2011 11th IEEE International Conference on Data Mining

Finding Communities in Dynamic Social Networks

Chayant Tantipathananandh and Tanya Y. Berger-Wolf
Department of Computer Science
University of Illinois at Chicago
851 S Morgan St
Chicago, IL 60607
Email: {ctanti2, tanyabw}@uic.edu

***Abstract*—Communities are natural structures observed in social networks and are usually characterized as "relatively dense" subsets of nodes. Social networks change over time and so do the underlying community structures. Thus, to truly uncover this structure we must take the temporal aspect of networks into consideration. Previously, we have represented framework for finding dynamic communities using the social cost model and formulated the corresponding optimization problem [33], *assuming that partitions of individuals into groups are given in each time step*. We have also presented heuristics and approximation algorithms for the problem, with the same assumption [32]. In general, however, dynamic social networks are represented as a sequence of graphs of snapshots of the social network and the assumption that we have partitions of individuals into groups does not hold. In this paper, we extend the social cost model and formulate an optimization problem of finding community structure from the sequence of arbitrary graphs. We propose a semidefinite programming formulation and a heuristic rounding scheme. We show, using synthetic data sets, that this method is quite accurate on synthetic data sets and present its results on a real social network.**

***Keywords*-community structure; dynamic social networks; semidefinite programming**

I. Introduction

Many systems in nature can be modeled as networks. Examples include smartphone communications, online social networking sites, animal sightings, and molecular interactions. One phenomenon that has been frequently observed across different kinds of networks is the presence of latent community structures. Nodes can be grouped in such a way that interactions among group members are apparent while interactions between different groups are less common. Such groups are customarily termed "communities", "modules", or "clusters" because they represent a notion of cohesiveness or homogeneity among the members [22, 35]. The problem of finding such latent community structure is non-trivial and is compounded when the interactions in the system depend on time. As a result, communities may persist just over a period of time, follow a periodic pattern, change member composition gradually or abruptly, and demonstrate many other dynamic behaviors. Thus, it is important to consider the temporal aspect of the system of interactions. We call such a system of interactions a "dynamic network" as opposed to a "static network."

Recently, researchers have proposed many methods for identifying dynamic communities [5]. All such methods rely on the notion of gradual change of community membership and some notion of persistence [2]. This line of research began with several methods that *track*, rather than *infer,* communities over time [1, 4, 29]. Palla *et al*. [23] use a very local and restrictive approach of finding overlapping cliques of predefined size in consecutive time steps. Falkowski *et al*. [9, 10] find static communities in each snapshot using Girvan-Newman algorithm [16] and then contract those into a community graph and apply Girvan-Newman algorithm once more to find dynamic communities. Together with Kempe, we represented dynamic communities using a social cost model, based on which we formulated a optimization problem [33], *assuming that partitions*[1] *of individuals into groups are given in all time steps*, and presented heuristics and approximation algorithms [32]. This approach is similar to that of Falkowski *et al*. in the sense that we apply any algorithm for inferring *static* communities to obtain partitions in groups. Then, apply our algorithms to obtain dynamic communities. We call this the *two-step* approach.

In this paper, we extend our previous model [5, 33] to arbitrary dynamic networks, getting rid of the assumption that the partitions in all time steps are given. Then, we formulate an optimization problem by generalizing the Correlation Clustering [3] problem to incorporate the temporal dimension. We then apprixmately optimize this using semidefinite programming relaxation and a rounding heursitic. We show that, while the two-step approach is computationally faster, the direct optimization method is more accurate.

A. Other Related Work

All of the above mentioned methods for inferring dynamic communities use graph clustering as an underlying concept. The appealing aspect of our social costs model is its roots

Work supported in part by NSF grants IIS-0705822 and CAREER IIS-0747369

[1]A partition of a set U is a collection of pairwise disjoint subsets of U whose union is U.

1550-4786/11 $26.00 © 2011 IEEE
DOI 10.1109/ICDM.2011.67

in the social sciences view of group dynamics [5, 24]. There are, however, several additional methods for inferring dynamic communities that we have not mentioned. They take very different approaches, for example, maximizing compression [30] and fitting a statistical communities model [15, 21, 27, 34, 36]. See [5] for survey on this topic.

