Semi-Data-Driven Network Coarsening

Li Gao[†], Jia Wu[‡], Hong Yang[♯], Zhi Qiao[§], Chuan Zhou[†*], Yue Hu[†]
[†]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[‡]Quantum Computation & Intelligent Systems Centre, University of Technology Sydney, Australia
[♯]MathWorks, Beijing, China
[§]Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
{gaoli, huyue}@iie.ac.cn, zhiqiao.ict@gmail.com, hong.yang@mathworks.cn, jia.wu@uts.edu.au

Abstract

Network coarsening refers to a new class of graph 'zoom-out' operations by grouping similar nodes and edges together so that a smaller equivalent representation of the graph can be obtained for big network analysis. Existing network coarsening methods consider that network structures are static and thus cannot handle dynamic networks. On the other hand, data-driven approaches can infer dynamic network structures by using network information spreading data. However, existing data-driven approaches neglect static network structures that are potentially useful for inferring big networks. In this paper, we present a new *semi-data-driven* network coarsening model to learn coarsened networks by embedding both *static* network structure data and *dynamic* network information spreading data. We prove that the learning model is convex and the Accelerated Proximal Gradient algorithm is adapted to achieve the global optima. Experiments on both synthetic and real-world data sets demonstrate the quality and effectiveness of the proposed method.

1 Introduction

Network coarsening refers to collapsing similar nodes and edges in a network so that the size of the network can be significantly reduced [Karypis and Kumar, 1998]. With massive data generated from online social networks, it is challenging to perform complicated network analysis on social networks. Network coarsening provides a new tool for big social network analysis by condensing a big network into a smaller approximating one without much loss of information.

Existing studies on network coarsening assume that network structures are known in advance and they merely focus on using popular network metrics, such as edge-cut based [Dhillon *et al.*, 2007], link based [Mathioudakis *et al.*, 2011], and heavy-edge-matching based [Karypis and Kumar, 1998]. In particular, a recent work [Purohit *et al.*, 2014] studies graph coarsening by grouping similar nodes together to find a succinct representation of subgraphs. However, all these models are based on static network structures.

*C. Zhou is the corresponding author, zhouchuan@iie.ac.cn.

On the other hand, dynamic network structure analysis has been widely studied in recent years, which refers to networks that either evolve by time (i.e., new links are continuously formed and existing links get obsolete) [Sun *et al.*, 2014; Bhat and Abulaish, 2015] or different subsets of links are activated at different time windows [Rodriguez *et al.*, 2011]. Many *data-driven* models [Rodriguez *et al.*, 2011; Cui *et al.*, 2013; Du *et al.*, 2014] have been proposed to enable dynamic network analysis by building maximum likelihood estimation models on information spreading data. However, existing data-driven approaches neglect the fact that static network structure data can be also beneficial for learning when the network is large and the training data set is relatively sparse.

Motivated by the above observations, we study in this paper a new problem of network coarsening which embeds both static network structures and dynamic network information spreading data. The basic idea is that if two nodes tightly connect with each other and co-occur frequently in the same information cascade, they are likely to be combined together. Because of the heavily skewed degree distribution in networks and a large portion of edges and nodes are relatively unimportant for network analysis [Purohit *et al.*, 2014], we aim to find a smaller equivalent network to describe the original large yet sparse network. Based on the smaller representation, sophisticated network analysis can be solved efficiently.

In deed, learning from both *static* network structures and *dynamic* information spreading data is a very challenging problem, including 1) Network structure data and network information spreading data are *heterogeneous data* that need to be modeled jointly; and 2) Combining two different data sources for network coarsening leads to a complicated optimization problem which requires efficient algorithm.

To address the challenges, we propose a semi-data-driven network coarsening approach (*Semi-NetCoarsen* for short) to condense the networks. The corresponding learning function is constructed by simultaneously minimizing the graph regularization on the static network structures and maximizing the network coarsening likelihood on information spreading data (Challenge #1). We prove that the learning function is convex and the Accelerated Proximal Gradient algorithm is adapted to obtain the global optima (Challenge #2). Moreover, as a case study, we conduct the influence maximization analysis on the coarsened network. Experiments on both synthetic and real-world data sets demonstrate the algorithm performance.

2 Preliminaries

Information Spreading data: In a network $G := (V, E, W)$, where V denotes the nodes set, E the edges set and W the weighted adjacency matrix, assume that a message c propagates through the network and leaves a trace of passed nodes $u_i \in V$ at time stamp t_i, denoted by $(u_i, t_i)_c$. An information spreading cascade $\mathbf{t}^c$ can be denoted by a N-dimensional vector $\mathbf{t}^c := (t_1^c, \ldots, t_N^c)$, where $t_i^c \in [0, T^c] \cup \{\infty\}$. The symbol ∞ labels nodes that the message does not reach during the observation window $[0, T^c]$, in which T^c corresponds to the underlying temporal dynamics [Rodriguez *et al.*, 2011]. A collection of cascades is denoted as $\mathcal{C} := \{\mathbf{t}^1, \ldots, \mathbf{t}^{|\mathcal{C}|}\}$.

Similar Nodes: Given a network G and a collection of cascades $\mathcal{C}$, if nodes i and j tightly connect with each other with higher $W_{i,j}$ and $W_{j,i}$, and also co-occur frequently in the same cascade $\mathbf{t}^c$, we call nodes i and j as similar nodes.

