f : \mathcal{P}(X) \to \mathbb{R}$ is called submodular if $f(\{x\} \cup A) - f(A) \geq f(\{x\} \cup B) - f(B)$ for all $x \in X/(A \cup B)$ 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. It is easy to see that the objective function is submodular up to a term that is constant with respect to the particular edges that are chosen. If the objective function is submodular, then the marginal gain of an item in one time step is an upper bound on its value in later time steps. Thus, if a given edge is optimal compared to a set of values from earlier iterations, it must remain optimal given the current values. Accordingly, it is only necessary for the evaluation of the *best* edge to be based on the current state; it is not necessary to reevaluate the entire edge set. This observation offers dramatic speedups in practice.

4 Experiments

Sparsification is a preprocessing step for further network analysis tasks. Accordingly, a sparsification algorithm should be evaluated based on how useful its output is for the problem under consideration. This paper focuses on applications to the influence maximization task. 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 a variety of existing methods to be used in the experiments. The next section introduces the influence maximization domain in more detail.

4.1 Influence maximization

The influence maximization problem, first formalized by Kempe, Kleinberger, and Tardos [13] 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) [12], nodes become active at each step once a sum of activations from their neighbors passes a given threshold. In the Independent Cascade Model (ICM) [9], 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.

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:

- **CELF**: The greedy algorithm of Leskovec et al. [15], which incorporates lazy evaluation of marginal gains.
- **SP1M**: An algorithm by Kimura and Saito [14] 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. [3]. PMIA restricts influence spread evaluations to a local area around each node.
- **InfluMax**: A recent algorithm developed by Gomez-Rodriguez and Schölkopf [10]. 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.

4.2 Datasets

There are four principal datasets used in this analysis, spanning a range of sizes and functions:

- **net-HEP**: 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. The data set has been frequently used as a test case for influence maximization [13, 3, 11] and is available from `http://research.microsoft.com/en-us/people/weic/projects.aspx`.
- **email-Enron**: the dataset of email correspondence from the Enron corporation. Nodes correspond to employees, and edges link those who have exchanged at least one email. It is available from the SNAP network repository [16].
- **DBLP**: a citation network from the DBLP computer science bibliography. Similarly available from [16].
- **Epinions**: a trust network from the Epinions

web site. Users can elect to trust each other, and these decisions correspond to edges. Available from [16].

Table 1 provides the size and a few key statistics for each of the datasets.

4.3 Benchmarks

We evaluate several sparsification algorithms based on how well they serve as a preprocessing step for influence maximization. As explained in Section 2, previous sparsification methods are not directly comparable to the algorithm developed here. The closest is that of Mathioudakis et al. [17], 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:

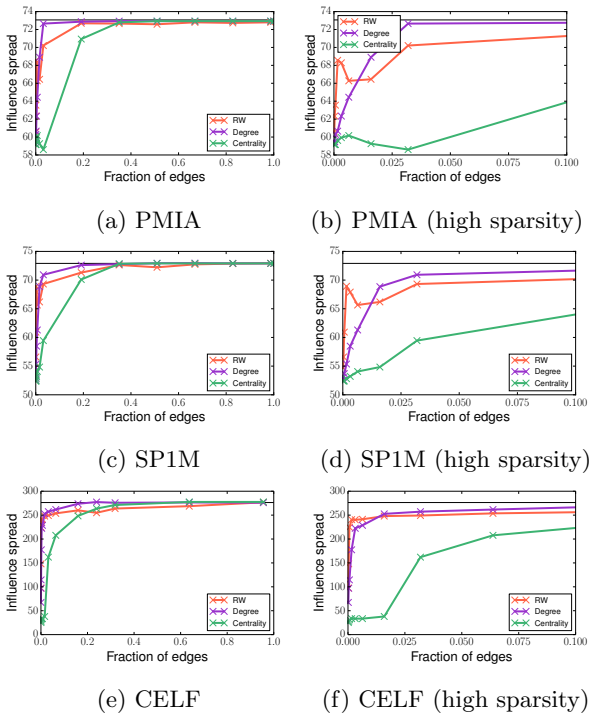
- **RW**: The random-walk based sparsifier proposed here.
- **Degree**: Rank edges according to the sum of the degrees of their endpoints and choose the top k.
- **Centrality**: Rank edges according to the sum of the eigenvector centrality of their endpoints and choose the top k.

5 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.

5.1 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. Different algorithms are applied to different datasets depending on their scalability: CELF, SP1M, and PMIA are applied to net-HEP, SP1M and PMIA are applied to email-Enron, and only PMIA is applied to DBLP and Epinions. InfluMax is not sufficiently scalable to be directly applied to any of the full datasets. Figures 1-4 show the results. 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.



