Fast Influence-based Coarsening for Large Networks

Manish Purohit[†]
manishp@cs.umd.edu

B. Aditya Prakash[*]
badityap@cs.vt.edu

Chanhyun Kang[†]
chanhyun@cs.umd.edu

Yao Zhang[*]
yaozhang@cs.vt.edu

V. S. Subrahmanian[†]
vs@cs.umd.edu

[*]Computer Science Department, Virginia Tech., USA
[†]Department of Computer Science, University of Maryland - College Park, USA

ABSTRACT

Given a social network, can we quickly 'zoom-out' of the graph? Is there a smaller equivalent representation of the graph that preserves its propagation characteristics? Can we group nodes together based on their influence properties? These are important problems with applications to influence analysis, epidemiology and viral marketing applications.

In this paper, we first formulate a novel Graph Coarsening Problem to find a succinct representation of any graph while preserving key characteristics for diffusion processes on that graph. We then provide a fast and effective near-linear-time (in nodes and edges) algorithm COARSENET for the same. Using extensive experiments on multiple real datasets, we demonstrate the quality and scalability of COARSENET, enabling us to reduce the graph by 90% in some cases without much loss of information. Finally we also show how our method can help in diverse applications like influence maximization and detecting patterns of propagation at the level of automatically created groups on real cascade data.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data Mining*

Keywords

Graph Mining; Propagation; Diffusion; Coarsening

1. INTRODUCTION

The unprecedented popularity of online social networking websites, such as Facebook, Google+, Flickr, and YouTube, has made it possible to analyze real social networks. Word of mouth marketing and viral marketing strategies have evolved to take advantage of this network structure by utilizing network effects. Similarly, understanding large-scale epidemiological datasets is important for designing effective propagation models and containment policies for public health. The sheer size of today's large social networks makes it challenging to perform sophisticated network analysis.

Given a propagation graph, possibly learnt from cascade analysis, is it possible to get a smaller nearly diffusion-equivalent representation for it? Getting a smaller equivalent graph will help multiple algorithmic and data mining tasks like influence maximization, immunization, understanding cascade data and data compression. In this paper, we study a novel graph coarsening problem with the aim of approximating a large social network by a much smaller graph that approximately preserves the network structure. Our primary goal is to find a compact representation of a large graph such that diffusion and propagation processes on the large graph can be studied by analyzing the smaller representation. Intuitively, most of the edges in a real network are relatively unimportant; hence we propose characterizing and "contracting" precisely such edges in a graph to obtain a coarse representation.

The main contributions of this paper are:

(a) *Problem Formulation:* We carefully formulate a novel *Graph Coarsening Problem* (GCP) to find a succinct representation of a given social network so that the diffusion characteristics of the network are mostly preserved.
(b) *Efficient Algorithms:* We develop COARSENET, an efficient (near-linear time) and effective algorithm for GCP, using careful approximations. We show that due to our novel scoring technique, the coarsened graph retains most of the diffusive properties of the original network.
(c) *Extensive Experiments:* We show that COARSENET is able to coarsen graphs *up to* 90% without much loss of key information. We also demonstrate the usefulness of our approach via a number of interesting applications. A major application we consider in this work is that of influence maximization in the Independent Cascade model. We propose a framework CSPIN that involves coarsening the graph and then solving influence maximization on the smaller graph to obtain high quality solutions. As the coarsened graph is much smaller than the original graph, the influence maximization algorithm runs orders of magnitude faster on the coarsened graph. Further using real cascade data from Flixster, we show how GCP can potentially help in understanding propagation data and constructing non-network surrogates for finding nodes with similar influence.

KDD'14, August 24–27, 2014, New York, NY, USA.

http://dx.doi.org/10.1145/2623330.2623701.

The rest of the paper is organized as follows: Section 2 gives related work and Section 3 briefly gives the notation and explains some technical preliminaries. Section 4 provides a formal definition of the Graph Coarsening Problem that we introduce, while Section 5 presents our approach and solution. In Section 6, we show how our coarsening framework can be applied to solve influence maximization on large networks. Finally, Section 7 gives experimental results while we conclude in Section 8.

2. RELATED WORK

The idea of coarsening a network for some task is not new, and has been used extensively in the popular community detection techniques (METIS [21] and GRACLUS [9]): nevertheless, they use different metrics for coarsening like cut-based, flow-based or heavy-edge matching-based conditions. In contrast we study diffusion-based metrics, and do not aim to find communities.

The related problem of graph sparsification has also been well studied in the theory community under the notion of "spanners" [10]. A spanner is a sparse subgraph that maintains the pairwise distances between all nodes within a multiplicative or additive factor. Fung et al. [12] study the cut-sparsifier problem which asks for a sparse weighted subgraph such that the weight of all cuts is maintained within a small multiplicative factor. Graph sparsification for influence analysis has emerged as a new tool for analyzing large networks. Mathioudakis et al. [28] propose an algorithm to find the sparse backbone of an influence network. The major difference is that graph sparsification *removes* edges (so the nodes stay the same), while we *coarsen* and *contract* node-pairs to reduce the graph. Another line of very recent work [30] tries to learn influence models at community-scale, using groups supplied by graph-partitioning algorithms like METIS. Our work is related in the sense that we also aim to 'group' nodes, but not based on link-based communities, instead automatically based on nodes' diffusion characteristics. In that sense we believe our work provides a complementary viewpoint: learn models directly at the node level, and then try to group them appropriately automatically.

The rest of the related work can be categorized into Epidemic Thresholds, Influence Maximization, Other Optimization problems, and General Information Diffusion.

Epidemic Thresholds. The classical texts on epidemic models and analysis are May and Anderson [1] and Hethcote [20]. Much research in virus propagation focuses on the so-called epidemic threshold, i.e. determining the conditions under which an epidemic will not break out. Widely-studied epidemiological models include *homogeneous models* [2, 29, 1] which assume that every individual has equal contact with others in the population. While earlier works [23, 31] focus on some specific types of graph structure (e.g., random graphs, power-law graphs, etc), Chakrabarti et al. [6] and Ganesh et al. [13] found that, for the flu-like SIS model, the epidemic threshold for any arbitrary graph depends on the leading eigenvalue of the adjacency matrix of the graph. Prakash et al. [32] further extended the result to a broad class of epidemic models.

Influence Maximization: The influence maximization problem was introduced by Domingos and Richardson [34]. Kempe et al. [22] formulated it as a combinatorial optimization problem under the Independent Cascade Model, proved it is NP-Hard and gave a simple $1 - 1/e$ approximation based on the submodularity of expected spread of a set of starting seeds. Numerous follow-up papers have looked at speeding-up the algorithm (e.g., [27, 16, 8, 24, 7]).

Other Optimization Problems. Another related problem is immunization, i.e, the problem of finding the best vertices for removal to stop an epidemic, with effective immunization strategies for static and dynamic graphs [19, 38, 4]. Other such problems where we wish to select a subset of '*important*' vertices on graphs, include 'outbreak detection' [27] and finding most-likely starting points ('culprits') of epidemics [26, 33].

General Information Diffusion. There is a lot of research interest in studying dynamic processes on large graphs, (a) blogs and propagations [18, 25, 22], (b) information cascades [3, 14, 17] and (c) marketing and product penetration [35]. These dynamic processes are all closely related to virus propagation. General algorithms for information diffusion based optimization include [36].

3. PRELIMINARIES

Table 1 gives some of the notation.