Supernodes and Superedges: Given a coarsened network $G_{coarsen} := (V', E', W')$, $m \in V^{'}$ represents a supernode, and $e \in E^{'}$ represents a superedge. The node set $V^{'}$ is obtained by merging all the similar nodes. The edge set $E^{'}$ is obtained by inferring the weighted adjacency matrix $W^{'}$ from the coarsened network.

In this paper, we merge similar nodes under two assumptions: 1) *One-time merge*. A pair of nodes can be activated for the network coarsening operation by only one time. If the merge does not succeed, the nodes will not be merged ever after; and 2) *One-way merge*. In a cascade $\mathbf{t}^c$, given two nodes j and i, if $t_j^c < t_i^c$, we merge node i into node j because node j is more likely to be the information source of the message c in terms of i.

3 Problem Formulation

3.1 Problem Statement

Given a network G, a collection of cascade data $\mathcal{C}$, and a network coarsening rate $0 \leq \alpha < 1$, the proposed *Network Coarsening* **aims** to infer a smaller coarsened network $G_{coarsen}$ by finding and merging the nodes that both tightly connect each other in the adjacency matrix W and frequently appear together in the cascade data $\mathcal{C}$, with $|V^{'}| = (1-\alpha)|V|$.

3.2 Principle

For a given α, we merge similar nodes and edges and remove redundant ones to obtain the coarsened network that contains K nodes, where $K = (1-\alpha)N$. Thus, the node i in G is eventually merged into a supernode s. The supernode can be regarded as a cluster with label l_s. All the nodes in s are labeled as l_s. The aim of network coarsening is to predict the class labels assigned to each node in G.

Given a non-negative vector $y_i = [y_{i,1}, \ldots, y_{i,K}]^T$ that denotes the distribution of class labels of the ith node, where $y_{i,m} \in (0, 1)$ represents the probability of node i being merged in the supernode m. The edge coarsening is determined accordingly by the merging of nodes. Assume $Y = [y_1, y_2, \ldots, y_N]^T \in \mathbb{R}^{N \times K}$, which models the class label distribution of all nodes of G. The key point is to optimize Y to ensure that nodes from the same class label are grouped together into the same supernode. Thus, if two nodes i and j are similar, y_i and y_j are more likely to be the same.

After defining the label distribution matrix Y, we formulate the learning objective function that maximizes the network coarsening likelihood on the network cascade data $\mathcal{C}$ and minimizes the graph regularization on the prior network structure data W. By solving the learning function, we can obtain the optimal Y^*, based on which we deliver the coarsened network $G_{coarsen}$.

3.3 Network Coarsening Model

Modeling Dynamic Information Spreading Cascades

If two nodes i and j co-occur frequently in the same cascade $\mathbf{t}^c$, they are likely to be coarsened together, *i.e.,* they have close distribution y_i and y_j. $y_i^T y_j$ is adopted to measure their similarity [Cai *et al.*, 2011].

Definition 1. *(*Node Merging Status*)*
For cascade $\mathbf{t}^c$, if $t_j^c < t_i^c$, we call node j as node i's parent and node i as node j's child. For the child node in an observation window T^c, the result of the merging status is binary, i.e., either merged or unmerged by a parent. For the child node outside T^c, the result of its merging is unmerged.

Let $f_{j,i}^c = f(t_i^c | t_j^c; y_i^T y_j, \sigma_c)$ be the probability density function that describes the conditional likelihood of merging node j and node i ($t_j^c < t_i^c$). The MLE of σ_c is $1/\overline{\Delta t}$, where σ_c is the diffusion speed and $\overline{\Delta t}$ is the average information propagation delay time between two neighboring nodes in $\mathbf{t}^c$ [Wang *et al.*, 2014]. Specifically, we use exponential distribution to model $f_{j,i}^c$, where $f_{j,i}^c = (y_i^T y_j + \sigma_c) \cdot exp(-(y_i^T y_j + \sigma_c)(t_i^c - t_j^c))$. For the given $f_{j,i}^c$, the symbol $P(t_j^c \to t_i^c | f_{j,i}^c)$ denotes the probability that node j merges node i.

From Definition 1, the *unmerged* status of node i towards one of its parents j denotes that label l_i is very different from l_j. By contrast, the *merged* status denotes that l_i is close to l_j. Thus, the label distribution of nodes of G can be determined by the node merging status.

Given cascades data $\mathcal{C}$, assume the independent cascade with respect to the network coarsening process, let $\prod_{\mathbf{t}^c \in \mathcal{C}} l(\mathbf{t}^c; Y, \sigma_c)$ represent the network coarsening likelihood. $l(\mathbf{t}^c; Y, \sigma_c)$ denotes the likelihood in terms of cascade $\mathbf{t}^c$, which is calculated by combining the coarsening likelihood of all the nodes, which is divided by T^c,

$$l(\mathbf{t}^c; Y, \sigma_c) = l(\mathbf{t}^c_{\leq T^c}; Y, \sigma_c) \times l(\mathbf{t}^c_{>T^c}; Y, \sigma_c). \quad (1)$$

where $l(\mathbf{t}^c_{>T^c}; Y, \sigma_c)$ is the likelihood function shown in Eq. (3) in terms of the cascade $\mathbf{t}^c_{>T^c} = (t_1^c, \ldots, t_N^c | t_i^c > T^c)$, which denotes that nodes outside T^c haven't been merged by any parents. $l(\mathbf{t}^c_{\leq T^c}; Y, \sigma_c)$, calculated in Eq. (2), denotes the network coarsening likelihood in terms of the cascade $\mathbf{t}^c_{\leq T^c} = (t_1^c, \ldots, t_N^c | t_i^c \leq T^c)$ for the given T^c.

