# A Particle-and-Density Based Evolutionary Clustering Method for Dynamic Networks

Min-Soo Kim
Department of Computer Science
University of Illinois at Urbana-Champaign
msk@cs.uiuc.edu

Jiawei Han
Department of Computer Science
University of Illinois at Urbana-Champaign
hanj@cs.uiuc.edu

## ABSTRACT

Recently, dynamic networks are attracting increasing interest due to their high potential in capturing natural and social phenomena over time. Discovery of evolutionary communities in dynamic networks has become a critical task. The previous evolutionary clustering methods usually adopt the temporal smoothness framework, which has a desirable feature of controlling the balance between temporal noise and true concept drift of communities. They, however, have some major drawbacks: (1) assuming only a fixed number of communities over time; and (2) not allowing arbitrary start/stop of community over time. The forming of new communities and dissolving of existing communities are very common phenomena in real dynamic networks. In this paper, we propose a new *particle-and-density based* evolutionary clustering method that efficiently discovers a variable number of communities of arbitrary forming and dissolving. We first model a dynamic network as a collection of lots of particles called *nano-communities*, and a community as a densely connected subset of particles, called a *quasi l-clique-by-clique* (shortly, *l-KK*). Each particle contains a small amount of information about the evolution of data or patterns, and the quasi *l-KK*s inherent in a given dynamic network provide us with *guidance* on how to find a variable number of communities of arbitrary forming and dissolving. We propose a density-based clustering method that efficiently finds temporally smoothed local clusters of high quality by using a *cost embedding technique* and optimal modularity. We also propose a *mapping method* based on information theory that makes sequences of smoothed local clusters as close as possible to data-inherent quasi *l-KK*s. The result of the mapping method allows us to easily identify the stage of each community among the three stages: *evolving*, *forming*, and *dissolving*. Experimental studies, by using various data sets, demonstrate that our method improves the clustering accuracy, and at the same time, the time performance by an order of magnitude compared with the current state-of-the art method.


*VLDB '09,* August 24-28, 2009, Lyon, France


## 1. INTRODUCTION

In recent years, social networks or information networks are of growing importance in a wide range of disciplines and application domains due to their ubiquity and generality of modeling. Especially, their time-evolving version, *dynamic networks* are attracting increasing interest due to their great potential in capturing natural and social phenomena over time. Examples of dynamic networks include network traffic data [20], telephone traffic data [23], bibliographic data [6], dynamic social networks [22], and time-series microarray data [29].

Clustering is the process of grouping similar objects into *clusters*. The clusters in network data called *communities* typically represent *cohesive subgroups of individuals* within a network, where each cohesive subgroup is a "subset of actors among whom there are relatively strong, direct, intense, frequent, or positive ties" [24, 15]. Identifying communities in network data has been paid much attention as an important research topic, but most studies have focused on static networks, not dynamic networks.

Recently, a new kind of clustering concept called *evolutionary clustering* has been proposed to capture the evolutionary process of clusters in temporal data. Chakrabarti et al. [4] first addressed this issue and proposed a framework called *temporal smoothness*. This framework assumes that the structure of clusters significantly changes in a very short time is less desirable, and so, it tries to smooth out each community over time. For smoothing, it trades off two kinds of qualities — the *snapshot quality* and the *history quality* — at every timestamp during clustering. The snapshot quality is about how accurately the clustering result captures the structure of current network, and the temporal quality about how similar the current clustering result is with the previous clustering result. This temporal smoothing allows us to control the balance between *temporal noise* and *true concept drift* of temporal patterns.

Several evolutionary clustering methods [5, 21, 15] have been proposed under the temporal smoothness framework. Especially, FacetNet [15] has several advanced features such as allowing attachment/detachment of nodes and explicitly handling general dynamic networks. However, the previous methods including FacetNet have several drawbacks: (1) assuming only a fixed number of communities over time; (2) not allowing arbitrary start/stop of community over time; and (3) not being scalable well with network size. Since the size of real dynamic networks tends to be large, and moreover, the forming of a new community and dissolving of an existing community are quite natural and common phenom-

ena in real dynamic networks [1], those existing methods would not be very useful in real applications. For example, they could not find a large number of evolving research groups of co-authors that form or dissolve at arbitrary time in the DBLP data.

In this paper, we propose a new *particle-and-density based* evolutionary clustering method that efficiently discovers a variable number of communities of arbitrary forming and dissolving. We first identify two sub-problems that arise when removing the constraint of the fixed number of communities. The first sub-problem is how to perform clustering with temporal smoothing when the number of communities varies, and the second sub-problem how to connect between local clusters across time when the number of local clusters are different. For solving them, we propose the concept of *nano-community* capturing how dynamic networks evolve over time at a *particle level*, and model a community as a dense subset of nano-communities forming an *l-clique-by-clique* (shortly, *l-KK*) topologically. The nano-communities and quasi *l-KK*s inherent in a given dynamic network provide *guidance* on how to find a variable number of communities of arbitrary forming and dissolving. For the first sub-problem, we present a density-based clustering method that efficiently finds smoothed local clusters of high quality using a *cost embedding technique* and optimal modularity. Our proposed cost embedding technique achieves flexible and efficient temporal smoothing by *pushing down* the cost formula from the clustering result level into the data level. For the second sub-problem, we propose a mapping method based on information theory, especially mutual information, that can identify the stage of each community at each timestamp among the following three stages: *evolving*, *forming*, and *dissolving*. Finding an optimal mapping of the maximum mutual information is a combinatorial optimization problem, and we propose a heuristic algorithm for it.

In summary, the contributions of this paper are as follows:

- We propose the concepts of nano-community and quasi *l-KK*, which enable us to discover a variable number of communities of arbitrary forming and dissolving by capturing the tendency inherent in a given dynamic network.
- We propose a cost-embedding technique that allows temporal smoothing independently of both the similarity measure and the clustering algorithm.
- We present an efficient density-based clustering method using optimal modularity that finds local clusters.
- We propose a mapping method based on information theory that makes sequences of local clusters close to data-inherent quasi *l-KK*s.
- We demonstrate, by using various data sets, that our method achieves better clustering accuracy, and at the same time, improves the time performance compared with the state-of-art method.

The rest of the paper is organized as follows. Section 2 presents the problem statement. Section 3 defines the concept of nano-community and quasi *l-KK*. Section 4 proposes a cost embedding technique and our density-based clustering method. Section 5 proposes the mapping method between local clusters. Section 6 presents the results of experimental evaluation. Section 7 discusses related work. Finally, Section 8 concludes the study.

**Table 1: Definitions of symbols**

| Symbols | Definitions |
|---|---|
| $\mathcal{G}$ | dynamic network |
| $t$ | timestamp |
| $G_t$ | network at timestamp $t$ of $\mathcal{G}$ |
| $V_t$ | set of nodes in the network $G_t$ |
| $E_t$ | set of edges in the network $G_t$ |
| $\mathcal{CR}_t$ | set of local clusters discovered on $G_t$ |
| $C_t$ | local cluster discovered on $G_t$ ($C_t \in \mathcal{CR}_t$) |
| $\mathcal{M}$ | community |
| $M_t$ | cross section of $\mathcal{M}$ at timestamp $t$ |

# 2. PROBLEM STATEMENT

## 2.1 Notation

We define a *dynamic network* $\mathcal{G}$ as a sequence of networks $G_t(V_t, E_t)$, *i.e.*, $\mathcal{G} = \{G_1, \ldots, G_t, \ldots\}$. Table 1 shows the symbols and their definitions. The dynamic network $\mathcal{G}$ allows new nodes to be attached to $G_t$ or existing nodes to be detached from $G_t$ at any timestamp $t$. We call a cluster $C_t$ of the clustering result $\mathcal{CR}_t$ for $G_t$ ($C_t \in \mathcal{CR}_t$) as a *local cluster* to discriminate it from the community itself. A community $\mathcal{M}$ is a sequence of $M_t$, where $M_t$ is a set of nodes composing the community $\mathcal{M}$ at timestamp $t$.

## 2.2 Temporal Smoothness

Evolutionary clustering under the temporal smoothness framework uses a cost function that can trade off the *history quality* with the *snapshot quality*. The cost function is composed of two sub-costs of a *snapshot cost* ($SC$) and a *temporal cost* ($TC$) as follows:

$$cost = \alpha \cdot SC(\mathcal{CR}_O, \mathcal{CR}_t) + (1-\alpha) \cdot TC(\mathcal{CR}_{t-1}, \mathcal{CR}_t) \quad (1)$$

In Eq. 1, the snapshot cost $SC()$ measures how similar the current clustering result $\mathcal{CR}_t$ is with the original clustering result $\mathcal{CR}_O$ that is obtained on $G_t$ without temporal smoothing. The smaller $SC()$ is, the better the snapshot quality is. The temporal cost $TC()$ measures how similar the current clustering result $\mathcal{CR}_t$ is with the previous clustering result $\mathcal{CR}_{t-1}$. The smaller $TC()$ is, the better the temporal quality is. The parameter $\alpha$ ($0 \leq \alpha \leq 1$) is used for controlling a level of preference to each sub-cost, *i.e.*, for trading off between two measures. The framework tries to find an optimal clustering $\mathcal{CR}_t$ that minimizes Eq. 1 at each timestamp $t$. When $\alpha = 1$, the framework produces the same clustering result with $\mathcal{CR}_O$, *i.e.*, $\mathcal{CR}_t = \mathcal{CR}_O$ in order to minimize $SC()$. When $\alpha = 0$, however, it produces the same clustering results with $\mathcal{CR}_{t-1}$, *i.e.*, $\mathcal{CR}_t = \mathcal{CR}_{t-1}$ in order to minimize $TC()$. In other cases of $0 < \alpha < 1$, it produces an intermediate result between $\mathcal{CR}_{t-1}$ and $\mathcal{CR}_O$.

