國立交通大學

資訊科學與工程研究所

碩 士 論 文

運用關係萃取策略於動態社群探勘之研究

Dynamic Community Detection via Relationship Extraction Strategy and Community Pedigree Mapping

研 究 生：彭誠毅

指導教授：李素瑛 教授

中 華 民 國 一百 年 七 月

運用關係萃取策略於動態社群探勘之研究
Dynamic Community Detection via Relationship Extraction Strategy and Community Pedigree Mapping

研 究 生：彭誠毅　　　　Student：Cheng-Yi Peng

指導教授：李素瑛　　　　Advisor：Suh-Yin Lee

國 立 交 通 大 學
資 訊 科 學 與 工 程 研 究 所
碩 士 論 文

A Thesis

Submitted to Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

July 2011

Hsinchu, Taiwan, Republic of China

中華民國一百年七月

運用關係萃取策略於動態社群探勘之研究

研究生：彭誠毅　　　　　　　　　　　　　　指導教授：李素瑛

國立交通大學資訊科學與工程研究所

摘要

近來在動態社會網路上，因為社群演化的廣大應用，動態社群的問題已經引起重大的關注。多數潛在的社會現象實際上可經由分析社群網路結構萃取出來。雖然在動態社群上已有多數的研究發表。它們一般而言均針對一連串的互動圖作社群分群，一連串的互動圖是通常用來表現動態社群網路的一種方式。然而互動圖展露在人與人之間的關係可能是不夠充足，因為它只是在一個時間切片上的快照。在這時間切片上兩人若沒有互動發生，並不代表這兩人沒有關係。本篇論文提出一個新穎的演算法，EPC(關係萃取和社群氏族的社群探勘者)，可被用來探勘社群演化。我們提出了一個關係萃取策略，在一個時間窗內產生出關係圖。EPC 架構於關係圖來產生社群分群，且利用社群氏族對映來發掘出動態社群在動態社群網路上的演化。實驗結果在合成資料和真實數據中顯示 EPC 的結果不僅在準確度比之前的方法佳，且在彈性和平滑程度也勝過之前的方法。

檢索詞：動態社群，關係萃取，衰減權重函數，社群氏族。

Dynamic Community Detection via Relationship Extraction Strategy and Community Pedigree Mapping

Cheng Yi Peng **Suh-Yin Lee**

Institute of Computer Science and Information Engineering
National Chiao-Tung University

Abstract

Recently, considerable attention has been paid to the issue of dynamic community in dynamic social network due to its widespread applications. Many potential social phenomena, in practice, can be extracted by analyzing the dynamic social structure over time. Although there have been many recent studies proposed on dynamic community, these works, generally, partition the community based on a sequence of interaction graphs, which is usually applied to express a dynamic social network. Nevertheless, the interaction graph may be insufficient to reveal the relationship among individuals, since, in a snapshot of time slice, no interaction among individuals does not indicate no actual relationship. In this thesis, a novel algorithm EPC, which stands for relationship Extraction and community Pedigree mapping Community miner, is developed to mine the evolution of community. We present a *Relationship Extraction* strategy to construct a relationship graph within a defined observation window. EPC partitions communities based on relationship graph and uses proposed *Community Pedigree Mapping* method to discover the evolution of dynamic community in dynamic social network. The experimental results on synthetic and real datasets show that EPC not only significantly outperforms the prior studies in accuracy but also possesses graceful scalability and smoothness.

Index terms: dynamic community, relationship extraction, decay weight function, pedigree of community

ACKNOWLEDGEMENT

I greatly appreciate the kind guidance of my advisor, Prof. Suh-Yin Lee. She not only helps with my research but also takes care of me. Her graceful suggestion and encouragement help me forward to complete this thesis.

Besides, I want to give my thanks to all members in the Information System Laboratory for their suggestion and instruction, especially Mr. Yi-Cheng Chen, Ms. Chao-Ying Wu, Mr. Wang-Zen Wang, Mr. Fan-Chung Lin and Mr. Li Yen Qua.

Especially I would express my thanks to Ms. Yu-Ru Lin for providing FacetNet code, Ms. Heli Sun for providing SHRINK code and Mr. Alan Mislove for providing the Facebook dataset.

Besides, I would express my thanks to the nicest and most beautiful lady Ms. Hui-yu Liou who accompanies me and always give me a happy smile.

Finally I would like to express my deepest appreciation to my parents. This thesis is dedicated to them.

Table of contents

List of Figures

List of Tables

Chapter 1.
Introduction

In recent years, social network analysis in e-commerce, social science and computer science has received significant attention. A social network is a social structure made up of individuals, which are tied by one or more specific types of relationship or interdependency, such as friendship, co-authorship, common interest or financial exchange. Clustering similar individuals into a group has been a big challenge. A cluster in a social network is typically called a community. Traditional static methods discovered communities using the aggregate interaction data and ignored the effect of time. However, social networks are dynamic and evolve over time. With the increasing popularity of social network websites, the use of dynamic social network analysis has increasingly been the focus of study in recent years.

Two major issues of dynamic community need to be addressed:

(1) Community discovery: Which nodes should be associated with each other to become a community at each timestamp? **(2) Evolution of community:** How to explain the evolution of community partition from previous timestamp to current timestamp?

Traditional methods of discovering dynamic communities such as [6, 7] are called **two-stage approach**. The community partition is detected at each timestamp using the interaction graph and the evolutions of communities between two consecutive timestamps are inferred successfully. However, the community partition discovered by current interaction data could distort the real community structure. The community partition presumed that there is no relationship between individual pairs while no interactions occur between them.

Recently, a new concept of **temporal smoothness** was proposed [1] based on two points of view. (1) Each community partition in the time sequence should be similar to the community partition at the previous timestamp. (2) The community partition should accurately reflect the change of the interaction networks. The concept of **temporal smoothness** tries to discover the communities which not only consider about interaction data of current timestamp but also historical interaction data to improve the weakness of two stage approach.

However, the methods [1, 2, 3, 5] based on temporal smoothness have the following drawbacks. It is a big issue that the community partition should well reflect the previous interaction data more or a little more reflect the current interaction data. If the community partition always well reflects the previous interaction data more, the new change of social network would be hard to detect. On the other hand, the community partition would be more similar to the community partition based on two-stage approach and have the same weakness as two-stage approach. This setting of temporal smoothness has a great effect on the result of the methods based on **temporal smoothness**.

Concerting the evolution of community, current methods [6, 13, 3, 5] focused on that one community of previous timestamp maps to one community of current timestamp (one-to-one mapping). We argue that mapping is not suitable for real dynamic community because mapping of communities is not always one-to-one.

In this thesis, we propose EPC, "relationship **E**xtraction and community **P**edigree dynamic **C**ommunity miner", which produces the community partition taking into account the evolution of community. Instead of interaction data, we proposed the *Relationship Extraction strategy* which constructs a weighted graph for each timestamp and the weight indicates the relationship strength between individuals; we use two current static clustering algorithms, SHRINK [12] and GSCAN [5], to discover the community based on Relationship graph. We also proposed the *Community Pedigree mapping* which uses a realistic way to explain the evolution of dynamic community.

The *Relationship Extraction strategy* produces the relationship graph which not only references the historical interaction data but also the ongoing interaction data. The relationship graph not only represents much realistic relationship between individuals but also express the dynamic property in relationship graph. The *Relationship Extraction strategy* combines the normalized decay weight function to simulate the change of relationship strength between individuals. The Community Pedigree mapping extends the human pedigree to illustrate the evolution of community over time. The states of a community could be Birth, Death, Alive, Child and Fission. Using the community pedigree mapping, we could simply determine the evolution of community.

In synthetic data experiment, our algorithm EPC not only has higher accuracy but also more smoothing than previous algorithms. The EPC also expresses the property of linearly scalability. We also apply EPC on real datasets, Enron email dataset; Facebook dataset and DBLP dataset. All datasets show the change of community partition in real dynamic social network is quite low. The change rates of communities in co-authorship and friendship are higher than the change rate in company since the variation of company network is lower than friendship network and co-authorship network.

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

(1) We propose a new technique of data smoothing, *Relationship extraction strategy*, to produce a relationship graph. We discover community partition using the relationship graph instead of current interaction to overcome the weakness of **two-stage approach**. The relationship graph not only represents much realistic relationship between individuals but also express the dynamic property in dynamic social network.

(2) We propose a new matching technique, *Community Pedigree mapping*, which extends the point of human pedigree to explain evolution of community.

(3) We propose EPC, "relationship **E**xtraction and community **P**edigree dynamic **C**ommunity miner", which has not only higher accuracy but also better smoothing than previous methods no matter the noise level of data is high or low.

(4) EPC is linearly scalable both on the execution time and memory usage.

The rest of this thesis is organized as follows. Chapter 2 provides the related work and motivation. Chapter 3 presents the notation and problem definitions. Chapter 4 introduces the

Relationship graph strategy. Chapter 5 describes the SHRINK [12] and GSCAN[5] cluster algorithms. Chapter 6 presents the evolution of community and the proposed algorithm EPC. Chapter 7 presents the experiments and performance study. The conclusion and future work is in chapter 8.

Chapter2

Related Works and Motivation

In this section, we introduce the related works and motivation of this thesis. We first introduce the community detection technique in static graphs and then describe that in dynamic graphs where the dynamic graphs could change over time. Section 2-3 describes the motivation of this thesis.

2.1 Community Detection in Static Graphs

In the study of community detection in a single static graph which doesn't change over time, several approaches have been proposed on static graph.

Graph Partitioning approach consists of dividing the vertices into k groups of predefined size, such that the number of inter-edges between the groups is small [29]. The Kernighan -Lin algorithm is one of the earliest methods. Another popular technique proposed by Barnes et al is the spectral bisection method, which is based on the properties of the spectrum of the *Laplacian matrix*. The *Laplacian matrix* L= D-A where D is the diagonal matrix whose element D_{ii} equals the degree of vertex i and A is the adjacency matrix of the graph. In particular, the eigenvector corresponding to the second smallest *eigenvalue* is used for graph bipartitioning.

However, the Graph partitioning based methods need to predefine the number of clusters or the size of clusters at the beginning and the predefined parameter has great effect upon the result of graph partitioning.

In modularity-based approach, the modularity measure has been widely used in community discovery for evaluating the quality of network partitions. Modularity-based approach [18, 19] assumes high value of modularity indicates good partitioning and the partition corresponding to its maximum modularity score on a given graph should be the best.

However, detection of a community whose size is smaller than a certain size is impossible. This serious problem is famously known as the resolution limit of modularity-based algorithms [10].

The density based approach applies a local cluster criterion. Clusters are regarded as regions in the data space in which the objects are dense, and which are separated by regions of low object density. Density of intra-edges is used to partition graph into clusters. Xu et al [21] proposed the Structural Clustering Algorithm for Network (SCAN). SCAN [21] needs two predefined parameters minimum similarity threshold (ε) and minimum neighbors of core node (u) which limits that a core node has at least u neighbors whose similarities are more than minimum similarity threshold (ε). Each cluster should contain at least one core node inside. SCAN is an efficient structural network clustering algorithm while the predefined parameters are appropriate.

However, SCAN [21] requires the predefinition of the minimum similarity parameter (ε) and minimum core size (u) and the setting of parameters would huge affect the result of cluster partitions.

Recently a structural clustering algorithm called SHRINK was proposed by Huang et al [12] to overcome the problem of predefining parameters (ε and u) of density-based clustering algorithm. Through the experiment, SHRINK was proven to be an efficient, parameter free and higher accuracy algorithm while comparing with the modularity-based approach [18, 19].

2.2 Community Detection in Dynamic Graphs

Real social networks change over time so the community partition changes over time. So community detection in dynamic graphs would be more realistic than community detection in static graphs. Several approaches for discovering dynamic communities have been studied in dynamic social networks. We would illustrate the concept of these approaches.

2.2.1 Two-Stage Approach

Two-stage approach has two steps: (1) Community partition is detected at each timestamp of the interaction graph and the results of community partition are independent at different timestamps. (2) While the relationships between the communities of two consecutive timestamps are determined, the community mapping of two consecutive timestamps are inferred successfully.

Palla et al [6] proposed a method called *Clique Percolation Method* (CPM) to analyze the real dynamic network such as co-authorship network and mobile phone network. The CPM tries to find all k-cliques that can be reached from each other through a series of adjacent k-cliques where the adjacency means sharing k-1 nodes. The adjacency k-cliques are considered as community in [6]. The community mapping of two consecutive timestamps are dependent on their relative node overlapping. They first described the events in community evolution including Birth, Death, Merging, Splitting, Growth and Contraction. They have shown that the lifetime of communities in these networks depends on the dynamic behavior of these communities, with large groups that alter their behavior persisting longer than others. On the other hand, small groups were found to persist longer if their membership remained unchanged [6].

Sun et al [7] considered dynamic networks such as Network traffic, email and cell-phone as bipartite graphs which treat source and destination separately. GraphScope [7] is based on the principle of *Minimum Description employs* (MDL) and employs lossless encoding scheme for a graph stream. The encoding scheme takes into account both the community structure and the community change in order to achieve a concise description of the data.

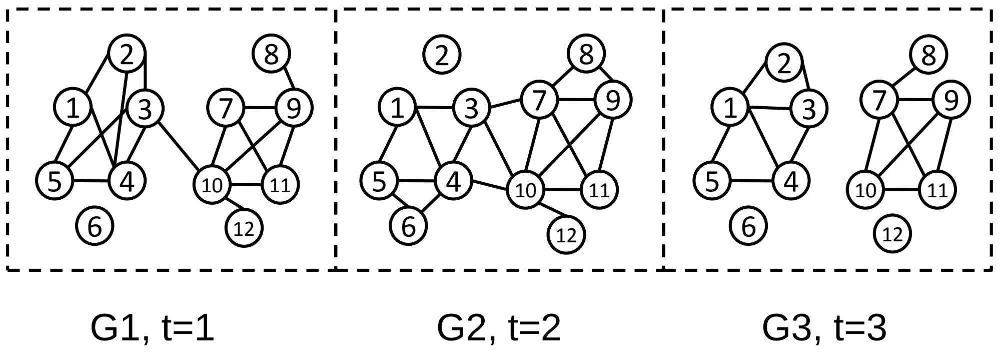


Figure 2-1 Interaction data example

We use Fig 2-1 and Fig 2-2 to illustrate the weakness of methods based on the **two stage approach**. Given the interaction data as shown in Fig 2-1, a node indicates an individual and an edge (u, v) indicates the interaction occurs between u and v. The community partition result of CPM [6] is shown in Fig-2-2. There are two communities discovered in time 1, one community discovered in time 2 and two community discovered in time 3.