$$l(\mathbf{t}^c_{\leq T^c}; Y, \sigma_c) = \prod_{t_i^c \leq T^c} f(t_i^c | t_1^c, \ldots, t_N^c \backslash t_i^c; Y, \sigma_c). \quad (2)$$

$$l(\mathbf{t}^c_{>T^c}; Y, \sigma_c) = \prod_{t_i^c \leq T^c} \prod_{t_m^c > T^c} (1 - P(t_i^c \to T^c | f_{i,m}^c)). \quad (3)$$

Obviously, $l(\mathbf{t}^c_{\leq T^c}; Y, \sigma_c)$ in Eq. (2) can be decomposed into

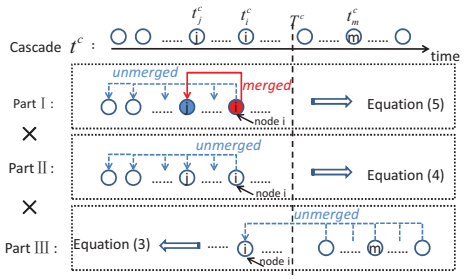

Figure 1: An simple illustration of the network coarsening process based on a cascade $\mathbf{t}^c$. The dashed blue lines in Part I and Part II represent that node i (in *unmerged* status when $t_i^c \leq T^c$) has not been merged by any node j ($t_j^c < t_i^c$). The dashed blue line in Part III represents that any node m (in *unmerged* status when $t_m^c > T^c$) has not been merged by the node i ($t_i^c \leq T^c$). The solid red line in Part I represents that node i (in *merged* status when $t_i^c \leq T^c$) has been merged by the node j ($t_j^c < t_i^c$).

two parts: 1) the first part is the nodes (in *unmerged* status) observed in the window T^c that have not been merged by any of its parents; 2) the second part is the nodes (in *merged* status) observed in T^c that have been merged by one of its parents, as shown in Fig. 1. Thus, the likelihood of the first part can be calculated as Eq. (4). Considering that the likelihood of a node being merged can be calculated by combining the probability of each its potential parent initializing the merge operation, thus, the likelihood of the second part is calculated as Eq. (5). Therefore, $l(\mathbf{t}^c_{\leq T^c}; Y, \sigma_c)$ in Eq. (2) can be computed by multiplying Eq. (4) and Eq. (5).

$$\prod_{u:t_u^c<t_i^c} (1 - P(t_u^c \to t_i^c | f_{u,i}^c)). \tag{4}$$

$$\sum_{j:t_j^c<t_i^c} f(t_i^c | t_j^c; y_i^T y_j, \sigma_c) \prod_{j\neq k, t_k^c<t_i^c} (1 - P(t_k^c \to t_i^c | f_{k,i}^c)). \tag{5}$$

Thus, Eq. (1) can be represented as,

$$\begin{aligned} l(\mathbf{t}^c; Y, \sigma_c) = & \prod_{t_i^c \leq T^c} \prod_{u:t_u^c<t_i^c} (1 - P(t_u^c \to t_i^c | f_{u,i}^c)) \times \\ & \prod_{k:t_k^c<t_i^c} (1 - P(t_k^c \to t_i^c | f_{k,i}^c)) \sum_{j:t_j^c<t_i^c} \frac{f(t_i^c | t_j^c; y_i^T y_j, \sigma_c)}{1 - P(t_j^c \to t_i^c | f_{j,i}^c)} \times \\ & \prod_{t_m^c > T^c} (1 - P(t_i^c \to T^c | f_{i,m}^c)). \end{aligned} \tag{6}$$

Modeling Static Network Structures

If two nodes i and j tightly connect with each other, they will have high probability to share the same class label (*i.e.,* graph regularization [Zhou *et al.*, 2005; Gu and Han, 2011]). The problem can be formulated as a graph clustering problem where the graph Laplacian can be taken as an effective tool.

We use $\|y_i - y_j\|^2$ to measure the "dissimilarity" between y_i and y_j [Cai *et al.*, 2011]. The formulation of graph regularization $R(Y, W)$ for network coarsening is shown in Eq. (7). Due to the fact that the closeness of y_i and y_j is symmetric, in this case, if $W_{i,j} \neq W_{j,i}$, we use the larger one to measure the connection between node i and j.

$$\begin{aligned} R(Y, W) &= \frac{1}{2} \sum_i \sum_j \|y_i - y_j\|^2 W_{i,j} \\ &= \sum_i y_i^T y_i D_{i,i} - \sum_i \sum_j y_i^T y_j W_{i,j} \\ &= Tr(Y^T D Y) - Tr(Y^T W Y) = Tr(Y^T U Y). \end{aligned} \tag{7}$$

where the matrix D is a diagonal matrix with entries being the row sums of W, $D_{i,i} = \sum_j W_{i,j}$, and U is the graph Laplacian [Chung, 1997] given by $U = D - W$.

3.4 Node Label Distribution Learning Function

In this paper, we maximize the network coarsening likelihood $\prod_{\mathbf{t}^c \in \mathcal{C}} l(\mathbf{t}^c; Y, \sigma_c)$ based on the dynamic network cascade data $\mathcal{C}$, together with minimizing the graph regularization $R(Y, W)$ based on the static network structure data W.