The temporal smoothness framework is originally devised for a fixed number of communities, *i.e.*, assumes that $|\mathcal{CR}_1| = \ldots = |\mathcal{CR}_t|$, and there is already one-to-one correspondence between $C_{t-1} \in \mathcal{CR}_{t-1}$ and $C_t \in \mathcal{CR}_t$. In contrast, our goal is removing the constraint on the fixed number of communities and allowing the forming of new communities and the dissolving of existing communities. By getting rid of the constraint, the following two problems spontaneously arise.

**Problem 1.** How to perform clustering on $G_t$ with temporal smoothing when $|\mathcal{CR}_{t-1}| \neq |\mathcal{CR}_t|$ and how to interpret the meaning of the result.

**Problem 2.** How to connect $C_{t-1} \in \mathcal{CR}_{t-1}$ with $C_t \in \mathcal{CR}_t$ when $|\mathcal{CR}_{t-1}| \neq |\mathcal{CR}_t|$ to determine the stage of each community among the following three stages: evolving, forming, and dissolving.

We present our solution for Problem 1 in Section 4 and that for Problem 2 in Section 5.

# 3. MODELING OF COMMUNITY

In this section, we model the structure of community, which is used as guidance for the clustering result. Section 3.1 proposes a concept of *nano-community* and Section 3.2 a quasi *l-clique-by-clique*.

## 3.1 Nano-Community

To capture the evolution of dynamic networks and communities over time at a fine granularity level, we propose the concept of *nano-community*, which is a kind of particle composing a whole dynamic network or a community. In fact, from the point of view of each individual node, a node already has its tiny local cluster together with its neighbor nodes. We could consider a sequence of such tiny local clusters having non-zero similarity score between time $t-1$ and $t$. We call this tiny community as a nano-community, which is defined as in Definitions 1∼2.

**Definition 1.** The *neighborhood* $N(v)$ of a node $v \in V_t$ is defined by $N(v) = \{x \in V_t \mid \langle v, x\rangle \in E_t\} \cup \{v\}$.

**Definition 2.** The *nano-community* $NC(v, w)$ of two nodes $v \in V_{t-1}$ and $w \in V_t$ is defined by a sequence $[N(v), N(w)]$ having a non-zero score for a similarity function $\Gamma : N(\cdot) \times N(\cdot) \rightarrow \mathbb{R}$.

For a similarity function $\Gamma()$, we propose three candidates as follows:

$$(1)\ \Gamma_I(N(v), N(w)) = \begin{cases} 1 & \text{if } v = w \\ 0 & \text{otherwise} \end{cases}$$

$$(2)\ \Gamma_E(N(v), N(w)) = \begin{cases} 1 & \text{if } v \in N(w) \text{ and } w \in N(v) \\ 0 & \text{otherwise} \end{cases}$$

$$(3)\ \Gamma_N(N(v), N(w)) = \begin{cases} 1 & \text{if } N(w) \cap N(v) \neq \emptyset \\ 0 & \text{otherwise} \end{cases}$$

The function $\Gamma_I()$ returns a non-zero value only if $v$ and $w$ are the same node at the different timestamps. $\Gamma_E()$ returns a non-zero value if $v$ and $w$ satisfy the condition of $\Gamma_I()$ (*i.e.*, $v = w$), or $N(v)$ and $N(w)$ have a common edge $\langle v, w\rangle$. $\Gamma_N()$ returns a non-zero value if $N(v)$ and $N(w)$ have a common node, which obviously includes the conditions of $\Gamma_I()$ and $\Gamma_N()$ for a non-zero score. Although we present $\Gamma()$ functions as ones to return just 1 for simplicity, they could be modified such that they return different values based on the number of common edges or nodes.

In each nano-community $NC(v, w)$, we consider there is a link between $v$ and $w$ having a weight of $\Gamma(N(v), N(w))$. Hereafter, we call a connection between a pair of nodes of different timestamps as a *link* to discriminate it from an *edge*, a connection between a pair of nodes within the same timestamp. By using links, we could conceptually construct a bipartite graph between $G_{t-1}$ and $G_t$, and further, construct a *t-partite graph* from $G_1$ to $G_t$.

The function $\Gamma()$ determines how many links there are in the $t$-partite graph. Among above three $\Gamma()$ functions, $\Gamma_I()$ actually plays the role of *Jaccard coefficient*, and so, the number of links when using $\Gamma_I()$ between two subgraphs $S_{t-1} \subset G_{t-1}$ and $S_t \subset G_t$ is equivalent to $Jaccard(S_{t-1}, S_t)$, which is usually not dense and rich enough to capture the evolution of either dynamic networks or communities well. On the other hand, the number of links when using $\Gamma_N()$ is usually too large to process efficiently. Therefore, we take $\Gamma_E()$ as a trade-off solution, which returns a non-zero score only if $v$ and $w$ are the same node, or have a common edge.

## 3.2 Quasi $l$-clique-by-clique

In this section, we model the topological structure of community $\mathcal{M}$ in the $t$-partite graph. Without losing generality, a cluster in a network represents a cohesive subgroup of nodes among whom there are relatively strong and dense connections. Naturally, the clique is the structure of the cluster having the highest density in networks. When the number of nodes in a clique is $s$, we denote the clique as $K_s$. Likewise, we could define the structure of the community having the highest density by generalizing the concept of *biclique*. A biclique (or a complete bipartite graph) is defined as a bipartite graph such that two nodes are connected if and only if they are in different partites [25]. When the partites have sizes $r$ and $s$, we denote the biclique as $K_{r,s}$. Figure 1(a) shows an example of a biclique. When using $\Gamma_E()$ function for constructing the $t$-partite graph, we note that the links between two cliques $K_s$ and $K_{s'}$ for $s = s'$ obviously form a biclique $K_{s,s'}$. We could easily extend the number of partites of biclique from two to $l$, which is the length of a community. Here, we consider that each partite of this structure corresponds to a cross section (*i.e.*, a local cluster) of a community, that is, each local cluster forms a clique, and a pair of adjacent local clusters forms a biclique. We call this structure as a *l-clique-by-clique*. When the sizes of partites (cliques) are $s_1, s_2, \ldots, s_l$, respectively, we denote the $l$-clique-by-clique as $KK_{[s_1,s_2,\ldots,s_l]}$. When we need only the length of a community, we denote it as $l$-$KK$. Figure 1(b) shows an example of 4-$KK$, where each circle indicates a clique.

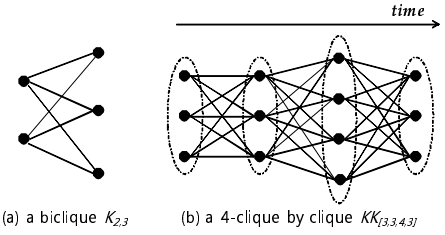


**Figure 1: Examples of a biclique and an $l$-clique-by-clique.**

A community of $l$-$KK$ is definitely the densest one among all communities of the same partite sizes. But, in real applications, most of communities have the looser structure, *i.e.*, quasi $l$-$KK$. In a quasi $l$-$KK$ community $\mathcal{M}$, let $M_t$ be the local cluster of $\mathcal{M}$ at time $t$, and $B_{t-1,t}$ the bipartite graph of $\mathcal{M}$ between $t-1$ and $t$. Without losing generality, the density of $M_t$ could be measured by the function $\Theta_K(M_t) = \frac{2|E_{M_t}|}{|V_{M_t}|(|V_{M_t}|-1)}$, where $E_{M_t}$ is the edge set of $M_t$, and $V_{M_t}$

the node set of $M_t$. Likewise, the density of $B_{t-1,t}$ could be measured by the function $\Theta_{KK}(B_{t-1,t}) = \frac{|L_{B_{t-1,t}}|}{|V_{M_{t-1}}||V_{M_t}|}$, where $L_{B_{t-1,t}}$ is the link set of $B_{t-1,t}$.

# 4. CLUSTERING WITH TEMPORAL SMOOTHING

In this section, we present a local clustering method with temporal smoothing when the number of local clusters varies. Section 4.1 proposes a *cost embedding technique* to perform smoothing at the *data level* instead of at the *result level*. Section 4.2 presents a density-based clustering algorithm using cost embedding, and Section 4.3 the optimal clustering method using *modularity*.

## 4.1 Cost Embedding Technique

The previous clustering methods using temporal smoothness perform temporal smoothing on the clustering result $\mathcal{CR}_t$. So, in general, they adjust $\mathcal{CR}_t$ iteratively so as to decrease the cost Eq. 1. However, such iterative process could degrade the clustering performance seriously. To solve this performance problem, and at the same time, find the smoothed local clusters in a flexible and natural way, we propose a temporal smoothing method on *data* $G_t$ instead of on *clustering result* $\mathcal{CR}_t$.

In fact, temporal smoothing can be performed not only at the community level (*i.e.*, each $C_t$) but also at the nano-community level (*i.e.*, each node). Figure 2 shows four cases of the relationship between two nodes $v$ at $t-1$ and $w$ at $t$. The thick solid lines indicate that $v$ and $w$ are similar (*i.e.*, distance is small) enough to be within the same local cluster, and the thin solid lines with breaker that $v$ and $w$ are too dissimilar (*i.e.*, distance is large) to be in different local clusters. The dotted lines represent links. In case 1, $v$ and $w$ are similar at both $t-1$ and $t$, and so, they would be in the same local cluster after temporal smoothing regardless of $\alpha$. In case 2, $v$ and $w$ are similar only at $t$. Thus, they would be in the same cluster at $t$ when $\alpha$ is large, but tend to be in different clusters at $t$ when $\alpha$ is small, in order to decrease the temporal cost $TC()$. That is, there is an effect that the *actual distance* between $v$ and $w$ becomes large by temporal smoothing when $\alpha$ is small in this case. Case 3 shows the reverse case of case 2. In case 4, $v$ and $w$ are dissimilar at both $t-1$ and $t$, and thus, they would be in different clusters regardless of $\alpha$. These four cases indicate that $\alpha$ determines not only the clustering result $\mathcal{CR}_t$, but also the actual individual distances between pairs of nodes at timestamp $t$.

