Online Community Transition Detection

Biying Tan[1], Feida Zhu[1], Qiang Qu[2], and Siyuan Liu[3]

[1] School of Information Systems
Singapore Management University
[2] Department of Computer Science
Aarhus University
[3] Heinz College
Carnegie Mellon University

Abstract. Mining user behavior patterns in social networks is of great importance in user behavior analysis, targeted marketing, churn prediction and other applications. However, less effort has been made to study the evolution of user behavior in social communities. In particular, users join and leave communities over time. How to automatically detect the online community transitions of individual users is a research problem of immense practical value yet with great technical challenges. In this paper, we propose an algorithm based on the Minimum Description Length (MDL) principle to trace the evolution of community transition of individual users, adaptive to the noisy behavior. Experiments on real data sets demonstrate the efficiency and effectiveness of our proposed method.

1 Introduction

Recent years have witnessed the growth of community detection as one of the major directions in social network mining. A community can be defined as a group of users sharing some common properties. As users migrate from a community to another, communities could form and dissolve. As a result, social networks in real life are highly dynamic with evolving communities. In this paper, we are particularly interested in the discovery of community transition of individual users in social networks. This study is important for understanding user behavior patterns, which can be used to support many real-life applications. For example, in targeted marketing, users' migration to another online community often foretells the emergence of new interests and, accordingly, new opportunities for marketing [1, 2]. Similar applications can also be found in churn prediction in which the detection of community transition within a sliding window could indicate a user's potential service switch. Furthermore, the discovery of impending migration is useful in social identity linkage across social networks [3].

Unfortunately, most of the existing methods on community mining assume an underlying social network which is static [4–9]. Though there are some recent studies to investigate the dynamics of social networks [10–13] and model dynamic communities based on community structure, most of them do not examine user interaction, which is in fact one of the most important dynamics of a social network [14]. For instance, in a social network like Twitter, interactions of tweeting and retweeting form conversations, which propagate information within and across communities. In this paper, we define a sequence of individual interaction networks snapshots to model dynamic communities.

To the best of our knowledge, very few studies have been devoted to the problem of discovering community transitions. The problem is challenging in that (I) The transition between communities is, in general, an irregular and infrequent event for an individual user, which adds an extra degree of difficulty for detection due to the absence of regular patterns; and (II) A single observed interaction may not substantiate a user's immediate inclination. For example, the chat with an insurance agent does not always indicate that the user is inclined to buy insurance.

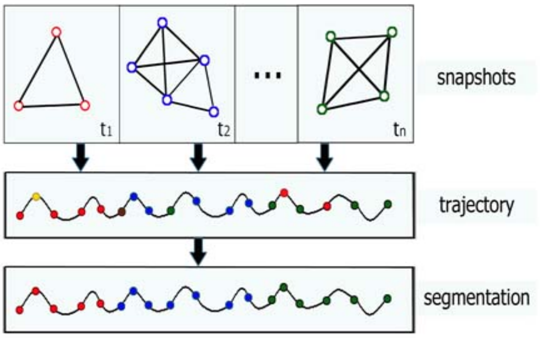

Fig. 1. The proposed framework

In summary, we make the following contributions in this work. We formalize the problem of community transition discovery for social network users. A trajectory is built to represent users' community evolution. We propose an automatic trajectory segmentation method based on the principle of Minimum Description Length (MDL) [15], which frees our solution from user-defined parameters such as the number of segments or the threshold to report a transition. Furthermore, our method can detect transitions in a noisy environment by replacing the noise with the underlying real observation. The framework of our method is illustrated in Figure 1. The experiments demonstrate the efficiency and effectiveness of our approach in discovering the community transition on real datasets.

The remainder of the paper is organized as follows. In Section 2, we define the underlying interaction network, community evolutionary trajectory and trajectory segmentation. Then we design an encoding scheme to determine a good segmentation of the community evolutionary trajectory in Section 3. Section 4 presents the corresponding algorithms, and we report the experimental results in Section 5. Section 6 reviews the related work and Section 7 concludes the paper.

2 Problem Statement

In this section, we present the preliminaries and formulate the problem.

