Physica A xx (xxxx) xxx–xxx

Contents lists available at ScienceDirect

Physica A

journal homepage: www.elsevier.com/locate/physa

A fast algorithm for community detection in temporal network

Q1 Jialin He, Duanbing Chen [*]

Web Sciences Center, University of Electronic Science and Technology of China, Chengdu 611731, People's Republic of China
Big Data Research Center, University of Electronic Science and Technology of China, Chengdu 611731, People's Republic of China

HIGHLIGHTS

- Our method takes advantage of community information at previous time step.
- Constructing a small network before detection at current time step.
- The CPU running time of our method improves as much as 69% over traditional one.
- Our method can achieve high quality of communities for slowly evolving networks.

ARTICLE INFO

Article history:
Received 5 October 2014
Received in revised form 20 December 2014
Available online xxxx

Keywords:
Complex systems
Temporal networks
Dynamic community
Community structure
Detection algorithm

ABSTRACT

Many complex systems can be investigated using the framework of temporal networks, which consist of nodes and edges that vary in time. The community structure in temporal network contributes to the understanding of evolving process of entities in complex system. The traditional method on dynamic community detecting for each time step is independent of that for other time steps. It has low efficiency for ignoring historic community information. In this paper, we present a fast algorithm for dynamic community detecting in temporal network, which takes advantage of community information at previous time step and improves efficiency while maintaining the quality. Experimental studies on real and synthetic temporal networks show that the CPU running time of our method improves as much as 69% over traditional one.

1. Introduction

Many complex systems can be represented as networks, sets of nodes joined in pairs by edges. Examples include co-authorship networks [1], in which the nodes are authors and the edges are co-authorships between authors, the blogosphere networks [2], in which the nodes are bloggers and the edges are hyperlinks in blogs, and social networks [3], in which the nodes are people and the edges represent social interactions. Community structure is a network characteristic describing the propensity of groups of nodes to form denser connections within groups than across. In a social network, a community might correspond to an actual community in real world, a group of people brought together by common interest, common location or workplace, or family ties [4].

When analyzing a social temporal network, researchers always treat it as a static network, which is derived from aggregation of data over all time [5]. In the past decades, many community detecting methods, such as modularity-based

[*] Corresponding author at: Web Sciences Center, University of Electronic Science and Technology of China, Chengdu 611731, People's Republic of China.
E-mail address: dbchen@uestc.edu.cn (D. Chen).

http://dx.doi.org/10.1016/j.physa.2015.02.069
0378-4371/© 2015 Published by Elsevier B.V.

methods [6–8], dynamic methods [9,10] and link community method [11], have been proposed based on static network. However, by discarding temporal information, the evolution pattern of community structures is lost [12].

During the past few years, there has been a growing body of work on the dynamic community detection in temporal networks. Greene et al. [12] described a model for tracking the progress of communities over time in a temporal network, where each community is characterized by a series of significant evolutionary events. Asur et al. [13] presented an event-based characterization of critical behavioral patterns for temporally varying interaction graphs. They used non-overlapping snapshots of interaction graphs and developed a framework for capturing and identifying interesting events which are used to characterize complex behavioral patterns of individuals and communities over time. Ning et al. [14] extended the standard spectral clustering to such evolving data by introducing the incidence vector/matrix to represent two kinds of dynamics in the same framework and by incrementally updating the eigenvalue system. Their incremental algorithm initialized by a standard spectral clustering, continuously and efficiently updates the eigenvalue system and generates instant cluster labels, as the data set is evolving. Palla et al. [15] developed a new algorithm based on clique percolation, that allows, for the first time, to investigate the time dependence of overlapping communities on a large scale and as such, to uncover basic relationships characterizing community evolution. Xu et al. [16] presented a state-space model for temporal networks that extends the well-known stochastic block model for static networks to the dynamic setting. An extended Kalman filter (EKF) augmented with a local search is used to fit the model in a near-optimal manner. Ilija Suba et al. [17] proposed a story tracking method based on the dynamics of keyword-association graphs. They created a graph representation of the story evolution called story graphs, and investigate how graph structure can be used for detecting and discovering new developments in the story. Konstantinos et al. [18] proposed an efficient methodology for performing event detection from large time-stamped web document streams. The methodology successfully integrates named entity recognition, dynamic topic map discovery, topic clustering, and peak detection techniques. All these studies, however, have a common weak point—when communities are extracted at a given time step, historic community structures, which contain valuable information related to current community structures, are not taken into account.