From the above observation, we propose a *cost embedding technique* that *pushes down* the cost formula from the local cluster level to the individual distance level. The new cost function at this nano level is defined as in Eq. 2. Here, $d_t(v,w)$ indicates the smoothed distance between $v$ and $w$ at time $t$, and $d_O(v,w)$ the original distance between $v$ and $w$ at time $t$ without temporal smoothing.

$$cost_N = \alpha \cdot SC_N(d_O(v,w), d_t(v,w)) + (1-\alpha) \cdot TC_N(d_{t-1}(v,w), d_t(v,w)) \quad (2)$$

There would be many candidate measures for $SC_N()$ and $TC_N()$, but, in this paper, we use one-dimensional Euclidean distance measure for simplicity, *i.e.*, $SC_N = |d_O(v,w) - d_t(v,w)|$, and $TC_N = |d_{t-1}(v,w) - d_t(v,w)|$. For those measures, we can easily find the optimal distance $d'_t(v,w)$ as in Eq. 3 that minimizes the cost of Eq. 2.

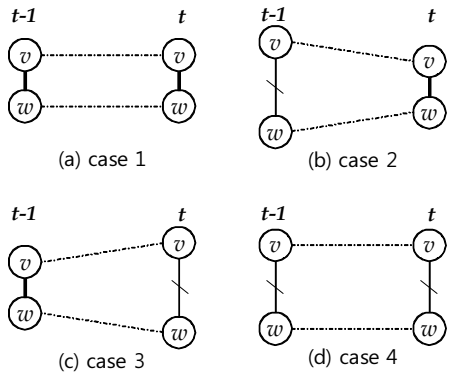


**Figure 2: Four cases of the relationship between two nodes $v$ and $w$ at timestamps $t-1$ and $t$.**

$$d'_t(v,w) = \alpha \cdot \{d_O(v,w) - d_{t-1}(v,w)\} + d_{t-1}(v,w) \quad (3)$$

When $\alpha = 1$, $d'_t(v,w)$ becomes $d_O(v,w)$, *i.e.*, $G_t$ have the original distances. Thus, no matter what kind of clustering algorithm is used, it produces $\mathcal{CR}_O$ as the result of clustering, which is consistent with the result using Eq. 1. Likewise, when $\alpha = 0$, $d'_t(v,w)$ becomes $d_{t-1}(v,w)$, *i.e.*, $G_t$ have the actual distances of the previous network $G_{t-1}$. Thus, it produces $\mathcal{CR}_{t-1}$, which is also consistent with the result using Eq. 1. When $0 < \alpha < 1$, $d'_t(v,w)$ becomes an intermediate value between $d_O(v,w)$ and $d_{t-1}(v,w)$, and so, it produces an intermediate result between $\mathcal{CR}_{t-1}$ and $\mathcal{CR}_O$. We note that $d'_t(v,w)$ becomes $d_t(v,w)$ at time $t+1$, that is, $d_t(v,w)$ partly depends on $d_{t-1}(v,w)$, which in turn partly depends on $d_{t-2}(v,w)$ an so on.

The cost embedding technique has two major advantages. First, it is independent of both the similarity measure $d()$ and the clustering algorithm to use. Thus, temporal smoothing could be performed no matter what kind of similarity measure or clustering algorithm we use. On the contrary, the previous evolutionary clustering methods perform temporal smoothing with a specific combination of similarity measure (*e.g.*, KL-divergence or chi-square statistics) and clustering algorithm (*e.g.*, k-means or spectral clustering) because they are required to adjust the result of clustering itself. Second, it makes temporal smoothing task efficient. While the previous methods adjust $\mathcal{CR}_t$ iteratively for smoothing, the method using cost embedding just needs to perform clustering once on the network that has been already smoothed on data level.

Now, we explain the effect of temporal smoothing when the number of local clusters varies and the role of $\alpha$. Figure 3 shows the result of temporal smoothing of our method on a tiny example. In Figure 3, there exist two communities — upper red one and lower blue one — continuing from $t-1$ to $t$. When $\alpha$ is large ($\alpha$=0.8) in Figure 3(a), a new green colored local cluster is found at $t$ in a clear shape, which indicates the start of a new community. But, when $\alpha$ is small ($\alpha$=0.2) in Figure 3(b), the size of the new local cluster becomes smaller, which would be totally disappeared if $\alpha = 0$. As shown in Figure 3, $\alpha$ controls not only the trade-off between $SC()$ and $TC()$, but also the frequency

in which a new community appears. A high $\alpha$ allows the forming of new communities *progressively*, whereas a low $\alpha$ allows it *conservatively*.

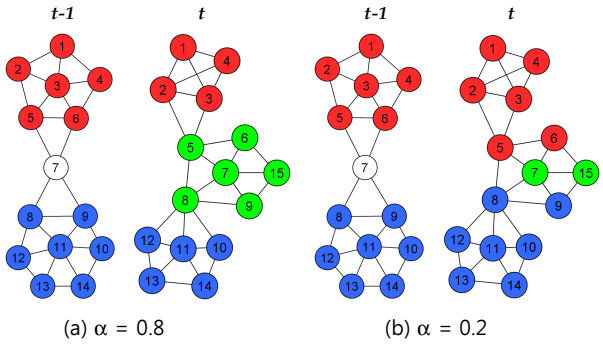


**Figure 3: The effect of temporal smoothing when $\alpha$ varies.**

## 4.2 Density-based Clustering

In this section, we present the clustering method using cost embedding for finding all local clusters at timestamp $t$, which are corresponding to temporally smoothed cross sections of quasi $l$-$KK$ communities. Clustering has been extensively studied so far, and there are three major approaches: (1) the partitioning approach (*e.g.*, $k$-means), (2) the hierarchical approach (*e.g.*, BIRCH), and (3) the density-based approach (*e.g.*, DBSCAN [8]). Since the partitioning approach needs to specify the number of clusters in advance, and the hierarchical approach usually does not scale well, we select the density-based approach, which has several advantages such as discovering an arbitrary number of clusters, handling noises, and being fast.

For similarity measure, we extend the *structure similarity* (or *cosine similarity*) [12, 26], which is one of the famous measures for network data. Eq. 4 shows the structure similarity. Intuitively, $\sigma(v, w)$ indicates how many nodes $v$ and $w$ share *w.r.t.* the overall number of their neighborhood nodes. By definition, $\sigma(v, w)$ becomes non-zero only if $v$ is directly connected to $w$ with an edge. The value of $\sigma(v, w)$ is in the range of 0.0∼1.0 and especially becomes 1.0 when both $v$ and $w$ are in a clique.

$$\sigma(v, w) = \frac{|N(v) \cap N(w)|}{\sqrt{|N(v)| \times |N(w)|}} \tag{4}$$

By the cost embedding technique, our method calculates the similarity between $v$ and $w$ by using Eq. 5 instead of Eq. 4. In Eq. 5, $\sigma_t(v, w)$ indicates $\sigma(v, w)$ in $G_t$.

$$\sigma'_t(v, w) = \alpha \cdot \{\sigma_t(v, w) - \sigma_{t-1}(v, w)\} + \sigma_{t-1}(v, w) \tag{5}$$

Eq. 5 needs to look up the value of $\sigma_{t-1}(v, w)$, *i.e.*, $\sigma(v, w)$ in $G_{t-1}$, as well as $\sigma_t(v, w)$. Since the clustering algorithm might require multiple calculations of $\sigma'_t(v, w)$, it would be better to store the value of $\sigma'_t(v, w)$ after the first calculation for improving the clustering performance. We note that $\sigma'_t(v, w)$ becomes $\sigma_t(v, w)$ at time $t$+1, and so, $\sigma_{t-1}(v, w)$ in our method is the value already computed and stored at time $t - 1$ by using $\sigma_{t-2}(v, w)$. Since $\sigma_t(v, w)$ is non-zero only if there is an edge between $v$ and $w$, we only need to calculate and store $\sigma'_t(v, w)$ as many as the number of edges in $G_t$, *i.e.*, $|E_t|$. Thus, both the time complexity and space complexity of calculating all similarity values required during the whole clustering task on $G_t$ is $O(|E_t|)$.

Now, we summarize the notions of density-based clustering using Eq. 5 through Definitions 3∼9. The similar notions are used in other density-based clustering methods such as DBSCAN [8], SCAN [26], and TRACLUS [11] for points, static network, and trajectory data, respectively. Intuitively, the clustering algorithm of these notions takes two density parameters, $\mu_t$ and $\varepsilon_t$, and discovers all *topologically dense subgraph* on $G_t$ as in Figure 3(a). We note that the density-based clustering approach does not partition the network, and there remain some noise nodes after clustering. In Figure 3, the white colored node '7' indicates a noise node. Since the density-based clustering of these notions visits each node only once and checks the connectivity between the node and its neighborhood nodes, the time complexity becomes $O(|E_t|)$. We omit the detailed algorithm because it is basically similar with the existing density-based clustering algorithms.

**Definition 3.** The *$\varepsilon$-neighborhood* $N_\varepsilon(v)$ of a node $v \in V_t$ is defined by $N_\varepsilon(v) = \{x \in N(v) \mid \sigma'_t(v, x) \geq \varepsilon_t\}$.

**Definition 4.** A node $v \in V_t$ is called a *core node w.r.t.* $\varepsilon_t$ and $\mu_t$ if $|N_\varepsilon(v)| \geq \mu_t$.