Traditionally, social network analysis stemmed from graph theory. So the representation of a social network is simply an undirected graph $G = (V, E)$. We call them static networks to emphasize the fact that they do not have the time dimension. For example, this includes a snapshot of dynamic network and multiple snapshots aggregated over some period of time. There exists a lot of work on finding communities in static network. Several recent surveys describe the multitude of algorithms for such problem [8, 11–13, 25].

Graph clustering and partitioning are related problems and are well studied areas in terms of approximation algorithms. Our optimization problem is similar to the MAX-k-CUT problem [14] and minimizing disagreements in CORRELATION CLUSTERING [3]. For the latter, Charikar *et al*. [7] gave an $O(\log n)$-approximation algorithm and showed that minimizing disagreements is APX-hard (a PTAS is unlikely).

II. PROBLEM FORMULATION

A. Preliminaries

We consider a dynamic social network to be a time series of discrete graphs. We assume discrete time steps $t = 1, \ldots, T$ and represent a dynamic social network as a sequence $\mathcal{G} = \langle G_1, \ldots, G_T \rangle$ of undirected graphs where all $G_t = (V, E_t)$ share the same vertex set V. We use $N = |V|$ to denote the number of individuals. For simplicity, we write uv for an (unordered) pair $\{u, v\}$. So the edges of the graph G_t are the pairs $uv \in E_T$. We call pairs $uv \notin E_t$ non-edges.

We use u and v to denote individuals and t and t' to denote time steps. We use individual-time pair (u, t) when we need to refer to individual u at time step t. We define a *community interpretation* as a coloring χ of individuals over time: $\chi : V \times \{1, \ldots, T\} \to \mathbb{N}$. Each color in Range$(\chi)$ represents a community. The affiliation of a node u at time t is given by the color $\chi(u, t)$. That is, (u, t) is in the same community as (v, t') if they have the same color: $\chi(u, t) = \chi(v, t')$. With respect to a community interpretation χ, we say that an edge $uv \in E_t$ is a *true positive* if $\chi(u, t) = \chi(v, t)$, otherwise uv is a *false positive*. Similarly, we say that a non-edge $uv \notin E_t$ is a *true negative* if $\chi(u, t) \neq \chi(v, t)$, otherwise uv is a *false negative*. In CORRELATION CLUSTERING terminology, the true positives and true negatives are called *agreements* and the false positives and false negatives are called *disagreements*.

B. Community Model

We assume the following behavioral model [5]:

Gradual changes: Individuals change community affiliation infrequently over time.

Reliable true positive: Members of the same community mostly interact with each other.

Negligible false positive: Members of different communities rarely interact with each other.

We translate these assumptions into social costs:

Switching cost: each individual u incurs C_{sw} when changing community affiliation: $\chi(u, t) \neq \chi(u, t + 1)$.

False negative cost: two individuals incurs C_{fn} when they belong to the same community but do not interact: $\chi(u, t) = \chi(v, t)$ and $uv \notin E_t$.

False positive cost: two individuals incurs C_{fp} when they belong to different communities but do interact: $\chi(u, t) \neq \chi(v, t)$ and $uv \in E_t$.

We define the NETWORK COMMUNITY INTERPRETATION optimization problem as finding a coloring χ that minimizes the total cost of switching, false negative, and false positive:

$$\begin{aligned} cost(\chi) = C_{\text{sw}} \sum_{t=1}^{T-1} |\,\{u \in V : \chi(u, t) \neq \chi(u, t + 1)\}\,| \\ + C_{\text{fn}} \sum_{t=1}^{T} |\,\{uv \notin E_t : \chi(u, t) = \chi(v, t)\}\,| \\ + C_{\text{fp}} \sum_{t=1}^{T} |\,\{uv \in E_t : \chi(u, t) \neq \chi(v, t)\}\,|. \end{aligned}$$

This problem is NP-hard by a straightforward reduction from the unweighted CORRELATION CLUSTERING problem.