There are also some community detection methods in temporal network which take advantage of historic community information. Nguyen et al. [19] presented an adaptive method which updates and discovers the new community structures based on its history together with the network changes. However, by treating network changes as a collection of simple events, the method must be computed for each simple event, which leads to low efficiency. Bassett et al. [20] analyzed the behavior of several null models used for optimizing quality functions in multilayer networks. That research mainly emphasizes that the appropriateness of different types of null-model networks depends on the network structures and the construction of representative partitions. Lin et al. [5] proposed a novel framework for analyzing communities and their evolutions through a robust unified process. In the framework, the historic evolution patterns are used to regularize current community structures so that they are less likely to deviate too dramatically from the most recent community structures. A limit of the framework is that when the number of communities is a variable at each time step, the method might be not effective on real networks.

In this paper, we present a quite simple but very efficient detection algorithm of dynamic community in temporal network. For each time step, our method first constructs a small network by advantage of community structures at previous time step and then detects communities in the new network. Experimental studies on real and synthetic temporal networks show that our method improves as much as 69% CPU running time and maintains the quality of communities compared with traditional one.

2. Dynamic community method

In this section, we first introduce Blondel method which is used to detect communities at each time step and then describe our dynamic community detection method.

2.1. Blondel algorithm

Blondel et al. [21] introduce a fast greedy approach based on modularity optimization for weighted network. The algorithm is divided into two phases that are repeated iteratively. Initially, each node in network is formed a community. Then, for each node i, one considers the neighbors j of i and computes the gain of modularity by putting i into the community which contains node j temporarily. The node i is then placed in the community of its neighbor that yields the largest increase of modularity, as long as it is positive. In the second phase, each community is considered as a new node, and the weights of the edges between new nodes are given by the sum of the weights of the edges between original nodes in the corresponding two communities. The two phases are repeated until there is no more improvement and maximal modularity is achieved. The computational time grows like $O(m)$, so the algorithm is extremely fast.

2.2. Constructing small network

In the past, many methods independently detect dynamic communities at each time step. By ignoring historic community information, the traditional dynamic community methods lead to redundant computations in large networks [22]. The

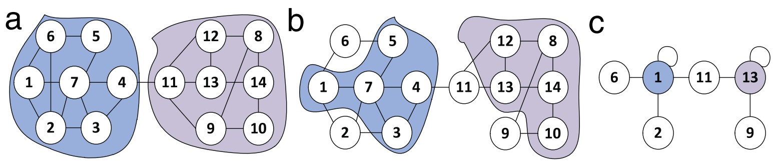

Fig. 1. A toy network and its corresponding reconstructed network. (a) Two communities at time step $t-1$, (b) a node still stays in the community at time step $t-1$ if its connections keep unchanged at time step t, (c) a small network constructed at time step t.

method proposed in this paper takes into account previous community structures and works only on the network changes, thus reduces computational cost significantly.

For two consecutive time steps, there are only small changes in the number of edges that do not affect the community structures dramatically. Because of this, we argue that if the connections of all the nodes in the same community at time step $t-1$ keep unchanged at time step t, they are still in the same community at time step t. We use a toy network with 14 nodes and 2 communities to illustrate our method, as shown in Fig. 1. Fig. 1(a) shows two communities labeled as 1–7 and 8–14 respectively at time step $t-1$. From Fig. 1(b), it can be seen that each node's connections with others at time step t are the same as time step $t-1$ except nodes 2, 6, 9 and 11, so these nodes (1, 3, 4, 5, 7 and 8, 10, 12, 13, 14) are still in the same community.