**Definition 5.** A node $x \in V_t$ is *direct reachable* from a node $v \in V_t$ *w.r.t.* $\varepsilon_t$ and $\mu_t$ if (1) $v$ is a core node and (2) $x \in N_\varepsilon(v)$.

**Definition 6.** A node $v_j \in V_t$ is *reachable* from a node $v_i \in V_t$ *w.r.t.* $\varepsilon_t$ and $\mu_t$ if there is a chain of nodes $v_i, v_{i+1}, \ldots, v_{j-1}, v_j \in V_t$ such that $v_{i+1}$ is direct reachable from $v_i$ $(i < j)$ *w.r.t.* $\varepsilon_t$ and $\mu_t$.

**Definition 7.** A node $v \in V_t$ is *connected* to a node $w \in V_t$ *w.r.t.* $\varepsilon_t$ and $\mu_t$ if there is a node $x \in V_t$ such that both $v$ and $w$ are reachable from $x$ *w.r.t.* $\varepsilon_t$ and $\mu_t$.

**Definition 8.** A non-empty subset $S \subseteq V_t$ is called a *connected cluster w.r.t.* $\varepsilon_t$ and $\mu_t$ if $S$ satisfies the following two conditions:

(1) *Connectivity*: $\forall v, w \in S$, $v$ is connected to $w$ *w.r.t.* $\varepsilon_t$ and $\mu_t$

(2) *Maximality*: $\forall v, w \in V_t$, if $v \in S$ and $w$ is reachable from $v$ *w.r.t.* $\varepsilon_t$ and $\mu_t$, then $w \in S$.

**Definition 9.** Let $P$ be a set of connected clusters found by Definition 8. A node $v \in V_t$ is a *noise* if $v$ is not contained in any cluster of $P$.

Sometimes, new nodes are attached to $G_t$ or existing nodes of $G_{t-1}$ are detached from $G_t$. In the case of detachment, we do not need to do something because Eq. 5 would be calculated for only pairs of nodes that exist in $G_t$. However, in the case of attachment, where $v, x \in V_t$, but $v \in V_{t-1}$ and $x \notin V_{t-1}$, *i.e.*, $x$ is attached to $G_t$, we need to determine the value of $\sigma_{t-1}(v, x)$ for calculating $\sigma'_t(v, x)$.

Our method sets $\sigma_{t-1}(v, x)$ to the largest similarity value to follow the semantics of temporal smoothing. Suppose $\sigma_{t-1}(v, x) = \varepsilon_t - 0.01$. When $\alpha = 1$, $\sigma'_t(v, x) = \sigma_t(v, x)$ in Eq. 5, and so, $\mathcal{CR}_t$ becomes $\mathcal{CR}_O$, which obeys the semantics of temporal smoothing. When $\alpha = 0$, $\sigma'_t(v, x) = \sigma_{t-1}(v, x)$,

and all attached nodes $\{x\}$ become noise nodes since their similarity values do not satisfy the condition of Definition 3. Accordingly, except for the detached nodes, $\mathcal{CR}_t$ becomes identical with $\mathcal{CR}_{t-1}$, which obeys the semantics of temporal smoothing as well. When $0 < \alpha < 1$, $\sigma'_t(v,x)$ becomes a value between $\sigma_{t-1}(v,x)\,(=\varepsilon_t - 0.01)$ and $\sigma_t(v,x)$. If we set $\sigma_{t-1}(v,x)$ to a very small value (*e.g.*, 0), the nodes $\{x\}$ tend not to appear in the resulting clusters even when $\alpha$ is fairly large. Thus, we set $\sigma_{t-1}(v,x) = \varepsilon_t - 0.01$.

## 4.3 Clustering of Optimal Modularity

The density-based clustering of Section 4.2 requires two kinds of user-defined parameters, $\mu_t$ and $\varepsilon_t$, where $\mu_t$ is for specifying the minimum size of cluster, and $\varepsilon_t$ for specifying the minimum similarity between nodes within a cluster. For two parameters, it has been known that the clustering result is sensitive to $\varepsilon_t$, but not much sensitive to $\mu_t$. We propose the method to determine $\varepsilon_t$ automatically by using the novel concept of *modularity* [19].

There are several well-known quality measures for graph partitioning (or clustering) such as min-max cut, normalized cut, and modularity. Among them, min-max cut and normalized cut can only measure the quality of a binary partitioning. Modularity, however, can measure the quality of a partitioning into multiple clusters, and thus, it is more suitable for our problem. Instead of the original modularity that uses the number of edges or node degrees in each cluster, we adopt the extended modularity [9] that uses the structure similarity, which has been known to be more effective than the original modularity. The extended modularity $Q_s$ is defined as follows:

$$Q_S = \sum_{c=1}^{NC} \left[ \frac{IS_c}{TS} - \left( \frac{DS_c}{TS} \right)^2 \right], \tag{6}$$

where $NC$ is the number of clusters, $TS$ the total similarity between all pairs of nodes in the graph, $IS_c$ the total similarity of a pair of nodes within a cluster $c$, $DS_c$ the total similarity between a node in the cluster $c$ and any node in the graph.

The optimal clustering is achieved by maximizing $Q_S$, which has been known to be NP-complete. There have been several heuristic algorithms for maximizing modularity. They all are classified into two categories: (1) bottom-up (agglomerative) clustering [18]; and (2) top-down (divisive) clustering [28]. The former starts from all nodes, *i.e.*, $|V_t|$ clusters, and the latter from a graph $G_t$ itself, *i.e.*, a single cluster. They merge or split the graph iteratively until it reaches the maximum modularity. The performance of both approaches could be degraded when the size of network is large because the suitable number of clusters lies between 1 and $|V_t|$ and increases as the network size gets larger, which would require a lot of iteration steps.

In lieu of both approaches, we propose another approach that directly *jumps into the middle point* between two extremes, *i.e.*, 1 and $|V_t|$. This approach takes advantage of the feature of density-based clustering that discovers all clusters above a given density threshold (*i.e.*, $\varepsilon_t$) quickly. Our approach first performs clustering with the initial density parameter $seed_{\varepsilon_t}$, and then, decreases or increases it until reaching the maximum modularity. In fact, the maximum point reached by our approach might be a local maximum point, not a global maximum one. The previous two approaches also suffer from the local maximum problem. Solving this problem is beyond the scope of this paper.

The more smartly we choose the seed density parameter, the more quickly we reach the maximum point. Here, we present a simple heuristic method that chooses a median value of $\mu_t$-th similarity values of the sample nodes picked from $V_t$. Here, a sampling rate of only about 5∼10% would be sufficient. Each node $v$ has $|N(v)|$ similarity values with its neighborhood nodes. The reason of using the $\mu_t$-th value among them is that the clustering result is mainly determined by the $\mu_t$-th value in Definition 4. The reason of choosing the median value is that it would produce an intermediate number of clusters between 1 and $|V_t|$.

For efficient clustering, our method increases or decreases the density by a unit $\Delta_\varepsilon$ (*e.g.*, 0.01 or 0.02) and maintains two kinds of heaps: (1) max heap $H_{max}$ for edges having similarity below $seed_{\varepsilon_t}$; and (2) min heap $H_{min}$ for edges having similarity above $seed_{\varepsilon_t}$. $H_{max}$ and $H_{min}$ are built during the initial clustering. After finding the initial clusters $\mathcal{CR}_t$ and calculating its modularity $Q_{mid}$, our method calculates two additional modularity values, $Q_{high}$ and $Q_{low}$. Here, $Q_{high}$ is calculated from $\mathcal{CR}_t$ with the edges having similarity of $[seed_{\varepsilon_t}, seed_{\varepsilon_t} + \Delta_\varepsilon]$ in $H_{min}$, and $Q_{low}$ calculated from $\mathcal{CR}_t$ except the edges having similarity of $[seed_{\varepsilon_t} - \Delta_\varepsilon, seed_{\varepsilon_t}]$ in $H_{max}$. If $Q_{high}$ is the highest among $Q_{high}$, $Q_{mid}$, and $Q_{low}$, our method increases the density by $\Delta_\varepsilon$. If $Q_{low}$ is the highest value, our method decreases the density by $\Delta_\varepsilon$. Otherwise, $seed_{\varepsilon_t}$ would be the best density parameter. The initial clustering result $\mathcal{CR}_t$ is continuously modified by adding edges from $H_{max}$ to $\mathcal{CR}_t$ or by deleting edges of $H_{min}$ from $\mathcal{CR}_t$. Actually, this algorithm corresponds to the bulk version of the incremental density-based clustering algorithm [7]. When the algorithm stops, the modified $\mathcal{CR}_t$ becomes the final clustering result. The basic outline of the algorithm is as follows:

1. Performs clustering on $G_t$ with $\varepsilon_t = seed_{\varepsilon_t}$, builds $H_{max}$ and $H_{min}$, and calculates $Q_{mid}$.
2. Calculates $Q_{high}$ and $Q_{low}$, and then,
   (1) if $Q_{high}$ is the highest value: increases $\varepsilon_t$ by $\Delta_\varepsilon$ and calculates a new modularity $Q'_{high}$.
   (2) if $Q_{low}$ is the highest value: decreases $\varepsilon_t$ by $\Delta_\varepsilon$ and calculates a new modularity $Q'_{low}$.
3. Repeats Step 2(1) or 2(2) until the modularity does not increase any more.