The maximum likelihood estimation (MLE) of Y is a solution to $min_Y - \sum_{\mathbf{t}^c \in C} \log l(\mathbf{t}^c; Y, \sigma_c)$. In addition, as a node in G is merged into only one supernode, it is expected that $Y_{\cdot i}$ and $Y_{\cdot j}$ is orthogonal, where $i \neq j$. We set constraints as $Y^T Y = I$ to ensure the above conditions. Meanwhile, the row vector $Y_{m\cdot}$ is expected to be sparse. Thus, the network node label distribution learning function is formulated as

$$\begin{aligned} \underset{Y}{\text{argmin}} \quad & - \sum_{\mathbf{t}^c \in \mathcal{C}} \log l(\mathbf{t}^c; Y, \sigma_c) + \lambda Tr(Y^T U Y) + \lambda_y \|Y\|_1 \\ \text{s.t.} \quad & Y^T Y = I. \end{aligned} \tag{8}$$

To solve the above problem efficiently, we relax the equality constraint as follows,

$$\begin{aligned} \underset{Y}{\text{argmin}} \quad & - \sum_{\mathbf{t}^c \in \mathcal{C}} \log l(\mathbf{t}^c; Y, \sigma_c) + \lambda Tr(Y^T U Y) \\ & + \lambda_y \|Y\|_1 + \rho \|Y^T Y - I\|_F^2. \end{aligned} \tag{9}$$

where λ, λ_y and ρ are non-negative parameters. When $\lambda = 0$, the model degenerates to *data-driven* method that only models the network coarsening process on the cascade data $\mathcal{C}$.

Theorem 1. *The problem in Eq.* (9) *is convex and the global optima can be obtained by using gradient-based algorithms.*

Obviously, the first term of the negative likelihood function $-\log l(\mathbf{t}^c; Y, \sigma_c)$ is convex, and the other three regularization terms $Tr(Y^T U Y)$, $\|Y\|_1$ and $\|Y^T Y - I\|_F^2$ are also convex, so the function is jointly convex.

4 The Semi-Data-Driven Framework

We first learn the optimal solution Y^* to the objective in Eq. (9) in Section 4.1. Then, we use the optimal Y^* to obtain the coarsened network $G_{coarsen}$ by applying the semi-data-driven network coarsening framework in Section 4.2. As a case study, in Section 4.3, we solve the influence maximization on the coarsened network $G_{coarsen}$.

4.1 Node Label Distribution Learning Algorithm

In this work, we adapt the Accelerated Proximal Gradient (APG) [Beck and Teboulle, 2009] to learn the optimal node label distribution Y^*. Let $\eta(Y) = - \sum_{\mathbf{t}^c \in \mathcal{C}} \log l(\mathbf{t}^c; Y, \sigma_c) +$

Algorithm 1: Network Node Label Distribution Learning

Input: $\gamma \in (0,1)$, $Y^0 \in S_N^+$, $Z^1 = Y^0$, Lipschitz constant L_f, $L = L_f$ and $t^1 = 1$, $k = 1$
Output: The optimal solution of Eq. (9)

for $j = 0, 1, \ldots,$ **do**
  $G = Z^k - \frac{1}{L}\nabla\eta(Z^k)$, compute $p_L(G)$ (Eq. (12))
  if $F(p_L(G)) \leq Q_L(p_L(G), Z^k)$ **then**
  ;
  set $L^k = L$, stop
  else $L = \gamma^{-1}L$;
Set $Y^k = p_L(G)$; $t^{k+1} = \frac{1+\sqrt{1+4(t^k)^2}}{2}$
Set $Z^{k+1} = Y^k + (\frac{t^k - 1}{t^{k+1}})(Y^k - Y^{k-1})$; $L = \gamma L^k$
$k = k + 1$
Stop if $\frac{F(Y^k) - F(Y^{k+1})}{F(Y^k)} \leq \epsilon$; otherwise, go to 1
return The optimal solution Y^*

$\lambda Tr(Y^T UY) + \rho\|Y^T Y - I\|_F^2$, and $\xi(Y) = \lambda_y \|Y\|_1$, then, Eq.(10) is the summation of $\eta(Y)$ and $\xi(Y)$.

$$F(Y) = \eta(Y) + \xi(Y). \tag{10}$$

The derivative of η is denoted as $\nabla\eta$ which is Lipschitz continuous on Y, $\|\nabla\eta(Y) - \nabla\eta(Z)\| \leq L_f\|Y - Z\|$, $Z \in S_N^+$, where S_N^+ is $N \times K$ non-negative matrix. Given Z, for $L > 0$, the quadratic approximation of $F(Y)$ can be defined as,

$$\begin{aligned} Q_L(Y, Z) &= \eta(Z) + < Y - Z, \nabla\eta(Z) > + \frac{L}{2}\|Y - Z\|_F^2 + \xi(Y) \\ &= \frac{L}{2}\|Y - G\|_F^2 + \xi(Y) + \eta(Z) - \frac{1}{2L}\|\nabla\eta(Z)\|_F^2. \end{aligned} \tag{11}$$

where $G = Z - \frac{1}{L}\nabla\eta(Z)$. To minimize $Q_L(Y, Z)$ with respect to Y, it is reduced to solve the following $p_L(G)$ by ignoring the constant terms in Z:

$$p_L(G) = \text{argmin}_{Y \in S_N^+} \frac{L}{2}\|Y - G\|_F^2 + \xi(Y). \tag{12}$$

Then, the solution of Eq. (12) is $p_L(G) = U\text{Diag}((\tau - \lambda_y/L)_+)V^T$ [Toh and Yun, 2010], where $G = U\Sigma V^T$, $\Sigma = \text{Diag}(\tau)$, $\tau \in \mathcal{R}^q$ is the vector of positive singular values arranged in a descending order and $x_+ = \max\{x, 0\}$. We use 0 to replace the negative entries in Y. The parameter learning is shown in Algorithm 1.