In order to reduce redundant computations, we first construct a small network by advantage of previous community structures and then detect current communities in the new network. If a node's connections change (adding edges or removing edges) at current time step, this single node is still taken as a node in the new network. If the connections of all the nodes in the same community at previous time step keep unchanged at current time step, they are regarded as a node with self-loops in the new network. The weight of an edge between two new nodes is given by the sum of the weights of the edges between original nodes in the corresponding two communities. Each node's degree in the new network is the sum of all nodes' degrees in its corresponding community. As shown in Fig. 1(c), because of changes in node's connections, nodes 2, 6, 9 and 11 are directly constructed as 4 nodes in the new network. These nodes in two communities (1, 3, 4, 5, 7 and 8, 10, 12, 13, 14) keep their connections unchanged and are constructed as two new nodes labeled as 1 and 13 respectively.

After constructing a small network at current time step, we use Blondel algorithm to detect communities in the new network. The dynamic community detection algorithm is described as follows:

1. The community structures of the network G_1 at time step 1 are initialized by using the Blondel algorithm.
2. For network G_t at time steps $t = 2, 3, \ldots, T$
   (a) Constructing a small network G_{t_new} according to the network structure in G_t and the community information in G_{t-1}.
   (b) Detecting communities in G_{t_new} by using Blondel algorithm.

By using Blondel method, both our method and traditional method have same time complexity $O(m)$ at each time step, where m is the number of edges in network. Because of constructing a small network, the computing time of our method mainly consumed in network reconstructing. However, For the traditional method, most of the computing time is taken in the first iteration, which is divided into two phases including modularity optimization and network reconstruction. Compared with traditional method, our method saves the time of modularity optimization. So the coefficient of time complexity for our method is smaller than that for traditional one.

3. Experimental results

In this section, we use real and synthetic temporal networks to evaluate the performance of our method.

3.1. Data description

ArXiv hep-ph network [23] is an author collaboration network from the arXiv High Energy Physics-Phenomenology (hep-ph) section. An edge between two authors represents a common publication. Timestamp denotes the date of a publication. The network has 28,093 nodes and 4,596,803 edges, spanning from year 1992 to 2002.

Facebook network [24] contains friendship data of users. A node represents a user and an edge represents a friendship between two users. The network has 63,731 nodes and 817,035 edges, spanning from year 1970 to 2009.

Enron email network [25] consists of 1,148,072 emails between 1999 and 2003. Nodes in the network are employees and edges are emails. It is possible to send an email to oneself, and thus this network contains self-loops.

Wikipedia conflict network [26] has 118,100 nodes and 2,917,785 edges, spanning from year 2001 to 2009. A node represents a user and an edge represents a conflict between two users, with the edge sign representing a positive or negative interaction.

We first clean four temporal networks by removing self-loops and outlier data. Then the time-scale is converted from millisecond to day. Four cleaned temporal networks are seen as Table 1. In the experiment, we partition each temporal network into 31 snapshots according to Moore visualization method [27]. Each snapshot is derived from aggregation of data over a time interval. In order to make comparable changes in the number of edges between two consecutive snapshots, the

Table 1
Four cleaned temporal networks in experiment.

Network	Number of nodes	Number of edges	Duration
Conflict	116 836	2027 871	2001.10–2009.03
Hepph	16 959	1194 440	1992.07–1999.12
Facebook	61 096	614 797	2006.09–2009.01
Enron	86 836	296 952	1998.01–2004.02

Table 2
The cumulative CPU running time(s) at time step 31 for four temporal networks evolving by 1 day, 5 days, 20 days and 25 days.

Network	Traditional method				Our method			
	1	5	20	25	1	5	20	25
Conflict	582.97	532.47	487.86	459.72	243.26	251.64	249.53	230.02
Hepph	290.21	276.80	201.24	192.41	177.33	164.17	126.03	116.22
Facebook	203.85	195.13	95.02	64.21	66.35	60.05	28.86	19.98
Enron	112.31	111.27	76.41	66.26	60.41	61.32	52.83	46.06

Table 3
The speedup ratios for four temporal networks evolving by 1 day, 5 days, 20 days and 25 days.