Consider a graph, G, consisting of a set, $V(G)$, of $|V(G)|$ nodes ($|V(G)| \geq 2$) and a set, $E(G)$, of edges linking pairs of nodes, $e(u_i, u_j) \in E(G)$ for $u_i, u_j \in V(G)$. We term a graph G as an ego network if a node $u_0 \in V(G)$ satisfying for any $e(u_0, u_i) \in E(G)$, there exists $u_i \in V(G)$. Given a set $\Sigma = \{\varsigma_1, \varsigma_2, \ldots, \varsigma_k\}$ of labels, a labeling function, $L(V) \mapsto \Sigma$, maps nodes in G to labels in Σ.

Given such an underlying ego network G, we define a_{ij} as an interaction between two users u_i and u_j, which is attached with a weight representing the frequency of interactions between the two users. Such interactions, for example, can be emails or replies.

Consider an ego network G which is evolving over time, we use $G^{(t)}$ to represent G at t-timestamp. Given a network snapshot $G^{(t)}$, we use a $|V^{(t)}| \times |V^{(t)}|$ matrix to represent an interaction network $I^{(t)} = G^{(t)}$, where an element $a_{ij}^{(t)}$ is a weight of the edge $e^{(t)}(u_i, u_j) \in E(G^{(t)})$ at timestamp t, and each user $u_i \in V(G^{(t)})$ is associated with a set of labels $\Sigma_{u_i}^{(t)}$.

In this paper, community is defined as the subgroup of users sharing the same properties, then we group the users with a same label into a community. Given a sequence of interaction network snapshots, i.e., $I = \{I^{(1)}, I^{(2)}, ..., I^{(t)}\}$, the $n^{(i)}$ communities detected at i-th snapshot are denoted by $C(i) = \{C_1^{(i)}, C_2^{(i)}, ..., C_{n_i}^{(i)}\}$, where $C_j^{(i)} \in C(i)$ is a subgraph of $I^{(i)}$. The members $V(C_j^{(i)})$ within $C_j^{(i)}$ share a same label, such as school, location, gender, thus we have the label of $C_j^{(i)}$, $L(C_j^{(i)}) = \bigcap L(V(C_j^{(i)}))$, $L(C_j^{(i)}) \neq L(C_{j'}^{(i)})$ for $j \neq j'$. As a user that interacts more often with a community has more to do with that community and therefore we introduce the notion of dominant community which denotes the community with maximal interactions.

Definition 1 ***(Dominant Community).*** *The community which has the maximal interactions with user u_0 at i-timestamp is defined as the dominant community $D^{(i)}$. We define the community activity to measure the frequency of interactions, which is computed as the ratio of intra-community interactions to the total number of interactions at i-timestamp, i.e.,*

$$CA(C_j^{(i)}) = \frac{\sum_{u_m, u_n \in V(C_j^{(i)})} a_{mn}^{(i)}}{\sum_{u_m, u_n \in I^{(i)}} a_{mn}^{(i)}}, \tag{1}$$

where the minimum is 0 indicating no interaction within this community, while the maximum is 1 representing no interaction outside this community.

Tracking dominant communities along the sequence of interaction network snapshots enables to discover valuable knowledge about the behavior evolution regarding to communities in social network. To motivate this, we construct a community evolutionary trajectory using the sequence of dominant communities.

Definition 2 ***(Community Evolutionary Trajectory).*** *A community evolutionary trajectory of length t for user u_0 is a sequence consisting of t dominant communities, i.e.,*

$$\mathcal{T} := \{L_1, L_2, ..., L_t\}, \tag{2}$$

where each trajectory element L_i is the label of the dominant community $D^{(i)}$.

There are two reasons to construct the evolutionary trajectory using dominant communities: 1) The communities within which the users interact frequently are more important, which reveals stronger correlation between users and communities. 2) Community evolution trajectory provides an efficient method to study the community evolution. Traditional algorithms with regard to community evolution problems are of high computational complexity. Tracking how communities evolve over time based on sequence reduces the complexity by mapping complex graph structure into a community label, which is more efficient, especially in large social networks.