Table 1: Symbols

Symbol	Definition and Description
$\mathbf{A}, \mathbf{B}, \ldots$	matrices (bold upper case)
$\vec{a}, \vec{b}, \ldots$	column vectors
a_j or $a(j)$	j^{th} element of vector a
n	number of vertices in the graphs
m	number of edges in the graphs
α	the reduction factor
$\lambda_{\mathbf{G}}$	first eigenvalue (in absolute value) of adjacency matrix of graph $\mathbf{G}$
$\vec{u}_{\mathbf{G}}, \vec{v}_{\mathbf{G}}$	Right and left first eigenvectors (for $\lambda_{\mathbf{G}}$) of adjacency matrix $\mathbf{G}$
IC Model	The Independent Cascade Model
GCP	Graph Coarsening Problem (see Definition 4.3)
coarseNet	Our algorithm for GCP

IC Model. A social network is a directed, weighted graph $G = (V, E, w)$. Usually each vertex $v \in V$ represents an individual of the network and edges represent influence relationships between these individuals. The Independent Cascade (IC) model is a popular diffusion model used to model the way influence propagates along the edges of a social network. In this setting, a vertex $v \in V$ is called *active* if it has been influenced and *inactive* otherwise. Once an inactive vertex becomes active, it always stays active, i.e. we focus only on *progressive* models. Given a seed set $S \subset V$ of initially active vertices, the Independent Cascade model proceeds in discrete time steps as follows. At time step t, let S_t denote the set of vertices activated at time t. Every vertex $u \in S_t$ is given a *single* chance to activate each currently inactive neighbor v with probability of success $w(u, v)$ independently of all other interactions. If u succeeds, then v becomes active at time $t + 1$. This diffusion process continues until no more activations are possible. The *influence spread* of seed set S, denoted by $\sigma(S)$, is the expected number of activated vertices at the end of the process.

4. PROBLEM FORMULATION

Motivated by the fact that in any real network, *most* edges and vertices are not important (due to the heavily skewed degree distributions), we propose a graph coarsening problem which involves pruning away precisely such edges (and vertices). We aim to coarsen the graph to obtain a much smaller representation which retains the diffusive properties. We *coarsen* a graph by successively merging adjacent node pairs. We attempt to quickly find "good" edges which have little effect on the network's diffusive properties. At first glance, this seems impossible as the diffusive properties of a graph are highly dependent on the connectivity of the vertices and edge weights. Further, determining which node pairs to merge and analyzing the effect of merging two nodes on diffusion are non-trivial. Informally, we study the following problem in this paper:

DEFINITION 4.1 (INFORMAL PROBLEM).
INPUT: *Weighted graph $G = (V, E, w)$ and a target fraction $0 < \alpha < 1$*
GOAL: *Coarsen G by repeatedly merging adjacent node pairs to obtain a weighted graph $H = (V', E', w')$ such that*

- $|V'| = (1 - \alpha)|V|$
- *Graph H approximates graph G with respect to its diffusive properties*

Role of Eigenvalues. In order to address the informal problem described above, we need a tractable way to characterize the diffusive properties of a network. Recent work [32] shows that for almost any propagation model (including the IC model), important diffusion characteristics (in particular the so-called epidemic threshold) of a graph (after removing self loops) are captured by the spectrum of the graph, specifically, by the first eigenvalue of the adjacency matrix. Thus it is natural to believe that if the first eigenvalue of the coarsened graph H (its adjacency matrix) is close to that of the original graph G, then H indeed approximates G well. Although the work of [32] deals with undirected graphs, their findings are also applicable to strongly connected directed graphs.

Merging node pairs. To explicitly formulate the problem in Definition 4.1, we also need to define what happens when a node pair is merged (i.e. an edge is contracted) in a weighted graph. More precisely, after merging neighboring vertices a and b to form a new node c, we need to determine the *new* edge weights of all incoming and outgoing edges of c. In order to maintain the diffusive properties of the network, we need to reweight the new edges appropriately.

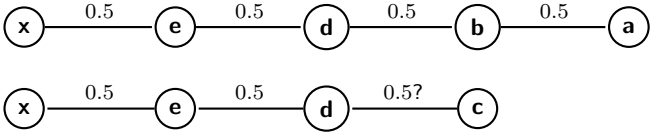


Figure 1: Why reweight?

To see why this is crucial, consider Figure 1. Assume that the IC model is being run. Suppose we need to pick the two best seeds (i.e. two nodes with the maximum influence spread as defined in the previous Section) from the top 5-vertex chain. Further assume that the graph is undirected and each edge has the same weight $\beta = 0.5$. Clearly, vertices b and e are the best. If we merge vertices $\{a, b\}$, we get the bottom 4-vertex chain. To still match the original solution, we correspondingly want $\{c, e\}$ to be the best seed-set in the new chain—but if edge $\{d, c\}$ remains the same weight, any of the pair of vertices $\{e, c\}$ or $\{x, d\}$ are the best seed sets in the 4-vertex chain. This motivates the need to reweight suitably so that new coarsened graph still retains the original characteristics.

The main insight is that if we select c as a seed, we are in-effect intending to choose only *one* of vertices a and b to be seeded (influenced); which suggests that the likelihood of d being influenced from c is *either* 0.5 or 0.25 (corresponding to when a or b is chosen respectively). Hence the weight of edge (c, d) should be modified to reflect this fact.

We propose the following solution: Suppose $e = (a, b)$ is contracted and a and b are merged together to form "supervertex" c (say). We reweight the edges adjacent to a and b while coarsening so that the edges now represent the average of the transmission probabilities via a or b. So in our example of Figure 1, edge $\{c, d\}$ would have weight 0.375 (average of 0.5 and 0.25). Further, we can verify that in this case $\{e, c\}$ will be the best seed-set, as desired.

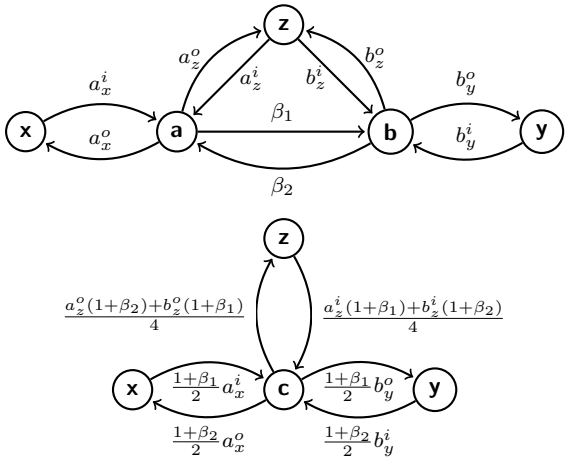


Figure 2: Reweighting of edges after merging node pairs

Extending the same principle, Figure 2 shows the general situation for any candidate node pair (a, b) and how a merge and re-weight (= contract) operation will look like. More formally, our contract operation is as follows:

DEFINITION 4.2 (MERGING NODE PAIRS). *Let $Nb^i(v)$ (respectively $Nb^o(v)$) denote the set of in-neighbors (resp. out-neighbors) of a vertex v. Let $v_u^i = w(u, v)$ and $v_u^o = w(v, u)$ denote the weight of the corresponding edges. If the node pair(a, b) is now contracted to a new vertex c, and $w(a, b) = \beta_1$ and $w(b, a) = \beta_2$, then the new edges are weighted as -*

$$c_t^i = \begin{cases} \frac{(1+\beta_1)a_t^i}{2} & \forall t \in Nb^i(a) \backslash Nb^i(b) \\ \frac{(1+\beta_2)b_t^i}{2} & \forall t \in Nb^i(b) \backslash Nb^i(a) \\ \frac{(1+\beta_1)(a_t^i)+(1+\beta_2)(b_t^i)}{4} & \forall t \in Nb^i(a) \cap Nb^i(b) \end{cases}$$

$$c_t^o = \begin{cases} \frac{(1+\beta_2)a_t^o}{2} & \forall t \in Nb^o(a) \backslash Nb^o(b) \\ \frac{(1+\beta_1)b_t^o}{2} & \forall t \in Nb^o(b) \backslash Nb^o(a) \\ \frac{(1+\beta_2)(a_t^o)+(1+\beta_1)(b_t^o)}{4} & \forall t \in Nb^o(a) \cap Nb^o(b) \end{cases}$$

Graph Coarsening Problem. We are now ready to state our problem formally. Motivated by the connections between the diffusive and spectral properties of a graph, we define the following Graph Coarsening Problem to find the set of node pairs which when merged (according to Definition 4.2) lead to the least change in the first eigenvalue. Further, since a vertex cannot influence itself, we assume without loss of generality that the graph G has no self loops.

DEFINITION 4.3 (GRAPH COARSENING PROBLEM). INPUT: *Directed, strongly connected, weighted graph* $G = (V, E, w)$ *without self loops and a target fraction* $0 < \alpha < 1$ OUTPUT: $E^* = \arg\min_{E' \subset E, |E'| = \alpha|V|} |\lambda_G - \lambda_{G'}|$, *where* G' *is obtained from* G *by merging all node pairs in* E'.

A related problem is Edge Immunization [37] that asks for a set of edges whose *removal* leads to the *greatest drop* in the first eigenvalue. In contrast, GCP seeks to find a set of edges whose *contraction* (Definition 4.2) leads to the *least change* in the first eigenvalue. The Edge Immunization problem is known to be NP-hard [37].

5. OUR SOLUTION

As obvious algorithms to GCP are clearly exponential, we propose a greedy heuristic that repeatedly merges a node pair which minimizes the change in the first eigenvalue. Let $G_{-(a,b)}$ denote the graph G after merging nodes a and b (and incorporating the re-weighting strategy), and λ_G denote the first eigenvalue of the adjacency matrix of G. We define the score of a node pair(a, b) as follows -

DEFINITION 5.1 (SCORE). *Given a weighted graph* $G = (V, E, w)$ *and an adjacent node pair*(a, b), *score*(a, b) *is defined by:*

$$score(a, b) = |\lambda_{G_{-(a,b)}} - \lambda_G| = \Delta\lambda_{(a,b)}$$

Intuitively, if $score(a, b) \approx 0$, it implies that edges (a, b) and (b, a) do not play a significant role in the diffusion through the graph and can thus be contracted. Figure 3 shows an example of our approach.

Naïve Algorithm: The above intuition suggests the following naïve algorithm for selecting node pairs for merging. At each stage, calculate the change in the eigenvalue due to merging each adjacent node pair, choose the node pair leading to the least change, merge the chosen nodes, and repeat until the graph is small enough. An implementation for this, even using the Lanczos algorithm for eigenvalue computation for sparse graphs, will be too expensive, taking $O(m^2)$ time. Can we compute (maybe approximately) the scores of each node pair faster?

Main Idea: We use a matrix perturbation argument to derive an expression for the change in eigenvalue due to merging two adjacent nodes. Using further information about the specific perturbations occurring due to merging two adjacent nodes, we show that the change in the eigenvalue can be approximated well in constant time. Thus, we obtain a linear ($O(m)$) time scheme to estimate the score of every pair of adjacent nodes.

5.1 Score Estimation

Let a and b denote the two neighboring vertices that we are trying to score. We assume that the first eigenvalue of the graph λ_G and the corresponding right and left eigenvectors $\vec{u}, \vec{v}$ are precomputed. Further since the graph G is strongly connected, by the Perron-Frobenius theorem, the first eigenvalue λ_G and the eigenvectors $\vec{u}$ and $\vec{v}$ are all real and have positive components. When it is clear from the context, we drop subscripts G and (a, b). In the proofs that follow $\lambda = \lambda_G$ and $\Delta\lambda = \Delta\lambda_{(a,b)}$ as there is no ambiguity. Let $\mathbf{A}$ denote the adjacency matrix of the graph. Further as $\vec{u}$ denotes the eigenvector of $\mathbf{A}$, let $u_a = u(a)$ denote the component of $\vec{u}$ corresponding to vertex a. Merging nodes changes the dimensions of the adjacency matrix $\mathbf{A}$ which we handle by viewing merging nodes a, b as adding b's neighbors to a and isolating node b.

Approximation 5.1 provides an equation for the change in the eigenvalue by a matrix perturbation argument. Proposition 5.2 and Proposition 5.3 show how our reweighting strategy helps us to approximate the $score(a, b)$ in constant time.

APPROXIMATION 5.1. *The change in eigenvalue* $\Delta\lambda$ *can be approximated by* $\Delta\lambda = \frac{\vec{v}^T\Delta\mathbf{A}\vec{u} + \vec{v}^T\Delta\mathbf{A}\Delta\vec{u}}{(\vec{v}^T\vec{u} + \vec{v}^T\Delta\vec{u})}$ *where* $\Delta\mathbf{A}$ *denotes an infinitesimally small change in the adjacency matrix* $\mathbf{A}$ *and* $\Delta\vec{u}$ *denotes the corresponding change in the eigenvector* $\vec{u}$.

JUSTIFICATION. By the definition of an eigenvalue and eigenvector of a matrix, we have

$$\mathbf{A}\vec{u} = \lambda\vec{u} \tag{1}$$

$$\vec{v}^T\mathbf{A} = \vec{v}^T\lambda \tag{2}$$

Perturbing all values of (1) infinitesimally, we get

$$(\mathbf{A} + \Delta\mathbf{A})(\vec{u} + \Delta\vec{u}) \approx (\lambda + \Delta\lambda)(\vec{u} + \Delta\vec{u})$$
$$\mathbf{A}\Delta\vec{u} + \Delta\mathbf{A}\vec{u} + \Delta\mathbf{A}\Delta\vec{u} \approx \lambda\Delta\vec{u} + \Delta\lambda\vec{u} + \Delta\lambda\Delta\vec{u}$$

Premultiplying by $\vec{v}^T$ and using (1) and (2),

$$\Delta\lambda(\vec{v}^T\vec{u} + \vec{v}^T\Delta\vec{u}) \approx \vec{v}^T\Delta\mathbf{A}\vec{u} + \vec{v}^T\Delta\mathbf{A}\Delta\vec{u}$$
$$\Delta\lambda \approx \frac{\vec{v}^T\Delta\mathbf{A}\vec{u} + \vec{v}^T\Delta\mathbf{A}\Delta\vec{u}}{(\vec{v}^T\vec{u} + \vec{v}^T\Delta\vec{u})} \tag{3}$$

□

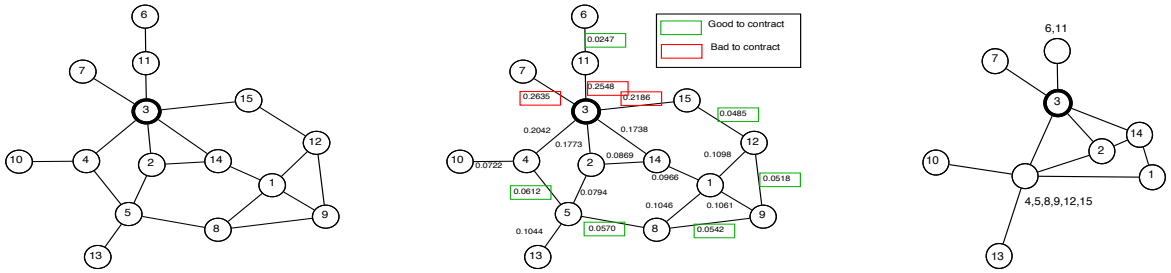
Using expression (3) along with prior knowledge about the perturbations to the adjacency matrix $\mathbf{A}$ and the eigenvector $\vec{u}$, we obtain an expression for computing the score of the node pair.