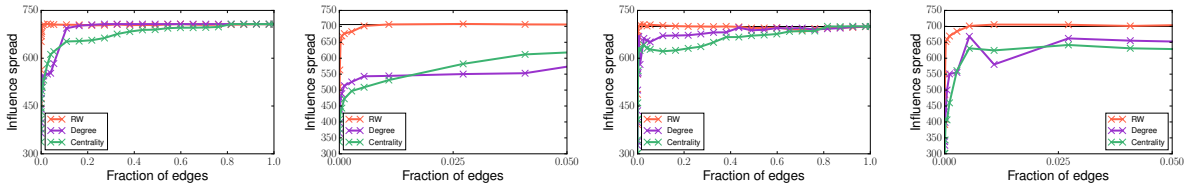
(a) PMIA (b) PMIA (high sparsity)

(c) SP1M (d) SP1M (high sparsity)

(e) CELF (f) CELF (high sparsity)

Figure 1: Influence spread for the net-HEP dataset

An immediate conclusion from these results is that sparsification can be extremely effective for the influence maximization task: only a small fraction of the original edges are needed to obtain the same quality of results on the net-HEP, DBLP, and email-Enron datasets. However, the sparsifiers under consideration do not converge equally quickly to the optimal value. In most cases, the RW algorithms converges

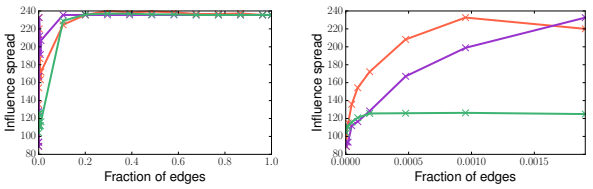


(a) PMIA (b) PMIA (high sparsity) (c) SP1M (d) SP1M (high sparsity)

Figure 2: Influence spread for the email-Enron dataset

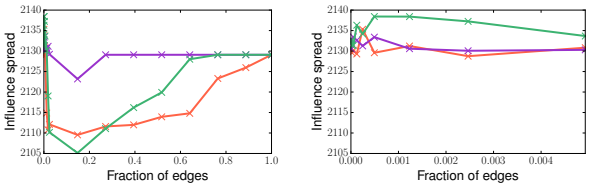
Table 1: Network datasets.

Name	Nodes	Edges	Clustering coefficient	Average degree
net-HEP	15,233	31,398	0.498	4.12
email-Enron	36,692	183,831	0.497	10.02
DBLP	317,080	1,049,866	0.632	6.62
Epinions	75,879	508,837	0.137	10.69



(a) PMIA (b) PMIA (high sparsity)

Figure 3: Influence spread on the DBLP dataset.



(a) PMIA (b) PMIA (high sparsity)

Figure 4: Influence spread on the Epinions dataset.

fastest, with the other two requiring several times more edges to reach the optimal influence spread.

The Epinions dataset appears to present a greater challenge to sparsification. Both RW and Centrality perform more poorly here, with practically the entire dataset needed to reach the optimum. This can potentially be explained by reference to the properties of the underlying network. As shown in Table 1, the Epinions network has by far the lowest clustering coefficient of any of the datasets under consideration. A high clustering coefficient indicates that a node's neighbors are likely to be connected themselves; that is, the local area around each node is likely to be dense. In this situation, which characterizes the other datasets, edges are likely more redundant because there are many roughly equivalent paths for influence to reach the same destination. However, when a network is not well-clustered, edges make more distinct contributions to influence spread in the network.

However, it is important to note that there is often a spike in performance when very few edges are used, in the range of 100 to 1000. For convenience, the right column of each figure magnifies this range of the graph. On the net-HEP dataset, the results are already close to optimal at only 20 edges (out of over 31,000), and reach optimality by 500 edges. On the email-Enron network, only 500 edges (out of over 118,000) are again required for the RW algorithm to reach the optimal value for both the PMIA and SP1M algorithms. The same approximate range (100-1000) edges also produces near-optimal values for the DBLp and Epinions datasets. It is striking that virtually the same number of edges is required as the size of the network increases by more than two orders of magnitude. This property, which is not always shared by the baseline algorithms, is extremely useful for sparsification 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 section.

It is interesting that performance often *decreases* as the number of edges is increased past this high-sparsity regime. In effect, sparsification performs best

when it identifies the edges which are unambiguously most important. Adding additional edges beyond this point just creates clutter which complicates the job of influence maximization algorithms. That is, sparsification is best used to prune away the bulk of edges which have little impact on the overall properties of influence propagation, leaving only a small core of the network for further analysis.