	1	5	20	25
Conflict	0.58	0.53	0.48	0.49
Hepph	0.38	0.40	0.37	0.39
Facebook	0.67	0.69	0.69	0.68
Enron	0.46	0.44	0.30	0.30

"evolving speed" is set by 1 day, 5 days, 20 days and 25 days. Because the number of snapshots is fixed, the time interval becomes small when the "evolving speed" increases. Taking facebook friendship network for example, it has 871 days in total and is equally partitioned into 31 time interval with the "evolving speed" set by 1 day (1–841, 2–842, ..., 31–871) or 5 days (1–721, 6–726, ..., 151–871), then the data of each time interval is represented as a snapshot.

3.2. *Efficiency analysis*

The method proposed in the paper and the traditional one have been compiled and tested on the laptop with a core i7-4702 MQ and 8 GB memory. In the implementation, two methods share the same Blondel algorithm. The only difference is that our method constructs a small network before using Blondel algorithm. The cumulative CPU running time at time step t for two methods is computed by Eqs. (1) and (2) respectively,

$$CT^t_{traditional} = \tau_{read} + \sum_{t=1}^{T}(\tau^t_{cut} + \tau^t_{traditional}) \tag{1}$$

$$CT^t_{our} = \tau_{read} + \sum_{t=1}^{T}(\tau^t_{cut} + \tau^t_{construct} + \tau^t_{our}) \tag{2}$$

where τ_{read} is the time reading a temporal network, τ^t_{cut} is the time partitioning the snapshot at time step t, $\tau^t_{traditional}$ is the computing time at time step t for traditional method and τ^t_{our} for our method, $\tau^t_{construct}$ is the time constructing a small network at time step t.

Q2 From Fig. 2, it can be seen that our method is always faster than traditional one for the cumulative CPU running time (s) at all time steps except the first time step. Tables 2 and 3 show the total cumulative CPU running time and corresponding speedup ratios respectively for four temporal networks evolving by 1 day, 5 days, 20 days and 25 days. The speedup ratios are defined by $(CT^{31}_{traditional} - CT^{31}_{our})/CT^{31}_{traditional}$. As shown in Table 3, our method speeds up significantly compared with traditional one in four temporal networks. One should note that the speedup ratios are related to network structure. For example, the speedup ratio improves as much as 69% for facebook, but only 40% for hepph. Meanwhile, for the same temporal network, different evolving speeds also have different speedup ratios such as 46% and 30% for enron evolving by 1 day (very static) and 25 days (very dynamic) respectively. However, it is not always true, for example, for facebook, the speedup ratios almost equal for all evolving speeds. Besides, we evaluate the performance of our method on different community structures and different number of communities respectively.

First, four specific networks generated by Girvan–Newman method [28] are used to test our method on different community structures. At step time 1, we set the number of communities to 50, the number of nodes in a community to 100 and fix the average degree of a node to 90 for four synthetic networks. As seen from Table 4, the only difference for four networks