III. ALGORITHM

Our algorithm follows a standard approach introduced by Goemans and Williamson [17] who used semidefinite programming (SDP) to derive an approximation algorithm for MAX-CUT. Our SDP formulation is similar to Swamy's formulation of CORRELATION CLUSTERING [31]. We first formulate a vector program and then relax it an SDP. We solve the SDP using CSDP [6], an implementation of the primal-dual interior point method. The resulting solution is fractional and needs to be rounded to an integral solution.

A. Vector Program Formulation

We have a vector $x_{u,t}$ representing the community membership of u at time t. Let $e_1, \ldots, e_{NT}$ be the standard basis for $\mathbb{R}^{NT}$. That is, e_i has 1 on the i^{th} coordinate and 0 elsewhere. So $e_i \cdot e_j = 1$ if and only if $i = j$. We reformulate NETWORK COMMUNITY INTERPRETATION as

$$\begin{aligned} \text{minimize:} \quad & C_{\text{sw}} \sum_{t=1}^{T-1} \sum_{u \in V} (1 - x_{u,t} \cdot x_{u,t+1}) \\ & + C_{\text{fn}} \sum_{t=1}^{T} \sum_{uv \notin E_t} x_{u,t} \cdot x_{v,t} \\ & + C_{\text{fp}} \sum_{t=1}^{T} \sum_{uv \in E_t} (1 - x_{u,t} \cdot x_{v,t}) \end{aligned}$$

$$\text{subject to:} \quad x_{u,t} \in \{e_1, \ldots, e_{NT}\} \qquad \forall u, t$$

The constraints $x_{u,t} \in \{e_1, \ldots, e_{NT}\}$ ensure that we can interpret the solution vectors $x_{u,t}$ as an equivalence relation: (u, t) and (v, t') belong to the same community if and only if $x_{u,t} \cdot x_{v,t'} = 1$. Since CSDP takes a maximization problem, we rewrite the objective function as follows, dropping constant terms.

$$\begin{aligned}
\text{maximize:} \quad & C_{\text{sw}} \sum_{t=1}^{T-1} \sum_{u \in V} x_{u,t} \cdot x_{u,t+1} \\
& - C_{\text{fn}} \sum_{t=1}^{T} \sum_{uv \notin E_t} x_{u,t} \cdot x_{v,t} \\
& + C_{\text{fp}} \sum_{t=1}^{T} \sum_{uv \in E_t} x_{u,t} \cdot x_{v,t} \\
\text{subject to:} \quad & x_{u,t} \in \{e_1, \ldots, e_{NT}\} \qquad \forall u, t
\end{aligned}$$

Then, we relax the constraints $x_{u,t} \in \{e_1, \ldots, e_{NT}\}$ by replacing them with:

- $x_{u,t} \cdot x_{u,t} = 1$ for all u, t.
- $x_{u,t} \cdot x_{v,t'} \geq 0$ for all u, t, v, t' such that $u \neq v$ or $t \neq t'$.

Note that, because of the symmetry, we actually need only half of the non-negative constraints. We keep the redundant constraints for brevity and clarity. As a consequence, all dot products take values between 0 and 1. This results in the following vector program,

$$\begin{aligned}
\text{maximize:} \quad & \text{(same as above)} \\
\text{subject to:} \quad & x_{u,t} \cdot x_{u,t} = 1 \quad \forall u, t \\
& x_{u,t} \cdot x_{v,t'} \geq 0 \ \forall u, t, v, t' s.t.\ u \neq v \text{ or } t \neq t' \\
& x_{u,t} \in \mathbb{R}^{NT} \quad \forall u, t
\end{aligned}$$

All feasible solutions to the integral program are still feasible solutions to the relaxed program. So the optimal value of the relaxed program is at least that of the integral program. This fractional optimal value will allow us to compute an upper bound on the original objective function in the minimization problem.