5.2 Speedup

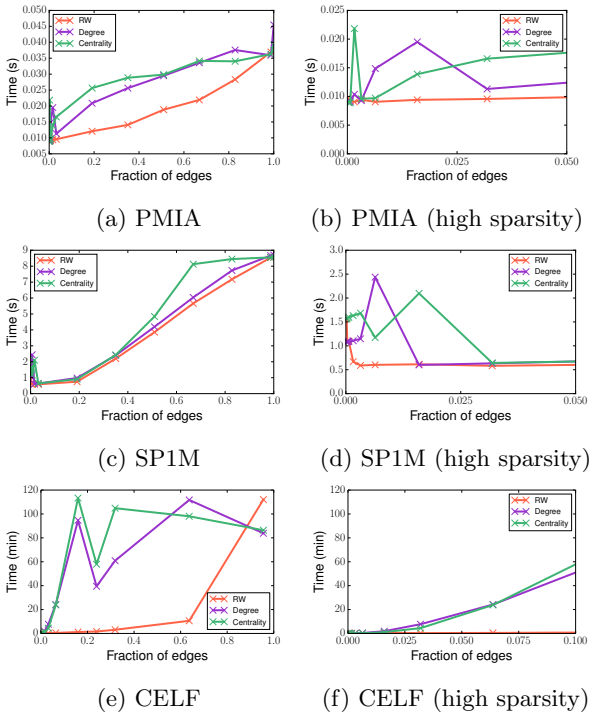
This section characterizes the tradeoff between the fraction of edges selected and the amount of time required to run the influence maximization algorithm. While both PMIA and SP1M are sufficiently scalable that the speedups are not necessary for the datasets which are considered, these results help show the utility of sparsification for addressing datasets that are currently out of reach for these algorithms. Figures 5-8 present the results for different datasets.

In most cases, 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 (in all but one case) 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 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. For instance, on the 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.

5.3 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.

CELF has previously been applied to the net-HEP dataset [3]. However, larger networks have either been dismissed as impractical [3], or else running CELF on them required amounts of time that are usually infeasible (7 days in [11]). Here, CELF is applied to the email-Enron network in a matter of

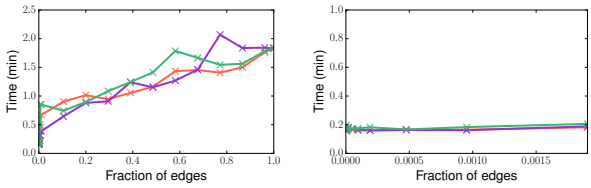


(a) PMIA (b) PMIA (high sparsity)

(c) SP1M (d) SP1M (high sparsity)

(e) CELF (f) CELF (high sparsity)

Figure 5: Runtime for the net-HEP dataset

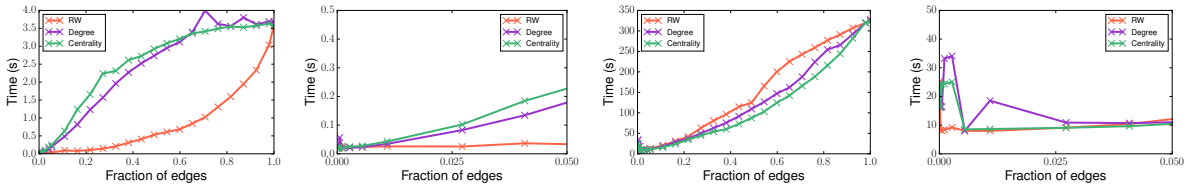


(a) SP1M (b) SP1M (high sparsity)

Figure 7: Runtime on the DBLP dataset.

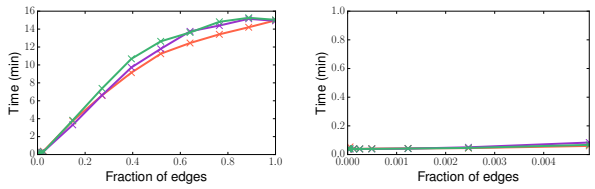
minutes. Figure 9 shows the runtime and influence spread of CELF applied to this dataset. The influence spread shown in Figure 9a increases only slightly as the algorithm runs, suggesting that a near-optimal seed set has been found. As can be seen in 9b, the runtime is uniformly less than five minutes.

InfluMax, while achieving influence spreads greater than that of PMIA or SP1M, has never been applied to a large network. Indeed, Du et al. [5] 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 contains over 15,000 nodes and 31,000 edges, again with a budget of 3 nodes. Figure 10c 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



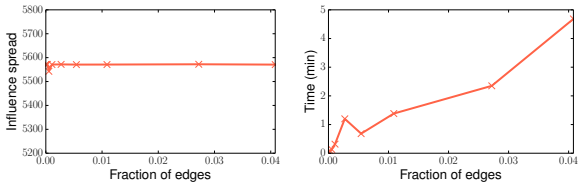
(a) PMIA (b) PMIA (high sparsity) (c) SP1M (d) SP1M (high sparsity)

Figure 6: Runtime for the email-Enron dataset



(a) SP1M (b) SP1M (high sparsity)

Figure 8: Runtime on the Epinions dataset.