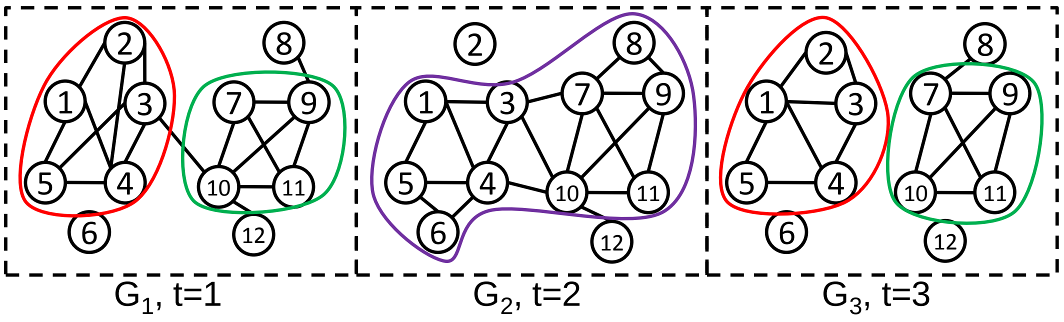


Figure 2-2 Community partition produced by CPM

While the interaction data change frequently, the community partition also change frequently. The node 2 is covered by a community at time 1 and time 3. However, the individual 2 disappeared at time t=2 because individual 2 has no interaction with other individuals at time t=2. There should be some relationship strength between individual 2 and other individuals even though no interactions occur at time t=2.

However, the methods [6,7] based on the **two-stage approach** presume that there is no relationship between individual pairs while no interactions occur between them and it often results in community structure with significant changes [3].

While the community partitions at all timestamps have been produced, how to illustrate the evolution of community is another issue. In the studies of evolution of community, Palla et al [6] proposed a point of view that the basic operation in a community life should contains birth, death, growth, contraction, merge and split.

Asur et al. [13] considered the issue of community evolution and they defined their own community similarity function. They proposed five critical events such as *Continue*, *K-merge*, *K-split*, *Form* and *Dissolve* to illustrate the evolution of communities between two consecutive timestamps.

Lin et al [3] proposed the *Evolution net* which is a bipartite graph to illustrate all relationship between the communities of every two consequent timestamps. However, Lin et al [3] didn't illustrate how a community evolution over time. Kim et al [5] also propose a heuristic mapping algorithm based on mutual information.

However the above methods [6, 13, 3, 5] made effort in the mapping a community at previous timestamp to a community at current timestamp **(1-1 mapping problem)**. We use Fig 2-3 to illustrate the point of view.

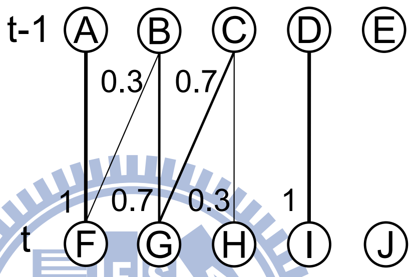


Figure 2-3 Evolution Net

The evolution net shown in Fig 2-3 is a bipartite graph from t-1 to t. There are 5 communities (A, B, C, D, E) at t-1 and 5 communities (F, G, H, I, J) at t. A node indicates a community and an edge from a community of t-1 to community of t indicates that there is a relationship between these two communities where the edge weight indicates the similarity between them. The community B having relationships with communities F and G could be considered as community B being divided into two parts, one part of B merges with community A into community F at timestamp t; the other part of B and one part of C are merged into community G. The merge and split operations of communities could happen at the same time.

Although the similarities between communities are determined and are shown an edge weight in Fig 2-3, there is no indication community F is a split part of B or F is a growth of community A. Similarly, we do not know if community C is dead while community G matches with community B. It is judged that the one to one mapping is not suitable for real dynamic networks.

2.2.2 The approach of Evolutionary Clustering

The approach of *Evolutionary clustering* was first proposed in [1] taking into account the concept of *temporal smoothness*. Each community partition in the time sequence should be similar to the community partition at the previous time slice. The community partition should accurately reflect the change of the interaction graph. For evolution clustering, [1] adopted the two widely used clustering algorithms, k-means and agglomerative hierarchical clustering incorporating temporal smoothness.

Fig 2-4 shows the details of the **concept of temporal smoothness**. Given the dynamic network G={G_1, G_2, …, G_{t-1}, G_t, …}. G_t is the interaction graph of time t. The CP_t indicates the community partition at time t and is affected by previous community partition CP_{t-1} and the current interaction graph G_t.

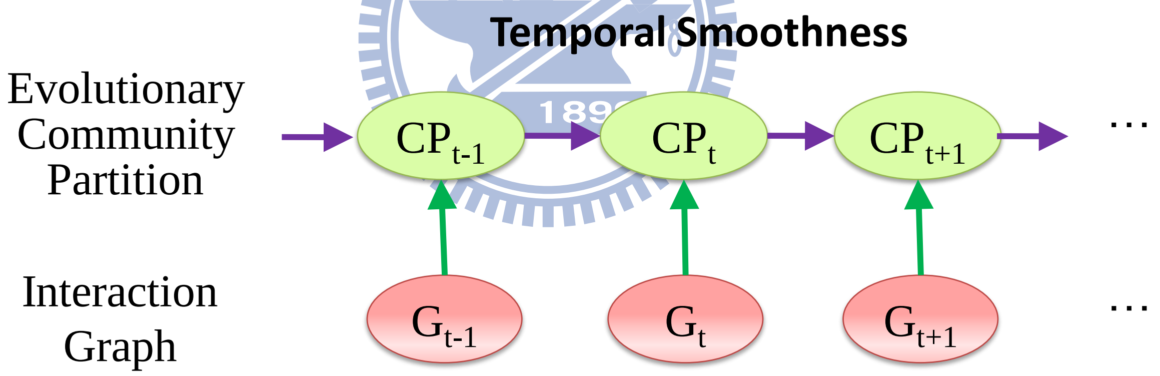


Figure 2-4 Concept of temporal smoothness

In order to measure the quality of community partition based on the concept of temporal smoothness, the objective function is defined:

$$\mathrm{Cost}(CP_t^i) = \alpha * SC(G_t, CP_t^i) + (1-\alpha) * TC(CP_{t-1}, CP_t^i) \quad (1)$$

Consider all possible community partitions AP = { CP_t^1, CP_t^2, …, CP_t^i,…}. For each community partition CP_t^i, the snapshot cost SC() measures the similarity between the interaction graph G_t and CP_t^i . The temporal cost TC() measures the similarity between the previous community partition CP_{t-1} and CP_t^i. While the SC() is lower, the quality of snapshot is higher. While the TC() is lower, the quality of temporal smoothness is higher. The optimal community partition $CP_t = CP_t^i \mid \min_{\forall CP_t^i \in AP}\{Cost(CP_t^i)\}$. The parameter α controls the emphasis of the result of community partition. While α =1, the community partition CP_t would be the same as community partition discovered in G_t . On the other hand, the CP_t would be the same as CP_{t-1} whileα =0.

Chi et al [2] extended the **concept of temporal smoothness** and considered that the characteristic change of dynamic community contains both long-term trend drift and short-term variation due to noise. They proposed two evolutionary spectral clustering algorithms called *Preserving Clustering Quality* (PCQ) and *Preserving Clustering Membership* (PCM). Their experiment shows that PCQ and PCM are less sensitive to short-term noises while at the same time are adaptive to long-term cluster drifts. The spectral clustering uses the eigenvectors of *Laplacian matrix* for clustering the graph nodes. The *Laplacian matrix* L= D-A where D is the diagonal matrix whose element D_{ii} equals the degree of vertex i and A is the adjacency matrix of the graph.

Lin et al [3], who are the first critic, argue that the methods of **two-stage approach** are inappropriate in applications with noise data. They considered an individual could be assigned to more than one community so they further assumed that each interaction graph G_t is combined effect by community partition CP_t. They extended a mixture model [30] and incorporated the concept of temporal smoothness. The mixture model is a probabilistic model for representing the presence of sub-populations within an overall population, without requiring that an observed data-set should identify the sub-population to which an individual observation belongs.

Given G_t as the interaction graph at time t and assume there are k communities at time t. Mixture model assumes that the edge G_{ij} of G_t is combined effect due to all the k communities so $G_{ij} \approx \sum_{r=1}^{k} P_r * P_{r\to i} * P_{r\to j}$ where P_r is the prior probability that the interaction G_{ij} is due to the r-th community, $P_{r\to i}$ and $P_{r\to j}$ are the probabilities that an interaction in r-th community involves node i and node j, respectively. Written in a matrix form, we have $Gt \approx X\Lambda X^T$ where X is a $n \times k$ non-negative matrix and X_{ij} indicates the probability of node i belonging to community j. In addition, Λ is a $k \times k$ non-negative diagonal matrix with $\Lambda_i = P_i$ where Λ_i indicates Λ_{ii}. We use Fig 2-5 to illustrate the mixture model [3].

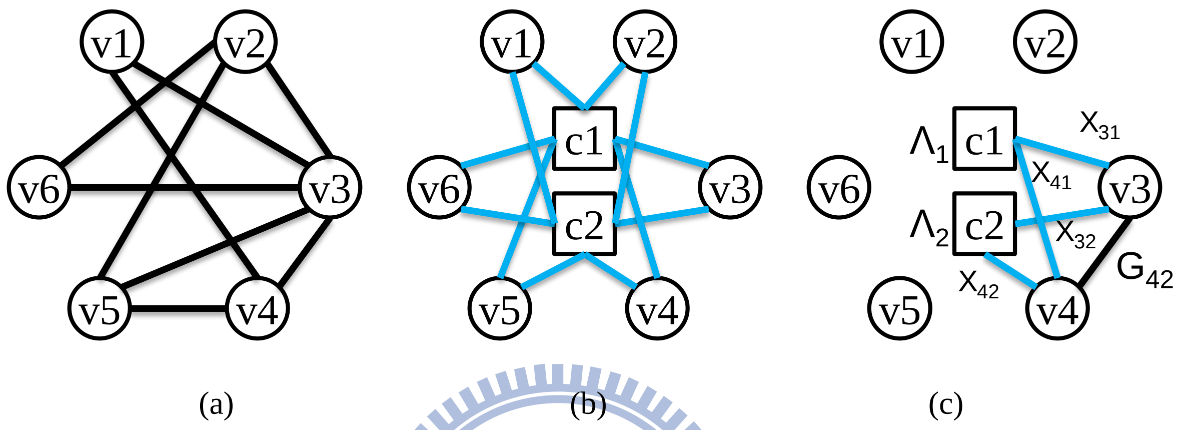


Figure 2-5 Mixture model(a) the original graph Gt (b) the bipartite graph with two communities c1 and c2 (c) How to approximate an edge (G_{42}).[3]

In Fig. 2-5, there are 6 nodes and 2 communities. For a general graph Gt in Fig 2-5(a), we use a special bipartite graph Fig 2-5(b) to approximate Gt. Note that (b) has two more nodes, i.e., c1 and c2, corresponding to the two communities. In (c), we show how an edge G_{34} is generated in the mixture model as the sum of $\Lambda_1 * X_{31} * X_{41}$ and $\Lambda_2 * X_{32} * X_{42}$ [3].

The Kullback-Liebler divergence was used to measure the difference between two community partitions *P* and *Q* to reconstruct the defined cost function. They proposed "*A Framework for Analyzing Communities and EvoluTions in dynamic NETworks*" (FacetNet), defined the *Community net* and *Evolution net* to represent the community structure and evolutions.

However, the methods in [1, 2, 3] assign a fixed number of communities over time and do not allow arbitrary start/stop of community over time [5].

Kim et al [5] overcome the problems of fixed number of community and arbitrary start/stop of community. They first model a dynamic network as a collection of particles called *nano-community*, and a community as a densely connected subset of particles, called a *quasi l-clique-clique*. They proposed a greedy algorithm called "*a Particle-and-Density Based Evolutionary Clustering Method*" (PD-Greedy). PD-Greedy extends density based clustering [20, 21] and incorporates with the **concept of temporal smoothness**. They used a cost embedding technique to efficiently find temporally smoothed local clusters of high quality.

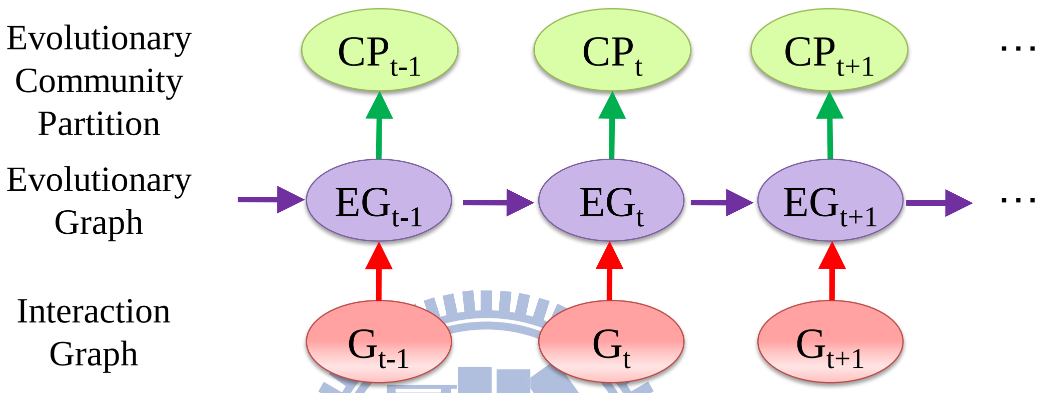


Figure 2-6 Flowchart of PD-Greedy

Fig. 2-6 shows the flowchart of PD-Greedy in which the working flows of **temporal smoothness** is reconstructed. The comparison subject is pushed from community level to down data level. At each timestamp, an evolution graph (EG_t), which reference both previous evolution graph (EG_{t-1}) and current interaction graph (Gt), is produced. Then the community partition based on the EG_t is discovered.

However, the methods including PD-Greedy [1, 2, 3, 5] based on **temporal smoothness** have several drawbacks. The parameter α of cost function might affect the community partition produced by the methods based on **temporal smoothness.**

Assume the interaction data is the same as in Fig 2-1, the previous community partition CP_1 is shown on the left of Fig 2-7 and the snapshot partition of G_2 is shown on the right of Fig 2-7.

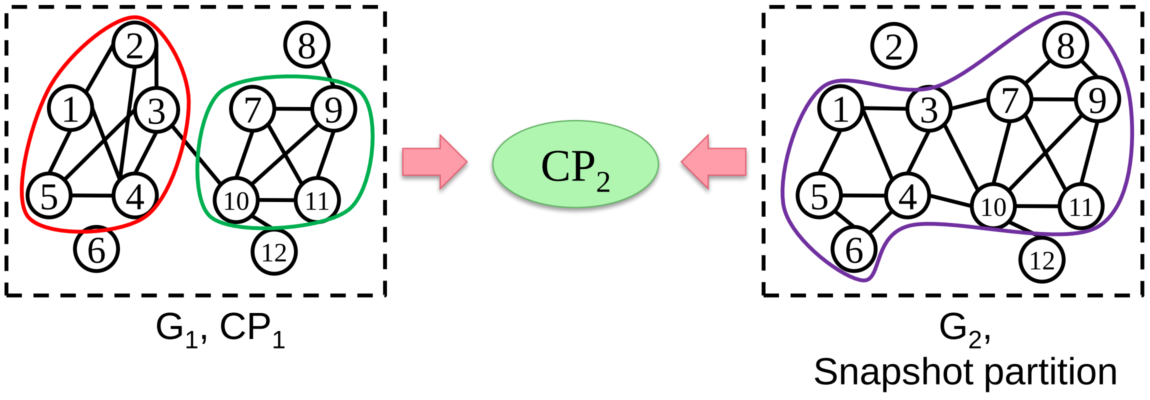


Figure 2-7 Community partition of temporal smoothness

Based on temporal smoothness, the community partition CP_2 is not only similar to CP_1 but the snapshot partition of G_2. if the parameter α of the cost function of the concept of temporal smoothness is close to one, the CP_2 is similar to the result of two-stage-approach and has the same weakness of two-stage approach. On the other hand, if the parameter α of the cost function of the concept of temporal smoothness is close to zero, it is hard to catch new community birth or change.