# 5. MAPPING OF LOCAL CLUSTERS

Now, we can find the temporally smoothed local clusters by using the method of Section 4, but do not know yet how to map between $C_{t-1} \in \mathcal{CR}_{t-1}$ and $C_t \in \mathcal{CR}_t$ for finding a variable number of communities of arbitrary forming and dissolving. We note that the $t$-partite graph already contains a fixed set of relatively densely connected subsets of nano communities, *i.e.*, *data-inherent* quasi $l$-$KK$s, each of which has its own rough start/stop points. After finding a set of local clusters (*i.e.*, cross sections) by a given $\alpha$, our method tries to make sequences of such local clusters as close as possible to data-inherent quasi $l$-$KK$s. That is, our method finds a set of quasi $l$-$KK$s that become different by different $\alpha$, at the same time, are close to data-inherent quasi $l$-$KK$s.

This is achieved by taking advantage of link density between timestamp networks. Each $C_t \in \mathcal{CR}_t$ is connected with some local clusters $C_{t-1} \in \mathcal{CR}_{t-1}$ through links, and

so, each $C_t$ has its own distribution of the number of links over $\mathcal{CR}_{t-1}$. Some mapping between $C_{t-1}$ and $C_t$ would have a lot of links, and so, the bipartite graph between $C_{t-1}$ and $C_t$ is dense. On the other hand, some mapping would have only a small number of links, and so, the link density is low.

In the model of quasi $l$-$KK$, we consider that, if the density of the bipartite graph between $M_{t-1}$ and $M_t$, *i.e.*, $\Theta_{KK}(B_{t-1,t})$ is high, then the community $\mathcal{M}$ would be continue (evolve) from $t-1$ to $t$. Such interpretation would be quite natural. If there is no local cluster $C_{t-1}$ that $C_t$ is densely connected with, then $C_t$ would be the begin of a new community $\mathcal{M}$ where $C_t = M_t$. Likewise, if there is no local cluster $C_t$ that $C_{t-1}$ is densely connected with, then $C_{t-1}$ would be the end of the existing community $\mathcal{M}$ where $C_{t-1} = M_{t-1}$.

We summarize all cases of relationships between $C_{t-1}$ and $C_t$ representing the *evolving, dissolving, or forming* of $\mathcal{M}$. Here, $\delta_{threshold}$ indicates a certain density threshold to determine the case. The stage of the evolving is again classified into three cases — *growing, shrinking, and drifting* — according to the relationship between $M_{t-1}$ and $M_t$. Sometimes, a community splits into multiple communities, or multiple communities merge into one community in real data, but we do not present such cases in this paper due to space limit.

(1) Evolving ($\Theta_{KK}(B_{t-1,t}) > \delta_{threshold}$): a community $\mathcal{M}$ *evolves* from $t-1$ to $t$.
   A. Growing ($M_{t-1} \subseteq M_t$): $\mathcal{M}$ *grows* between $t-1$ and $t$.
   B. Shrinking ($M_{t-1} \supseteq M_t$): $\mathcal{M}$ *shrinks* between $t-1$ and $t$.
   C. Drifting ($|M_{t-1} \cap M_t| \neq 0$): $\mathcal{M}$ *drifts* between $t-1$ and $t$.

(2) Forming ($\neg\exists C_{t-1} \in \mathcal{CR}_{t-1}$ *s.t.* $\Theta_{KK}(B_{t-1,t}) > \delta_{threshold}$): a community $\mathcal{M}$ *forms* at time $t$.

(3) Dissolving ($\neg\exists C_t \in \mathcal{CR}_t$ *s.t.* $\Theta_{KK}(B_{t-1,t}) > \delta_{threshold}$): a community $\mathcal{M}$ *dissolves* at time $t$.

For explanation, we use a fixed threshold $\delta_{threshold}$ above, but actually it might be very difficult or impossible to determine a single suitable $\delta_{threshold}$ since there could be multiple mappings between $\{C_{t-1}\}$ and $C_t$ or between $C_{t-1}$ and $\{C_t\}$ satisfying the condition of evolving case. For solving this, we propose a mapping method based on information theory, especially *mutual information*, in Sections 5.1∼5.3.

## 5.1 Link Counting

The mapping task is performed based on the number of links between a pair of local clusters. That is, we need to count the number of links for all pairs of local clusters, $\mathcal{CR}_{t-1} \times \mathcal{CR}_t$. We propose a method that counts the number of links in a single scan even when there are no stored links. Under the model of $t$-partite graph and quasi $l$-$KK$, we could derive the relationship between the number of nodes/edges and the number of links as in Lemma 1.

**Lemma 1.** Given a pair of local clusters $C_{t-1} \in \mathcal{CR}_{t-1}$ and $C_t \in \mathcal{CR}_t$, the number of links between $C_{t-1}$ and $C_t$, *i.e.*, $|B_{t-1,t}|$ satisfies Eq. 7, where $||_{node}$ indicates the number of nodes, and $||_{edge}$ the number of edges.

$$|B_{t-1,t}| = |C_{t-1} \cap C_t|_{node} + 2 \cdot |C_{t-1} \cap C_t|_{edge} \quad (7)$$

PROOF. First, there is a link between every pair of nodes of the same ID by the definition of $\Gamma_E()$. Thus, there is a total of $|C_{t-1} \cap C_t|_{node}$ links for them. In Figure 4, the thick dash lines indicate such links. Second, there are two links between every common edge $\langle x, y\rangle$ by the definition of $\Gamma_E()$, where one link is between $x$ at time $t-1$ and $y$ at time $t$, and another link between $y$ at time $t-1$ and $x$ at time $t$. Thus, there is a total of $2 \cdot |C_{t-1} \cap C_t|_{edge}$ links for them. In Figure 4, the thin dash lines indicate such links. □

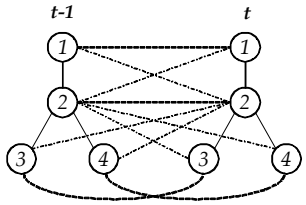


**Figure 4: An example for link counting.**

For mapping task, we record the number of links in a matrix of size $|\mathcal{CR}_{t-1}| \times |\mathcal{CR}_t|$ denoted as $Mat_{t-1,t}$. After initializing all cells as zero, our method scans each node and edge of $G_t$ and checks whether there is the same node or edge in $G_{t-1}$. If there is the same node, our method increases the link count of the corresponding cell of $Mat_{t-1,t}$ by 1, and if there is the same edge, it increases the link count by 2. For example, the upper left 6-cell matrix in Figure 5 shows the result of link counting for $G_{t-1}$ and $G_t$ in Figure 3. Here, $R_t$, $Y_t$, and $B_t$ indicate the local clusters of red color, green color, and blue color, respectively. The value in the cell $Mat_{t-1,t}[R_{t-1}][R_t]$, *i.e.*, 14 is the number of links between $R_{t-1}$ and $R_t$. We note that the value of each cell is changed by the parameter $\alpha$ even though the links themselves between $G_{t-1}$ and $G_t$ are invariable.

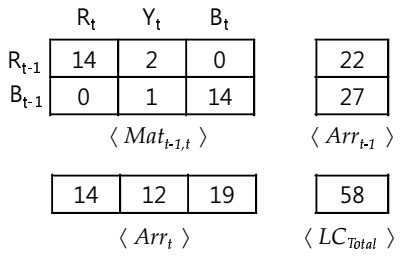


**Figure 5: A matrix and arrays of link counting.**

In addition to the link count between $C_{t-1}$ and $C_t$, we need to count the number of links from $C_{t-1}$ to $G_t$ and that from $C_t$ to $G_{t-1}$ for analysis of mutual information. In Figure 5, the upper right 2-cell array $Arr_{t-1}$ shows the link counts from $C_{t-1}$ to $G_t$, and the lower left 3-cell array $Arr_t$ that from $C_t$ to $G_{t-1}$. We note that the link count between $C_{t-1}$ and $C_t$ is symmetric, but that between $C_t$ and $G_{t-1}$ or that between $C_{t-1}$ and $G_t$ is non-symmetric. In Figure 5, the lower right 1-cell $LC_{Total}$ indicates the number of total links between $G_{t-1}$ and $G_t$. We also note that there is a relationship of Eq. 8 among those link counts, where Eq. 8 is still valid even if we replace $Arr_{t-1}$ with $Arr_t$. This relationship comes from the differences in the scope of link counting. For example, in Figure 3(a), the edge $\langle 3, 5\rangle$ in $R_{t-1}$ has a counterpart common edge in the scope of the whole network

$G_t$, and so, is counted in $Arr_{t-1}$. The edge, however, does not have any counterpart in the scope of the local cluster either $R_t$ or $Y_t$, and so, is not counted in $Mat_{t-1,t}$.

$$\sum_{i,j} Mat_{t-1,t}[i][j] \;\leq\; \sum_{k} Arr_{t-1}[k] \;\leq\; LC_{Total} \tag{8}$$

## 5.2 Mutual Information

The mutual information is one of the important entropy measures in information theory. Eq. 9 shows the equation of the mutual information. It originally means the dependency degree between two random variables $X$ and $Y$. From a different angle, it means the distance (especially called the Kullback-Leibler distance) between two distributions $P(X,Y)$ and $P(X)P(Y)$. If the distribution of $P(X)$ and $P(Y)$ is purely random as in Figure 6(a), $MI(X;Y)$ becomes 0. On the contrary, if the distribution of $P(X)$ and $P(Y)$ is skewed as in Figure 6(b), $MI(X;Y)$ becomes high. In Figure 6(b), if we set the relatively low probability value as zero, *i.e.*, *purify* the distribution more (or decrease the entropy), $MI(X;Y)$ increases. We take advantage of this feature of mutual information for mapping local clusters.

$$MI(X;Y) = \sum_i \sum_j P(x_i, y_j)\ log \frac{P(x_i, y_j)}{P(x_i)P(y_j)} \tag{9}$$