In order to track the evolution of communities transitions, we partition the community trajectory into a finite set of segments, and each segment has the maximal discriminating power. The goodness measurement of the segmentation will be made precisely in the next section, which formulates our cost objective function.

Definition 3 ***(Community Trajectory Partitions).*** *For a trajectory $\mathcal{T}$ of length t, L_i with $0 < i \leq t$ denotes its i-element. A trajectory segment is denoted by $L_{i...j}$ for $0 < i \leq j \leq t$. The set of consecutive communities which are assigned into the p-th segment $1 \leq p \leq n$ is denoted by $\mathcal{T}_p$. The partitions do not overlap, in the sense that $\mathcal{T}_p \bigcap \mathcal{T}_{p'} = \oslash$ for $p \neq p'$. Given two consecutive communities $\mathcal{T}_1 = L_{i...j}$, $\mathcal{T}_2 = L_{j+1...k}$, the labels of two segments are denoted by $L(\mathcal{T}_1) = L_i \bigcup ...L_j$, $L(\mathcal{T}_2) = L_{j+1} \bigcup ...L_k$, respectively. We say that a community transition has occurred if $\frac{L(\mathcal{T}_1) \bigcap (\mathcal{T}_2)}{L(\mathcal{T}_1) \bigcup (\mathcal{T}_2)} < \epsilon$, and we call $j+1$ a change point.*

Based on the above definition, the partition of a sequence can be regarded as a 1-dimensional classificaiton problem, and the best partition is a set of homogeneous segments, within which there exists the communities with the same label. We treat a community evolutionary trajectory as a sequence, an encoding scheme is proposed to describe the sequence. As a perfect segment is the repetitions of the same character, we can use fewer characters to describe the sequence, which leads to a smaller encoding cost. According to the Minimal Description Length (MDL) principle, the best encoding scheme is the one which leads to the minimal encoding cost. The problem of finding a good partition can be converted to the problem of determining a good encoding method. Thus, we define the problem as follows:

The goal: Given a community evolutionary trajectory $\mathcal{T}$, find the best partition $P(\mathcal{T})$ by identifying a set of change points, which leads to the minimal encoding cost.

3 Methodology

To achieve the goal that finding the best partition with minimal encoding cost, a lossless encoding scheme is introduced. Consider a lossless compressor that output s' from s, s' can be seen as another description of s. If s' is shorter than s, then s' is a better description than s, which gives a smaller encoding cost. Given a community evolutionary trajectory, our target is to identify a set of change points that leads to the optimal partition. The problem can be rephrased as to pick up the unreasonable change points. The unreasonable change points mainly consist of the noisy ones. For example, given a trajectory $T = NNNUNNNAUUUUUUU$, the first occurring "U" and "A" can be considered as such noisy change points, and the optimal partition should be $\{\{NNNUNNNA\}, \{UUUUUUU\}\}$. Under the MDL principle, we design a cost objective function with two parts, the first one is the cost to encode the optimal segmentation, and the second is the cost to identify the unreasonable change points.

3.1 Segment Encoding

We start by converting the trajectory into a sequence whose elements are numerical characters. Let Num be the function that maps each community label to a numerical character $L_i \mapsto Char(j)$. For example, given a trajectory $\mathcal{T} = NNNNNSNNUUUU$, and Num maps each label to a character as follows: $(S, 0), (N, 1), (U, 2)$, so that we have the corresponding sequence $S = Num(\mathcal{T}) =$ 111110112222. After converting the trajectory, we introduce how to encode the trajectory segments.

As our goal is to decompose the trajectory into best segments, each segment can be seen as a repetition of some characters. For example, given an input sequence: $S = 111110112222$, our encoding schema partitions this sequence into five consecutive segments: $P(S) = \{\{11111\}, \{0\}, \{11\}, \{2222\}\}$. A segment can be encoded into two fields, encoding the repeating character symbol into one field, and the length of the segment in the other:

- $Char^{S_p}(j)$: the repeating character of segment S_p,
- $Len(S_p)$: the length of segment S_p.