Tang et al [4] proposed the algorithm of *Evolutionary Multi-mode Clustering* where multi-mode network typically consists of multiple heterogeneous social actors among which various types of interactions could occur. However, we consider the network which has only single kind of interaction and this network is different from multi-mode networks in [4].

Besides, there is another issue, **analyzing all interaction data** [8]. Interaction data could frequently change violently over time so analyzing the interaction data at a single time slice may miss important tendencies of a dynamic network. An individual tends not to change he's "home community" too frequently. An individual tends to interact with the member of his "home community" most of the time where the "home community means the original community of the specific individuals [8]. They proved that find the most explanatory community structure is NP-hard and APX-hard problem where the class APX (approximate) is the set of NP optimization problems that allow polynomial-time approximation algorithms with approximation ratio bounded by a constant (or constant-factor approximation algorithms for short)[31]. They also proposed a greedy heuristic approximation algorithm using individual coloring and group coloring to identify the dynamic communities at each timestamp where the same colored individuals and group means the same community.

However, the greedy heuristic algorithm in [8] is not appropriate for large dynamic networks and mining the communities with all timestamps of interaction graph would take huge computation time.

2.3 Motivation

To summarize, the methods [6,7] based on the **two stage approach** presume that there is no relationship between individuals while no interactions occurs between them and it often results in community structure with significant changes [3]. The methods based on the **approach of analyzing all interaction data** take huge computation cost and are not linearly scalable. The methods based on the **approach of Evolutionary clustering** which discovers the community partition based on the property of current interaction graph have the same weakness as **two stage approach**. And the parameter α of cost function based on temporal smoothness also huge affects community partition.

Besides, on the study of evolution of community, current researches [6, 5, 13, 3] are not suitable for real dynamic community due to the one-to-one mapping in which one community of previous timestamp maps to one community of current timestamp.

For dynamic community detection, our goal is to develop a framework which not only has higher accuracy, linearly saleable execution time but also realistic community partition result which reference multiple interaction graphs. Besides discovering dynamic communities at each timestamps, there should be a general method to explain the evolution of communities.

Chapter3

Notation and Problem Definition

In this section, we formally introduce necessary notation and formulate the problems.

3-1 Notation and Symbol definition

We define a dynamic social network G as a sequence of interaction graphs.

Definition 1 (*Interaction Graph*)

An interaction graph $G_t = (V_t, E_t)$ is an un-weighted undirected graph where a node indicates an individual and an edge indicates that an interaction occurs between two individuals at time t.

Definition 2 (*Dynamic Social Network*)

A dynamic social network G is a sequence of interaction graph Gt. i.e. $G = \{G_1, G_2, \ldots, G_t, \ldots\}$ where the G_t (V_t,E_t) is an interaction graph and the t indicates the t-th time point.

For example, the Fig 2-1 shows the first three graphs in a dynamic social network.

Definition 3 (*Observation Window*)

The observation window Ψ of current time point tc is defined by observation eyeshot (wr). i.e. $\Psi = \{\text{time point } t \mid tc - wr \le t \le tc + wr\}$ where the tc indicates the current timestamp.

We use the interaction graphs whose time points are included by observation window Ψ to produce the relationship graph RG_t.

Definition 4 (*Relationship Graph*)

A relationship graph RG_t is a weighted undirected graph at time t. i.e. $RG_t = (V_t^R, E_t^R)$ where V_t^R indicates the set of individuals of RG_t; E_t^R indicates the edges of RG_t and the edge weight indicates the relationship strength between the individuals. Let $W_t(u, v)$ represent the relationship strength between individuals u and v at time t.

Definition 5 (*Community Partition)*

A community partition CP_t is the a sequence of communities at time t. i.e. CP_t

$=\{C_t^1, C_t^2, \ldots, C_t^j, \ldots\}$ where C_t^j is the j-th community of CP_t .

3-2 Problem Statement

Definition 6 (*Dynamic Community Identification*)

Given a dynamic social network $G = \{G_1, G_2, \ldots, G_t, \ldots\}$, How to produce the community partition CP_t of each timestamp? While the community partitions of each timestamp have been discovered, How to determine the evolution between the communities of every two consecutive timestamps?

Chapter 4

Relationship Extraction Strategy

In this chapter, we first illustrate the framework of EPC, "relationship **E**xtraction and community **P**edigree dynamic **C**ommunity miner", and then we introduce how the *Relationship graph* is constructed. The *Relationship Extraction strategy* extracts the relationship strength using interaction data and combines the normalized decay weight function to simulate the change of relationship strength within a fix time observation window.

4-1 Proposed Framework

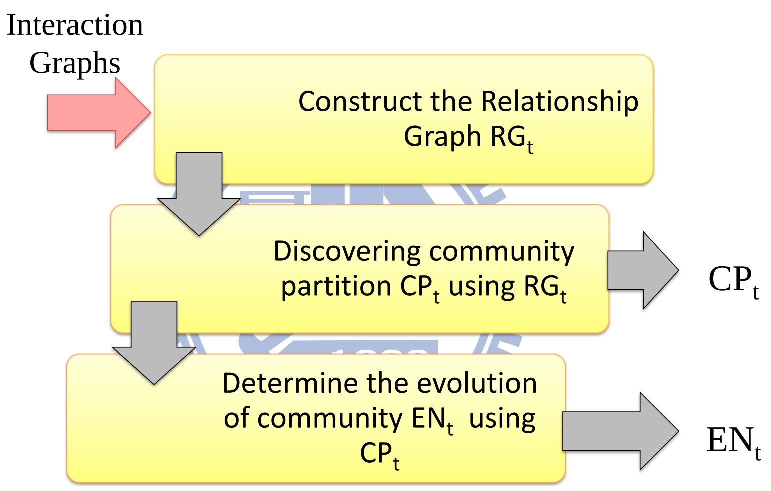

Interaction Graphs
Construct the Relationship Graph RG_t
Discovering community partition CP_t using RG_t
CP_t
Determine the evolution of community EN_t using CP_t
EN_t

Fig 4-1 Flowchart of EPC, "relationship **E**xtraction and community **P**edigree dynamic **C**ommunity miner"

The flowchart of EPC, "relationship **E**xtraction and community **P**edigree dynamic **C**ommunity miner", shown in Fig 4-1 and consists of three phases. (1) Construct the relationship graph RGt which is using a set of interaction graphs within observation window. (2) Use static community detection methods to discover the community partition CP_t based on the *Relationship Graph* produced in first step. (3) Determine the evolution of community using the community partitions.

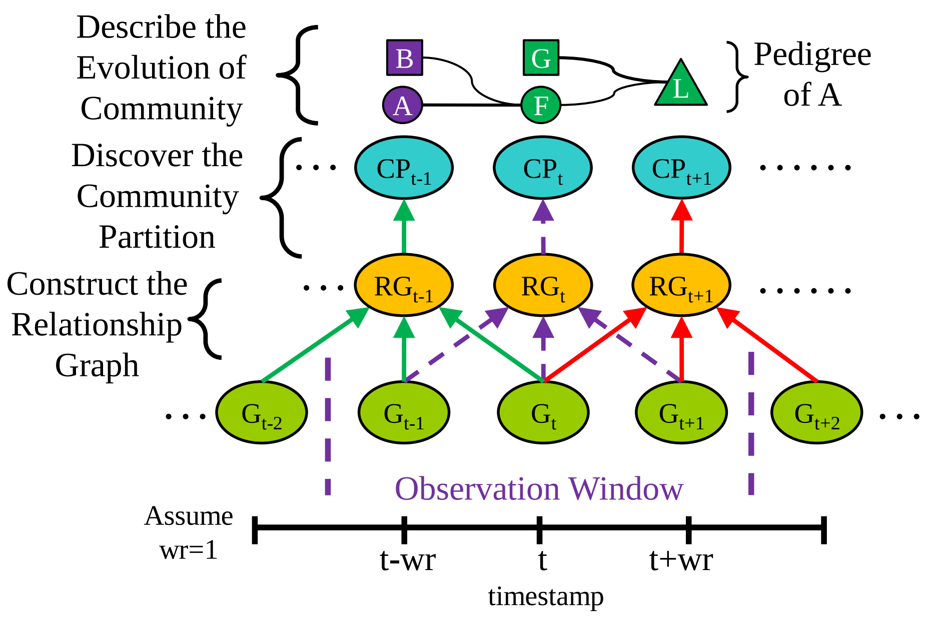


Fig 4-2 Framework of EPC, "relationship **E**xtraction and community **P**edigree dynamic **C**ommunity miner"

Our framework of EPC is shown in Fig 4-2. Assuming the observation eyeshot wr = 1, tc=t, so the relationship graph RG$_t$ is constructed using interaction graphs G$_t$, G$_{t-1}$ and G$_{t+1}$. Then we use static community detection method to generate the community partition CP$_t$ based on the relationship graph RG$_t$. While the community partitions of each timestamp have been generated, we determine the relationship between communities at each consecutive time points using the *Community Pedigree Mapping*.

The graph on the top of Fig 4-2 is the pedigree of community A where a node indicates a community and an edge indicates the similarity strength between communities. The "pedigree of community A" shows all the communities which have relationship with community A. A square shape indicates the spouse community of A and the triangle shape indicates this community would be dead at next timestamp. There are 5 community spread on the timestamps {t-1, t, t+1}. The community F is similar to previous community A and B so F is the child of A and B. The community L is the child of G and F. The community B is the spouse of A and G is the spouse of F. While we want to monitor some communities to figure out if these communities are involved with each other, the pedigree of community would be a good way to illustrate.

We present the **Relationship Extraction strategy** to construct the relationship graph RG_t in chapter 4. We present **current static clustering method, SHRINK** [12]**,** in chapter 5 and we propose the **Community pedigree Mapping** to solve the problem of evolution of communities in chapter 6.

4.2. Generating Relationship Graph?

The relationship graph RG_t is constructed from interaction graphs which are most to the current time point t. We use the observation eyeshot wr to control the observation window Ψ. For example, Assuming wr=2 and tc =3, then $\Psi = \{1,2,3,4,5\}$ and the relationship graph RG_3 is constructed using interaction graph G_3 and those interaction graphs 2 time units before (G_1, G_2) and after (G_3, G_4) current time tc. Then we determine the relationship strength $W_{tc}(u,v)$ between individuals u and v using the predefined normalized weight function.

Here we propose a naïve normalized weight function, normalized Equal weight function (EQL) as follows.

$$N_Q(t,tc) = \frac{1}{|\Psi|}, \forall\, t \in \Psi \qquad (2)$$

EQL considers the interaction graph of each time point within Ψ having the same weight and makes sure the weight summation would be equal to 1. Using normalized weight function to determine the relationship strength of each pair of individuals is just like the function as follows:

$$W_{tc}(u,v) = \sum_{\forall\, t \in \Psi} I_{u,v}(t) \times N(t,tc), \qquad (3)$$

where $I_{u,v}(t) = \begin{cases} 1, & \text{if there exists an edge (u,v)within interactiongraph } G_t \\ 0, & \text{otherwise} \end{cases}$ and $N(t,tc)$ is the predefined normalized weight function.

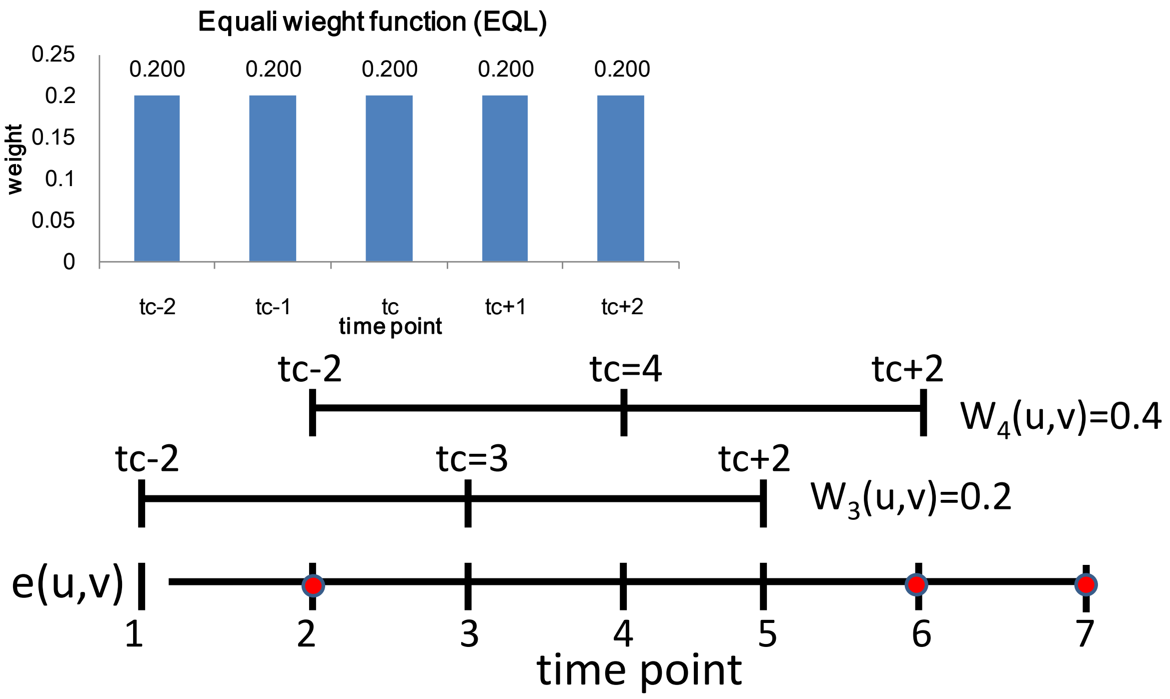


Figure 4-3 Example of determining the relationship strength

We use Fig 4-3 to illustrate how the relationship strength determined. Assuming the observation eyeshot wr equals to 2 and the interaction between individual u and v occurs at time 2, 6 and 7. The relationship strength between individual u and v at tc = 3, $W_3(u, v)$ = $I_{u,v}(1) * \mathrm{N_Q}(1,3)$ + $I_{u,v}(2) * \mathrm{N_Q}(2,3)$ + $I_{u,v}(3) * \mathrm{N_Q}(3,3)$ + $I_{u,v}(4) * \mathrm{N_Q}(4,3)$ + $I_{u,v}(5) * \mathrm{N_Q}(5,3)$ = (0*0.2)+ (1*0.2)+ (0*0.2)+ (0*0.2)+ (0*0.2) =0.2 . Using the same process we calculate $W_4(u,v) = 0.4$, $W_5(u,v) = 0.4$.