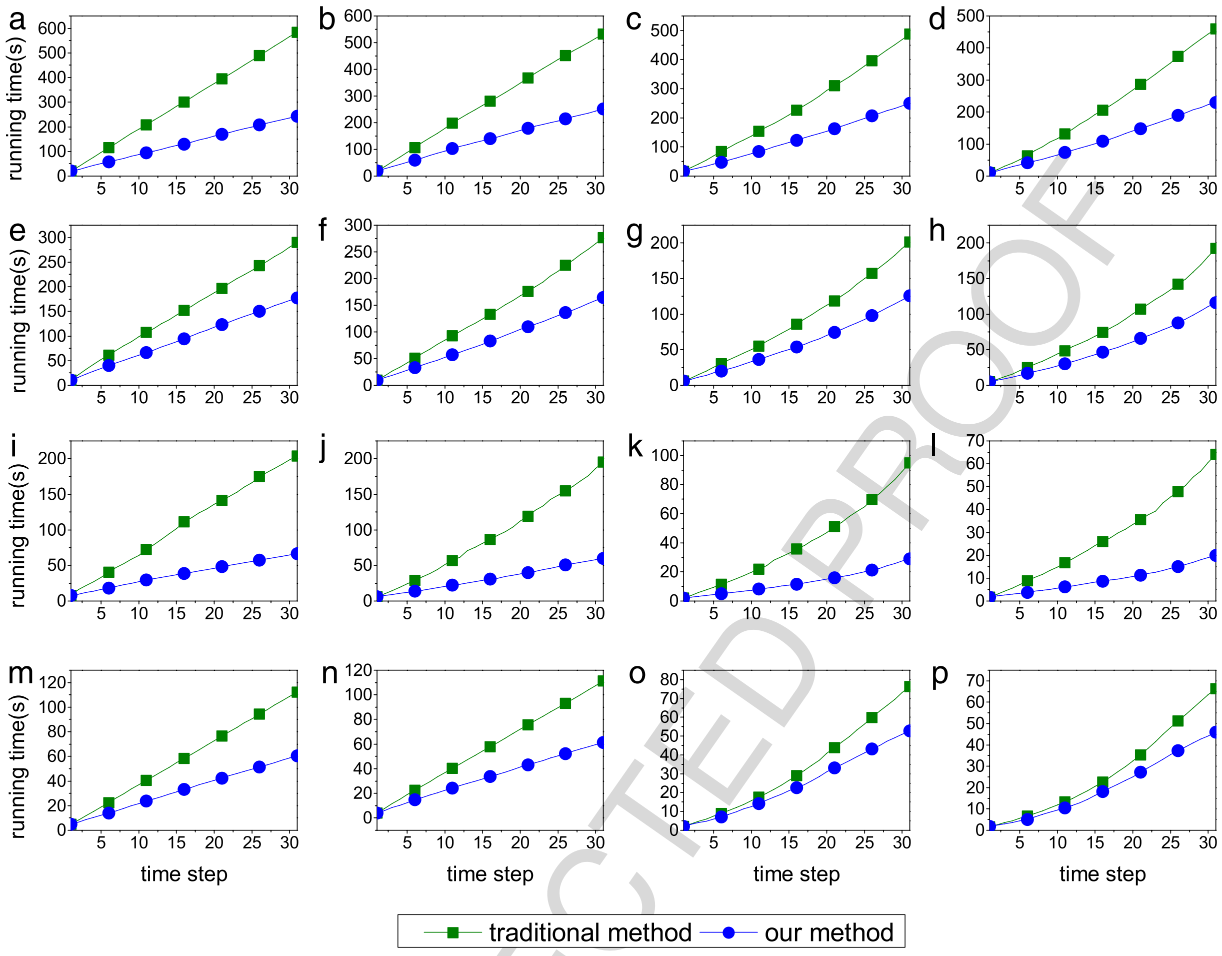

Fig. 2. The cumulative CPU running time(s) on four temporal networks with evolving by 1 day, 5 days, 20 days and 25 days respectively. (a)–(d) Conflict network, (e)–(h) hepph network, (i)–(l) facebook network, and (m)–(p) enron network.

Table 4
Four networks with different community structures at time step 1.

Network	Nodes/edges	Communities	Internal degree	External degree
gn1	5000/224955	50	80	10
gn2	5000/225104	50	70	20
gn3	5000/225074	50	45	45
gn4	5000/225628	50	40	50

is expected internal degree of a node. The larger the expected internal degree is, the stronger the community structure is. At each time step from 2 to 31, dynamics are introduced in the following way: in order to make the network to evolve slowly, we only randomly remove an edge and add an edge between two random chosen nodes. From Table 5, it can be seen that community structure plays an important role in the cumulative CPU running time and speedup ratio. With the increase of a node's external degree, the CPU running time for two methods increases gradually. For example, the computing time of our method on four networks is 15.40 s, 16.33 s, 17.82 s and 18.35 s respectively. Meanwhile, although our method performs faster than traditional one on four networks, the speedup ratio on the networks with weak-defined community is higher than that on the networks with well-defined community, such as 0.38 for gn4 and 0.27 for gn1. The reason is that more external edges lead to more computing time in the first iteration of Blondel method. However, the saving computing time of our method just rely on the modularity optimization of the first iteration, so the higher speedup ratio can be obtained on the networks with weak-defined community than those with well-defined community.

Second, we generate other four specific networks by Girvan–Newman method to test the performance of our method on different number of communities. As seen from Table 6, the expected internal degree and external degree of a node are same for four networks at time step 1. However, the number of communities for four networks is different, which is set to 10, 15, 100 and 150 respectively. At each time step from 2 to 31, dynamics are introduced with the same way as first case. Table 7

Table 5
The experimental results for different community structures, including the cumulative CPU running time (s), the average number of detected communities (groups), modularity (Q) and the speedup ratio.