In this relaxed vector program, the vector variables appear only in the form of dot products. Thus, the program has an equivalent formulation as a positive semidefinite program. We replace each dot product $x_{u,t} \cdot x_{v,t'}$ with a new variable $y_{(u,t),(v,t')}$. We put the new variables in a matrix $Y = [y_{(u,t),(v,t')}]$ whose rows are indexed by (u, t) and whose columns are indexed by (v, t'). The matrix Y is positive semidefinite (denoted $Y \succeq 0$) because $Y = W^T W$ where W is the matrix whose columns are $x_{u,t}$. So we have the following semidefinite program,

$$\begin{aligned}
\text{maximize:} \quad & C_{\text{sw}} \sum_{t=1}^{T-1} \sum_{u \in V} y_{(u,t),(v,t+1)} \\
& - C_{\text{fn}} \sum_{t=1}^{T} \sum_{uv \notin E_t} y_{(u,t),(v,t)} \\
& + C_{\text{fp}} \sum_{t=1}^{T} \sum_{uv \in E_t} y_{(u,t),(v,t)} \\
\text{subject to:} \quad & y_{(u,t),(u,t)} = 1 \quad \forall u, t \\
& y_{(u,t),(v,t')} \geq 0 \quad \forall u, t, v, t' s.t.\ u \neq v \text{ or } t \neq t' \\
& [y_{(u,t),(v,t')}] \succeq 0
\end{aligned}$$

This program can be numerically solved to any arbitrary precision by the interior-point method [18] in time polynomial in the size of the program and the required precision. This SDP is still impractical to solve for large networks, one reason being that there are many inequality constraints: $y_{(u,t),(v,t')} \geq 0$. Each inequality constrain introduces a slack variable, resulting in a huge symmetric matrix Y of dimension $NT + \binom{NT}{2}$. To reduce the size of the vector program, we relax this SDP further by eliminating the constraints $y_{(u,t),(v,t')} \geq 0$ for those terms $y_{(u,t),(v,t')}$ which do not appear in the objective function. The result is still a relaxation of the maximization vector program. Thus, the optimal value of this final SDP is still an upper bound on the optimal value of the maximization vector program.

B. Rounding Heuristic

Once we found an optimal solution $[y_{(u,t),(v,t')}]$ to the SDP relaxation, we perform single-linkage clustering [19] which is essentially Kruskal's algorithm for finding a minimum spanning tree. For the similarity function we use those variables $y_{(u,t),(v,t')}$ that appear in the objective function, ignoring the rest. Initially, each (u, t) is in its own cluster. Then, we consider (u, t, v, t') in non-decreasing order of $y_{(u,t),(v,t')}$, breaking ties arbitrarily. In this order, we replace the clusters of (u, t) and (v, t') with their union, maintaining the clusters using the Union-Find algorithm. At each iteration, we compute the cost of the community interpretation, keeping as a solution the one with the smallest cost, breaking ties by taking a community interpretation with fewer communities (more parsimonious).

Once we produce a community interpretation χ, we upper bound its approximation ratio as follows. Suppose OPT_{SDP} is the optimal value of the SDP relaxation and OPT is the cost of an optimal community interpretation. As argued earlier, OPT_{SDP} is an upper bound on the optimal solution to (maximixation) integral vector program and gives a *lower bound* L on OPT: $L := C_{\text{sw}}(T-1)N + C_{\text{fp}} \sum_{t=1}^{T} |E_t| - \text{OPT}_{\text{SDP}}$. This in turn gives an upper bound on the approximation ratio: $\frac{cost(\chi)}{\text{OPT}} \leq \frac{cost(\chi)}{L}$.

IV. Experimental Results

We now present the results of our algorithm on several synthetic and real datasets. First, we generate synthetic datasets with known community structures to test the accuracy of our method and to compare it to alternative

approaches. We then apply the algorithms to real dynamic networks. Although our algorithm take three parameters C_{sw}, C_{fn}, C_{fp}, it is the ratios between them that affect the optimal community structure since scaling the parameters by any positive constant does not change the optimal solution.

A. Generating Synthetic Test Data

Due to space limitation, we briefly describe our synthetic data generator. As a hidden Markov model, it first randomly assigns each individual at time $t = 1$ to one of the k communities. For each time step $t \geq 1$, it generates an observed network G_t based on the probability p_{fn} of false negatives and the probability p_{fp} of false positives. Then, it reassigns community affiliation for the next time step $t+1$ based on the probability p_{sw} of switching.