PROPOSITION 5.2 (SCORE ESTIMATE). *Under Approximation 5.1, the score of a node pair* $score(a, b)$ *can be approximated as*

$$\Delta\lambda_{(a,b)} = \frac{-\lambda(u_a v_a + u_b v_b) + v_a\vec{u}^T\vec{c^o} + \beta_2 u_a v_b + \beta_1 u_b v_a}{\vec{v}^T\vec{u} - (u_a v_a + u_b v_b)}$$

(ignoring second order terms).

PROOF. Approximation 5.1 provided an expression for $\Delta\lambda$ in terms of the change in the adjacency matrix and the eigenvector. Now $\Delta\mathbf{A}$, i.e., change in the adjacency matrix



(a) Original Network (b) Assigning scores to each edge (c) Coarsened Network

Figure 3: Our approach on an example network. All edges have weight 0.5 in the original graph. We do not show the new edge weights in the coarsened graph for clarity.

can be considered as occurring in three stages namely (i) Deletion of a, (ii) Deletion of b, (iii) Insertion of c. Assume that c is inserted in place of a. Thus we obtain,

$$\Delta\mathbf{A} = -\left(\vec{a^i}\vec{e_a}^T + \vec{e_a}\vec{a^o}^T\right) - \left(\vec{b^i}\vec{e_b}^T + \vec{e_b}\vec{b^o}^T\right) + \left(\vec{c^i}\vec{e_a}^T + \vec{e_a}\vec{c^o}^T\right) \quad (4)$$

where $\vec{e_v}$ denotes a vector with a 1 in the v^{th} row and 0 elsewhere. Further, as we modify only two rows and columns of the matrix, this change $\Delta\mathbf{A}$ is very small.

Also, deletion of vertices a and b cause a^{th} and b^{th} components of $\vec{u}$ and $\vec{v}$ to be zero. $\Delta\vec{u}$, i.e, change in the eigenvector $\vec{u}$ can thus be considered as setting u_a and u_b to zero, followed by small changes to other components and to u_a due to addition of c. Thus we obtain,

$$\Delta\vec{u} = -u_a\vec{e_a} - u_b\vec{e_b} + \vec{\delta} \quad (5)$$

Although $\Delta\vec{u}$ cannot be considered as small, we assume that the changes $\vec{\delta}$ after setting u_a and u_b components to zero are very small.

Substituting for $\Delta\mathbf{A}$, we get

$$\vec{v}^T\Delta\mathbf{A}\vec{u} = \vec{v}^T(-(\vec{a^i}\vec{e_a}^T + \vec{e_a}\vec{a^o}^T) - (\vec{b^i}\vec{e_b}^T + \vec{e_b}\vec{b^o}^T + (\vec{c^i}\vec{e_a}^T + \vec{e_a}\vec{c^o}^T))\vec{u}$$

Since $\vec{v}^T\vec{e_a} = v_a$ and similarly,

$$\vec{v}^T\Delta\mathbf{A}\vec{u} = -u_a\vec{v}^T\vec{a^i} - v_a\vec{a^o}^T\vec{u} - u_b\vec{v}^T\vec{b^i} - v_b\vec{b^o}^T\vec{u} + u_a\vec{v}^T\vec{c^i} + v_a\vec{c^o}^T\vec{u}$$

But $\vec{v}^T\vec{a^i} = \lambda v_a$ and $\vec{a^o}^T\vec{u} = \lambda u_a$ and similarly,

$$\vec{v}^T\Delta\mathbf{A}\vec{u} = -2\lambda\left(u_av_a + u_bv_b\right) + u_a\vec{v}^T\vec{c^i} + v_a\vec{c^o}^T\vec{u} \quad (6)$$

Now using (4) and (5) consider,

$$\vec{v}^T\Delta\mathbf{A}\Delta\vec{u} = \vec{v}^T\Delta\mathbf{A}(-u_a\vec{e_a} - u_b\vec{e_b} + \vec{\delta})$$

Since $\Delta\mathbf{A}$ and $\vec{\delta}$ are both very small, we ignore the second order term $\vec{v}^T\Delta\mathbf{A}\vec{\delta}$.

$$\begin{aligned}\Rightarrow \vec{v}^T\Delta\mathbf{A}\Delta\vec{u} &= \vec{v}^T\Delta\mathbf{A}(-u_a\vec{e_a} - u_b\vec{e_b}) \\ &= \vec{v}^T(-(\vec{a^i}\vec{e_a}^T + \vec{e_a}\vec{a^o}^T) - (\vec{b^i}\vec{e_b}^T + \vec{e_b}\vec{b^o}^T \\ &\quad + (\vec{c^i}\vec{e_a}^T + \vec{e_a}\vec{c^o}^T))(-u_a\vec{e_a} - u_b\vec{e_b})\end{aligned}$$

Since self loops do not affect diffusion in any way, we can assume without loss of generality that G has no self loops.

Further, simplifying using definitions of eigenvalue we get,

$$\vec{v}^T\Delta\mathbf{A}\Delta\vec{u} = \lambda(u_av_a + u_bv_b) + \beta_2u_av_b + \beta_1u_bv_a - u_a\vec{v}^T\vec{c^i} \quad (7)$$

Ignoring small terms, we also have,

$$\vec{v}^T\Delta\vec{u} = \vec{v}^T(-u_a\vec{e_a} - u_b\vec{e_b} + \vec{\delta}) = -(u_av_a + u_bv_b) \quad (8)$$

Substituting (6),(7) and (8) in Approximation 5.1, we get

$$\Delta\lambda = \frac{-\lambda\left(u_av_a + u_bv_b\right) + v_a\vec{u}^T\vec{c^o} + \beta_2u_av_b + \beta_1u_bv_a}{\vec{v}^T\vec{u} - (u_av_a + u_bv_b)}$$

□

Note that every term in this expression is a simple product of scalars, except for the $\vec{u}^T\vec{c^o}$ term. We now show that even $\vec{u}^T\vec{c^o}$ can in fact be expressed in terms of scalars and can thus be computed in constant time.

PROPOSITION 5.3. *Using the re-weighting scheme as defined in Definition 4.2, if c denotes the new vertex created by merging nodes $\{a, b\}$ and $\vec{c^o}$ denotes the out-adjacency vector of c, $\vec{u}^T\vec{c^o} = \frac{(1+\beta_2)}{2}(\lambda u_a - \beta_1u_b) + \frac{(1+\beta_1)}{2}(\lambda u_b - \beta_2u_a)$ where β_1 is the weight of edge (a, b) and β_2 is the weight of the edge (b, a).*

PROOF. Let $X = Nb^o(a)\backslash Nb^o(b), Y = Nb^o(b)\backslash Nb^o(a), Z = Nb^o(a) \cap Nb^o(b)$. Since, c is adjacent only to neighbors of a and b, we have

$$\vec{u}^T\vec{c^o} = \sum_{t\in X} u_tc_t^o + \sum_{t\in Y} u_tc_t^o + \sum_{t\in Z} u_tc_t^o + u_cW$$

where W is the weight of a self loop added at c. Note that a self loop does not affect diffusion in any way (as a node can not influence itself). We use a self loop only in the analysis so as to compute the scores efficiently.