Network	Traditional method			Our method			Speedup ratio
	Time	Groups	Q	Time	Groups	Q	
gn1	21.19	50	0.87	15.40	50	0.87	0.27
gn2	22.46	50	0.76	16.33	50	0.76	0.27
gn3	26.28	36	0.47	17.82	36	0.47	0.32
gn4	29.57	29	0.42	18.35	29	0.42	0.38

Table 6
Four networks with different number of communities at time step 1.

Network	Nodes/edges	Communities	Internal degree	External degree
cn1	1000/45238	10	80	10
cn2	1500/67552	15	80	10
cn3	10000/450043	100	80	10
cn4	15000/675983	150	80	10

Table 7
The experimental results for different number of communities, including the cumulative CPU running time (s), the average number of detected communities (groups), modularity (Q) and the speedup ratio.

Network	Traditional method			Our method			Speedup ratio
	Time	Groups	Q	Time	Groups	Q	
cn1	4.59	10	0.79	3.37	10	0.79	0.26
cn2	7.14	15	0.82	5.37	15	0.82	0.25
cn3	47.76	100	0.87	34.86	100	0.87	0.27
cn4	74.77	150	0.88	56.61	150	0.88	0.24

shows that with the increase of the number of communities, the speedup ratio has only small changes. So the number of communities has little influence on speedup ratio.

3.3. *Quality analysis*

To analyze the quality of communities detected by our method, we compare the modularity of our method with that of traditional one on four temporal networks evolving by 1 day, 5 days, 20 days and 25 days. As seen in Fig. 3, the maximum modularity obtained by two methods varies with different evolving speeds for four temporal networks. For the large evolving speeds such as 20 days and 25 days, the modularity of our method is lower than that of traditional one. Because of large changes in edges, a previous community may split into several smaller communities at current time step. When constructing a new network, some nodes in these smaller communities might be considered as a node, which leads to low quality of communities. On the contrary, For the small evolving speed such as 1 day, the two methods have nearly the same modularity at every time step and the difference in modularity value is less than 0.01. However, as Blondel et al. point out, the result of the Blondel algorithm depends on the ordering of nodes, but it does not have significant influence on modularity. Because of constructing a small network at current time step, the sweeping order of nodes for our method is different from that for traditional one. So the modularity achieved by our method is slightly higher than that of traditional one at some time step. From the above analysis, our method is suitable for the temporal networks that evolve slowly.

4. Conclusion

In this paper, we suggest a fast detection method of dynamic community in temporal network. Our method first constructs a small network by advantage of previous community structures and then detects communities at current time step. There are three experimental results in our paper. First, when small changes in the number of edges happen between two consecutive time steps, our method not only improves efficiency but also maintains quality of communities compared with traditional one. Second, the performance of our method is related to community structure. For two networks with approximately equal number of edges, our method performs better in the network with weak-defined community structure than that with well-defined community structure. Third, the number of communities has little influence on the speedup ratio of our method.

There are two open issues needing further studying. First, if the community structure changes dramatically, a previous community structure may split into several smaller communities at current time step. Some nodes in these smaller communities might be considered as a node when constructing a new network. So how to improve the accuracy of our method at processing community splitting is worth studying in the future. Second, apart from community structure, there are many other network characteristics which have an influence on the speed ratio. Much work is needed to find them in the future.