B. SDP Results

The first question is: If we know the true community structure, how close to the truth is the community structure produced by the SDP method? To answer this, we generate dynamic networks using different p_{sw}, p_{fn}, p_{fp}, and run the SDP method using different C_{sw}, C_{fn}, C_{fp}. Since the results are similar, show only one representative case $p_{\text{sw}} = 0.3$. In Figure 1, the left column shows the distance (Rand index [26]) to the ground truth. In the top row, the SDP method can find the true structure (distance=0), given the right parameter setting. In the other rows, it finds solutions close to the truth, but not identical. This is because of the high perturbation. However, the relationship between ratios $C_{\text{fn}}/C_{\text{fp}}$ and $p_{\text{fn}}/p_{\text{fp}}$ is clear. We observe how the probability ratio $p_{\text{fn}}/p_{\text{fp}}$ (which varies from the top to the bottom rows) affects the cost ratio $C_{\text{fn}}/C_{\text{fp}}$ at which the SDP method achieves the minimum distance to the ground truth. This conforms with the intuition that when $C_{\text{fn}} < C_{\text{fp}}$, the algorithm trades false positives for false negatives (since $p_{\text{fn}} > p_{\text{fn}}$). The right column of Figure 1 shows the upper bound on the approximation ratio of the SDP algorithm to the optimal solution (as discussed in Section III-B). Note that in some cases, this upper bound is exactly 1, indicating that the SDP method has found an optimal solution. However, the distance to the ground truth (in the corresponding plots in the left column) is greater than 0, indicating that the optimal structures found under those parameter settings do not match the ground truth.

C. Comparison with other methods

We compared our SDP algorithm with other competitive methods for detecting dynamic communities using the two-step approach. First, we ran methods for detecting static communities on each snapshot graph to find a group structure. We use Girvan-Newman (GN) and Clauset-Newman-Moore (CNM), in particular, the implementations from SNAP [20]. Once we discovered the group structure in each time step, we ran our branch-and-bound algorithm [33]

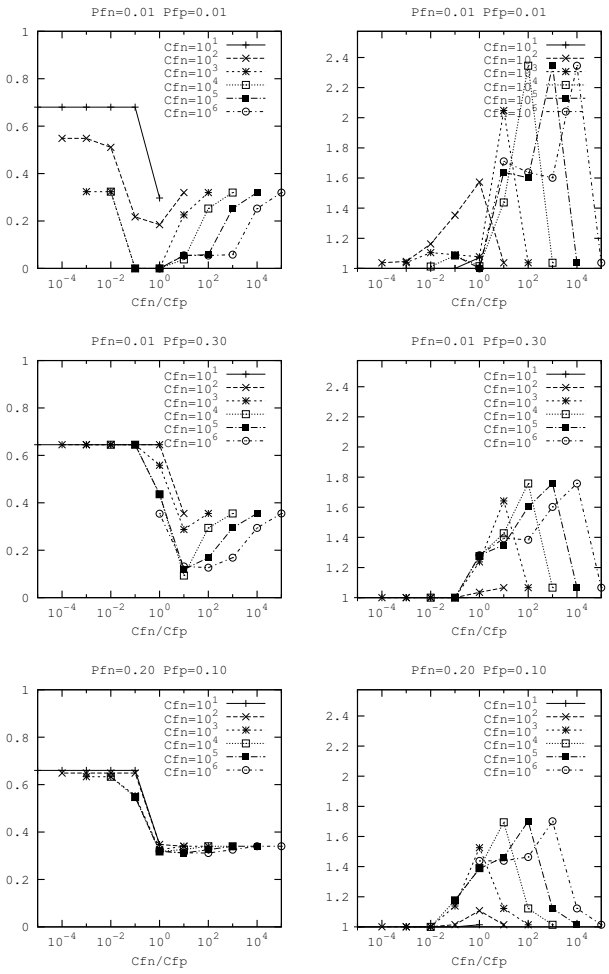


Figure 1. Rand index distance to ground truth (left column) and the upper bound on the approximation ratio (right column) of the SDP algorithm for $p_{\text{sw}} = 0.30, N = 15, T = 10, k = 3$.

to find an optimal community interpretation.[2] We vary the parameter settings within a certain range. Although this performs an exhaustive search, our test data sets are sufficiently small and the branch-and-bound algorithm can find the optimal solution within one minute in most cases with only few cases that take longer.