As per our reweighting scheme (See Definition 4.2)

$$\vec{u}^T\vec{c^o} = \sum_{t\in X}\frac{(1+\beta_2)}{2}u_ta_t^o + \sum_{t\in Y}\frac{(1+\beta_1)}{2}u_tb_t^o + \sum_{t\in Z}(\frac{(1+\beta_2)}{4}a_t^o + \frac{(1+\beta_1)}{4}b_t^o)u_t + u_cW \quad (9)$$

But, by definition of eigenvalues, we know that

$$\lambda u_a = \sum_{t\in V} u_t a_t^o = \sum_{t\in X} u_t a_t^o + \sum_{t\in Z} u_t a_t^o + u_b\beta_1$$

$$\sum_{t\in X} u_t a_t^o = \lambda u_a - \sum_{t\in Z} u_t a_t^o - \beta_1 u_b$$

$$= \lambda u_a - a^o(Z) - \beta_1 u_b \quad (10)$$

where $a^o(Z) = \sum_{t\in Z} u_t a_t^o$

Similarly, we get

$$\sum_{t\in Y} u_t b_t^o = \lambda u_b - b^o(Z) - \beta_2 u_a \quad (11)$$

Substituting Equations (10), (11), in (9),

$$\vec{u}^T\vec{c^o} = \frac{(1+\beta_2)}{2}(\lambda u_a - a^o(Z) - \beta_1 u_b) + \frac{(1+\beta_1)}{2}(\lambda u_b - b^o(Z) - \beta_2 u_a) + \frac{(1+\beta_2)}{4}a^o(Z) + \frac{(1+\beta_1)}{4}b^o(Z) + u_c W$$

We now choose $W = (-\frac{(1+\beta_2)}{4}a^o(Z) - \frac{(1+\beta_1)}{4}b^o(Z))/u_c$, so that we get

$$\vec{u}^T\vec{c^o} = \frac{(1+\beta_2)}{2}(\lambda u_a - \beta_1 u_b) + \frac{(1+\beta_1)}{2}(\lambda u_b - \beta_2 u_a)$$

□

COROLLARY 5.1. *Given the first eigenvalue λ and corresponding eigenvectors $\vec{u}, \vec{v}$, the score of a node pair score(a, b) can be approximated in constant time.*

PROOF. Substituting for $\vec{u}^t\vec{c^o}$ in Proposition 5.2 using Proposition 5.3, we obtain an expression for $score(a, b)$ that is composed entirely of scalar terms. Thus we can estimate the edge score in constant time. □

5.2 Complete Algorithm

Using the approximation described in the previous section, we assign a score to every pair of adjacent nodes of the graph. We then sort these node pairs in ascending order of the absolute value of their scores. Intuitively, we would like to merge a node pair if it has minimal score. Given an upper bound of α, the graph is then coarsened by contracting αn node pairs one by one in this order ignoring any pairs that have already been merged. We give the pseudo-code of our algorithm COARSENET in Algorithm 1.

LEMMA 5.1 (RUNNING TIME). *The worst case time complexity of our algorithm is $O(m\ln(m) + \alpha n n_\theta)$ where n_θ denotes the maximum degree of any vertex at any time in the coarsening process.*

PROOF. Computing the first eigenvalue and eigenvector of the adjacency matrix of the graph takes $O(m)$ time (for example, using Lanczos iteration assuming that the spectral gap is large). As shown in Section 5.1, each node pair can be assigned a score in constant time. In order to score all m adjacent pairs of nodes of the graph, we require linear i.e. $(O(m))$ time. The scored node pairs are sorted in $O(m\ln(m))$ time. Merging two nodes (a, b) requires $O(deg(a) + deg(b)) = O(n_\theta)$ time. Since we each merge at most αn pairs of nodes, the merging itself has time complexity $O(\alpha n n_\theta)$.

Therefore, our worst-case time complexity is $O(m\ln(m) + \alpha n n_\theta)$. □

Algorithm 1 Coarsening Algorithm - COARSENET (G, α)

Input: A directed, weighted graph G=(V,E,w), a reduction factor α
Output: Coarsened graph G^α_{coarse}=(V',E',w')

```
i = 0
n = |V|
G' = G
for each adjacent pair of nodes a, b ∈ V do
    Compute score(a, b) using Section 5.1
π ← ordering of node pairs in increasing order of score
while i ≤ αn do
    (a, b) = π(i)
    G' ← Contract_{G'}(a, b)
    i++
return G^α_coarse = G'
```

6. SAMPLE APPLICATION: INFLUENCE MAXIMIZATION

The eigenvalue based coarsening method described above aims to obtain a small network that approximates the diffusive properties of the original large network. As an example application, we now show how to apply our graph coarsening framework to the well studied influence maximization problem. Recall that given a diffusion model (IC model in our case) and a social network, the influence maximization problem is to find a small seed set of k nodes such that the expected number of influenced nodes is maximized.

Since we have designed our coarsening strategy such that nodes and edges important for diffusion remain untouched, we expect that solving influence maximization on the coarsened graph is a good proxy for solving it on the much larger original network. The major challenge in this process is to determine how to map the solutions obtained from the coarsened graph back onto the vertices of the original network. But due to the carefully designed coarsening strategy which tries to keep important, candidate vertices unmerged, we observe that a simple random pull back scheme works well in practice.

More formally, we propose the following multi-stage approach to solve influence maximization:

1. **Coarsen** the social network graph G by using Algorithm 1 to obtain a much smaller graph G_{coarse}. Let $\mu : V \to V_{coarse}$ denote a mapping from vertices of the original graph to those of the coarsened graph.
2. **Solve** the influence maximization problem on G_{coarse} to get k vertices $s_1, \ldots, s_k$ in the coarsened graph that optimize the desired objective function. We can use any off-the-shelf algorithm for influence maximization in this step. Since G_{coarse} is much smaller than G, traditional algorithms for influence maximization can provide high quality solutions in little time.
3. **Pull back** the solutions on to the vertices of the original graph. Given a seed s_i in G_{coarse}, we need to select a vertex $v \in \mu^{-1}(s_i)$ from G as a seed. Multiple strategies can be considered here such as $v =$

$\arg\max_{u \in \mu^{-1}(s_i)}(\sigma(u))$ where $\sigma(u)$ is the expected influence by seeding u. However, thanks to our careful coarsening framework, we show that a simple strategy of selecting a seed uniformly at random from $\mu^{-1}(s_i)$ for every seed s_i performs very well in practice.

Algorithm 2 describes our strategy to solve influence maximization problems. Note that a similar strategy can be applied to study other problems based on diffusion in networks.