In experiments, we empirically set $\gamma = 0.4$, $\epsilon = 0.001$, and $L_f = 10^{-6}NN_c$, where N_c is the cardinality of $\mathcal{C}$.

4.2 The Semi-NetCoarsen Algorithm

Algorithm 2 lists the steps of the network coarsening method *Semi-NetCoarsen*. Given an upper bound of α, the network is coarsened by merging αN nodes. Based on Y^*, for each node i, we merge i into the supernode m with the largest $y_{i,m}$, where $m = \text{argmax}_k\{y_{i,k}\}, k = 1, \ldots, (1 - \alpha)N$.

Next, based on the previous work [Purohit *et al.*, 2014], we use the following principle to assign new weights between supernodes. Let $y_s^{'}$ denote the label distribution of the node s in the node merging process, $y_s^{'} = \frac{\sum_k y_k^*}{n}$, where n is the number of node k that has been merged into s. We also denote $\psi^i(v)$ (respectively $\psi^o(v)$) as a set of in-neighbors and out-neighbors [Purohit *et al.*, 2014] of a node v. Let $v_u^i = W_{u,v}$ and $v_u^o = W_{v,u}$ denote the weight of the corresponding edges. If nodes a and b are merged into a new node c, the edge weight c_t^i is defined in Eq. (13). c_t^o is corresponding defined.

Algorithm 2: The *Semi-NetCoarsen*$(G, \mathcal{C}, \alpha)$ algorithm

Input: Network G, cascades $\mathcal{C}$, the coarsening rate α
Output: The coarsened network $G_{coarsen}$

$V^{'} = \emptyset$; Network coarsening based on Eq. (9)
Update label distribution matrix Y^* using Algorithm 1
for *i* = 1 *to* N **do**
  Assign label l_m to node i with $m \leftarrow \text{argmax}_k\{y_{i,k}^*\}$
  $V^{'} \leftarrow \{l_m\} \cup V^{'}$
while *($|V^{'}| < K$)* **do**
  for *j* = 1 *to* K **do**
    if $l_j \notin V^{'}$ **then**
      $i \leftarrow \text{argmax}_i\{y_{i,j}^*\}$, $c = 2$
      while *(l_i labels only one node)* **do**
        $i \leftarrow$ the row-coordinate of c-th larger of j-th column in Y^*; $c = c + 1$
      Assign label l_j to node i
      Update $V^{'}$ (remove i from the previous labeled class); $V^{'} \leftarrow \{l_j\} \cup V^{'}$
for *(Each pair of supernodes s_1 and s_2 in $V^{'}$)* **do**
  Assign weights to $s_1 \rightarrow s_2$ and $s_2 \rightarrow s_1$
return $G_{coarsen}$

$$c_t^i = \begin{cases} \frac{(1+b_a^i)(a_t^i + y_a^{'T} y_t^{'})}{3}, & \forall t \in \psi^i(a) \backslash \psi^i(b) \\ \frac{(1+a_b^i)(b_t^i + y_b^{'T} y_t^{'})}{3}, & \forall t \in \psi^i(b) \backslash \psi^i(a) \\ \frac{(1+b_a^i)(a_t^i + y_a^{'T} y_t^{'})}{6} & \\ + \frac{(1+a_b^i)(b_t^i + y_b^{'T} y_t^{'})}{6}, & \forall t \in \psi^i(a) \cap \psi^i(b) \end{cases} \tag{13}$$

Complexity Analysis: It takes $O(N_c NK)$ time to model the network coarsening process and assign labels to nodes. In the coarsening process, merging node i and j takes time $O(deg(i) + deg(j)) = O(n_\theta)$, where n_θ is the maximum degree of any nodes at any time. To sum up, the total worst-case time complexity of Algorithm 2 is $O(N_c NK + \alpha N n_\theta)$.

4.3 Semi-NetCoarsen Influence Maximization

As a case study, we show how to apply the network coarsening approach *Semi-NetCoarsen* to the well studied influence maximization [Lu *et al.*, 2015; Zhou *et al.*, 2015], which aims to select the most influential nodes to maximize the spread of influence. The coarsened network is expected to approximate the original network and speeds up the calculation. Specifically, we design a framework *CFSInflu* based on the *Semi-NetCoarsen* algorithm, including the following main steps.

- *Coarsen the network:* Given cascades data $\mathcal{C}$, we coarsen the large original network G using the proposed algorithm *Semi-NetCoarsen* to obtain a coarsened network $G_{coarsen}$. A mapping function $\omega : G \rightarrow G_{coarsen}$ from nodes in G to that in $G_{coarsen}$ is required.