We generated 10 dynamic networks for each generator parameter values $p_{\text{sw}}, p_{\text{fn}}, p_{\text{fp}}$. For each trial, we ran each method over different parameter settings and take the one that is closest to the ground truth. Figure 2 shows this result. Overall, SDP method outperforms the two other methods.

D. Experiments on Real Data Set

We have done experiments on two Bluetooth traces from the Haggle project. The results on both traces are similar so we present only one of them [28]. We consider only the

[2]This is available at http://compbio.cs.uic.edu/~chayant/commdy/

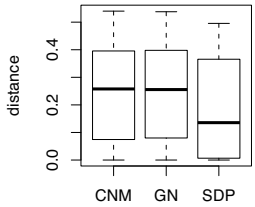
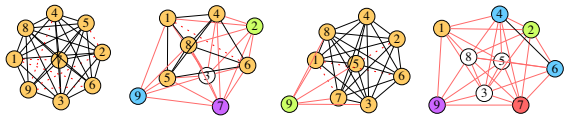


Figure 2. The distance to the ground truth of three algorithms with the best parameter settings, over 10 trials of synthetic data sets with $N = 10, T = 10, k = 2$.

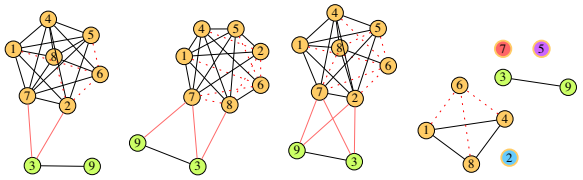
Bluetooth devices in the project and ignore the other devices since we do not have their complete traces. For each one hour interval, we create an undirected edge (u, v) at the corresponding time step if u has seen v in this interval. This result in a dynamic network with one has 9 nodes, 467 edges, 84 time steps. The number of time steps is still too large for the SDP method to analyze in its entirety, so we split the timeline into four smaller chunks, each is ≈ 20 time steps. Figure 3 shows the network at time step 0 with nodes color-coded by community affiliation detected by the SDP method. Black solid lines are true positives. Red solid lines are false positives. Red dotted lines are false negatives. As can be seen in the figure, the higher the ratio $C_{\text{fn}}/C_{\text{fp}}$ is, the fewer the false negatives and the more the false positives there are. The fewer the false negatives decreases there are, the smaller and denser the clusters become.



$(5, 1, 5)$ $(5, 5, 1)$ $(5, 5, 5)$ $(5, 10, 1)$

Figure 3. Community structures in the Haggle at time step 0 detected under various $(C_{\text{sw}}, C_{\text{fn}}, C_{\text{fp}})$.

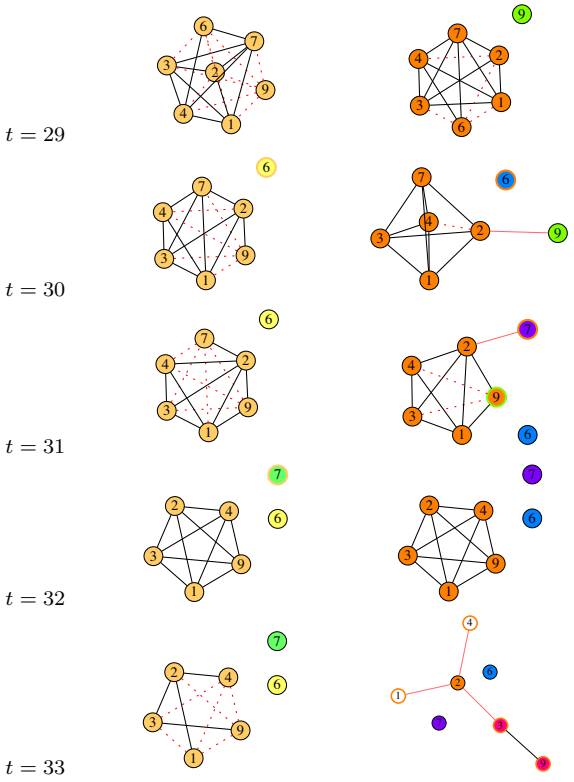
Next we show the community structure in the first few time steps detected with $(C_{\text{sw}}, C_{\text{fn}}, C_{\text{fp}}) = (5, 5, 5)$, which seems to be a good setting. Figure 4 shows that there is cluster $\{1, 2, 4\text{–}8\}$ which is quite stable over time steps 4–6.