Using Eq 3 to determine the relationship graph RG_2 of Fig 2-1 and the RG_2 is shown on the bottom of Fig 4-4. A node indicates an individual and the edge weight $W_2(u, v)$ indicates the relationship strength between individuals u and v.

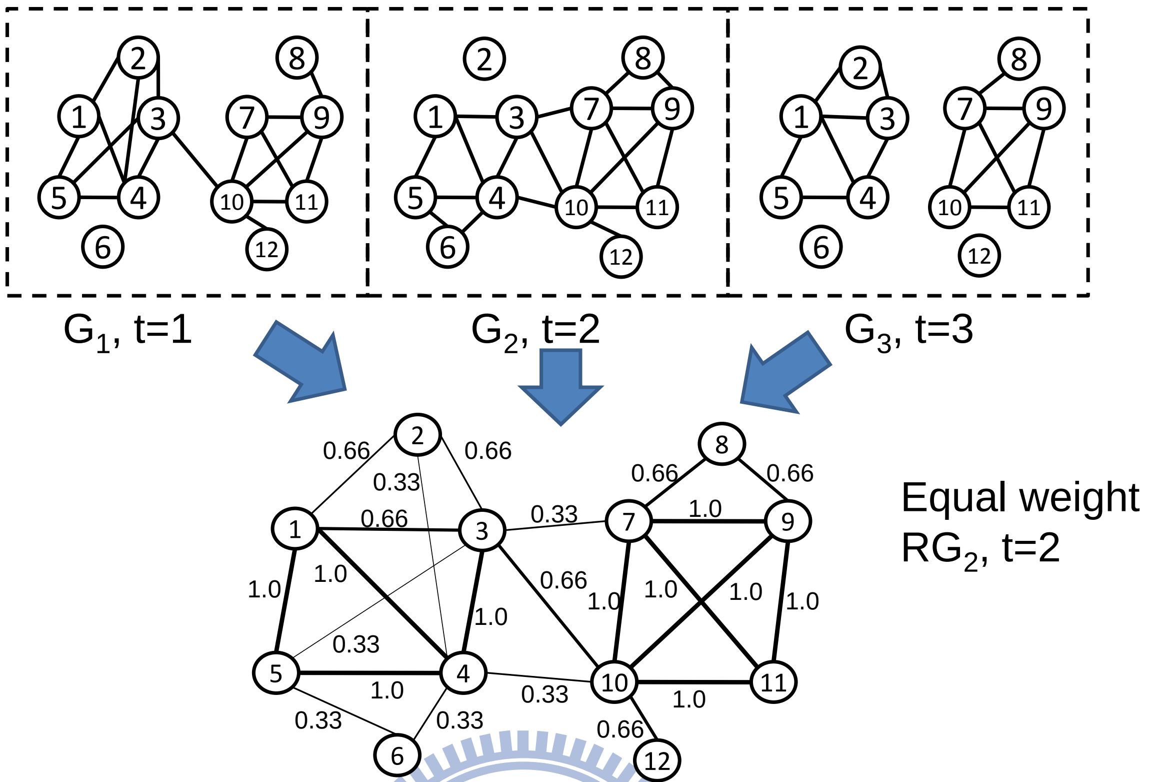


Figure 4-4 Using the EQL weight function to construct the Relationship graph RG2 of sample interaction data of Fig 2-1

4.3 Normalized Decay Weight Function

Assuming the observation eyeshot wr and the *Observation window* Ψ is predefined. We propose three *Normalized Decay weight functions*:

Linear Decay weight function (LIN):

$$N_L(t, tc) = \frac{-1}{wr+1} * |t - tc| + 1, \forall\, t \in \Psi \tag{4}$$

Wave Decay weight function:

$$N_W(t, tc) = 0.5 * \left(\sin\left(2 * \pi * \left(\frac{|t-tc|}{2*wr} + \varphi\right)\right) + 1\right), \forall\, t \in \Psi \tag{5}$$

$$, where\ \varphi = 0.25$$

Exponential Decay weight function:

$$N_E(t, tc) = exp\left(\frac{-|t-tc|}{wr}\right), \forall\, t \in \Psi \tag{6}$$

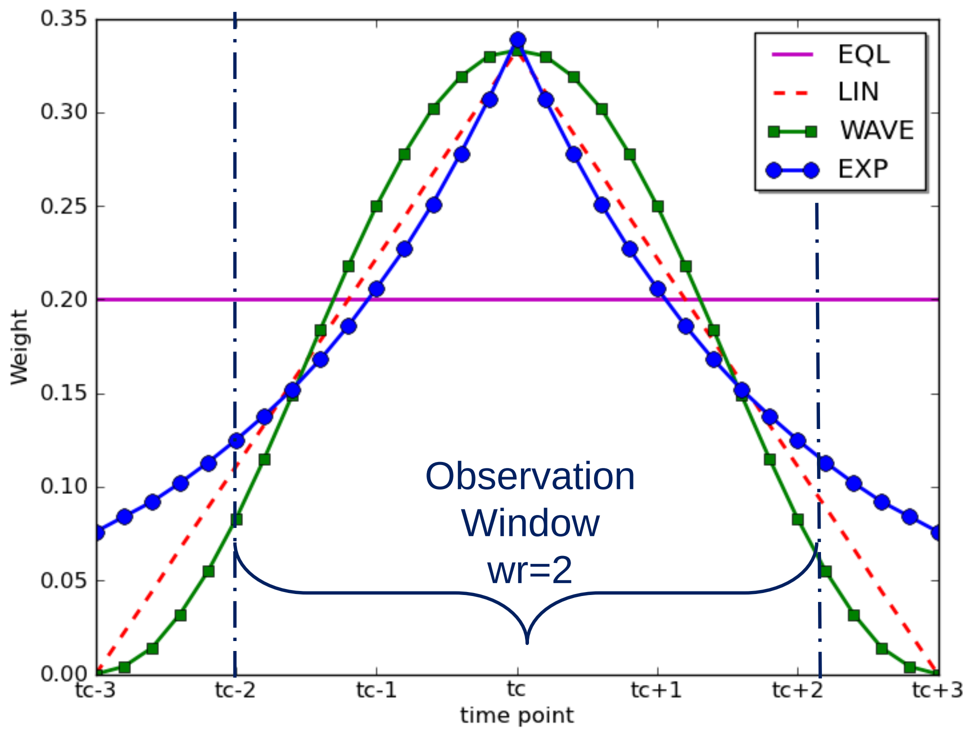


Fig 4-5 Normalized Decay Weight Function (wr=2)

The $N_L()$ is based on linear decay and the weight distribution is shown in the curve (LIN) in Fig 4-5. Using $N_L()$ to calculate the relationship strength is the same as the example in section 4-2. We multiply the weight $N_L(t,tc)$ with the interaction occurring in Ψ and sum all the values.

The $N_w()$ is based on the sine function of trigonometric functions to produce the relationship graph. The weight distribution is shown in the curve (WAVE) in Fig 4-5.

The $N_E()$ is based on the approach of exponential decay function. If the weight decreases at a rate proportional to its value, it is called exponential decay [11]. The processes can be modeled by the following differential equation.

$$\frac{dN}{dt} = -\lambda N \tag{7}$$

where N is the weight quantity and λ is called decay constant.

The decay constant λ controls the decay rate of the exponential decay and we use

$\lambda = 1/wr$ in our work. The weight distribution is shown in curve (EXP) in Fig 4-5.

For each normalized decay weight distribution, if there are some interactions whose time point is out of the *Observation Window*, the weight is assigned zero. Note that exponential decay weight distribution the weight of time point out of the observation window Ψ is non-zero but we simply assume the weight is zero.

4.4 Discussion of Relationship Extraction Strategy

In real world, the relationship between individuals could decay over time and the interaction at each time point within Ψ should be considered an energy which increases the relationship strength between individuals. So we presume each interaction has the same lifetime equal to the size of observation window. Then the lifetime of each interaction would be extended from 1 to the size of observation window. This property could overcome the weakness that there is no relationship between individuals while no interactions occur.

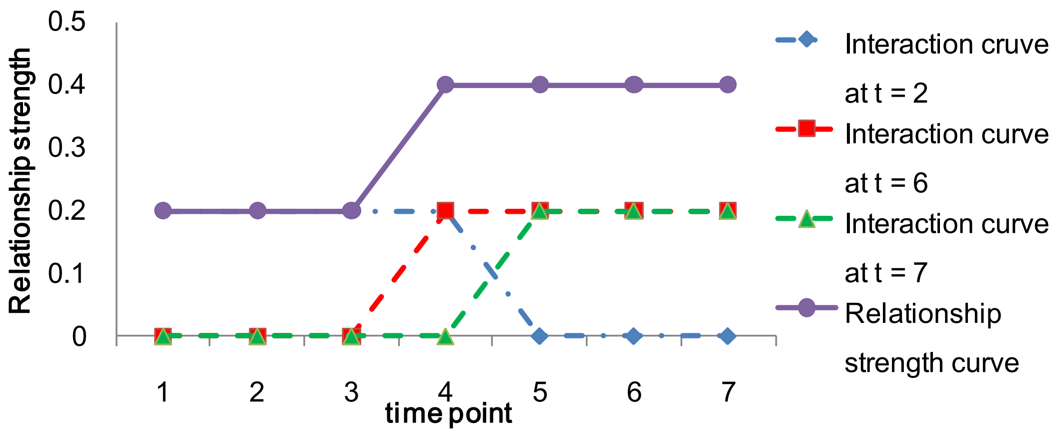


Fig 4-6. Relationship strength curve between u and v by extraction from the interaction data of Fig 4-2 based on Normalized Equal Weight Function (wr=2)

Fig 4-6 shows the evolution of relationship strength of individual u and v in the interaction data of Fig 4-3 using normalized equal weight function. The dotted line implies the interaction occurring at time points 2, 6 and 7. The interactions at all time points have the same lifetime equal to the size of observation window Ψ. The solid line sums up the curves of all interactions and represent the relationship strength of individual u and v over time.

However, the solid line in Fig 4-6 which shows the relationship curve is higher at time t=4, 5, 6 and 7. The relationship strength curve does not match any interaction data occurred and this curve does not have the property of dynamics.

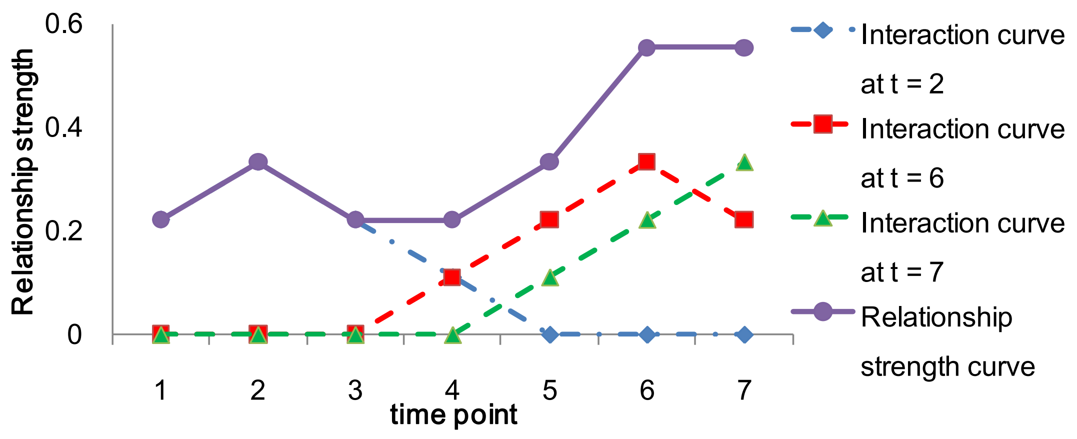


Fig 4-7. The Relationship strength curve of the interaction data of Fig4-2 based on Normalized Linear Decay Weight function (wr=2)

We change the equal weight function to the linear decay weight function and the Relationship strength curve is shown in Fig 4-7. The solid curve in Fig 4-7 indicates the relationship strength of individuals and the curve is high at t=2, 5, 6, 7 and the solid curve matches with the timestamps interaction occurred. The difference between Fig 4-6 and Fig 4-7 is that the relationship strength curve in Fig 4-7 demonstrates more dynamic property than that in Fig 4-6 so the normalized decay weight function would be more realistic than equal weight function.

Chapter 5
Current Static Community Detection methods

After generating relationship graphs at each time point, we choose static community methods for discovering the community partition at each timestamp. Although many of studies have focused on community detection on static networks, not every method is suitable for relationship graph. Two issues need to be considered about: (1) Discovering communities using the weighted graph (relationship graph). (2) Detecting the noisy vertices whose relationship strength is too low to belong to any community. The SHRINK algorithm [12] overcomes the problem of parameters pre-definition, such as minimum similarity threshold (ε) and minimum core size (μ), in density-based clustering algorithms and the predefined number of clusters in partitioning-based clustering algorithm. Through their experiment, SHRINK is an efficient, parameter free and high accuracy algorithm while comparing with other algorithm [18, 19]. So we choose SHRINK as our clustering algorithm. For comparison, we also use the greedy density-based clustering method (GSCAN) used in PD-Greedy [5, 21].

In section 5.1 we represent the detailed definition of GSCAN and section 5.2 represents the definition of SHRINK. Section 5.3 illustrates the quality measurement of community partition. Section 5.4 describes the detail algorithm of SHRINK algorithm and the algorithm of GSCAN is represented in section 5-5.

5.1 GSCAN

In this chapter we present the definition of GSCAN and related notation. Let (V, E, ω) be a weighted undirected network where ω is the weight set of edge set E, GSCAN uses the structure similarity as similarity measure and the related definition is as follows:

Definition 7 [5]. (*Neighborhood*)

Given G= (V, E, ω), for a node $u \in V$ and the adjacent nodes of u are neighbors of u ($\Gamma(u)$). i.e.: $\Gamma(u) = \{v \in V | (u, v) \in E\} \cup \{u\}$.