Algorithm 2 CSPIN: Influence Maximization Framework

Input: A weighted graph G=(V,E,w), the number of seeds k, a reduction factor α
Output: A seed set S of k seeds
1: G^{α}_{coarse}, $\mu \leftarrow$ COARSENET (G,α) (See Algorithm 1)
2: $s'_1, s'_2, \ldots, s'_k \leftarrow$ InfluenceMaximization(G^{α}_{coarse}, k)
3: **for** $i = 1, \ldots, k$ **do**
4: $s_i \leftarrow$ random sample from $\mu^{-1}(s'_i)$
5: return $S = \{s_1, s_2, \ldots, s_k\}$

7. EXPERIMENTAL EVALUATION

We performed several experiments to show the effectiveness of COARSENET algorithm and also the GCP framework for cascade analysis.

Table 2: Datasets: Basic Statistics

Dataset	#Vertices	#Edges	Mean Degree
Flickr_small	500,038	5,002,845	20.01
Flickr_medium	1,000,001	14,506,356	29.01
Flickr_large	2,022,530	21,050,542	20.82
DBLP	511,163	1,871,070	7.32
Amazon	334,863	1,851,744	11.06
Brightkite	58,228	214,078	7.35
Portland	1,588,212	31,204,286	39.29
Flixster	55,918	559,863	20.02

Datasets. All experiments were conducted on an Intel Xeon machine (2.40 GHz) with 24GB of main memory[1]. We used a diverse selection of datasets from different domains to test our algorithm and framework (see Table 2). These datasets were chosen for their size as well as the applicability to the diffusion problem. COARSENET was tested on data from Flickr, DBLP, Amazon, Brightkite and Portland epidemiology data. In the Flickr data, vertices are users, and links represent friendships [5]. In the DBLP data, vertices represent authors and edges represent co-authorship links. Brightkite is a friendship network from a former location-based social networking service provider Brightkite. In the Amazon dataset, vertices are products and an edge represents that the two products are often purchased together. The Portland dataset is a social contact graph of vertices representing people and edges representing interactions—it represents a synthetic population of the city of Portland, Oregon, and has been used in nation-wide smallpox studies [11]. Finally, we also used a *real* cascade dataset Flixster[2], where cascades of movie ratings happen over a social network.

[1]Code at:
`http://www.cs.vt.edu/~badityap/CODE/coarsenet.tgz`
[2]`http://www.cs.ubc.ca/~jamalim/datasets/`

7.1 Performance for the GCP problem

We want to measure the performance of COARSENET algorithm on the GCP problem. In short, we can coarsen up to 70% of node-pairs using COARSENET, and still retain almost the same eigenvalue.

7.1.1 Effectiveness

As a baseline we used RANDOM, a random node-pair coarsening algorithm (randomly choose a node-pair and contract), used in some community detection techniques. Figure 4 shows the values of λ as the reduction factor α increases when we ran COARSENET and RANDOM on three datasets (we set a weight of 0.02 for this experiment). We observed that in all datasets, as the reduction factor α increases, the values of λ barely change for COARSENET, showing that the diffusive properties are maintained even with almost 70% contraction; while RANDOM destroyed the eigenvalue very quickly with increasing α. This shows that (a) large graphs *can* in fact be coarsened to large percentages while maintaining diffusion; and (b) COARSENET effectively solves the GCP problem. As we show later, we apply the GCP problem and COARSENET on a detailed sample application of influence maximization.

7.1.2 Scalability

Figure 5 shows the running times of COARSENET w.r.t. α and n. To analyze the runtime of COARSENET with respect to graph size (n), we extracted 6 connected components (with 500K to 1M vertices in steps of 100K) of the Flickr_large dataset. As expected from Lemma 5.1, we observe that in all datasets, as the reduction factor α increases, the running time increases linearly (figures also show the linear-fit, with R^2 values), and scale near-linearly as the size of the graph increases. This demonstrates that COARSENET is scalable for large datasets.

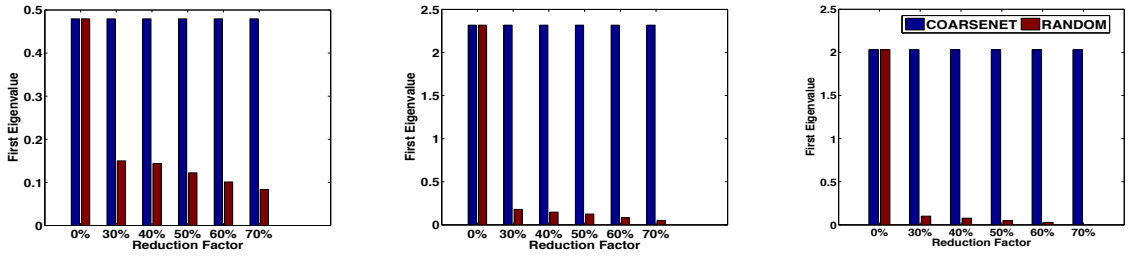
7.2 Application 1: Influence Maximization

Here we demonstrate in detail a concrete application of our GCP problem and COARSENET algorithm to diffusion-related problems. We use the well-known Influence Maximization problem. The idea as discussed before is to use the Coarsen-Solve-Project CSPIN framework (see Section 6). In short we find that we obtain 300× speed-up on large networks, while maintaining the quality of solutions.

Propagation probabilities: Since accurate propagation probabilities for these networks are not available, we generate propagation probabilities according to two models following the literature.

- **Uniform:** Each edge is assigned a low propagation probability of 0.02. In most real social networks, the propagation probabilities are known to be low. For example, [5] find that the propagation probability in the Flickr network is about 1-2%.
- **Trivalency:** We also test on the trivalency model studied in [7]. For every edge we choose a probability uniformly at random from the set $\{0.1, 0.01, 0.001\}$ which correspond to the edge having high, medium and low influence respectively.

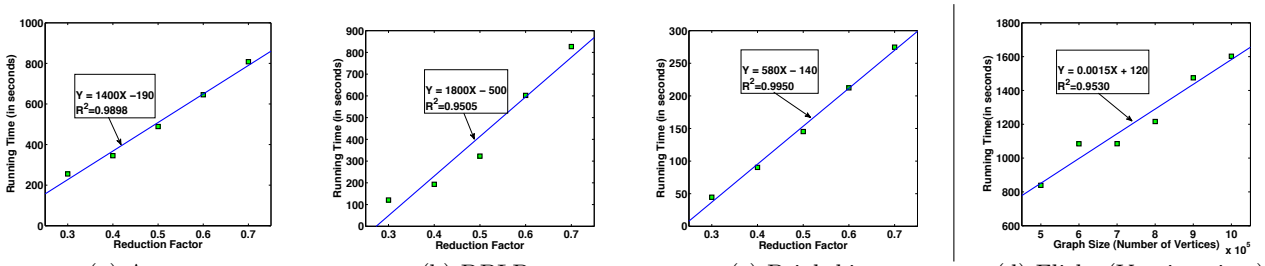
Algorithms and setup: We can use any off-the-shelf algorithm to solve Inf. Max. problem on the smaller coarsened network. Here, we choose to use the fast and popular PMIA [7] algorithm. We then compared the influence spreads



(a) Amazon (b) DBLP (c) Brightkite

Figure 4: Effectiveness of COARSENET for GCP. λ vs α for COARSENET and RANDOM. COARSENET maintains λ values.

.



(a) Amazon (b) DBLP (c) Brightkite (d) Flickr (Varying sizes)

Figure 5: Scalability of COARSENET for GCP. (a,b,c) Linear w.r.t. α. (d) Near-linear w.r.t. size of graph.

and running-times of the CSPIN framework with the plain PMIA algorithm to demonstrate gains from using GCP.