- *Find the most influential nodes:* Find the most influential k nodes $s_1, \ldots, s_k$ in $G_{coarsen}$ to maximize the spread of influence. A ranking function $rank(s)$ that ranks the supernode s of $G_{coarsen}$, in terms of the number of nodes of G merged in s, is required. We select the nodes $s_1, \ldots, s_k$ based on $rank(s)$ which gives the top-k answers as the seed nodes of $G_{coarsen}$.
- *Select the most influential nodes in G:* Given the seed node s_i in $G_{coarsen}$, we select a node $v \in \omega^{-1}(s_i)$ in G as a new seed. We should select v for $v = \text{argmax}_{u \in \omega^{-1}(s_i)}(\sigma(u))$, where $\sigma(u)$ is the expected influence function of node u. Based on the network coarsening process, we select a seed v with the maximal value of $\mathcal{I}(v) = \frac{\sum_{i=1}^{m} \sigma_i(v)}{m}$ from $\omega^{-1}(s_i)$, *i.e.*, $s_i^{'} = \text{argmax}_{v \in \omega^{-1}(s_i)} \mathcal{I}(v)$, where $\sigma_i(v)$ denotes the number of nodes generated from v in the cascade $\mathbf{t}^i$, and m denotes the number of cascades generated from v.

Note that "coarsen the network, solve the problem in the coarsened network, and then project the solutions back towards the original graph", can be applied to other social analysis [Karypis and Kumar, 1995; Purohit *et al.*, 2014].

5 Experiments

We evaluate the effectiveness of the proposed *Semi-NetCoarsen* approach on both the synthetic and real-world data sets. All experiments are conducted on a Linux system with 6 cores 1.4GHZ AMD CPUs and 32GB memory.

5.1 Data Sets

For synthetic data, we consider the Kronecker graph model [Leskovec *et al.*, 2010] with a parameter matrix $[0.9, 0.1; 0.1, 0.9]$ to generate synthetic Kronecker data, and the Forest Fire model [Barabási and Albert, 1999] to generate FFire data with forward burning probability 0.2, backward burning probability 0.17. For real-world data, we collect Twitter data[1] [Zhang *et al.*, 2013]. Following the work [Wang *et al.*, 2014], we select users with a large number of friends (213 friends in our work) as the nodes and extract their social connections as edges. For each tweet record, we extract its diffusion path after eliminating all the isolated nodes and edges. Table 1 gives the statistics of the data sets.

5.2 Experimental Settings

As the network coarsening rate α increases, the network is condensed smaller. With randomly selected k nodes, if the influence spread of the selected k nodes in the original network is the same as that in the coarsened network, then the network is taken as accurately coarsened.

Given a node set S, for the original and coarsened networks, we calculate the expected number of influenced nodes under the Independent Cascade model [Goldenberg *et al.*, 2001]. Specifically, for each influenced supernode m, let $\mathcal{C}(m)$ represent a set of cascades generated from m. We uniformly sample one cascade from $\mathcal{C}(m)$ to calculate the expected number of influenced nodes $\mathcal{G}(m)$ which have been merged into m. We repeat the process by 100 times and report the average results. We repeat to randomly select the seed sets S by 100 times. Thus, we obtain the average result $\mathcal{G}_c(S)$ for $G_{coarsen}$ and $\mathcal{G}(S)$ for G.

[1] http://aminer.org/billboard/Influencelocality

Table 1: Data Statistics

Dataset	#Nodes	#Edges	#Cascades
Kronecker	1,000,000	8,048,000	834,000
FFire	1,000,000	14,770,000	5,000,000
Twitter	288,416	2,265,210	2,683,729

Metrics

We use two metrics to evaluate the performance, 1) the running time; and 2) the average influence spread error rate $\mathcal{E}(S)$, which is calculated as $\mathcal{E}(S) = \frac{1}{n} \sum_i^n \frac{|\mathcal{G}_c(S) - \mathcal{G}(S)|}{\mathcal{G}(S)}$. n is set to 100. The smaller $\mathcal{E}(S)$ is favored in comparisons.

Baseline Methods

We implement the following methods for comparisons.

- *Random:* A random network coarsening algorithm that randomly chooses node pairs for join.
- *CoarseNet:* A recent network coarsening method based on static network structures [Purohit *et al.*, 2014].
- *Data-driven:* A data-driven approach that only uses information cascade data $\mathcal{C}$ for network coarsening.

Parameter Setup

The parameter λ is searched from $\lambda \in \{0.1, 1, 10, 100\}$, λ_y is searched from $\lambda_y \in \{0.01, 0.1, 1\}$, and ρ is selected from $\rho \in \{0.01, 0.1, 1\}$. The parameters are tuned by the smallest $\mathcal{E}(S)$ on a sub-graph with 1,000 nodes and 2,000 cascade data for each dataset when $\alpha = 0.5$. Table 2 reports the results.

5.3 Experimental Results

Fig. 2 reports the results under the metrics with varying α from $\{0.3, 0.4, 0.5, 0.6, 0.7\}$, in which we randomly select 20 source nodes, $|S| = 20$. From Fig. 2, we have the following observations. 1) The average influence spread error rate of *Semi-NetCoarsen* is much smaller than the baselines, and shows robustness and consistency across both synthetic and real-world datasets. For example, for the Twitter dataset when $\alpha = 0.3$, the result of *Semi-NetCoarsen* is 0.0214, while it is 0.0911, 0.1802, 0.6029 respectively for *Data-driven*, *CoarseNet* and *Random* method. The main reason is that our method embeds both the static network structure and dynamics occurring on the network (*i.e.,* information cascade). 2) The *Semi-NetCoarsen* method uses more time than its peers, which is a trade-off between the mean influence spread error rate and running time. 3) To sum up, our