Definition 8 [5]. (*Structural Similarity*)

Let G= (V, E, ω) be a weighted undirected network. The structural similarity between two adjacent nodes u and v is defined as below:

$$\sigma(u,v) = \frac{\sum_{x \in \Gamma(u) \cap \Gamma(v)} \omega(u,x) * \omega(v,x)}{\sqrt{\sum_{x \in \Gamma(u)} \omega^2(u,x)} * \sqrt{\sum_{x \in \Gamma(v)} \omega^2(v,x)}} \quad (8)$$

where $\omega(u,v)$ indicate the weight of $edge(u,v)$

GSCAN applies a minimum similarity threshold ε to the computed structural similarity when assigning cluster membership as formalized in the following ε -Neighborhood definition:

Definition 9[5]. (*ε -Neighborhood*)

For a node $v \in$ V, the **ε** *-Neighborhood* $N_{\varepsilon}(v)$ of a node v is defined by $N_{\varepsilon}(v) = \{x \in \Gamma(v) | \sigma(v, x) > \varepsilon\}$

When a vertex shares structural similarity with enough neighbors, it becomes a seed for a cluster. Such a vertex is called a *core node*, *Core nodes* are a special class of vertices that have a minimum limit of μ neighbors with a structural similarity that exceeds the threshold ε [21].

Definition 10[5]. (*Core node*)

A node $v \in$ V is called a *core node* w.r.t. ε and μ, if $|N_{\varepsilon}(v)| \geq \mu$.

Definition 11[5]. (*Directly reachable*)

A node $x \in$ V is directly reachable from a node $v \in$ V w.r.t. ε and μ if (1) node v is core node. (2) $x \in N_{\varepsilon}(v)$.

Definition 12[5]. (Reachable)

A node $v_j \in$ V is reachable from a node $v_i \in$ V w.r.t. ε and μ if there is a chain of nodes $v_i, v_{i+1}, \ldots, v_{j-1}, v_j$ such that v_{i+1} is directly reachable from v_i (i < j) w.r.t. ε and μ.

Definition 13[5]. (Connected)

A node $v \in$ V is connected to a node $u \in$ V w.r.t. ε and μ if there is a node x ∈ V such that both v and u are reachable from x w.r.t. ε and μ.

Definition 14[5] (*Connected cluster*)

A non-empty subset S ⊆ V is called a connected cluster w.r.t. ε and μ if S satisfies the following two conditions:

(1) Connectivity: ∀ v, u ∈ S, v is connected to w w.r.t ε and μ.

(2) Maximality: ∀ v, u ∈ V, if v ∈ S and u is reachable from v w.r.t ε and μ, then u ∈ S.

Using the above definition, a structure-connected cluster with respect to ε , μ is uniquely determined by any cores of this cluster.

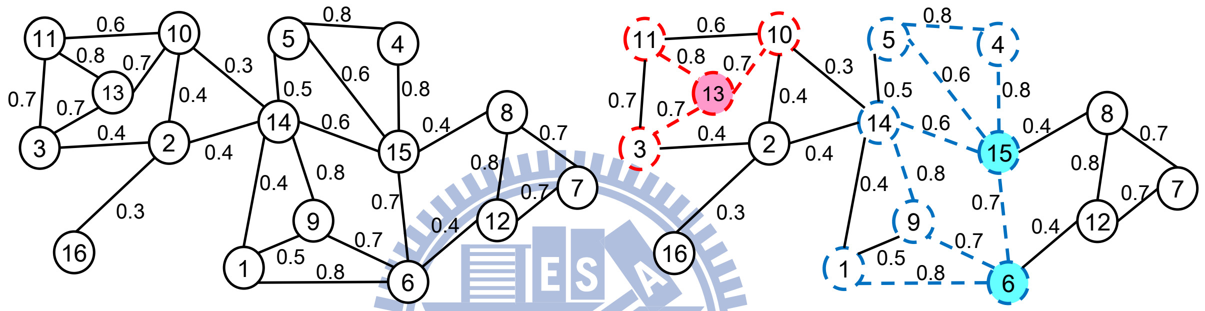


Figure 5-1 (a) Sample network G (b) Connected cluster of Sample network G

We use the Fig 5-1 to illustrate the related definition**.** In Fig 5-1, a node indicates an individual and an edge indicates the structural similarity between individuals. Let ε = 0.6 and μ = 3, we could evaluate the *Core node*s as node 13, node 15 and node 6. The node 10 is *Directly reachable* from node 13 due to that node 13 is a core node and $\sigma(13{,}10) \geq$ ε. The node 4 is *Reachable* from node 6 due to the *Directly reachable* chain (node 6, node 15, node4). Based on the definition of ***Connected cluster,*** there are two clusters, {3, 10, 11, 13} and {1, 4, 5, 6, 9, 14, 15}.

5.2 SHRINK

In this section we introduce SHRINK and related notation. For structural similarity measure, SHRINK uses the same cosine similarity as GSCAN and the related definition is as follows:

Definition 15[12]. (*Dense Pair*) $\boldsymbol{u \leftrightarrow_{\varepsilon} v}$

Given a network G(V,E), If $\sigma(u,v)$ is the largest similarity between nodes u and v and their adjacent neighbor nodes. i.e.: $\sigma(u,v) = max\,\{\sigma(x,y)|(x = u, y \in \Gamma(u) - \{u\}) \vee (x = v, y \in \Gamma(v) - \{v\})\}$ then $\{u,v\}$ is called a *Dense Pair* in G, denoted by $u \leftrightarrow_{\varepsilon} v$, where $\varepsilon = \sigma(u,v)$ is the largest similarity between nodes u and v and their adjacent neighbor nodes.

Definition 16[12]. (*Micro-community*)

Given a network G=(V,E), *MC*(a)= (V', E') is a connected sub-graph which is represented by node a in network G. *MC*(a) is a local *Micro-community* if and only if

$$\begin{cases} (1)\ \ \mathrm{a} \in \mathrm{V}'; \\ (2)\text{for all } u \ \in \mathrm{V}', \exists v \ \in \mathrm{V}'(u \leftrightarrow_{\varepsilon} v) \\ (3)\nexists u \in \mathrm{V}(u \leftrightarrow_{\varepsilon} v \wedge u \in \mathrm{V}' \wedge v \notin \mathrm{V}') \end{cases}$$

where $\varepsilon = \sigma(u,v)$ represents the largest similarity between nodes u and v and their adjacent neighbor nodes.

We use the Fig 5-1(a) to illustrate the ***Dense Pair*** and ***Micro-Community***. All the ***Dense Pair*** within the Fig 5-1(a) are shown in Fig 5-2. The dot line nodes and dot line edges indicate that these nodes are ***Dense Pair*** and could be grouped into a *Micro-Community*.

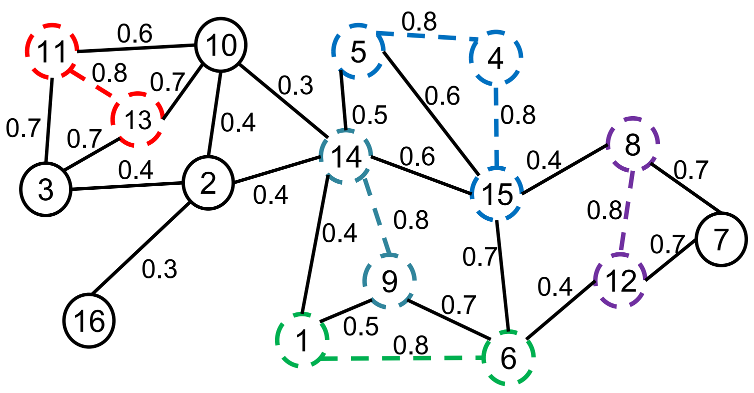


Figure 5-2 All ***Dense Pairs*** within the sample network G in Fig 5-1(a)

Definition 17[12]. (*Super-network*)

Given a network G=(V, E, σ), $\widetilde{\mathrm{V}} = \{\mathrm{V}_1, \mathrm{V}_2, \dots, \mathrm{V}_\mathrm{K}\}$ is a community partition of the node set V and $\forall \mathrm{V}_\mathrm{i} \in \widetilde{\mathrm{V}}$, the sub-network *MC*$_\mathrm{i}$=(V$_\mathrm{i}$, E$_\mathrm{i}$) induced by the node set V$_\mathrm{i}$ is a local

Micro-community in G. Define $\widetilde{\mathrm{E}} = \{(u,v) \in \mathrm{E} | \exists u \in \mathrm{V_i}, \exists v \in \mathrm{V_j}\}$ and $\widetilde{\sigma}(\mathrm{V_i}, \mathrm{V_j}) = \max\{\sigma(u,v) | u \in \mathrm{V_i}, v \in \mathrm{V_j}\}$; then $\widetilde{\mathrm{G}} = (\widetilde{\mathrm{V}}, \widetilde{\mathrm{E}}, \widetilde{\sigma})$ is called a ***Super-network*** of G.

Especially, the algorithm not only discovers all the communities but also the hubs and outliers in the network. A hub is called an overlap community and a hub plays a special role in many real networks such as search engines of web page network and the communication center of protein. An outlier does not belong to any communities because the similarities between it and other nodes are too small. This algorithm does not partition all the nodes into communities and this property is just perfect for our requirement.

5.3 Measurement of Partitioning Quality

Although several well-known quality measures such as normalized cut [24] and modularity [18] have been proposed, the modularity is the most popular measure by far. GSCAN and SHRINK both use the same similarity based modularity function Qs [25].

$$\mathrm{Q_s} = \sum_{\mathrm{i}=1}^{\mathrm{k}} \left[\frac{\mathrm{IS_i}}{\mathrm{TS}} - \left(\frac{\mathrm{DS_i}}{\mathrm{TS}}\right)^2\right] \qquad (9)$$

Assume the community partition has k communities $\{C^1, C^2, \ldots, C^k\}$, $IS_i = \sum_{u,v\in C^i} \sigma(u,v)$ is the total similarity of the nodes within cluster C^i, $DS_i = \sum_{u\in C^i, v\in V} \sigma(u,v)$ is the total similarity between the nodes in cluster C^i and any nodes in the network, and $TS = \sum_{\mu,v\in V} \sigma(\mu,v)$ is the total similarity between any two nodes in the network.

SHRINK is based on this quality function (Qs) and incrementally calculates the increment of the modularity quality. Given two adjacent local communities C^i and C^j, the modularity gain can be computed by

$$\Delta\mathrm{Q_s} = \mathrm{Q_S}^{C^i \cup C^j} - \mathrm{Q_S}^{C^i} - \mathrm{Q_S}^{C^j} = \frac{2\mathrm{US_{ij}}}{\mathrm{TS}} - \frac{2\mathrm{DS_i} * \mathrm{DS_j}}{(\mathrm{TS})^2} \qquad (10)$$

Where $US_{ij} = \sum_{\mu\in C^i, v\in C^j} \sigma(\mu,v)$ is the summation of similarity of total edges between two communities C^i and C^j.

Based on Eq (10), assume that the micro-community MC_i is constructed by h clusters i.e.: $MC_i = \{ C^{i1}, C^{i2}, \ldots, C^{ih} \}$, the modularity gain ΔQ_s for merging a micro-community MC_i into a *super-node* can be easily computed as

$$\Delta Q_s(MC_i) = \frac{\sum_{x,y \in \{i1,i2,\ldots,ih\}} US_{xy}}{TS} - \frac{\sum_{x,y \in \{i1,i2,\ldots,ih\}} DS_x * DS_y}{(TS)^2} \qquad (11)$$

SHRINK uses the modularity gain to control the shrinkage of the micro-communities. Only while the modularity gain is positive ($\Delta Q_s(MC) > 0$), these communities within micro-community (*MC)* could be merged into a *super-node*.

5.4 Algorithm of SHRINK

We use Fig 5-1 ~ Fig 5-5 to illustrate the key point of SHRINK. Given a simple network G as shown in Fig 5-1, nodes indicate the individuals and the weight of an edge indicates the *Structural Similarity* between individuals.

Each round of process of SHRINK has two phases. (1) For each node u we considered *u* as a *micro-community MC*(*u*), determine if each neighbor of the node x within *MC*(*u*) is the *Dense pair*. If *x* within *MC*(*u*) and a node *v* of the neighbors of *x* is *Dense pair*, then push *v* into the *micro-community MC*(*u*). The example is shown the Fig 5-3(a) and 5-4(a). The dotted lines indicate all *Dense pairs* found in the network G. (2) SHRINK determines the $\Delta Q_s(MC_i)$ for all *micro-communities* {*MC*$_1$, *MC*$_2$, …*MC*$_k$} and only while the $\Delta Q_s(MC_i) > 0$, all the nodes of the *micro-community MC*$_i$ would be merged into a *super-node* which contains more than one node at next round. Fig 5-3(b) and 5-4(b) show the second process of SHRINK. Fig 5-5(a) shows the result of third round of the SHRINK process and Fig 5-5(b) shows the fourth round of the SHRINK process. Fig 5-5(b) displays that the process of SHRINK terminates of $\sum \Delta Q_s(MC_i) < 0$. The network shrieked from G is called *Super-network* as shown in Fig 5-3(a) and Fig 5-4(a). Fig 5-5(b) shows the final result of SHRINK and there is a hub (node 2) and an outlier (node16).