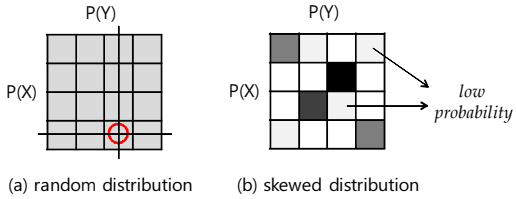


**Figure 6: Two distributions of $P(X)$ and $P(Y)$.**

We take the distributions of link counts of $\mathcal{CR}_{t-1} \times \mathcal{CR}_t$ into account for utilizing the mutual information. If $x_i$ and $y_i$ in Eq. 9 are subsets of objects, Eq. 9 could be rewritten as in Eqs. 10∼11. In Eq. 11, $N$ indicates the total number of objects. In our problem, $x_i$ and $y_i$ correspond to sets of links of $(C_{t-1} \rightarrow G_t)$ and $(G_{t-1} \leftarrow C_t)$, respectively, and $(x_i \cap y_j)$ to a set of links of $(C_{t-1} \leftrightarrow C_t)$. Thus, by using $Mat_{t-1,t}$, $Arr_{t-1}$, $Arr_t$, and $LC_{Total}$, we could derive Eq. 12 that calculates the mutual information of links between $G_{t-1}$ and $G_t$.

$$MI(X;Y) = \sum_i \sum_j P(x_i \cap y_j)\ log \frac{P(x_i \cap y_j)}{P(x_i)P(y_j)} \tag{10}$$

$$= \sum_i \sum_j \frac{|x_i \cap y_j|}{N}\ log \frac{N \cdot |x_i \cap y_j|}{|x_i| \cdot |y_j|} \tag{11}$$

$$= \sum_i \sum_j \frac{Mat_{t-1,t}[i][j]}{LC_{Total}}\ log \frac{LC_{Total} \cdot Mat_{t-1,t}[i][j]}{Arr_{t-1}[i] \cdot Arr_t[j]} \tag{12}$$

A mapping between $C_{t-1}$ and $C_t$ means getting rid of links of $(C_{t-1} \rightarrow G_t)$ and $(G_{t-1} \leftarrow C_t)$ except links of $(C_{t-1} \leftrightarrow C_t)$. This is equivalent to making the values of $Mat_{t-1,t}[i][C_t]$ zero $(1 \leq i \leq |\mathcal{CR}_{t-1}|$ and $i \neq C_{t-1})$ and making the values of $Mat_{t-1,t}[C_{t-1}][j]$ zero $(1 \leq j \leq |\mathcal{CR}_t|$ and $j \neq C_t)$. For example, Figure 7(a) shows the result after mapping between $R_{t-1}$ and $R_t$. This purifying process increases the mutual information. The initial mutual information in Figure 5 is 0.287, which increases to 0.315 after mapping in Figure 7(a). Likewise, in Figure 7(b), the mutual information increases to 0.345 by mapping between $B_{t-1}$ and $B_t$.

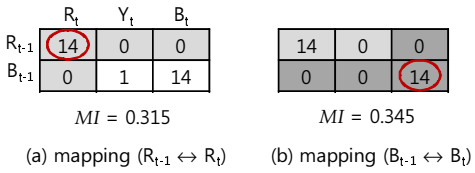


**Figure 7: $Mat_{t-1,t}$ and $MI$ during mapping process.**

Once a mapping between $C_{t-1}$ and $C_t$ is done, other local clusters of $\mathcal{CR}_{t-1}$ cannot be connected with $C_t$, or other local clusters of $\mathcal{CR}_t$ with $C_{t-1}$, in our semantics. Thus, at most $min(|CR_{t-1}|, |CR_t|)$ mapping operations are possible. As mentioned before, our method tries to make communities as coincident as possible with data-inherent quasi $l$-$KK$s. We consider that the higher mutual information the set of mapping operations achieves, the more coincident with data-inherent quasi $l$-$KK$s the resulting communities are. We can choose at most $min(|CR_{t-1}|, |CR_t|)$ pairs among $|\mathcal{CR}_{t-1}| \times |\mathcal{CR}_t|$ pairs to achieve the maximum mutual information. It is a combinatorial optimization problem, and in general, very difficult to find an optimal combination. We present a heuristic algorithm for this in Section 5.3.

## 5.3 Heuristic Algorithm

Our proposed greedy algorithm first performs mapping between two local clusters producing the maximum gain of the mutual information. Then, it repeatedly performs the same mapping operation for the remaining pairs of local clusters. Algorithm 1 outlines the pseudo-code of this algorithm. $Unit_{MI}()$ indicates a function that calculates the body part of Eq. 12 for $Mat_{t-1,t}[i][j]$. Here, we denote the result of $Unit_{MI}()$ as $Unit_{MI}[][]$. We note that $Unit_{MI}[i][j]$ is not necessarily proportional to the number of the links between $C_{t-1}$ and $C_t$. For example, in Figure 5, $Mat_{t-1,t}[R_{t-1}][R_t] = Mat_{t-1,t}[B_{t-1}][B_t] = 14$, but $Unit_{MI}[R_{t-1}][R_t] = 0.234 \neq Unit_{MI}[B_{t-1}][B_t] = 0.111$. The algorithm uses a max heap $H_{max}$ to choose the matrix cell having the largest unit $MI$.

Sometimes, the mapping operation produces the minus gain of mutual information, *i.e.*, $\Delta_{MI} < 0$, which typically occurs in a situation where the link distributions of the corresponding row and column is not skewed, but random, like the circle in Figure 6(a). It means the mapping would not be a good choice in the aspect of information theory. In an early stage of the mapping process, $\Delta_{MI}$ is fairly high, but it becomes smaller as the mapping is performed repeatedly. If $\Delta_{MI}$ becomes less than zero, the algorithm stops even if $H_{max}$ is not empty.

We can determine the case of each community by looking at the purified $Mat_{t-1,t}$. Let $N_{map}$ be the number of mapping operations performed until the stop. Then, it indicates there is a total of $N_{map}$ communities that evolve (continue) from $t-1$ to $t$. It also indicates there is a total of $(|\mathcal{CR}_{t-1}| - N_{map})$ communities that dissolve at time $t$, and a total of $(|\mathcal{CR}_t| - N_{map})$ communities that newly form at time $t$. For

each dissolved community of the above ($|\mathcal{CR}_{t-1}| - N_{map}$) communities, its $\Delta_{MI} < 0$ indicates that there is no corresponding $C_t$ having a relatively high $\Theta_{KK}(B_{t-1,t})$ value. Likewise, for each forming community, its $\Delta_{MI} < 0$ indicates that there is no corresponding $C_{t-1}$ having a relatively high $\Theta_{KK}(B_{t-1,t})$ value. Therefore, we could say that the result of the mapping method based on the mutual information is quite reasonable and coincides with the meaning of real community.

**Algorithm 1** Greedy Algorithm

**Input:** (1) Matrix $Mat_{t-1,t}$,
(2) Arrays $Arr_{t-1}$ and $Arr_t$,
(3) Total number of links $LC_{Total}$.
**Output:** Purified version of $Mat_{t-1,t}$.

```
/* Initialization */
Compute all Unit_MI[i][j] = Unit_MI(Mat_{t-1,t}[i][j]);
Insert all (Unit_MI[i][j], i, j) into a max heap H_max;
/* Iterative phase */
while H_max ≠ ∅ do
    Choose (Unit_MI[p][q], p, q) ∈ H_max with the largest
    Unit_MI[p][q] value;
    Δ_MI = −Σ_i Unit_MI[i][q] − Σ_j Unit_MI[p][j]
           + 2 * Unit_MI[p][q];
    if Δ_MI > 0 then
        Set all Mat_{t-1,t}[p][j] and all Mat_{t-1,t}[i][q]
        as zero value except Mat_{t-1,t}[p][q];
        Delete all (Unit_MI[p][j], p, j) from H_max;
        Delete all (Unit_MI[i][q], i, q) from H_max;
    else
        Break the while loop;
return Mat_{t-1,t};
```

# 6. EXPERIMENTAL EVALUATION

In this section, we evaluate the effectiveness and efficiency of our method. We conduct all the experiments on a Pentium Core2 Duo 2.0 GHz PC with 2 GBytes of main memory, running on Windows XP. We implement our algorithm in C++ using Microsoft Visual Studio 2005. In all experiments, we use $\mu_t = 2$.

## 6.1 Synthetic Data

We compare the effectiveness and efficiency of our method and the previous method FacetNet [15] by using synthetic data. FacetNet is the current state-of-art method for discovering evolutionary clusters for dynamic networks. We generate two kinds of data sets: (1) dynamic network of a fixed number of communities (*SYN-FIX*); and (2) dynamic network of a variable number of communities (*SYN-VAR*).

### 6.1.1 Data generation and quality measure

For SYN-FIX, we use the data generating method proposed by Newman et al. [19], which has been also used in FacetNet. One network contains 128 nodes, which are divided into 4 communities of 32 nodes each. We generate such network for 10 consecutive timestamps (*i.e.*, $\mathcal{G} = \{G_1, \ldots, G_{10}\}$). In order to introduce dynamics into $\mathcal{G}$, we randomly select 3 nodes from each community in $G_{t-1}$ and make those nodes leave their original communities and join randomly the other three communities in $G_t$. Edges are placed independently and randomly between a pair of nodes within the same community with a higher probability $p_{in}$ and between a pair of nodes of different communities with a lower probability $p_{out}$. The value of $p_{in}$ and $p_{out}$ are chosen such that the average degree of each node is set to be 16. In order to control the noise level in the dynamic network $\mathcal{G}$, we introduce a single parameter $z_{out}$, which represents the average number of edges from a node to nodes in other communities. If we decrease $z_{out}$, then $p_{in}$ becomes larger, and at the same time, $p_{out}$ becomes smaller such that the average degree of each node becomes 16. That is, the noise level decreases. On the other hand, if we increase $z_{out}$, the noise level in $\mathcal{G}$ increases.