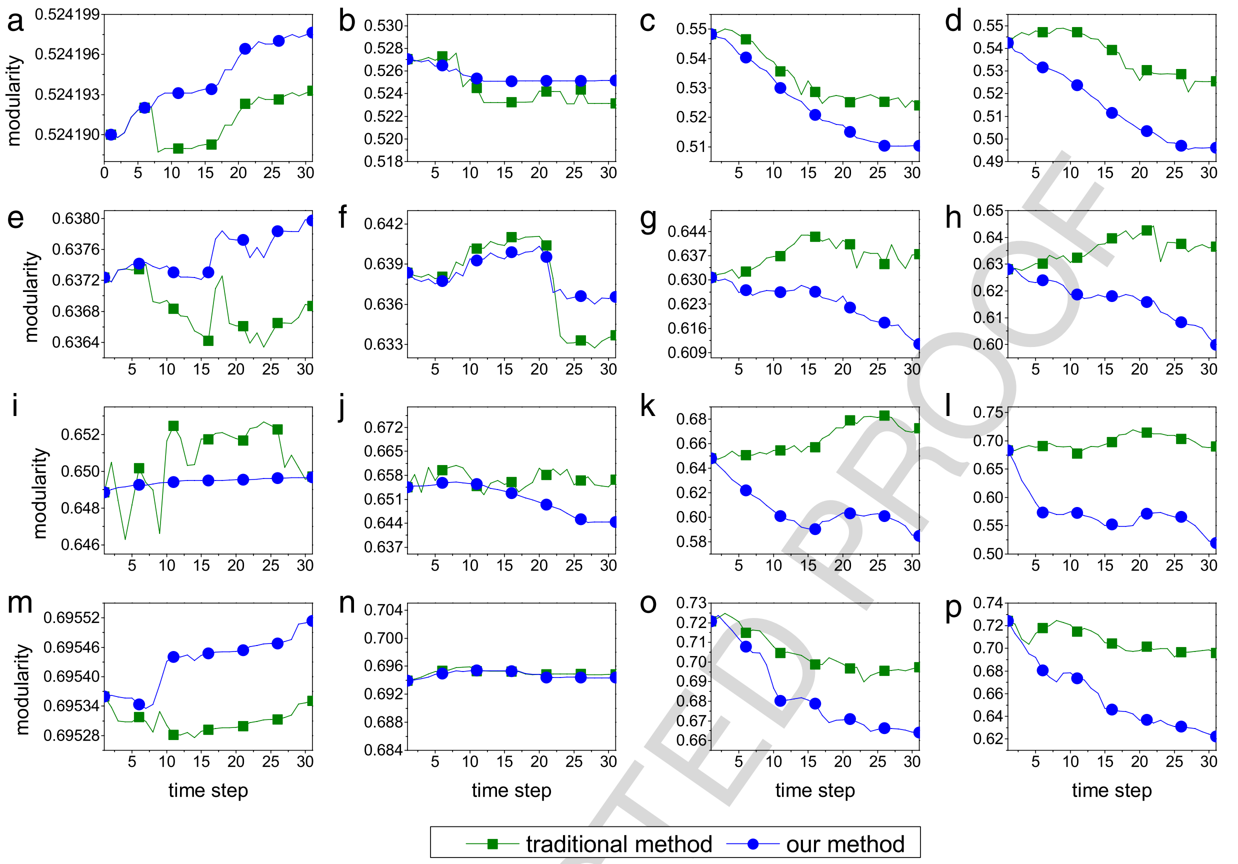

Fig. 3. Modularity on four temporal networks with evolving by 1 day, 5 days, 20 days and 25 days respectively. (a)–(d) Conflict network, (e)–(h) hepph network, (i)–(l) facebook network, (m)–(p) enron network.

Acknowledgments

This work is supported by the National Natural Science Foundation of China with Grant Nos. 61433014, 61103109 and 61300018, and by the Tsinghua-Tencent Joint Laboratory for Internet Innovation Technology. Q3