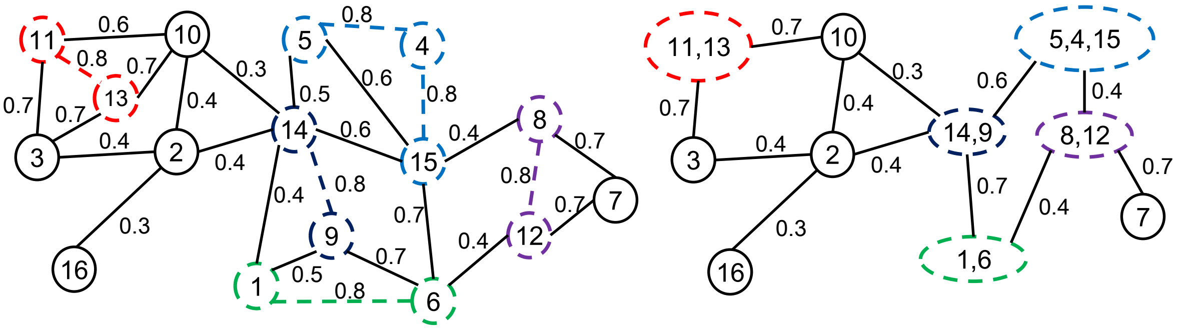


Fig 5-3 Round 1(a): Find all Dense pairs in G (b)Δ Q>0, Shrink

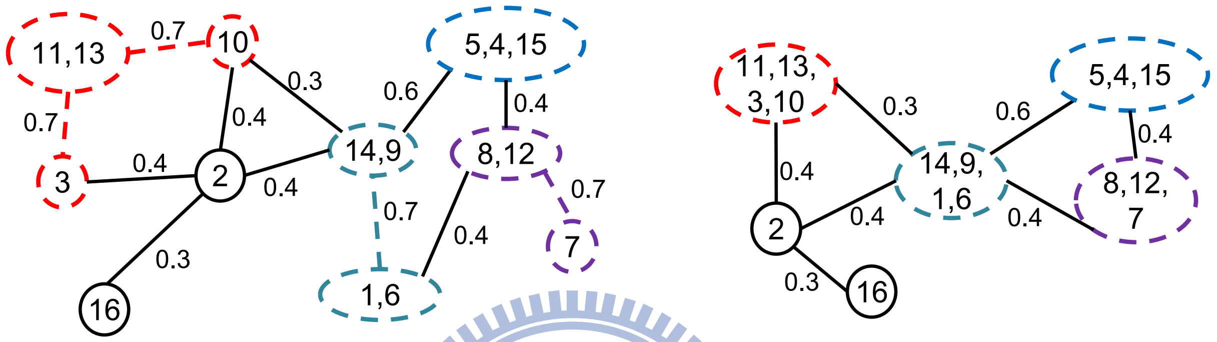


Fig 5-4 Round 2(a): Find all Dense pairs in G (b)Δ Q>0, Shrink

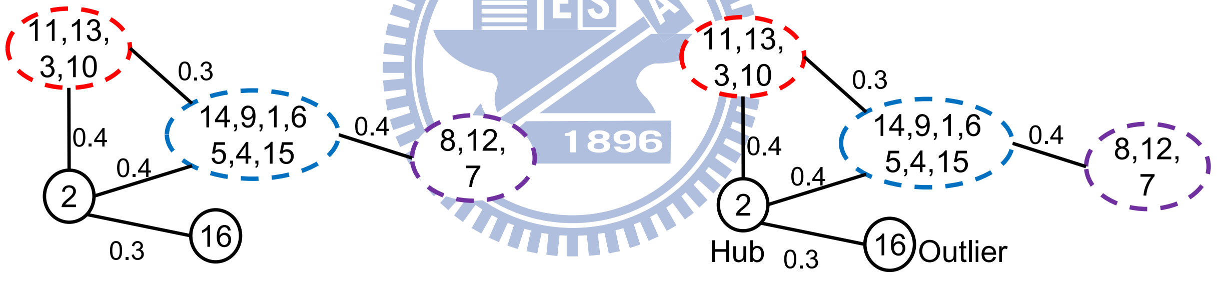


Fig 5-5 (a): Round 3 shrink becauseΔ Q>0 (b) Round 4 no-shrink becauseΔ Q<0

Algorithm1 : SHRINK[12]

```
Input: weighted networks G = (V, E)
Output: Set of clusters CP = {C_1, C_2, …, C_k};   Set of hubs and outliers N;
1:  CP← {{vi}|vi ∈ V} ;
2:  while true do
3:      //Phase 1: Detect the all Dense Pairs of G
4:      Micro-community MC (v) ← {v};
5:      for each unclassified  v ∈ V  do
6:          temp community C(v)  ←  ∅  ;
7:          Classify  v;   Queue q;   q.insert(v) ;
8:          ε ← max {σ(v, x)|x ∈ Γ(v) − {v}} ;
9:          While q.empty()  ≠  true do
10:             u← q.pop();
11:             if u=v ∨ max{σ(u, x)|x ∈ Γ(v) − {u}}=ε
12:                 C(v) ← C(v)∪{u} ;
13:                 for each w  ∈ Γ(u) − u
14:                     if  σ(w, u)  =  ε
```

```
                            q.insert ($w$) ;
                        end
                    end
                end
            end
            MC ← MC ∪ C($v$)
        end
        //Phase 2.2: Shrink micro-community
        $\Delta Q_S \leftarrow 0$ ;
        for each C ∈ MC do
            if $|C| > 1 \wedge \Delta Qs(MC) > 0$ then
                CP ← (CP− $\bigcup_{vi \in C}\{\{vi\}\}) \cup \{\tilde{v}\}$ ) ;
                $\Delta Q \leftarrow \Delta Q + \Delta Qs(MC)$;
            end
        end
        if $\Delta Q = 0$ then
            break;
        end
    end
    N ←∅
    for each C ∈ CP do
        if |C|=1
            CP ← CP − C ;
            N ← N ∪ C ;
        end
    end
    return CP, N ;
```

Figure 5-6 Algorithm of SHRINK

5.5 Algorithm of GSCAN

In this section, we describe the algorithm GSCAN. GSCAN is extended from SCAN [21] and combines with greedy heuristic setting of ε . SCAN performs one pass scan on each node of a network and finds all structure connected clusters for a given parameter setting. The pseudo code of the algorithm SCAN is presented in Fig 5-7. Given a weighted undirected graph G (V, E), at the beginning all nodes are labeled as unclassified. For each node that is not yet classified, SCAN checks whether this node is a ***core node***. If the node is a core, a new cluster is expanded from this node. Otherwise, the vertex is labeled as a non-member.

GSCAN use greedy heuristic setting of ε to optimize the modularity score Q of clustering result. GSCAN adjusts the ε with change of modularity score Q and decreases or increasesε until Q reaching the local maximum modularity [5].

Algorithm2 : SCAN [21]

```
Input: weighted networks G = (V, E), $\varepsilon$ , $\mu$
Output: Set of clusters CP = {$C_1, C_2, \ldots, C_k$} , modularity score Q .
1:  // all nodes in V are labeled as unclassified;
2:  CP ←∅;
3:  for each unclassified node v ∈ V do
4:      // STEP 1. check whether v is a core;
5:      if $CORE_{\varepsilon,\mu}$ (v) then
6:          // STEP 1.1. if v is a core node, a new cluster is expanded;
7:          set C ←∅;
8:          insert all x ∈ $N_\varepsilon$ (v) into queue Q;
9:          while Q ≠ 0 do
10:             y = first vertex in Q;
11:             R = {x ∈ V | $DirREACH_{\varepsilon,\mu}$ (y, x)};
12:             for each x ∈ R do
13:                 if x is unclassified or non-member then
14:                     C ←C ∪ x;
15:                 end
16:                 if x is unclassified then
17:                     insert x into queue Q;
18:                 end
19:             end
20:             remove y from Q;
21:         end
22:         CP ←CP ∪ C;
23:     end
24: end
25: // 1.2 determine the modularity score of CP;
26: Q = $Q_S$(CP);
27: return CP , Q
```

Figure 5-7 Algorithm of SCAN

GSCAN chooses a median of similarity values of the sample nodes picked from V and the sampling rate is only about 5~10%. GSCAN increases or decreases the ε by a unit $\Delta\varepsilon$ = 0.01 or 0.02 and maintains two kinds of heaps: (1) max heap H_{max} for edges having similarity below seedε ; and (2) min heap H_{min} for edges having similarity above seedε . Hmax and Hmin are built during the initial clustering. After finding the initial clusters CP and calculating its modularity Q_{mid}, GSCAN calculates two additional modularity values, Q_{high} and Q_{low}. Here, Q_{high} is calculated from CP with the edges having similarity of range [seedε , seedε + $\Delta\varepsilon$] in H_{min}, and Q_{low} calculated from CP except the edges having similarity of range [seedε - $\Delta\varepsilon$, seedε] in H_{max}. If Q_{high} is the highest among Q_{high}, Q_{mid}, and Q_{low}, GSCAN increases the density by $\Delta\varepsilon$. If Q_{low} is the highest value, GSCAN decreases the density by $\Delta\varepsilon$. Otherwise, seedε would be the best density parameter. The initial clustering CP is continuously modified by adding edges from H_{max} to CP or by deleting edges of H_{min} from CP. The detailed algorithm of GSCAN is in Fig 5-8[5]:

Algorithm2 : GSCAN [5]

```
Input: weighted networks $G = (V, E)$, $\varepsilon$ , $\mu$
Output: Set of clusters $CP = \{C_1, C_2, \ldots, C_k\}$, modularity score Q.
1:  $CP_{mid} \leftarrow \emptyset$;  $CP_{high} \leftarrow \emptyset$ ;  $CP_{low} \leftarrow \emptyset$ ;
2:  $Q_{mid} \leftarrow 0$ ;  $Q_{high} \leftarrow 0$ ;  $Q_{low} \leftarrow 0$ ;
3:  $(CP_{mid}, Q_{mid}) \leftarrow$ SCAN$(G, \varepsilon , \mu)$ ;
4:  $(CP_{high}, Q_{high}) \leftarrow$ SCAN$(G, \varepsilon + \Delta\varepsilon, \mu)$ ;
5:  $(CP_{low}, Q_{low}) \leftarrow$ SCAN$(G, \varepsilon - \Delta\varepsilon, \mu)$ ;
6:  while ( $Q_{mid}$ != max $(Q_{mid}, Q_{high}, Q_{low})$) do
7:      if  $Q_{high}$ == max $(Q_{mid}, Q_{high}, Q_{low})$
8:          $\varepsilon \leftarrow \varepsilon + \Delta\varepsilon$ ;
9:          $(CP_{mid}, Q_{mid}) \leftarrow (CP_{high}, Q_{high})$;
10:         $(CP_{low}, Q_{low}) \leftarrow (CP_{mid}, Q_{mid})$ ;
11:         $CP_{high}, Q_{high} \leftarrow$ SCAN$(G, \varepsilon + \Delta\varepsilon, \mu)$ ;
12:     else if  $Q_{min}$ == max $(Q_{mid}, Q_{high}, Q_{low})$
13:         $\varepsilon \leftarrow \varepsilon + \Delta\varepsilon$ ;
14:         $(CP_{high}, Q_{high}) \leftarrow (CP_{mid}, Q_{mid})$;
15:         $(CP_{mid}, Q_{mid}) \leftarrow (CP_{low}, Q_{low})$ ;
16:         $(CP_{low}, Q_{low}) \leftarrow$ SCAN$(G, \varepsilon - \Delta\varepsilon, \mu)$ ;
17:     end
18: end
19: return $CP_{mid}, Q_{mid}$ ;
```

Figure 5-8 Algorithm of **GSCAN**

Chapter 6
The Community Pedigree Mapping

Current methods [6, 13, 3, 5] made effort in the problem of mapping one community of previous timestamp to one community of current timestamp (1-1 **mapping problem**). We argue these methods are not suitable for real community evolution as discussed in section 2.3. So we propose a community pedigree mapping to express the evolution of communities. In this chapter we illustrate the evolution of communities between two consecutive time points. Section 6.1 provides the description of community similarity and section 6.2 presents the states of a community during its lifetime of community. Section 6.3 introduces the details of *Community Pedigree Mapping* and section 6.4 presents "relationship Extraction and community Pedigree dynamic Community miner" (EPC).

6.1 The Similarity between Communities

Social networks are dynamic and different amount of individuals are alive at each time point. If some individuals disappear at the time point either t-1 or t, we assume those are negative ones. On the other hand, the positive individuals who we care about are alive at both two consecutive time points t-1 and t. We further define those individuals as *Influence individuals.*

Definition 11 (*Influence individuals*)

Given the relationship graphs $RG_{t-1}(V_{t-1}^R, E_{t-1}^R)$ and $RG_t(V_t^R, E_t^R)$. The V_{t-1}^R is the node set of the relationship graph RG_{t-1} and V_t^R is the node set of RG_t. We define the *Influence individuals* at time points t-1 and t are:

$$\Pi(t) = \{u \in V | u \in V_{t-1}^R \cap V_t^R\} \tag{12}$$

Our Community Pedigree Mapping is based on *Influence individuals* to determine the similarity of community.

Definition 12 (*The similarity between the Communities at different timestamps*)

Given the i-th community $C_{t-1}^{i}=(V_{t-1}^{i}, E_{t-1}^{i})$ of community partition CP_{t-1} at time t-1 and the j-th community $C_t^j = (V_t^j, E_t^j)$ of CP_t at time t. The similarity between community C_{t-1}^{i} and community C_t^j is the number of individuals who are alive at both time points t-1 and t and is defined below.

$$\Phi\left(C_{t-1}^{i}, C_t^j\right) = \frac{V_{t-1}^{i} \cap V_t^j \cap \Pi(t)}{\max\left(\left|V_{t-1}^{i} \cap \Pi(t)\right|, \left|V_t^j \cap \Pi(t)\right|\right)} \tag{13}$$

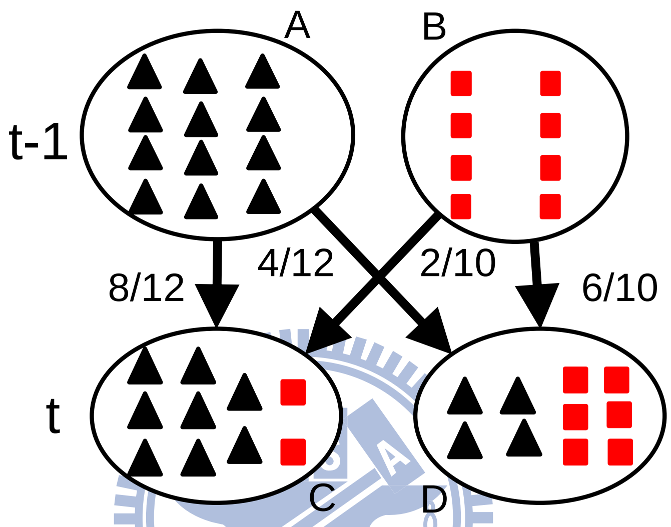
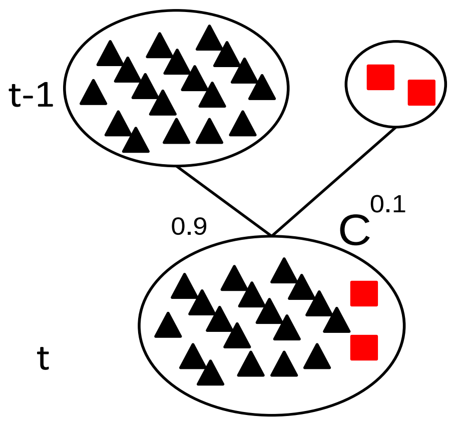


Figure 6-1 Example of community similarity calculation

Take Fig 6-1 as an example, we using it illustrate the function 6-2. There are two communities A and B at time t-1 and two communities C and D at time t. There are 8 members overlapped between community A and community C. The similarity between A and C is $\frac{8}{12}$ since the community size of A is larger then C (12> 10).



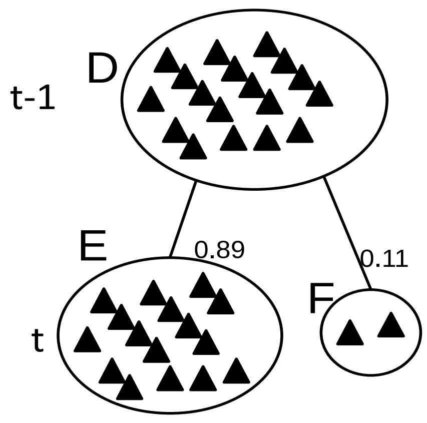


Figure 6-2 (a) Merge example (b) Split example

There is a serious problem, splitting or merging of small community would cause the real alive state disappear, while we determine the similarity between the communities. We use the Fig 6-2 to illustrate the situation. Considering the two communities A and B at time t-1 and C at time t in Fig 6-2(a), C is still the combination of A and B even though the members of community C are almost the members of A while there is no constrain for similarity generation; In Fig 6-2(b), Communities E and F both are the splitting of D even though F has few members of community D. The above phenomenon is not reasonable so we predefine the *Minimum community similarity threshold* (ρ) to avoid this situation. On the other hand, we set the $\Phi\left(C_{t-1}^{i}, C_{t}^{j}\right) = 0$ while the $\frac{V_{t-1}^{i} \cap V_{t}^{j} \cap \Pi(t)}{\max\left(\left|V_{t-1}^{i} \cap \Pi(t)\right|, \left|V_{t}^{j} \cap \Pi(t)\right|\right)} <$ *Minimum community similarity threshold* (ρ).