(a) Influence spread (b) Runtime

Figure 9: CELF applied to the email-Enron dataset

spread of a seed set in the continuous time model used by InfluMax is intractable for large networks. However, Figure 10a 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 since additional edges increase the number of nodes that are reachable. Figure 10b 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.

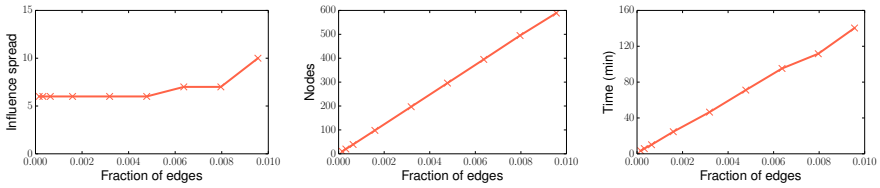
6 Conclusions

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 paper proposes 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 four real-world network datasets. 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 other network analysis tasks.

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 typi-



(a) Influence spread (b) Number of nodes (c) Runtime

Figure 10: InfluMax applied to the net-HEP dataset

cally 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, examining additional tasks could help us better characterize the applicability of sparsification for preprocessing networks.

References

[1] E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing. Mixed membership stochastic blockmodels. In *Advances in Neural Information Processing Systems*, pages 33–40, 2009.

[2] A. Benczur and D. R. Karger. Randomized approximation schemes for cuts and flows in capacitated graphs. *arXiv preprint cs/0207078*, 2002.

[3] W. Chen, C. Wang, and Y. Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1029–1038. ACM, 2010.

[4] A. Clauset, M. E. Newman, and C. Moore. Finding community structure in very large networks. *Physical Review E*, 70(6):066111, 2004.

[5] N. Du, L. Song, M. Gomez-Rodriguez, and H. Zha. Scalable influence estimation in continuous-time diffusion networks. In *Advances in Neural Information Processing Systems*, pages 3147–3155, 2013.

[6] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010.

[7] N. J. Foti, J. M. Hughes, and D. N. Rockmore. Nonparametric sparsification of complex multiscale networks. *PloS One*, 6(2):e16431, 2011.

[8] W. S. Fung, R. Hariharan, N. J. Harvey, and D. Panigrahi. A general framework for graph sparsification. In *Proceedings of the ACM Symposium on Theory of Computing*, pages 71–80. ACM, 2011.

[9] J. Goldenberg, B. Libai, and E. Muller. Talk of the network: A complex systems look at the underlying process of word-of-mouth. *Marketing Letters*, 12(3):211–223, 2001.

[10] M. Gomez-Rodriguez and B. Schölkopf. Influence maximization in continuous time diffusion networks. In *Proceedings of the International Conference on Machine Learning*, 2012.

[11] A. Goyal, W. Lu, and L. V. Lakshmanan. Simpath: An efficient algorithm for influence maximization under the linear threshold model. In *IEEE International Conference on Data Mining*, pages 211–220, 2011.

[12] M. Granovetter. Threshold models of collective behavior. *American Journal of Sociology*, pages 1420–1443, 1978.

[13] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 137–146. ACM, 2003.

[14] M. Kimura and K. Saito. Tractable models for information diffusion in social networks. In *Knowledge Discovery in Databases: PKDD 2006*, pages 259–271. Springer, 2006.

[15] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-effective outbreak detection in networks. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 420–429. ACM, 2007.

[16] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.

[17] M. Mathioudakis, F. Bonchi, C. Castillo, A. Gionis, and A. Ukkonen. Sparsification of influence networks. In *Proceedings of the ACM SIGKDD International*

Conference on Knowledge Discovery and Data Mining, pages 529–537. ACM, 2011.

[18] M. E. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(2):026113, 2004.

[19] F. Radicchi, J. J. Ramasco, and S. Fortunato. Information filtering in complex weighted networks. *Physical Review E*, 83(4):046101, 2011.

[20] V. Satuluri, S. Parthasarathy, and Y. Ruan. Local graph sparsification for scalable clustering. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 721–732, 2011.

[21] M. Á. Serrano, M. Boguñá, and A. Vespignani. Extracting the multiscale backbone of complex weighted networks. *Proceedings of the National Academy of Sciences*, 106(16):6483–6488, 2009.

[22] D. A. Spielman and S.-H. Teng. Spectral sparsification of graphs. *SIAM Journal on Computing*, 40(4):981–1025, 2011.