Table 2: Parameters Setup

Dataset	λ	λ_y	ρ	T^c	σ_c
Kronecker	1	0.1	0.1	600	10
FFire	10	0.01	0.01	600	10
Twitter	100	1	0.01	720	0.2

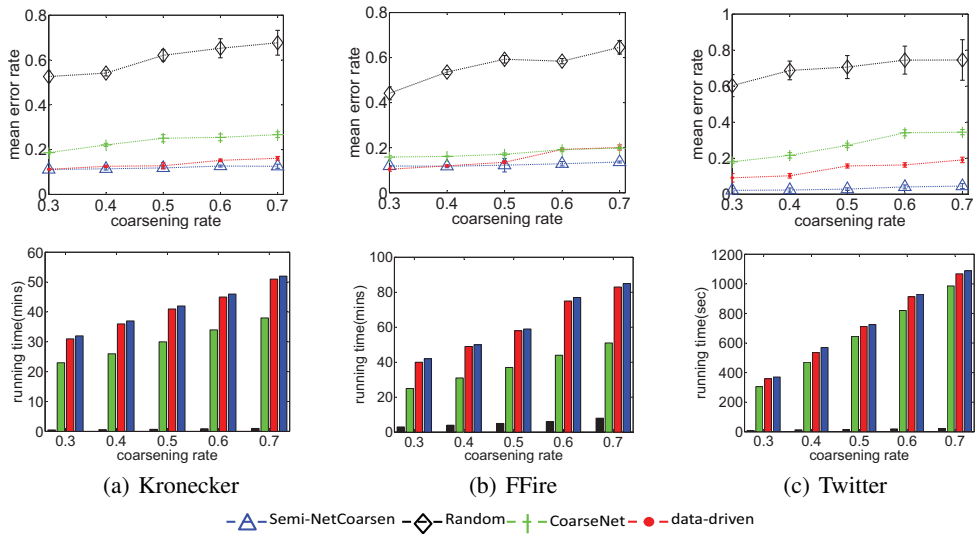

Figure 2: Performance comparisons of the network coarsening approaches.

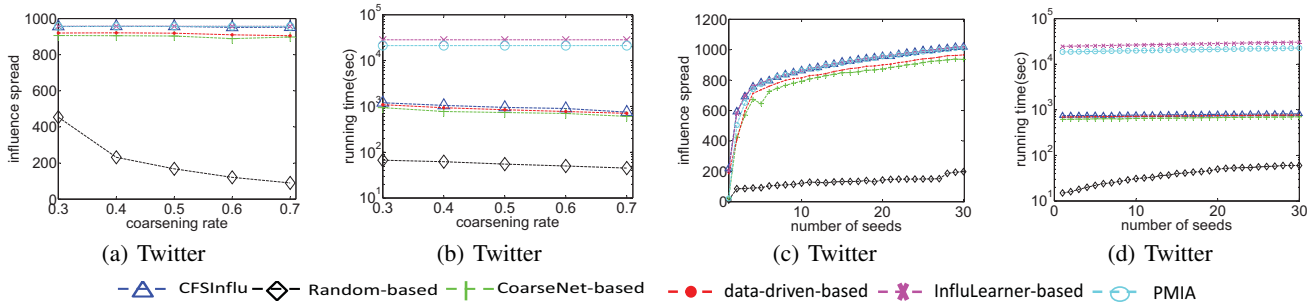

Figure 3: Performance comparisons on a case study: influence maximization on the Twitter dataset.

method outperforms baseline methods in terms of accuracy without significantly raising time cost.

5.4 Influence Maximization: A Case Study

We implement the following influence maximization methods for comparison with the *CFSInflu* method proposed in Section 4.3: *Random-based*, *CoarseNet-based* [Purohit *et al.*, 2014], *Data-driven-based*, *InfluLearner-based* [Du *et al.*, 2014] and *PMIA* [Chen *et al.*, 2010]. Among these baselines, the *InfluLearner-based* and *PMIA* approaches are carried out directly on the original networks. The rest three approaches work on the coarsened networks based on the corresponding network coarsening methods.

Fig. 3 reports the experimental results, where we vary α from $\{0.3, 0.4, 0.5, 0.6, 0.7\}$ (the number of seeds $k = 20$), and k from 1 to 30 ($\alpha = 0.5$). Due to the limited space, we only report the results on the Twitter dataset.

With respect to influence spread metric. The *InfluLearner-based* and *PMIA* methods directly work on the original networks, and thus obtain the best results. Fig. 3(c) shows that the influence spread derived from the proposed *CFSInflu* method approximates the result on the original network. Meanwhile, *CFSInflu* method outperforms all the baseline methods that conducted on the coarsened networks.

With respect to running time. When more pairs of nodes are merged, the coarsened graph is smaller. Thus, the supernodes in $G_{coarsen}$ can be found faster and we can conduct the influence maximization analysis efficiently on the coarsened network. For example, Fig. 3(b) shows that the running time of the *CFSInflu* method drops with increasing α. Furthermore, Fig. 3(d) shows our method can obtain about 100 $\times$ speedup based on the coarsened network. we can conclude that our method runs orders of magnitude faster than the *InfluLearner-based* or *PMIA* methods while maintaining the influence spread results.