6.2 The State of Communities

Dynamic Community could change over time and we further define five community states: *Birth, Death, Alive, Child and Division*.

Definition 13 (*Birth*)

A new community C_t^j is born at time t iff there is no similarity between C_t^i and any communities $C_{t-1}^i$of CP_{t-1}. i.e.: $\exists C_t^j \in CP_t,\ \ \nexists C_{t-1}^i \in CP_{t-1}: \Phi\left(C_{t-1}^i, C_t^j\right) > 0$

Definition 14 (*Death*)

An old community C_{t-1}^i is dead at time point t iff there are none similarity between C_{t-1}^i and any communities of community partition CP_t. i.e.:$\exists C_{t-1}^i \in CP_{t-1}, \nexists C_t^j \in CP_t: \Phi\left(C_{t-1}^i, C_t^j\right) > 0$

Definition 15 (*Alive*) A current community C_t^j is alive iff there is similarity between one community C_{t-1}^i of community partition CP_{t-1} and one community C_t^j of CP_t. i.e.: $\exists C_t^j \in CP_t, \exists!\, C_{t-1}^i \in CP_{t-1}: \Phi\left(C_{t-1}^i, C_t^j\right) > 0$

Definition 16 (*Child*)

A current community C_t^j is a child of $\{C_{t-1}^1, C_{t-1}^2, \dots, C_{t-1}^h\}$ iff there exist more than one community C_{t-1}^{i} in CP_{t-1} having similarity to a community C_{t}^{j} of CP_{t}. i.e.: $\exists C_{t-1}^{i} \in CP_{t-1,i=1,\dots,h}, \exists! \, C_{t}^{m} \in CP_{t}: \Phi_{i\in\{1,2,\dots,h\}}\left(C_{t-1}^{i}, C_{t}^{j}\right) > 0$

Definition 17 (*Fission*)

A current community $\{C_{t}^{1}, C_{t}^{2}, \dots, C_{t}^{h}\}$ are fissions of C_{t-1}^{i} iff there exist communities $\{C_{t}^{1}, C_{t}^{2}, \dots, C_{t}^{h}\}$ of CP_{t} similar to a single community C_{t-1}^{i} of CP_{t-1} $\wedge$ $\forall C_{t}^{i} \in CP_{t}$. i.e.: $\exists C_{t}^{1}, C_{t}^{2}, \dots, C_{t}^{h} \in CP_{t}, \exists! \, C_{t-1}^{i} \in CP_{t-1}: \Phi_{i\in\{1,2,\dots,h\}}\left(C_{t-1}^{i}, C_{t}^{i}\right) > 0$

We use the Fig 6-3 to illustrate the five states of community. A node indicates a community and an edge weight indicates the similarity between communities. There are 5 communities (A, B, C, D, E) at time point t-1 and 5 communities (F, G, H, I, J) at time point t. The Fig 6-3(a) is the original graph and the Fig 6-3(b) is the evolution result of our community pedigree Mapping.

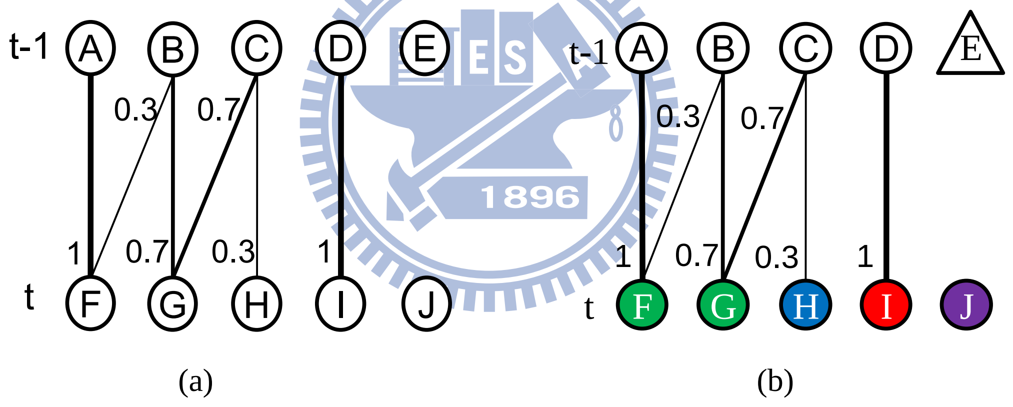


Figure 6-3 Example of evolution of communities

(1) Birth: A circle colored purple indicates that a community is born at current time point t and the example is shown as the community J in Fig 6-3(b). (2) Death: A triangle indicates that a community would be dead at next time point and the example is shown as the community E in Fig 6-3(b). (3) Alive: A circle of colored red indicates that a community is Alive from only one community at time t-1 to only one community at time t. The example is shown as the community I at time t which is Alive from the community D at time t-1 in Fig 6-3(b). (4) Child: A circle of colored green indicates that a community is a child of some communities at time t-1. The example is shown as the communities F which is the child of A

and B in Fig 6-3(b); (5) Fission: A circle of colored blue indicates that a community is a fission of a single community of time t-1 and the example is shown as the community H which is a fission of C in Fig 6-3(b).

6.3 Community Pedigree Mapping

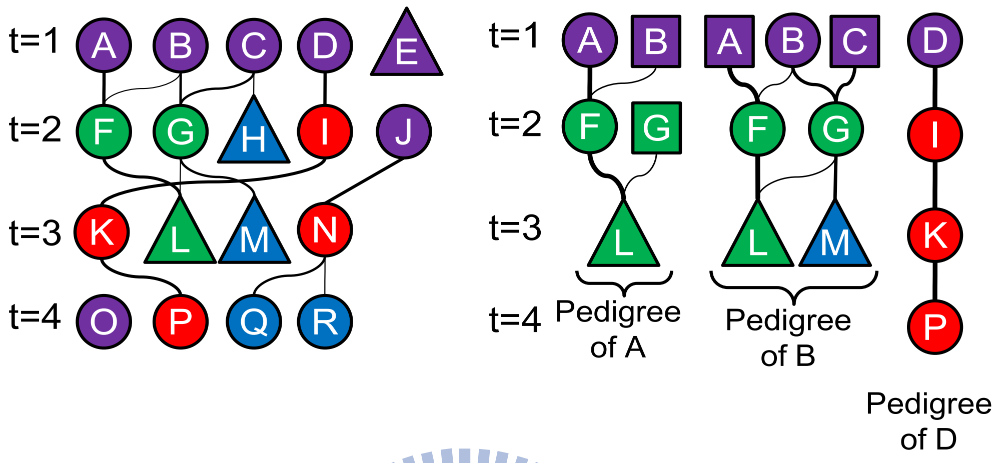


Figure 6-4 (a) Evolution net (b) Pedigree of single Community

After we determine the similarities between communities and the states of communities, the states of a community express the evolution of all communities as in 6-4(a). Besides, the evolution of single community we called pedigree. We use the same states of a community to express the evolution of single community. Fig 6-4(a) shows the similarities between communities and the state of communities from t=1 to t=4 and is called *evolution net* [3]. Fig 6-4(b) shows the pedigrees of community A, B and D. In the pedigree of specify community, a circle shape indicates that this community has blood relationship with the specific community. A square shape indicates that this community is non-blood relationship spouses of blood relationship communities.

In the pedigree of A, communities B is a non-blood relationship spouse of A. Their child F has a spouse G at time t=2 and their child is L. The community L is dead at time 4 so the pedigree of community A ends at time t=4. In the middle of Fig 6-4(b), the pedigree of community B, F is the child of A and B; G is the child of B and C; the community L is the child of F and G. The community G is blood relationship spouse of F so the shape of G is circle; Community M is fission of G. We could see the pedigree of community B ends at time t=4. In the pedigree of community D. Community I is Alive of D; K is Alive of I and P is Alive of K. The pedigree of D develops and does not finish.

Though the illustration, we could observe the life time of community. Community could be alive, split and merge over time. The proposed *community pedigree Mapping* expresses the evolution of community and solves the **“one to one mapping problem”.**

6.4 Proposed Algorithm: “relationship Extraction and community Pedigree dynamic Community miner” (EPC)

Assume a dynamic network G={G_1,G_2,…,G_t,…} where G_t is the interaction graph at t, the observation eyeshot wr, the selected normalized weight function (N(t,tc)) and ***Minimum community similarity threshold*** ρ, we start the algorithm EPC. For each time point tc, we determine the ***observation window*** Ψ=[tc-wr, tc+wr] and EPC could be divided into three steps:

(1)Construct the relationship graph RG_{tc}:

For each edge within interaction graphs G_t where time point t is belong to observation window Ψ, we calculate the relationship strength between individuals (u,v) using Eq (3): $W_{tc}(u,v) = \sum_{\forall\, t \in \Psi} I_{u,v}(t) \times N(t,tc)$ and then we contruct the Relationship graph RG_{tc}.

(2)Use the clustering method, SHRINK [12], to discover the community partition CP_{tc} based on relationship graph RG_{tc}:

(3)Determine the evolution net (EN_{tc}) for every two consequent timestamps tc-1 and tc:

Based on the predefined *Minimum community similarity threshold* (ρ) and the community partition results at time tc-1 and tc, we calculate the similarity between communities of tc-1 and the communities of tc using Eq (13). For each community, we determine the states of communities. The detail algorithm is shown in Fig 6-5.

```
Algorithm 2 : EPC
Input: Dynamic networks $G = \{G_1, G_2, \ldots, G_t, \ldots\}$ where $G_t$ is the interaction graph of time t, observation
eyeshot (wr), Selected normalized weight function (N(t,tc)), Minimum community similarity threshold (ρ)
Output: Community partition of each time point $\{CP_1, CP_2, \ldots, CP_t, \ldots\}$, the evolution net of all communities
at all time points $\{EN_1, EN_2, \ldots, EN_t, \ldots\}$
1:  for each tc $\in$ T do
2:      $V_{Rtc-1}$ = Vertex set of Relationship graph $RG_{tc}$ ;
3:      Relationship graph $RG_{tc} \leftarrow \varnothing$;
4:      Evolution net $EN_{tc} \leftarrow \varnothing$;
5:      Community partition $CP_{tc-1} \leftarrow CP_{tc}$;  $CP_{tc} \leftarrow \varnothing$;
6:      //Phase 1: Construct the relationship graph $RG_{tc}$
7:      for each edge $(u, v) \in G_t$, where t $\in \Psi$
8:          Determine the relationship strength $W_{tc}(u,v) = \sum_{\forall t \in \Psi} I_{u,v}(t) \times N(t, tc)$ ;
9:          Insert the edge $(u, v, weight = W_{tc}(u,v))$ into Relationship graph $RG_{tc}=(V_{Rtc}, E_{Rtc})$ ;
10:     end
11:     //Phase 2: Mining the community using SHRINK
12:     $CP_{tc}$ = SHRINK ($RG_{tc}$) or GSCAN($RG_{tc}$, ε, μ);
13:     //Phase 3: Determine the evolution of community
14:     $V_{Rtc}$ = Vertex set of $RG_{tc}$;
15:     Calculate the Influence individuals $\Pi(tc) = V_{Rtc} \cap V_{Rtc-1}$ ;
16:     Determine the community similarity $\Phi(C^n_{t-1}, C^m_t)$ for all $C^n_{tc-1} \in CP_{tc-1}$ and $C^m_{tc} \in CP_{tc}$;
17:     for each community $C^n_{tc-1} \in CP_{tc-1}$ do
18:         if $C^n_{tc-1}$ has no similarity with any $C^m_{tc} \in CP_{tc}$ then
19:             The state of $C^n_{tc-1}$ in evolution net $EN_{tc} \leftarrow$ 'Death';
20:         end
21:     end
22:     for each $C^m_{tc} \in CP_{tc}$ do
23:         if $C^m_{tc}$ has no similarity with any $C^n_{tc-1} \in CP_{tc-1}$ then
24:             The state of $C^m_{tc}$ in evolution net $EN_{tc} \leftarrow$ 'Birth';
25:         else if $C^m_{tc}$ has only one similarity with $C^n_{tc-1} \in CP_{tc-1}$ then
26:             if $C^n_{t-1}$ has only one similarity with $C^m_{tc}$ then
27:                 The state of $C^m_t$ in evolution net $EN_{tc} \leftarrow$ 'Alive';
28:             else
29:                 The state of $C^m_t$ in evolution net $EN_{tc} \leftarrow$ 'Fission';
30:             end
31:         else if $C^m_t$ has more than one similarities with some $C^n_{tc-1} \in CP_{tc-1}$ then
32:             The state of $C^m_t$ in evolution net $EN_{tc} \leftarrow$ 'Child';
33:         end
34:     end
35:     Output $CP_{tc}$, $EN_{tc}$;
36: end
```

Figure 6-5 Algorithm of EPC

For each iteration of EPC, the time complexity of step 1 is $O(|E^R_{tc}|)$ where the E^R_{tc} is the edge set of relationship graph RG_{tc}=(V^R_{tc} E^R_{tc}); the time complexity of SHRINK in step 2 is $O(|E^R_{tc}|\log(|V^R_{tc}|))$ and the time complexity of step 3 is $O(|V^R_{tc}|)$. The total time complexity of each iteration of EPC is $O(|E^R_t|\log(|V^R_t|))$ and the efficiency bottleneck of EPC depends on the clustering algorithm chosen.

Chapter 7
Experimental results and Performance study

In this chapter, the accuracy and efficiency of EPC would be examined. The environment is on a AMD Athlon(tm) Π X2 240 CPU of 2.8 GHz with 2GBytes of main memory, running on Windows XP. The proposed EPC is implemented using C++. We compare EPC with the FacetNet [3] and PD-Greedy [5] by using 2 synthetic datasets SYN-FIX and SYN-VAR. SYN-FIX generates the dynamic network of a fixed number of communities and fixed number of nodes over time. SYN-VAR generates the dynamic network of a variable number of communities and variable number of nodes over time. For accuracy comparison, we use the mutual information to evaluate the performance.

Section 7.1 describes the details of synthetic dataset generator and quality measurement using mutual information. Section 7.2 presents the discussion of all parameters of all comparison algorithms. Section 7.3 presents the accuracy of synthetic data experiment. Section 7.4 presents the smoothness quality of synthetic data experiment. Section 7.5 presents the scalability of synthetic data experiment and section 7.6 presents the result of real data experiment.