Note that, $Char^{S_p}(j)$ and $Len(S_p)$ is constant, which has no effect on the final segmentation. To facilitate the encoding, we define the entropy for a sequence under a partition S_p as

$$E(S) = -\sum_{x \in S} p(x) \log p(x), \tag{3}$$

where x is the element in S, $p(x)$ is the probability of x that appears in S. $E(S)$ measures how homogeneous the sequence is. The minimum of $E(S)$ is 0 when all the elements are the same, and the maximum is $\log |S|$ when the sequence is a pure random sequence. Thus, we obtain the segment encoding cost as the follows:

Definition 4 ***(Segments Encoding Cost)***

$$C^{S_p} = \sum_{S_P} m + E(S), \tag{4}$$

where m is the number of bits needed to encode a segment. The cost is a sum of encoding cost of all the segments.

3.2 Identifying the Unreasonable Change Points

Given a trajectory with a set of change points, the cost of identifying the change point is heavily related to the occurrence frequency of the corresponding community. For instance, given a trajectory $T = NNNUNNNNAUUUUUUU$, obviously, the cost of identifying the time that "A" appears as an unreasonable change point is much less than the one to identify the first time "U" appears, as we need to consider eight candidate "U" in order to identify. Thus it leads to more cost to substitute the community which occurs frequently in the trajectory. It is also sound to interpret this with our problem statement, more penalty should be given to treat a frequently occurring community as a noise. The encoding scheme for a substitution is encoded in a way similar as a segment encoding, which includes:

– $Sub^{S_p}(k, k')$: the substitution of character k' for k,
– $Pos^{S_p}(k)$: the position of character k in the sequence.

Similarly, the description complexity for substitution $\mathcal{T}$ is constant, which is equal to the encoding cost of a segment. Thus, we obtain the substitution encoding cost as the follows:

Definition 5

$$C^{S_b} = \sum_{S_b} m, \tag{5}$$

where m is the number of bits needed to encode a substitution. The cost is a sum of encoding cost of all the substitutions.

Given a community evolutionary sequence S, our goal is to partition it into a number of segments, and compress each segment which leads to a minimal encoding cost. The total encoding cost is the sum of the segments encoding cost and the cost of all the substitutions. We formulate the total encoding cost in the following:

Definition 6 ***(Total Encoding Cost)***

$$C = \sum_{S_p} C^{S_p} + \sum_{S_b} C^{S_b}, \tag{6}$$

where C^{S_p} is the encoding cost of p-th partition, and C^{S_b} is the encoding cost of b-th substitution.

In the next section, we present a search algorithm to find the optimal solution, based on the cost objective function.

4 Algorithm

In this section, based on the encoding scheme and cost function introduced in section 3, an algorithm is introduced to find the optimal segmentation in the presence of noise. Algorithm 1 is to find all the change points existing in the trajectory. The change point is a point in the trajectory that is different from the one located in the immediate left/right. The algorithm we apply to find the optimal segmentation is basically to substitute some unreasonable change points in order to find the optimal partition under the measurement of the cost function. Algorithm 2 presents the approximate algorithm to achieve the target. Basically, we use Depth-First Search (DFS) to search for the optimal partition from the answer space. Procedure OP presents the search logic with one node in the search tree. Given the current partition, OP constructs the change point sets for both two directions, PL and PR. Then OP tries to substitute each of them with the character which makes it distinguished, i.e., the substitution merges one change point into the segment left or right to it. After the substitution, OP computes the encoding cost of the new partition. If the encoding cost is smaller than the value of the original partition passed to OP, OP is recursively called with the new partition. Global value minCost and minString are used to store the answer. They are kept updating during the whole search procedure, with the minimal cost and its corresponding partition.