6 Conclusions

In this paper, we propose a new semi-data-driven framework *Semi-NetCoarsen* to study the network coarsening problem by both maximizing the network coarsening likelihood on the dynamic information cascade data and minimizing the graph regularization on the static network structure data. We apply the coarsened network to conduct case study on influence maximization analysis. Experiments and comparisons on both synthetic and real-world data demonstrate the effectiveness of the proposed method.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments and suggestions. This work was supported by the 973 project (No. 2013CB329605), NSFC (No. 61502479 and 61370025), Strategic Leading Science and Technology Projects of CAS (No. XDA06030200), and Australia ARC Discovery Project (DP140102206).

References

[Barabási and Albert, 1999] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[Beck and Teboulle, 2009] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[Bhat and Abulaish, 2015] Sajid Yousuf Bhat and Muhammad Abulaish. Hoctracker: Tracking the evolution of hierarchical and overlapping communities in dynamic social networks. *Knowledge and Data Engineering, IEEE Transactions on*, 27(4):1019–1013, 2015.

[Cai *et al.*, 2011] Deng Cai, Xiaofei He, Jiawei Han, and Thomas S Huang. Graph regularized nonnegative matrix factorization for data representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(8):1548–1560, 2011.

[Chen *et al.*, 2010] Wei Chen, Chi Wang, and Yajun Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *KDD*, pages 1029–1038, 2010.

[Chung, 1997] Fan RK Chung. *Spectral graph theory*, volume 92. American Mathematical Soc., 1997.

[Cui *et al.*, 2013] Peng Cui, Shifei Jin, Linyun Yu, Fei Wang, Wenwu Zhu, and Shiqiang Yang. Cascading outbreak prediction in networks: a data-driven approach. In *KDD*, pages 901–909, 2013.

[Dhillon *et al.*, 2007] Inderjit S Dhillon, Yuqiang Guan, and Brian Kulis. Weighted graph cuts without eigenvectors a multilevel approach. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(11):1944–1957, 2007.

[Du *et al.*, 2014] Nan Du, Yingyu Liang, Maria Balcan, and Le Song. Influence function learning in information diffusion networks. In *ICML*, pages 2016–2024, 2014.

[Goldenberg *et al.*, 2001] Jacob Goldenberg, Barak Libai, and Eitan Muller. Using complex systems analysis to advance marketing theory development: Modeling heterogeneity effects on new product growth through stochastic cellular automata. *Academy of Marketing Science Review*, 9(3):1–18, 2001.

[Gu and Han, 2011] Quanquan Gu and Jiawei Han. Towards feature selection in network. In *CIKM*, pages 1175–1184, 2011.

[Karypis and Kumar, 1995] George Karypis and Vipin Kumar. Metis-unstructured graph partitioning and sparse matrix ordering system, version 2.0. 1995.

[Karypis and Kumar, 1998] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing*, 20(1):359–392, 1998.

[Leskovec *et al.*, 2010] Jure Leskovec, Deepayan Chakrabarti, Jon Kleinberg, Christos Faloutsos, and Zoubin Ghahramani. Kronecker graphs: An approach to modeling networks. *The Journal of Machine Learning Research*, 11:985–1042, 2010.

[Lu *et al.*, 2015] Wei-Xue Lu, Peng Zhang, Chuan Zhou, Chunyi Liu, and Li Gao. Influence maximization in big networks: an incremental algorithm for streaming subgraph influence spread estimation. In *IJCAI*, pages 2076–2082, 2015.

[Mathioudakis *et al.*, 2011] Michael Mathioudakis, Francesco Bonchi, Carlos Castillo, Aristides Gionis, and Antti Ukkonen. Sparsification of influence networks. In *KDD*, pages 529–537, 2011.

[Purohit *et al.*, 2014] Manish Purohit, B Aditya Prakash, Chanhyun Kang, Yao Zhang, and VS Subrahmanian. Fast influence-based coarsening for large networks. In *KDD*, pages 1296–1305, 2014.

[Rodriguez *et al.*, 2011] Manuel Gomez Rodriguez, David Balduzzi, and Bernhard Schölkopf. Uncovering the temporal dynamics of diffusion networks. In *ICML*, pages 561–568, 2011.

[Sun *et al.*, 2014] Yizhou Sun, Jie Tang, Jiawei Han, Cheng Chen, and Madhu Gupta. Co-evolution of multi-typed objects in dynamic star networks. *Knowledge and Data Engineering, IEEE Transactions on*, 26(12):2942–2955, 2014.

[Toh and Yun, 2010] Kim-Chuan Toh and Sangwoon Yun. An accelerated proximal gradient algorithm for nuclear norm regularized linear least squares problems. *Pacific Journal of Optimization*, 6(615-640):15, 2010.

[Wang *et al.*, 2014] Senzhang Wang, Xia Hu, Philip S Yu, and Zhoujun Li. Mmrate: inferring multi-aspect diffusion networks with multi-pattern cascades. In *KDD*, pages 1246–1255, 2014.

[Zhang *et al.*, 2013] Jing Zhang, Biao Liu, Jie Tang, Ting Chen, and Juanzi Li. Social influence locality for modeling retweeting behaviors. In *IJCAI*, pages 2761–2767, 2013.

[Zhou *et al.*, 2005] Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. Learning from labeled and unlabeled data on a directed graph. In *ICML*, pages 1036–1043, 2005.

[Zhou *et al.*, 2015] Chuan Zhou, Peng Zhang, Wenyu Zang, and Li Guo. On the upper bounds of spread for greedy algorithms in social network influence maximization. *Knowledge and Data Engineering, IEEE Transactions on*, 27(10):2770–2783, 2015.