7.2.1 *Effectiveness*

Quality of solution (Influence spread). In all experiments, the influence spread generated by our CSPIN approach is within 10% of the influence spread generated by PMIA. In some cases, we even perform slightly better than PMIA. Figure 6(a) shows the expected spread obtained by selecting $k = 1000$ seeds on five datasets. For these experiments, the percentage of edges to merged is set at 90% and we use the uniform propagation model.

Quality w.r.t α. We find that we can merge up to 95% of the edges while still retaining influence spread. As more edges are merged, the coarsened graph is smaller; so the superseeds in G^{α}_{coarse} can be found faster and thus we expect our running time to decrease. We ran tests on the Flickr_medium dataset for 1000 seeds and varied α from 80% to 95%. Figure 6(b) shows the ratio of the expected influence spread obtained by CSPIN to that obtained by PMIA is almost 1 with varying α.

Quality of solution: Effect of unbiased random pullback. COARSENET groups nodes which have similar diffusion effects, hence choosing *any one* of the nodes randomly inside a group will lead to similar spreads (hence we do the random pullback in CSPIN). Note we do not claim that these groups belong to link-based communities—only that their diffusive effects are similar. To demonstrate this, we performed 100 trials of the random pullback phase for the Flickr_small graph. For these trials, 1000 superseeds were found by coarsening 90% of the edges. In each trial, we use these same superseeds to find the 1000 seeds independently and uniformly at random. Table 3 shows that the coefficient of variation of the expected spread is only 5.061×10^{-5}.

Table 3: Insensitivity of CSPIN to random pullback choices : Expected influence spread does not vary much.

#Trials	Maximum Spread	Minimum Spread	Coefficient of variation ($\frac{\sigma}{\mu}$)
100	58996.6	58984.8	5.061×10^{-5}

7.2.2 *Scalability*

Scalability w.r.t number of seeds (k). As the budget k increases, we see dramatic performance benefits of CSPIN over PMIA. We run experiments on Flickr_small, and Portland by setting $\alpha = 90\%$, and k varied from 0.01% to 1% of $|V|$. Figure 7(a,b) shows the total running times (including the coarsening). Due to lack of space we show only the results for the trivalency model (the uniform case was similar). In all datasets, as k increases, the running time of CSPIN increases very slowly. Note that we get *orders of magnitude* speed-ups: e.g. on Flickr PMIA takes *more than 10 days* to find 200+ seeds, while CSPIN runs in 2 minutes.

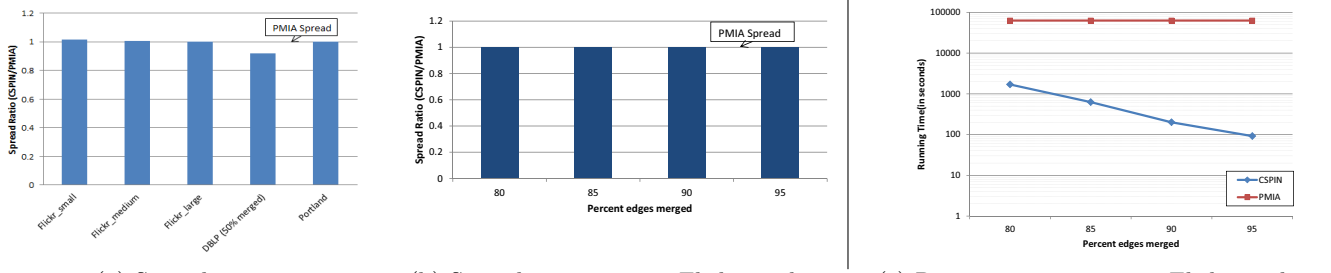
Scalability w.r.t α. We can see that the running time also drops with increased coarsening as seen in Figure 6(c).

Scalability w.r.t n. We ran CSPIN on the components of increasing size of Flickr_large with $k = 1000$ and $\alpha = 90\%$. Figure 7(c) plots the running times: CSPIN obtains a speed up of around 250× over PMIA consistently.

7.3 Application 2: Diffusion Characterization

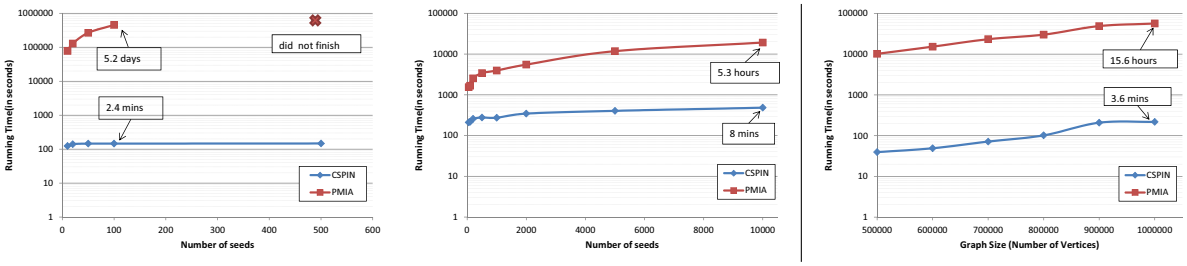
We now briefly describe how the GCP problem can help in understanding cascade datasets in an exploratory setting.

Methodology: We used a Flixster dataset, where users can share ratings of movies with friends. There is a log-file which stores ratings actions by each user: and a cascade is supposed to happen when a person rates the same movie soon after one of her friends. We use the methodology of [15] to learn influence probabilities of a IC-model over the edges of the friendship network from the traces. We then coarsen



(a) Spread ratio (b) Spread ratio vs α on Flickr_medium (c) Running times vs α on Flickr_medium

Figure 6: Effectiveness of CSPIN. Ratio of influence spread between CSPIN and PMIA for (a) different datasets; (b) varying α. (c) Running time vs α.



(a) Flickr_small (trivalency model) (b) Portland (trivalency model) (c) Flickr (Varying sizes)

Figure 7: Scalability of CSPIN. (a,b) vs k; (c) vs size of graph. CSPIN gets increasing orders-of-magnitude speed-up over PMIA.

the resulting directed graph using COARSENET to $\alpha = 50\%$, and study the formed groups (supernodes). Note that this is in contrast to the approaches where the group information is supplied by a graph-partitioning algorithm (like METIS), and then a group-based IC model is learnt. The base network had 55,918 nodes and 559,863 edges. The trace-log contained about 7 *million* actions over 48,000 movies. We get 1891 groups after removing groups with only one node, with mean group size 16.6 with the largest group having 22061 nodes (roughly 40% of nodes).

Distribution of movies over groups: Figure 8 shows the histogram of the # of groups reached by the movie propagations (following [30], we assume that a movie reaches a group if at least 10% of its nodes rated that movie). We show only the first 100 points of the distribution. We observe that a very large fraction of movies propagate in a small number of groups. Interestingly we observe a *multi-modal* distribution, suggesting movies have multiple scales of spread.

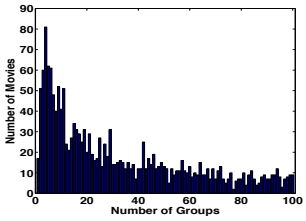


Figure 8: Distribution of # groups entered by movie traces.

Groups through the lens of surrogates: An important point to note is that our groups *may not* be link-based communities: we just ensure that nodes in a group have the same diffusive properties. We validated this observation in the previous section (Table 3). Hence a natural question is if groups found in Flixster have any other natural structure (e.g. demographics)—if they do, we can get a non-network external *surrogate* for similar diffusive characteristics. Fortunately, the Flixster does contain a couple of auxiliary features for its users (like ID, Last Login, Age). We calculated the Mean Absolute Error (MAE) for 'Age' *inside* each group, and compared it with the MAE across groups. We found that the average MAE inside the group is very small (within 2 years) compared to a MAE of almost 8 outside, which implies that ages are concentrated within groups and can act as surrogates for diffusive characteristics.