Algorithm 1. Find potential change points (left)

```
 1: procedure FCPL($S$, $P$)
 2:     $P \leftarrow \oslash$
 3:     $cur \leftarrow S_0$
 4:     for $i \leftarrow 1, |S|$ do
 5:         if $cur \neq S_i$ then
 6:             $cur = S_i$
 7:             $P \leftarrow P \bigcup S_i$
 8:         end if
 9:     end for
10: end procedure
```

Algorithm 2. Find optimal partition

```
 1: $optimalStr = S$
 2: $minCost = Cost(S)$
 3: procedure OP($S$)
 4:     $PL \leftarrow$ FCPL($S$)
 5:     $PR \leftarrow$ FCPR($S$)
 6:     $min = Cost(S)$
 7:     for each $p$ in $PL(PR)$ do
 8:         $p' \leftarrow$ the point locates in the immediate left(right) of $p$
 9:         $S' \leftarrow Substitute(S, p, p')$
10:         $cost = Cost(S')$
11:         if $cost \geqslant min$ then return
12:         else
13:             if $cost < minCost$ then
14:                 $optimalStr = S'$
15:                 $minCost = cost$
16:             end if
17:             OP($S'$)
18:         end if
19:     end for
20: end procedure
```

5 Experimental Results

In this section, we will evaluate our method on real, large social network datasets. We perform the experiments on a PC with a Intel(R) Core(TM) i5-2300 2.80GHz CPU and 12GB RAM and the algorithm is implemented with JAVA.

Table 1. The statistics of Renren dataset

	Max	Avg
Number of Communities/User	10	2
Number of Timestamps/User	1105	580

5.1 Dataset

We conduct our study on the Renren Network dataset[1]. Renren Network is a leading social networking service in China, which is similar as Facebook. As Renren is one of the largest real name based social network in China, and Renren user provides a list of education background in their profiles, thus we can observe the evolution of their communities through the observation of interactions with different groups of friends along with the evolution of users' identities. The dataset consists of more than 16 million replies to 6,437 users' status posts, and the profile information of 690,926 unique users are involved in the dataset.

The interaction network snapshot $I^{(t)}$ is constructed from the observed reply messages, each edge in the network corresponding to a reply between two users. We use a window size of one day to construct the interaction network snapshots, and then generate the community evolutionary trajectory based on the network snapshots. The education and occupation information are chosen as the labels of users, and each label is associated with a time period constraint. The users with the same label are grouped into a community in each network snapshot, the community with the maximal community activity is selected as the element of the trajectory. The statistics of Renren dataset is shown in Table 1.

5.2 Effectiveness

In this subsection, we evaluate the effectiveness of our method for finding optimal segmentation in the following two experiments.

In the first experiment, we match the reasonable change points identified by our method with the ground truth provided by the labels in users' profiles. More specifically, we compute the smallest difference (day) between the results with any change point in the ground truth. The variance of the results is reported in Figure 2. We can observe that most dots locate in the left side, indicating our method has a high accuracy. Many dots gather in the lower left corner, which demonstrates our method can detect the transition happened in the latest two

[1] `www.renren.com`

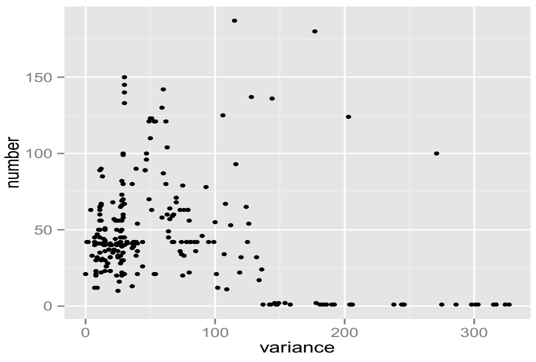

Fig. 2. Variance distribution of the results

months. The dots in the right part present the cases of low accuracy, which could be due to the delayed update or data missing of their profiles for some users. The other reason is due to the short sequence generated from inactive users.

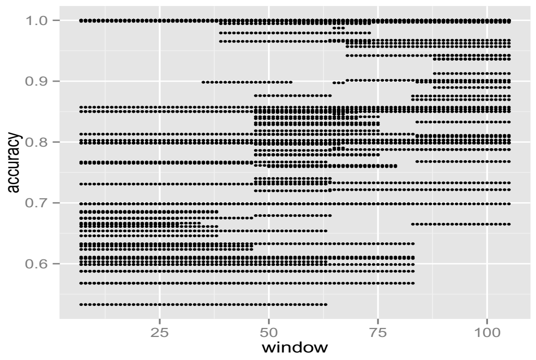

Fig. 3. Comparison of accuracy with different tolerance windows

Secondly, we estimate our method under different tolerance window. The tolerance window is created by moving the change point on the left and right with the distance of the tolerance value. Then we determine if any change point in ground truth falls within that window. The comparison of the accuracy with different tolerance from 7 to 105 days is listed in Figure 3. We can observe an obvious increase of accuracy along with increase of the tolerance window. The results show that our method can achieve 60%∼70% accuracy when the window is smaller than 35 days, and the accuracy can reach to 75%∼87% when the size of tolerance window is between 47 and 62 days. The results showed in Figure 3 is consist with what we observe in Figure 2.