$t = 4$ $t = 5$ $t = 6$ $t = 7$

Figure 4. Community structures in the Haggle data set with parameters $(C_{\text{sw}}, C_{\text{fn}}, C_{\text{fp}}) = (5, 5, 5)$.

Next, we vary $C_{\text{sw}} \in \{1, 5\}$ and fix $C_{\text{fn}} = 1$ and $C_{\text{fp}} = 5$. Figure 5 illustrates that that, when switching is cheap ($C_{\text{sw}} = 1$ v.s. $C_{\text{sw}} = 5$), clusters are more stable over time with few nodes changing affiliation. As can be seen in the first column at $t = 33$, the big cluster still exists even when most of its connections are gone. The expensive C_{sw} cost slows down the rate of affiliation changes.



$t = 29$ $t = 30$ $t = 31$ $t = 32$ $t = 33$

Figure 5. Community structures in Haggle data set at time steps 29–33. The first column is detected with $(C_{\text{sw}}, C_{\text{fn}}, C_{\text{fp}}) = (5, 1, 5)$ and the second column is detected with $(C_{\text{sw}}, C_{\text{fn}}, C_{\text{fp}}) = (1, 1, 5)$.

V. Conclusion

We extend our previous work on finding communities in dynamic social networks [5] to arbitrary networks, getting rid of the assumption that the individuals in each time step are partitioned into groups. To the best of our knowledge, our method is the first that is based on social theory. We formulate the Network Community Interpretation problem and devise an approximation algorithm via SDP relaxation and a heuristic rounding scheme. The algorithm produces a community interpretation with an approximation guarantee. We show empirically that the method produces solutions which are near optimal and close to the ground truth of the synthetic data. Our method outperforms the two-step approach. We also show how the SDP method can be used to find dynamic communities in a real dataset. A natural question one may have is: How good is the theoretical approximation guarantee? Another future direction is the available SDP solvers are not scalable and speeding up SDP solvers is an active area of research. Thus, speeding up our algorithm will need a new angle.

REFERENCES

[1] C. C. Aggarwal and P. S. Yu. Online analysis of community evolution in data streams. In *SIAM ICDM*, 2005.

[2] L. Backstrom, D. Huttenlocher, J. Kleinberg, and X. Lan. Group formation in large social networks: Membership, growth, and evolution. In *ACM KDD*, 2006.

[3] N. Bansal, A. Blum, and S. Chawla. Correlation clustering. *Machine Learning*, 56, 2004.

[4] T. Y. Berger-Wolf and J. Saia. A framework for analysis of dynamic social networks. In *ACM KDD*, 2006.

[5] T. Y. Berger-Wolf, C. Tantipathananandh, and D. Kempe. Community identification in dynamic social networks. In C. F. Philip S. Yu, Jiawei Han, editor, *Link Mining: Models, Algorithms and Applications*, chapter 8. Springer, 2010.

[6] B. Borchers and J. Young. Implementation of a primal–dual method for sdp on a shared memory parallel architecture. *Computational Optimization and Applications*, 37, 2007.

[7] M. Charikar, V. Guruswami, and A. Wirth. Clustering with qualitative information. In *IEEE FOCS*, 2003.

[8] L. Danon, A. Díaz-Guilera, J. Duch, and A. Arenas. Comparing community structure identification. *J. of Statistical Mechanics: Theory and Experiment*, 9008, 2005.