References

[1] M.E. Newman, The structure of scientific collaboration networks, Proc. Natl. Acad. Sci. 98 (2) (2001) 404–409.
[2] D. Obradović, S. Baumann, A. Dengel, A social network analysis and mining methodology for the monitoring of specific domains in the blogosphere, Soc. Netw. Anal. Min. 3 (2) (2013) 221–232.
[3] W.W. Zachary, An information flow model for conflict and fission in small groups, J. Anthropol. Res. (1977) 452–473.
[4] M. Newman, Communities, modules and large-scale structure in networks, Nat. Phys. 8 (1) (2012) 25–31.
[5] Y.-R. Lin, Y. Chi, S. Zhu, H. Sundaram, B.L. Tseng, Facetnet: a framework for analyzing communities and their evolutions in dynamic networks, in: Proceedings of the 17th International Conference on World Wide Web, ACM, 2008, pp. 685–694.
[6] R. Guimera, L.A.N. Amaral, Functional cartography of complex metabolic networks, Nature 433 (7028) (2005) 895–900.
[7] A. Clauset, M.E. Newman, C. Moore, Finding community structure in very large networks, Phys. Rev. E 70 (6) (2004) 066111.
[8] M.E. Newman, Modularity and community structure in networks, Proc. Natl. Acad. Sci. 103 (23) (2006) 8577–8582.
[9] J.-R. Chen, Z.-M. Hong, L.-N. Wang, L. Wu, Dynamic evolutionary community detection algorithms based on the modularity matrix, Chin. Phys. B 23 (11) (2014) 118903.
[10] L.G. Moyano, M.L. Mouronte, M.L. Vargas, Communities and dynamical processes in a complex software network, Physica A 390 (4) (2011) 741–748.
[11] W. Liu, M. Pellegrini, X. Wang, Detecting communities based on network topology, Sci. Rep. 4 (2014) 5739.
[12] D. Greene, D. Doyle, P. Cunningham, Tracking the evolution of communities in dynamic social networks, in: Advances in Social Networks Analysis and Mining (ASONAM), 2010 International Conference on, IEEE, 2010, pp. 176–183.
[13] S. Asur, S. Parthasarathy, D. Ucar, An event-based framework for characterizing the evolutionary behavior of interaction graphs, ACM Trans. Knowl. Discov. Data (TKDD) 3 (4) (2009) 16.
[14] H. Ning, W. Xu, Y. Chi, Y. Gong, T.S. Huang, Incremental spectral clustering with application to monitoring of evolving blog communities, in: SDM, SIAM, 2007, pp. 261–272.
[15] G. Palla, A.-L. Barabási, T. Vicsek, Quantifying social group evolution, Nature 446 (7136) (2007) 664–667.
[16] K.S. Xu, A.O. Hero, Dynamic stochastic blockmodels for time-evolving social networks.
[17] I. Subašić, B. Berendt, Story graphs: tracking document set evolution using dynamic graphs, Intell. Data Anal. 17 (1) (2013) 125–147.
[18] K.N. Vavliakis, A.L. Symeonidis, P.A. Mitkas, Event identification in web social media through named entity recognition and topic modeling, Data Knowl. Eng. 88 (2013) 1–24.

[19] N.P. Nguyen, T.N. Dinh, Y. Shen, M.T. Thai, Dynamic social community detection and its applications, PLoS One 9 (4) (2014) e91431.
[20] D.S. Bassett, M.A. Porter, N.F. Wymbs, S.T. Grafton, J.M. Carlson, P.J. Mucha, Robust detection of dynamic community structure in networks, Chaos 23 (1) (2013) 013142.
[21] V.D. Blondel, J.-L. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks, J. Stat. Mech. Theory Exp. 2008 (10) (2008) P10008.
[22] S. Bansal, S. Bhowmick, P. Paymal, Fast community detection for dynamic complex networks, in: Complex Networks, Springer, 2011, pp. 196–207.
[23] J. Leskovec, J. Kleinberg, C. Faloutsos, Graph evolution: densification and shrinking diameters, ACM Trans. Knowl. Discov. Data (TKDD) 1 (1) (2007) 2.
[24] B. Viswanath, A. Mislove, M. Cha, K.P. Gummadi, On the evolution of user interaction in facebook, in: Proceedings of the 2nd ACM Workshop on Online Social Networks, ACM, 2009, pp. 37–42.
[25] B. Klimt, Y. Yang, The enron corpus: a new dataset for email classification research, in: Machine Learning: ECML 2004, Springer, 2004, pp. 217–226.
[26] U. Brandes, J. Lerner, Structural similarity: spectral methods for relaxed blockmodeling, J. Classif. 27 (3) (2010) 279–306.
[27] J. Moody, D. McFarland, S. Bender-deMoll, Dynamic network visualization1, Am. J. Sociol. 110 (4) (2005) 1206–1241.
[28] M.E. Newman, M. Girvan, Finding and evaluating community structure in networks, Phys. Rev. E 69 (2) (2004) 026113.