For SYN-VAR, we modify the generating method for SYN-FIX so as to introduce the forming and dissolving of communities and the attaching and detaching of nodes. The initial network contains 256 nodes, which are divided into 4 communities of 64 nodes each. We generate 10 consecutive networks. We randomly choose 8 nodes from each community and make those 32 nodes a new community, which is lasting for 5 timestamps and its nodes return to the original communities. We perform this creation of a new community once at each timestamp between $2 \leq t \leq 5$, and so, the numbers of communities between $1 \leq t \leq 10$ are 4, 5, 6, 7, 8, 8, 7, 6, 5, and 4, respectively. We set the average degree of each node to be a half of the size of the cluster where the node belongs to. We also randomly delete exiting 16 nodes from each network and randomly add new 16 nodes to each network for $2 \leq t \leq 10$.

Since we have the ground truth answer for communities and their membership at each timestamp, we can directly measure the accuracy of the clustering result. For the measure, we use the normalized mutual information ($NMI$), which is a well-known external criteria and has been also used in FacetNet. $NMI$ is defined as in Eq. 13, which is the normalization of $MI$ by the average of two entropies $H(X)$ and $H(Y)$. The reason of using $NMI$ instead of $MI$ is that the method producing more clusters tends to obtain a higher score than that producing less clusters. The value of $NMI$ is in the range of 0.0∼1.0 by normalization, and a higher $NMI$ indicates better accuracy. We note that the purpose of using $NMI$ here is totally different from that of using $MI$ in Section 5. This $NMI$ is about distance between the clustering result and the ground truth at each time. On the contrary, $MI$ in Section 5 is about distance between two consecutive networks, which is moreover continuously changed through the purifying process.

$$NMI(X;Y) = \frac{MI(X;Y)}{[H(X) + H(Y)]/2} \tag{13}$$

### 6.1.2 Effectiveness

Figure 8 shows the accuracies of clustering results for SYN-FIX. When $z_{out} = 3$ in Figure 8(a), *i.e.*, when it is easy to detect the community structure due to the low noise level, both FacetNet and our method achieve very high accuracy. When $z_{out} = 5$ in Figure 8(b), *i.e.*, when the noise level is high, however, the accuracies of both methods become much lower. We note that the accuracies of both methods continuously increase as time goes on in Figure 8(b). It is because the clustering result or the individual distance at time $t-1$ affects on that at time $t$ by temporal smoothing. There

is a similar phenomenon at the initial timestamps in Figure 8(a), but the accuracies are not improved continuously because they are already so high.

Figure 9 shows the accuracies of clustering results for SYN-VAR. In both figures, our method significantly improves the accuracy compared with FacetNet. When new communities form or existing communities dissolve, our method recognizes the change of the number of communities and the start/stop points of communities, so achieves high accuracy. FacetNet, however, could not deal with such things, and thus, its accuracy becomes largely degraded. Especially, the accuracy of FacetNet around at the middle timestamps ($accumulatedt = 4 \sim 6$) is worse than that at the initial timestamps or the final timestamps. It is because the difference between the numbers of communities in the ground truth and the clustering result of FacetNet is maximized at those timestamps. The accuracy when $z_{out} = 5$ is much lower than that when $z_{out} = 3$, which is a similar result with Figure 8, and it does not increase over time. The reason is that temporal smoothing for SYN-VAR hardly gets the accumulated benefits due to the continuous change of the community structure and the attachment/detattchment of nodes.

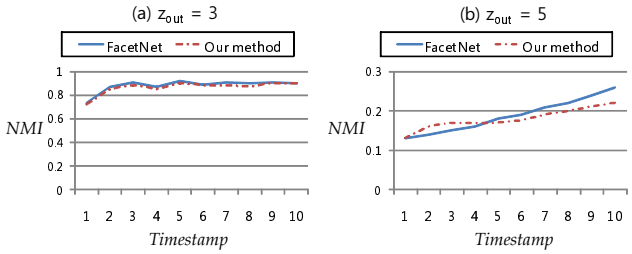


**Figure 8: Accuracies of clustering results for SYN-FIX ($\alpha = 0.8$).**

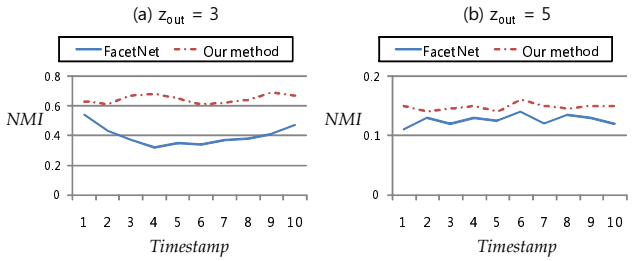


**Figure 9: Accuracies of clustering results for SYN-VAR ($\alpha = 0.8$).**

### 6.1.3 *Efficiency*

Figure 10(a) shows the time performances of our method and FacetNet for SYN-VAR as the size of network varies. Here, the size of network indicates the number of nodes per timestamp network. We generate four data sets of different sizes such that the number of clusters increase. We perform clustering over 10 timestamps, and present the average result. We use $z_{out} = 3$ and $\alpha = 0.8$, but the variation of $z_{out}$ or $\alpha$ did not influence the performance. In Figure 10(a), our method improves the performance by an order of magnitude compared with FacetNet. While FacetNet performs a lot of iterations (*e.g.*, 600) of matrix computation for temporal smoothing, our method performs clustering once due to the cost embedding technique. This result shows our method is more suitable for large-scale data. We skip the result for SYN-FIX and the result when the size of each cluster increases, which are similar with that for SYN-VAR in Figure 10(a).

In addition to SYN-VAR, Figure 10(b) shows the performance result for the real DBLP co-authorship data, where an author is a node, and a co-authorship is an edge. Our method still significantly improves the performance compared with FacetNet. We note that the running times increase a little more rapidly than in Figure 10(a). The reason is that the time complexity of our density-based clustering algorithm is $O(|E_t|)$, and $|E_t|$ increases a little more rapidly than $|V_t|$ in real networks due to the *densification laws* of networks [14]. The running time of FacetNet is also proportional to $|E_t|$ since it uses internally a soft clustering algorithm [27] that has the time complexity of almost $O(|E_t|)$.

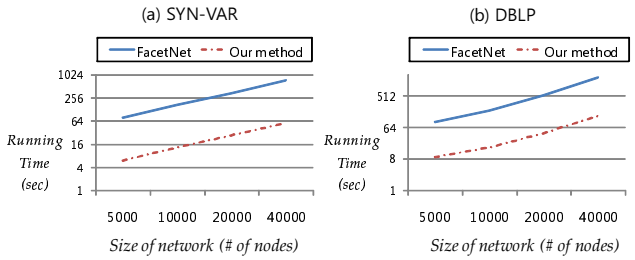


**Figure 10: Running times as the size of network varies.**

## 6.2 Real Data

We evaluate some features of our method by using two real data sets: DBLP data [6] and the football data [17]. The football data is the NCAA Football Division 1-A schedule data, where nodes are football teams, and edges are matches between teams. This data has been used for several network clustering studies [18, 26]. In particular, we study the data of the year 2006, which has been also used in SCAN [26]. In this data, there is a total of 179 nodes, and the number of clusters of the ground truth is eleven, which represent real eleven conferences like "Big Ten". For the DBLP data, we extract the co-authorship information from the original DBLP data, related to database or information area of the last 10 years from 1999 to 2008. This DBLP data contains a total of 127,214 unique authors.

### 6.2.1 *Football Data*

In Section 4.3, we have presented the clustering method that determines $\varepsilon_t$ so as to maximize modularity $Q_s$. Figure 11(a) and Figure 11(b) show the modularity $Q_s$ and the number of clusters, respectively, as $\varepsilon_t$ varies for the football data. When $\varepsilon_t = 1$, there is no cluster, *i.e.*, each node itself is a cluster since the density threshold is too high. Here, $Q_s$ becomes zero. On the other hand, when $\varepsilon_t = 0$, there is only a single cluster since the threshold is too low. Our method directly jumps into the middle point between $\varepsilon_t = 0$ and $\varepsilon_t = 1$. We choose a median value, 0.62, as a start point by the proposed heuristic method. Since the decreasing $\varepsilon_t$ can obtain higher $Q_s$, we continuously decrease $\varepsilon_t$ by $\Delta_\varepsilon = 0.02$

until $Q_s$ reaches the maximum point. The algorithm stops when $\varepsilon_t = 0.46$ and produces ten clusters, which are the same result with the ground truth except only one conference. The best $\varepsilon_t$ obtained manually is 0.5. We note that our method finishes clustering within eight iterative steps from 0.64 to 0.46, whereas the previous divisive method [28] finishes it with about 200 iterative steps.

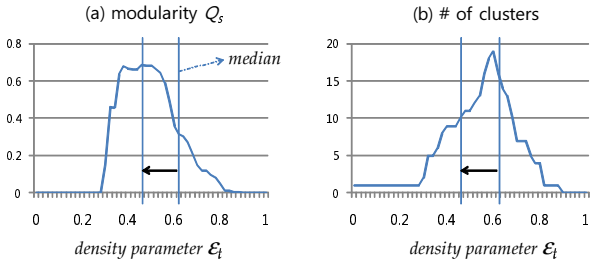


**Figure 11: Modularity $Q_s$ and the number of clusters as $\varepsilon_t$ varies ($\mu = 2$, $\Delta_\varepsilon = 0.02$).**

### 6.2.2 DBLP Data

In this section, we evaluate the effect of $\alpha$ using the DBLP data. The parameter $\alpha$ controls the trade-off between the snapshot quality and the history quality, and at the same time, the frequency of forming/dissolving of communities. Actually, the trade-off and the frequency are closely related with each other in the case of allowing arbitrary start/stop of communities. Figure 12 shows the average length of communities and the number of communities as $\alpha$ varies between 0.1 and 0.9. When $\alpha = 0.9$, our method mainly focuses on the snapshot quality, and so, there is a good chance that a local cluster is not connected with other local cluster due to the low density between them. Thus, the average lifetime of community decreases while the number of communities increases. In contrast, when $\alpha = 0.1$, there is a good chance that a local cluster is connected with other local cluster, and so, the average lifetime of community increases while the number of communities decreases.