Furthermore, we display a comparison of the original trajectory and the trajectory generated by our method in Figure 4. Figure 4(a) shows an example of the original trajectory, each colour area corresponds to a segment of consecutive identical dominant communities. A rough transformation of communities can be observed in Figure 4 (a). Our method substitutes the noise observation with the underlying real behavior, and nicely generates a set of segments representing a community evolution as shown in Figure 4 (b): $SchoolA \rightarrow SchoolB \rightarrow SchoolC$.

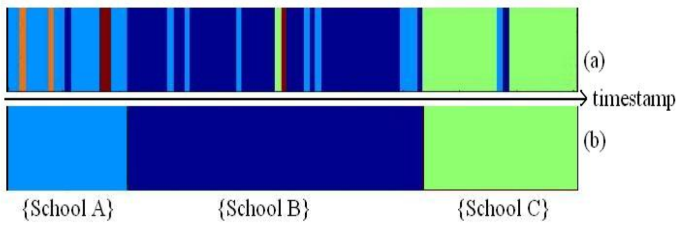

Fig. 4. Comparison of the original trajectory (a) with the trajectory (b) generated by our method

5.3 Scalability

We illustrate the running time for different number of snapshots in Figure 5. As shown in Figure 5, the near linear run time complexity demonstrates the high scalability of our method.

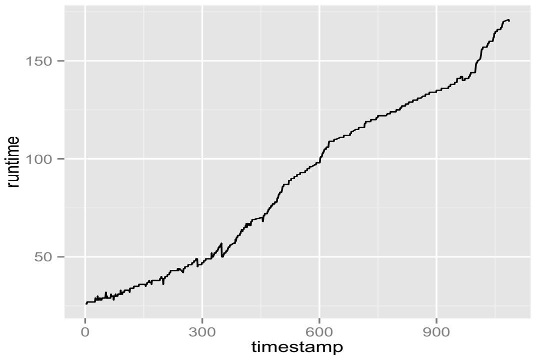

Fig. 5. Runtime (ms) vs number of timestamps (day)

6 Related Work

Several papers have studied on the community evolution in social networks. In general, we consider three broad class of temporal analysis on online communities: 1) properties or phenomena of evolving community analysis, 2) stage of dynamic community identification, 3) dynamic community detection and the community changes identification. In the first class work, Leskovec et al. [16] discovered the shrinking diameter phenomena on time-evolving networks. In the second class work, Palla et al. [17] quantifies the events in community evolution: growth, merging, birth, contraction, splitting and death. In the last class work, Tang et al. [11] proposed a clustering algorithm to mine the evolution of communities in social network with multiple entities. Aggarawal and Yu [18] proposed a method to find community changes in dynamic graphs, which requires user-specified parameters. Sun. et al. [19] and Ferlez at al. [20] proposed a parameter-free framework to discover communities and community changes in dynamic networks, relying on the MDL principle. Overall, all the above studies have a drawback that they cannot determine the evolution of users across communities individually, they all track the evolution of the whole network.

7 Conclusions and Future Work

In this paper, we propose a parameter-free method to discover the community transition for individual users in dynamic networks. We start by constructing a trajectory to represent the evolution of communities, then a trajectory segmentation approach is proposed to discover the best partition, based on the MDL principle. Our method is automatic, which requires no user-specified parameters, like the threshold of a transition and the number of segments. Furthermore, our method is adaptive to distinguish between the noise and underlying true behavior. The experiments on real dataset show that our method has a high accuracy to find the community transitions, and our method is fast and scalable.