[9] T. Falkowski. *Community Analysis in Dynamic Social Networks*. Dissertation, University Magdeburg, 2009.

[10] T. Falkowski, J. Bartelheimer, and M. Spiliopoulou. Mining and visualizing the evolution of subgroups in social networks. *Web Intelligence*, 2006.

[11] S. Fortunato. Community detection in graphs. *Physics Reports*, 486, February 2010.

[12] S. Fortunato and C. Castellano. Community structure in graphs. *eprint arXiv*, 2007.

[13] L. Freeman. Finding social groups: A meta-analysis of the southern women data. In R. Breiger, K. Carley, and P. Pattison, editors, *Dynamic Social Network Modeling and Analysis*. The National Academies Press, 2003.

[14] A. M. Frieze and M. Jerrum. Improved approximation algorithms for max k-cut and max bisection. In *IPCO*, 1995.

[15] W. Fu, L. Song, and E. P. Xing. Dynamic mixed membership blockmodel for evolving networks. In *ICML*, 2009.

[16] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *PNAS*, 99, 2002.

[17] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42, Nov 1995.

[18] C. Helmberg, F. Rendl, R. J. Vanderbei, and H. Wolkowicz. An interior-point method for semidefinite programming. *SIAM J. on Optimization*, 6(2), 1996.

[19] A. K. Jain and R. C. Dubes. *Algorithms For Clustering Data*. Prentice Hall, 1988.

[20] J. Leskovec. Stanford network analysis platform (SNAP), Oct 2010. http://snap.stanford.edu/.

[21] Y.-R. Lin, Y. Chi, S. Zhu, H. Sundaram, and B. L. Tseng. Facetnet: a framework for analyzing communities and their evolutions in dynamic networks. In *WWW*, 2008.

[22] M. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys. Rev. E*, 69, 2004.

[23] G. Palla, I. Dernyi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435, 9 June 2005.

[24] M. Pearson and P. West. Drifting smoke rings: Social network analysis and markov processes in a longitudinal study of friendship groups and risk-taking. *Connections*, 25(2), 2003.

[25] M. A. Porter, J.-P. Onnela, and P. J. Mucha. Communities in networks. *arXiv preprint*, Feb 2009.

[26] W. M. Rand. Objective Criteria for the Evaluation of Clustering Methods. *J. Am Stat. Assoc.*, 66(336), 1971.

[27] P. Sarkar and A. Moore. Dynamic social network analysis using latent space models. *ACM SIGKDD Explorations Newsletter*, 7(2), December 2005.

[28] J. Scott, R. Gass, J. Crowcroft, P. Hui, C. Diot, and A. Chaintreau. CRAWDAD haggle/imote/intel (2006-01-31).

[29] M. Spiliopoulou, I. Ntoutsi, Y. Theodoridis, and R. Schult. MONIC: modeling and monitoring cluster transitions. In *ACM KDD*, 2006.

[30] J. Sun, C. Faloutsos, S. Papadimitriou, and P. S. Yu. Graphscope: parameter-free mining of large time-evolving graphs. In *ACM KDD*, 2007.

[31] C. Swamy. Correlation clustering: maximizing agreements via semidefinite programming. In *ACM-SIAM SODA*, 2004.

[32] C. Tantipathananandh and T. Berger-Wolf. Constant-factor approximation algorithms for identifying dynamic communities. In *ACM KDD*, 2009.

[33] C. Tantipathananandh, T. Berger-Wolf, and D. Kempe. A framework for community identification in dynamic social networks. In *ACM KDD*, 2007.

[34] H. Tong, S. Papadimitriou, J. Sun, P. S. Yu, and C. Faloutsos. Colibri: fast mining of large static and dynamic graphs. In *ACM KDD*, 2008.

[35] S. Wasserman and F. K. *Social Network Analysis.* Cambridge University Press, 1994.

[36] T. Yang, Y. Chi, S. Zhu, Y. Gong, and R. Jin. A bayesian approach toward finding communities and their evolutions in dynamic social networks. In *SIAM ICDM*, February 2009.