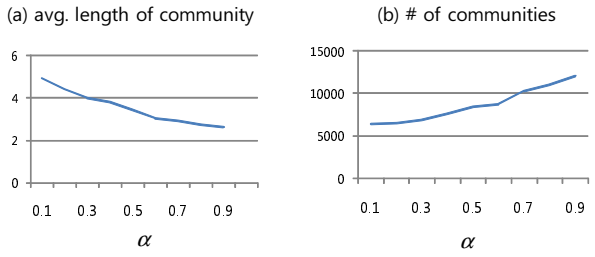


**Figure 12: Average length of communities and the number of communities as $\alpha$ varies ($0.1 \leq \alpha \leq 0.9$).**

## 7. RELATED WORK

There has been a lot of studies that analyze the characteristics of communities in dynamic networks. Several studies focus on identifying the properties or key features influencing the behavior of communities in dynamic networks [2, 10, 1, 13]. Especially, Asur et al. [1] has introduced a set of key events that occur in real dynamic networks. According to those events, communities can newly form (*i.e.*, start) or dissolve (*i.e.*, stop) at any time as well as continue with some change (*i.e.*, evolve). Some other studies such as GraphScope [20] focus on discovering communities in dynamic networks without temporal smoothing. GraphScope identifies the subgroup structure and the change points where the subgroup structure is largely changed, by employing the Minimum Description Length (MDL) principle. GraphScope, however, has a major weak point that it cannot catch the evolution of community individually.

Recently, a new clustering concept called *evolutionary clustering* and a framework called *temporal smoothness* have been proposed by Chakrabarti et al. [4]. Chi et al. [5] proposed two spectral clustering algorithms PCM and PCQ to incorporate this temporal smoothness framework. Lin et al. [15] proposed the more advanced algorithm called *FacetNet* for general dynamic networks. This is the current state-of-art method for discovering evolutionary clusters in dynamic networks and most closely related to our work. Tang et al. [21] proposed an evolutionary clustering algorithm based on a little different framework to handle multi-mode networks. However, all above studies have some drawbacks that they handle only a fixed number of communities over time, they do not allow arbitrary start/stop of community over time, or they are not very scalable due to the high computation cost caused by a lot of iterations of matrix calculation until converge. Chakrabarti et al. [4] used a similar technique with our cost embedding technique for preparing a similarity matrix, but their technique was not for temporal smoothing and was not presented in a formal definition.

In addition to evolutionary clustering, there have been few studies that discover a kind of communities over temporal text data [16] or blogosphere data [3]. Mei et al. [16] proposed an algorithm for discovering evolutionary theme from temporal text data. This algorithm extracts themes at each timestamp, constructs a $t$-partite graph by connecting relevant themes between adjacent two timestamps, and finds a sequence of themes as an evolutionary theme. Bansal et al. [3] also constructs a $t$-partite graph from temporal blogosphere data and finds the most *stable* top-k clusters for stable topics in blogoshere. These methods can find some evolutionary patterns of a variable number and arbitrary start/stop. However, they do not consider temporal smoothness at all. Furthermore, since they find communities in two separated stages without temporal smoothing, the resulting communities are rough and coarse granular. On the other hand, our method finds the communities at a fine granularity level by temporal smoothing based on nano communities.

## 8. CONCLUSION

In this paper, we have proposed a novel evolutionary clustering method, the particle-and-density based method, for efficiently discovering a variable number of communities in dynamic networks. Since real dynamic networks tend to be large, and at same time, the forming of new communities and dissolving of existing communities are very common phenomena in real data, a scalable and powerful evolutionary clustering method is required. Our method has modeled a dynamic network as a collection of lots of particles called nano-communities, and a community as a densely connected subset of such particles. Each particle contains a small amount of information about the evolution of data or communities. For flexible and efficient temporal smoothing, we have proposed a cost embedding technique, which is independent of both the similarity measure and the cluster-

ing algorithm. The density-based clustering algorithm using the cost embedding technique and optimal modularity measure efficiently discovers temporally smoothed local clusters of high quality. We have also proposed a mapping method based on information theory that can identify the stage of each community among the three stages — evolving, forming, and dissolving — by purifying link distribution between consecutive networks so as to maximize the mutual information. Experimental studies, by using two kinds of synthetic data sets and two kinds of real data sets, demonstrate that our method improves the clustering accuracy, and at the same time, the time performance by an order of magnitude compared with the state-of-the art method FacetNet.

## 9. ACKNOWLEDGMENT

The work was supported in part by the Korea Research Foundation grant funded by the Korean Government (MOE-HRD), KRF-2007-357-D00203, by the U.S. National Science Foundation grants IIS-08-42769 and BDI-05-15813, and the Air Force Office of Scientific Research MURI award FA9550-08-1-0265. Any opinions, findings, and conclusions expressed here are those of the authors and do not necessarily reflect the views of the funding agencies. We thank Yu-Ru Lin for providing us with the FacetNet code. We also thank the anonymous reviewers for their valuable comments that helped enhance the quality of this paper.

## 10. REFERENCES

[1] S. Asur, S. Parthasarathy, and D. Ucar. An event-based framework for characterizing the evolutionary behavior of interaction graphs. In *Proc. KDD 2007*, pages 913–921.

[2] L. Backstrom, D. Huttenlocher, X. Lan, and J. Kleinberg. Group formation in large social networks: Membership, growth, and evolution. In *Proc. KDD 2006*, pages 44–54.

[3] N. Bansal, F. Chiang, N. Koudas, and F. W. Tompa. Seeking stable clusters in the blogosphere. In *Proc. VLDB 2007*, pages 806–817.

[4] D. Chakrabarti, R. Kumar, and A. Tomkins. Evolutionary clustering. In *Proc. KDD 2006*, pages 554–560.

[5] Y. Chi, X. Song, D. Zhou, K. Hino, and B. L. Tseng. Evolutionary spectral clustering by incorporating temporal smoothness. In *Proc. KDD 2007*, pages 153–162.

[6] DBLP bibliographic data. http://www.informatik.uni-trier.de/ ley/db, 2009.

[7] M. Ester, H.-P. Kriegel, J. Sander, M. Wimmer, and X. Xu. Incremental clustering for mining in a data warehousing environment. In *Proc. VLDB 1998*, pages 323–333.

[8] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. KDD 1996*, pages 226–231.

[9] Z. Feng, X. Xu, N. Yuruk, and T. A. J. Schweiger. A novel similarity-based modularity function for graph partitioning. In *Proc. DaWaK 2007*, pages 385–396.

[10] R. Kumar, J. Novak, and A. Tomkins. Structure and evolution of online social networks. In *Proc. KDD 2006*, pages 611–617.

[11] J.-G. Lee, J. Han, and K.-Y. Whang. Trajectory clustering: A partition-and-group framework. In *Proc. SIGMOD 2007*, pages 593–604.

[12] E. A. Leicht, P. Holme, and M. E. J. Newman. Vertex similarity in networks. *Physical Review*, E73:026120, 2006.

[13] J. Leskovec, L. Backstrom, R. Kumar, and A. Tomkins. Microscopic evolution of social networks. In *Proc. KDD 2008*, pages 462–470.

[14] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: Densification laws, shrinking diameters and possible explanations. In *Proc. KDD 2005*, pages 177–187.

[15] 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 *Proc. WWW 2008*, pages 685–694.

[16] Q. Mei and C. Zhai. Discovering evolutionary theme patterns from text - an exploration of temporal text mining. In *Proc. KDD 2005*, pages 198–207.

[17] NCAA Football Division 1-A schedule data. http://www.jhowell.net/cf/scores/scoresindex.htm, 2009.

[18] M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Physical Review*, E69:066133, 2004.

[19] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review*, E69(2):026113, 2004.

[20] J. Sun, C. Faloutsos, S. Papadimitriou, and P. S. Yu. GraphScope: Parameter-free mining of large time-evolving graphs. In *Proc. KDD 2007*, pages 687–696.

[21] L. Tang, H. Liu, J. Zhang, and Z. Nazeri. Community evolution in dynamic multi-mode networks. In *Proc. KDD 2008*, pages 677–685.

[22] C. Tantipathananandh, T. Y. Berger-Wolf, and D. Kempe. A framework for community identification in dynamic social networks. In *Proc. KDD 2007*, pages 717–726.

[23] Telephone traffic data. http://reality.media.mit.edu/download.php, 2007.

[24] S. Wasserman and K. Faust. *Social Network Analysis: Methods and Applications*. Cambridge University Press, 1994.

[25] D. B. West. *Introduction to Graph Theory*. Prentice Hall, second edition, 2001.

[26] X. Xu, N. Yuruk, Z. Feng, and T. A. J. Schweiger. SCAN: A structural clustering algorithm for networks. In *Proc. KDD 2007*, pages 824–833.

[27] K. Yu, S. Yu, and V. Tresp. Soft clustering on graphs. *NIPS*, 2005.

[28] N. Yuruk, M. Mete, X. Xu, and T. A. J. Schweiger. A divisive hierarchical structural clustering algorithm for networks. In *Proc. ICDMW 2007*, pages 441–448.

[29] L. Zhao and M. J. Zaki. Tricluster: An effective algorithm for mining coherent clusters in 3D microarray data. In *Proc. SIGMOD 2005*, pages 694–705.