Our paper is a preliminary study on the detection of community transition, which still needs a lot of improvements. First, we should improve the method by incorporating all the communities evolved in the user's interactions, and detecting long-term transition and short-term transition. Second, we should enlarge the experiment by evaluating our method on different datasets, and comparing some state-of-art methods.

Acknowledgement. This research has been supported by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office, Media Development Authority (MDA) and the Pinnacle Lab at Singapore Management University.

References

1. Dasgupta, K., Singh, R., Viswanathan, B., Chakraborty, D., Mukherjea, S., Nanavati, A.A., Joshi, A.: Social ties and their relevance to churn in mobile telecom networks. In: Proceedings of the 11th International Conference on Extending Database Technology: Advances in Database Technology, pp. 668–677. ACM (2008)
2. Richter, Y., Yom-Tov, E., Slonim, N.: Predicting customer churn in mobile networks through analysis of social groups. In: SDM, pp. 732–741 (2010)

3. Liu, S., Wang, S., Zhu, F., Zhang, J., Krishnan, R.: Hydra: Large-scale social identity linkage via heterogeneous behavior modeling
4. Duch, J., Arenas, A.: Community detection in complex networks using extremal optimization. Physical Review E 72(2), 27104 (2005)
5. Fortunato, S., Barthelemy, M.: Resolution limit in community detection. Proceedings of the National Academy of Sciences 104(1), 36–41 (2007)
6. Lancichinetti, A., Fortunato, S.: Community detection algorithms: a comparative analysis. Physical Review E 80(5), 56117 (2009)
7. Fortunato, S.: Community detection in graphs. Physics Reports 486(3), 75–174 (2010)
8. Kovács, I.A., Palotai, R., Szalay, M.S., Csermely, P.: Community landscapes: an integrative approach to determine overlapping network module hierarchy, identify key nodes and predict network dynamics. PloS One 5(9), e12528 (2010)
9. Leskovec, J., Lang, K.J., Mahoney, M.: Empirical comparison of algorithms for network community detection. In: Proceedings of the 19th International Conference on World Wide Web, pp. 631–640. ACM (2010)
10. Kim, M.-S., Han, J.: A particle-and-density based evolutionary clustering method for dynamic networks. Proceedings of the VLDB Endowment 2(1), 622–633 (2009)
11. Tang, L., Liu, H., Zhang, J., Nazeri, Z.: Community evolution in dynamic multimode networks. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 677–685. ACM (2008)
12. Tantipathananandh, C., Berger-Wolf, T., Kempe, D.: A framework for community identification in dynamic social networks. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 717–726. ACM (2007)
13. Yang, T., Chi, Y., Zhu, S., Gong, Y., Jin, R.: Detecting communities and their evolutions in dynamic social networks a bayesian approach. Machine Learning 82(2), 157–189 (2011)
14. Wilson, C., Boe, B., Sala, A., Puttaswamy, K.P., Zhao, B.Y.: User interactions in social networks and their implications. In: Proceedings of the 4th ACM European Conference on Computer Systems, pp. 205–218. ACM (2009)
15. Barron, A., Rissanen, J., Yu, B.: The minimum description length principle in coding and modeling. IEEE Transactions on Information Theory 44(6), 2743–2760 (1998)
16. Leskovec, J., Kleinberg, J., Faloutsos, C.: Graphs over time: densification laws, shrinking diameters and possible explanations. In: Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, pp. 177–187. ACM (2005)
17. Palla, G., Barabási, A.L., Vicsek, T.: Quantifying social group evolution. Nature 446(7136), 664–667 (2007)
18. Aggarwal, C.C., Yu, P.: Online analysis of community evolution in data streams. In: Proceedings of the SIAM International Conference on Data Mining (SDM 2005), pp. 56–67 (2005)
19. Sun, J., Faloutsos, C., Papadimitriou, S., Yu, P.S.: Graphscope: parameter-free mining of large time-evolving graphs. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 687–696. ACM (2007)
20. Ferlez, J., Faloutsos, C., Leskovec, J., Mladenic, D., Grobelnik, M.: Monitoring network evolution using mdl. In: IEEE 24th International Conference on Data Engineering, ICDE 2008, pp. 1328–1330. IEEE (2008)