8. CONCLUSIONS

We propose influence-based coarsening as a fundamental operation in the analysis of diffusive processes in large networks. Based on the connections between influence spread and spectral properties of the graph, we propose a novel Graph Coarsening Problem and provide an effective and efficient heuristic called COARSENET. By carefully reweighting the edges after each coarsening step, COARSENET attempts to find a succinct representation of the original network which preserves important diffusive properties.

We then describe the CSPIN framework to solve influence maximization problems on large networks using our coarsening strategy. Experimental results show that CSPIN indeed outperforms traditional approaches by providing high quality solutions in a fraction of the time.

Finally we show that our COARSENET framework can also be used for examining cascade datasets in an exploratory setting. We observe that in our case study the nodes merged together form meaningful communities in the sense of hav-

ing similar diffusive properties which can serve as surrogates using external demographic information.

Future work can consist of resolving the complexity of GCP and investigating more applications of our framework to tasks where spectral properties may need to be preserved.

Acknowledgements. The authors would like to thank Christos Faloutsos for discussions. This material is based upon work supported by the US Army Research Office under Grant No. W911NF0910206, by the NSF under Grant No. IIS-1353346, by the NSA (under a 'Science of Security' lablet) and by the VT College of Engineering.

9. REFERENCES

[1] R. M. Anderson and R. M. May. *Infectious Diseases of Humans*. Oxford University Press, 1991.

[2] N. Bailey. *The Mathematical Theory of Infectious Diseases and its Applications*. Griffin, London, 1975.

[3] S. Bikhchandani, D. Hirshleifer, and I. Welch. A theory of fads, fashion, custom, and cultural change in informational cascades. *Journal of Political Economy*, 100(5):992–1026, October 1992.

[4] L. Briesemeister, P. Lincoln, and P. Porras. Epidemic profiles and defense of scale-free networks. *WORM 2003*, Oct. 27 2003.

[5] M. Cha, A. Mislove, and K. P. Gummadi. A Measurement-driven Analysis of Information Propagation in the Flickr Social Network. In *In Proceedings of the 18th International World Wide Web Conference (WWW'09)*, Madrid, Spain, April 2009.

[6] D. Chakrabarti, Y. Wang, C. Wang, J. Leskovec, and C. Faloutsos. Epidemic thresholds in real networks. *ACM TISSEC*, 10(4), 2008.

[7] W. Chen, C. Wang, and Y. Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. *KDD*, 2010.

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

[9] I. S. Dhillon, Y. Guan, and B. Kulis. Weighted graph cuts without eigenvectors a multilevel approach. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(11):1944–1957, 2007.

[10] M. Elkin and D. Peleg. Approximating k-spanner problems for k> 2. *Theoretical Computer Science*, 337(1):249–277, 2005.

[11] S. Eubank, H. Guclu, V. S. Anil Kumar, M. V. Marathe, A. Srinivasan, Z. Toroczkai, and N. Wang. Modelling disease outbreaks in realistic urban social networks. *Nature*, 429(6988):180–184, May 2004.

[12] W. S. Fung, R. Hariharan, N. J. Harvey, and D. Panigrahi. A general framework for graph sparsification. In *Proceedings of the 43rd annual ACM symposium on Theory of computing*, pages 71–80. ACM, 2011.

[13] A. Ganesh, L. Massoulié, and D. Towsley. The effect of network topology on the spread of epidemics. In *IEEE INFOCOM*, Los Alamitos, CA, 2005. IEEE Computer Society Press.

[14] 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*, 2001.

[15] A. Goyal, F. Bonchi, and L. V. Lakshmanan. Learning influence probabilities in social networks. WSDM '10, 2010.

[16] A. Goyal, W. Lu, and L. V. S. Lakshmanan. Simpath: An efficient algorithm for influence maximization under the linear threshold model. *ICDM*, 2011.

[17] M. Granovetter. Threshold models of collective behavior. *Am. Journal of Sociology*, 83(6):1420–1443, 1978.

[18] D. Gruhl, R. Guha, D. Liben-Nowell, and A. Tomkins. Information diffusion through blogspace. In *WWW '04*, 2004.

[19] Y. Hayashi, M. Minoura, and J. Matsukubo. Recoverable prevalence in growing scale-free networks and the effective immunization. *arXiv:cond-mat/0305549 v2*, Aug. 6 2003.

[20] H. W. Hethcote. The mathematics of infectious diseases. *SIAM Review*, 42, 2000.

[21] G. Karypis and V. Kumar. METIS: Unstructured graph partitioning and sparse matrix ordering system. *The University of Minnesota*, 2, 1995.

[22] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146, New York, NY, USA, 2003. ACM.

[23] J. O. Kephart and S. R. White. Measuring and modeling computer virus prevalence. *IEEE Computer Society Symposium on Research in Security and Privacy*, 1993.

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

[25] R. Kumar, J. Novak, P. Raghavan, and A. Tomkins. On the bursty evolution of blogspace. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 568–576, New York, NY, USA, 2003. ACM Press.

[26] T. Lappas, E. Terzi, D. Gunopulos, and H. Mannila. Finding effectors in social networks. In *Proceedings of the 16th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Washington, DC*, pages 1059–1068, 2010.

[27] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. S. Glance. Cost-effective outbreak detection in networks. In *KDD*, pages 420–429, 2007.

[28] M. Mathioudakis, F. Bonchi, C. Castillo, A. Gionis, and A. 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.

[29] A. G. McKendrick. Applications of mathematics to medical problems. In *Proceedings of Edin. Math. Society*, volume 44, pages 98–130, 1925.

[30] Y. Mehmood, N. Barbieri, F. Bonchi, and A. Ukkonen. Csi: Community-level social influence analysis. In *Machine Learning and Knowledge Discovery in Databases*, volume 8189 of *Lecture Notes in Computer Science*. 2013.

[31] R. Pastor-Santorras and A. Vespignani. Epidemic spreading in scale-free networks. *Physical Review Letters 86*, 14, 2001.

[32] B. A. Prakash, D. Chakrabarti, M. Faloutsos, N. Valler, and C. Faloutsos. Threshold conditions for arbitrary cascade models on arbitrary networks. In *ICDM*, 2011.

[33] B. A. Prakash, J. Vreeken, and C. Faloutsos. Spotting culprits in epidemics: How many and which ones? In *ICDM*, 2012.

[34] M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 61–70. ACM, 2002.

[35] E. M. Rogers. *Diffusion of Innovations, 5th Edition*. Free Press, August 2003.

[36] P. Shakarian, M. Broecheler, V. Subrahmanian, and C. Molinaro. Using generalized annotated programs to solve social network optimization problems. *ACM Transactions on Computational Logic*, 2012.

[37] H. Tong, B. A. Prakash, T. Eliassi-Rad, M. Faloutsos, and C. Faloutsos. Gelling, and melting, large graphs by edge manipulation. In *ACM CIKM*, 2012.

[38] H. Tong, B. A. Prakash, C. E. Tsourakakis, T. Eliassi-Rad, C. Faloutsos, and D. H. Chau. On the vulnerability of large graphs. In *ICDM*, 2010.