7.1 Synthetic Data generation

7.1.1 SYN-FIX

Parameter	Description	Default
n	Initial number of vertices.	128
s_c	Initial size of community.	32
n_c	Initial number of communities.	4
Avg_v_deg	Average vertex degree.	16
Avg_v_out_deg	Average vertex degree out of original community.	3 ~ 5
Ran_sel	Random select some vertices out of original community.	3

Table 7-1 Parameters of SYN-FIX

The data generator SYN-FIX has been released in [26] and the original idea is proposed in [18]. The same idea is also discussed in [5]. The SYN-FIX produces an environment of fixed number of nodes and communities over time. It generates a network which has 128 nodes, four communities of 32 nodes each and average vertex degree (Avg_v_deg) 16. Table 7-1 describes the parameters of SYN-FIX.

In SYN-FIX, the parameter Avg_v_out_deg is the average out-degree of all nodes in network. The Avg_v_out_deg controls the number of inter-edges placed between different communities so the number of intra-edge placed in a community is decided at the same time. The number of intra-edges is increasing while Avg_v_out_deg is decreasing. We generate two datasets of SYN-FIX (SYN-FIX-VOD_3 and SYN-FIX-VOD_5) and produce such networks for twenty consecutive timestamps. The SYN-FIX-VOD_3 uses the Avg_v_out_deg = 3 and the SYN-FIX-VOD_5 uses the Avg_v_out_deg = 5. At each timestamp 3 randomly selected nodes would leave original community and randomly join the other three communities.

7.1.2 SYN-VAR

Parameter	Description	Default
n	Initial number of vertices.	256
s_c	Initial size of community.	[32, 64]
n_c	Initial number of communities.	[4, 8]
Avg_c_e-ratio	Average community edge ratio.	[0.2, 0.8]
Avg_c_out_e-ratio	Average community edge ratio out of original community.	[0.3, 0.5]
Ran_sel	Random select some nodes out of original community.	[8, 20]
Add_v_at	Add some new vertices at each time point.	16
Add_new_c_at	Add new community at some time points.	
Remove_min_c_at	Remove min-community at some time points.	

Table 7-2 Parameters of SYN-VAR

SYN-VAR is first discussed in [5]. The SYN-VAR produces an environment of variable number of nodes and communities over time. The Average community edge ratio (Avg_c_e-ratio) is the ratio of the maximum intra-edges to the node number of the selected community. The average community edge ratio out of original community (Avg_c_out_e-ratio) means the ratio of inter-edges to total edges of the community. That is we generate the network containing 256 nodes, 4 communities of 64 nodes each. Set the average community edge-ratio (Avg_c_e-ratio) to 0.5 and average community edge ratio out of original community (Avg_c_out_e-ratio) to 0.3. The total number of edges of a single community is 0.5*(64*63/2) = 1008. The number of inter-edges is 1008*0.3 = 432 and the number of intra-edges is 1008-432 = 676.

All datasets produced by SYN-FIX and SYN-VAR is shown as in Table 7-3.

Type	Synthetic Dataset	Description
SYN-FIX	SYN-FIX-VOD_3	Fixed node number, fixed community number and average vertex out degree=3.
	SYN-FIX-VOD_5	Fixed node number, fixed community number and average vertex out degree=5.
SYN-VAR	SYN-VAR-COE_0_3_REG	Dynamic changed node number, dynamic changed community number, average community out edge ratio = 0.3, creating new communities at time points {3, 5, 7, 9, 11, 13, 15, 17, 19} and deleting the smallest community at time{4, 6, 8, 10, 12, 14, 16, 18, 20}
	SYN-VAR-COE_0_5_REG	Dynamic changed node number, dynamic changed community number, average community out edge ratio = 0.5, creating new communities at time points {3, 5, 7, 9, 11, 13, 15, 17, 19} and deleting the smallest community at time{4, 6, 8, 10, 12, 14, 16, 18, 20}
	SYN-VAR-COE_0_3_RAN	Dynamic changed node number, dynamic changed community number, average community out edge ratio = 0.3, randomly creating 7 new communities at time points {2, 3, 4, 6, 9, 10, 14} and deleting the smallest community at time{7, 9, 14, 15, 18, 19, 20} 20}
	SYN-VAR-COE_0_5_RAN	Dynamic changed node number, dynamic changed community number, average community out edge ratio = 0.5, creating new communities at time points {1, 2, 3, 6, 8, 11, 12} and deleting the smallest community at time{7, 8, 10, 11, 14, 16, 17}

Table 7-3 Synthetic datasets generated by SYN-FIX and SYN-VAR

We generate four datasets of SYN-VAR and produced such networks for twenty consecutive timestamps. At each timestamp SYN-VAR randomly selects some nodes to be removed and randomly selects some nodes to be added to the network. We simulate the dynamic property of social networks using three parameters: "the average community out edge ratio" (Avg_c_out_e-ratio), "add new community at some time points" (Add_new_c_at) and "remove min-community at some time points" (Remove_min_c_at). New communities would be created at randomly selected time points. The smallest community would be removed at randomly selected time points. New communities would be constructed by parts of the larger communities at previous time. For example: the dataset SYN-VAR-COE_0_3_REG indicates the average community out edge ratio = 0.3, regularly creates communities at time points {3, 5, 7, 9, 11, 13, 15, 17, 19} and regularly deletes the smallest community at time {4, 6, 8, 10, 12, 14, 16, 18, 20}. The description of other datasets would be clear in Table 7-3.

The dataset generators, SYN-FIX and SYN-VAR, provide the ground truths of the communities while they produce the interaction graphs of dynamic networks. The accuracy of EPC can be compared with these ground truths. We use the normalized mutual information (NMI) as the performance of accuracy since NMI is well-known to evaluate the quality of clusters produced by clustering algorithm [27] as is defined as:

$$\mathrm{NMI} = \frac{\mathrm{MI(X,Y)}}{[\mathrm{H(X)+H(Y)}]/2} \tag{14}$$

where MI(X, Y) is the Mutual information of two random variables X and Y. MI(X,Y) measures the mutual dependency of X and Y and is defined as:

$$\mathrm{MI(X,Y)} = \sum_{y\in \mathrm{Y}}\sum_{x\in \mathrm{X}} \mathrm{p}(x,y) \times \log\left(\frac{\mathrm{p}(x,y)}{\mathrm{p}_1(x)\times \mathrm{p}_2(y)}\right) \tag{15}$$

where p(x, y) is the joint probability distribution function of X and Y, and $p_1(x)$ and $p_2(y)$ are the marginal probability distribution functions of X and Y respectively.

H(X) is the entropy of X and H(Y) is the entropy of Y. Entropy is a measure of the uncertainty associated with a random variable. The score of NMI is normalized to 0.0~1.0 and the accuracy is higher while the score is higher.

7.2 Synthetic Data Experiment

We compare EPC algorithm with two algorithms, FacetNet [4] and PD-Greedy [5]. First we would experiment and discuss the effect of all different parameters of EPC-SHRINK, EPC-GSCAN, FacetNet and PD-Greedy.

7.2.1 Parameter of FacetNet: α

FacetNet is based on the concept of temporal smoothness and uses the parameterα to control the community partitioning. However, the parameter α could affect the community partitioning. So, we use different α to observe the change of the community partitioning.

As shown in Fig 7-1, FacetNet are tested using SYN-FIX and SYN-VAR datasets. The vertical axis is the average NMI and the horizontal axis is the parameter α from 0.1 to 0.9. Each line within Fig 7-1 is the performance of different datasets. The accuracy of FacetNet is stable whatever the value of α is. That is the FacetNet is not influenced by the value of α .

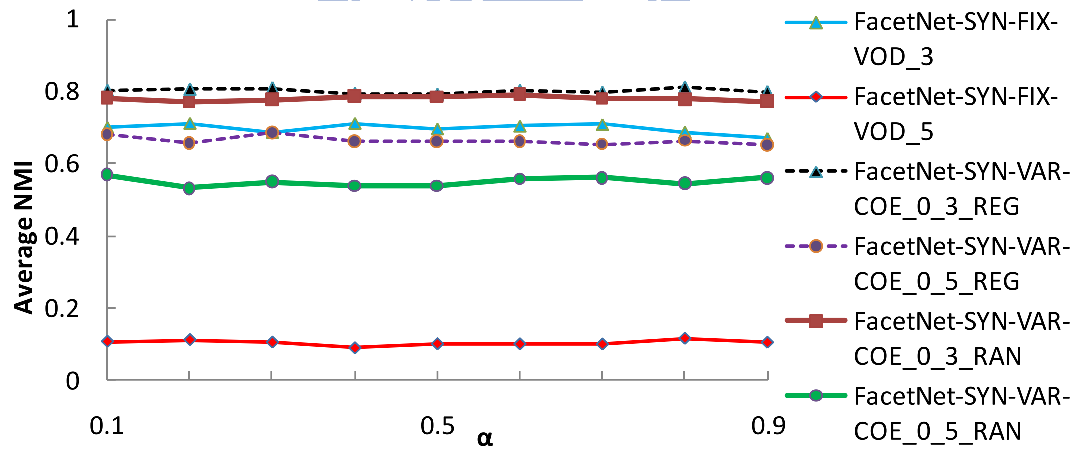


Figure 7-1 Parameter of FacetNet: α

7.2.2 Parameters of PD-Greedy[5]: α and μ

PD-Greedy has two parameters: α the parameter of temporal smoothness and μ the constraint of community size.

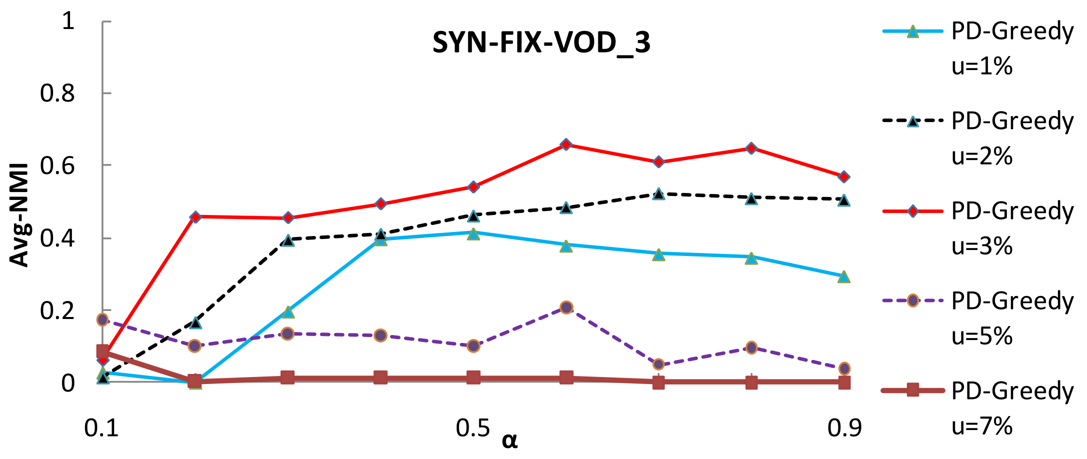


Figure 7-2 Parameters of PD-Greedy: α, μ on dataset SYN-FIX-VOD_3

Fig 7-2 shows the experimental result of PD-Greedy on α and μ. The vertical axis is the average NMI and the horizontal axis is the parameter α from 0.1 to 0.9. Each line within Fig 7-2 is the performance for different parameters μ on dataset SYN-FIX-VOD_3. The relevance between Avg-NMI and parameter μ is low and the presetting of parameter μ has dependency on datasets. While the parameter α is below 0.5, the Avg-NMI grows with α. While the parameter α is above 0.5, the trend of Avg-NMI changed fuzzily.

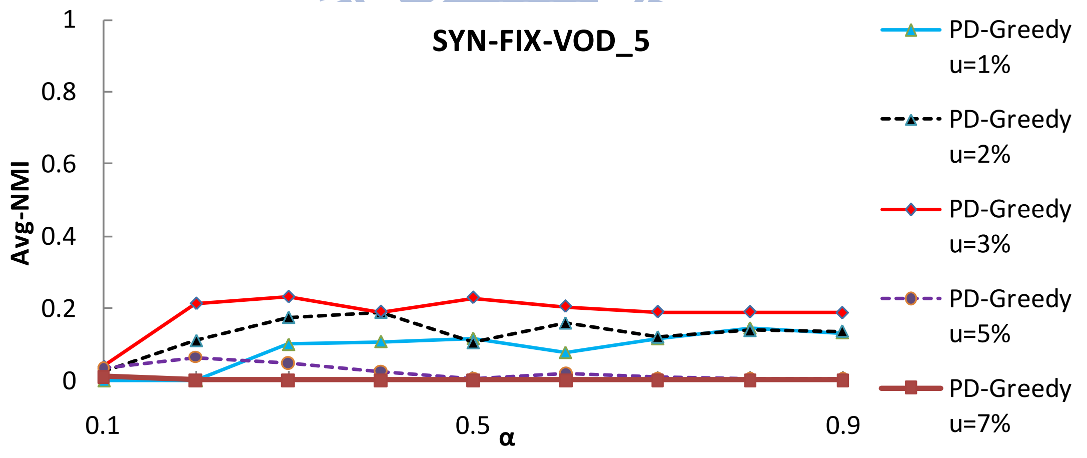


Figure 7-3 Parameters of PD-Greedy: α, μ on dataset SYN-FIX-VOD_5

Fig 7-3 shows the result of PD-Greedy on dataset SYN-FIX-VOD_5. Each line has low Avg-NMI due to the noise of dataset SYN-FIX-VOD_5 is higher than that of SYN-FIX-VOD_3. Avg-NMI has ambiguous correlation with parameter α.

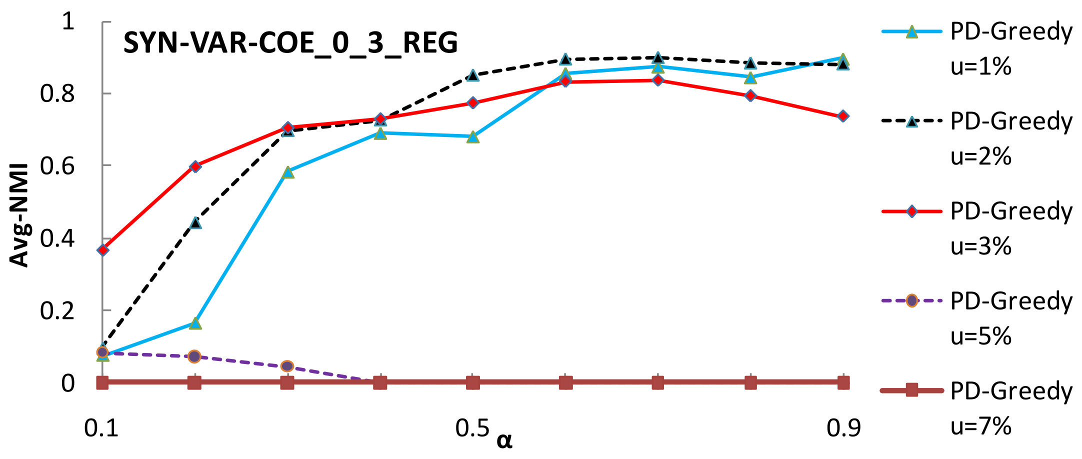


Figure 7-4 Parameters of PD-Greedy: α , μ on dataset SYN-VAR-COD_0_3_REG

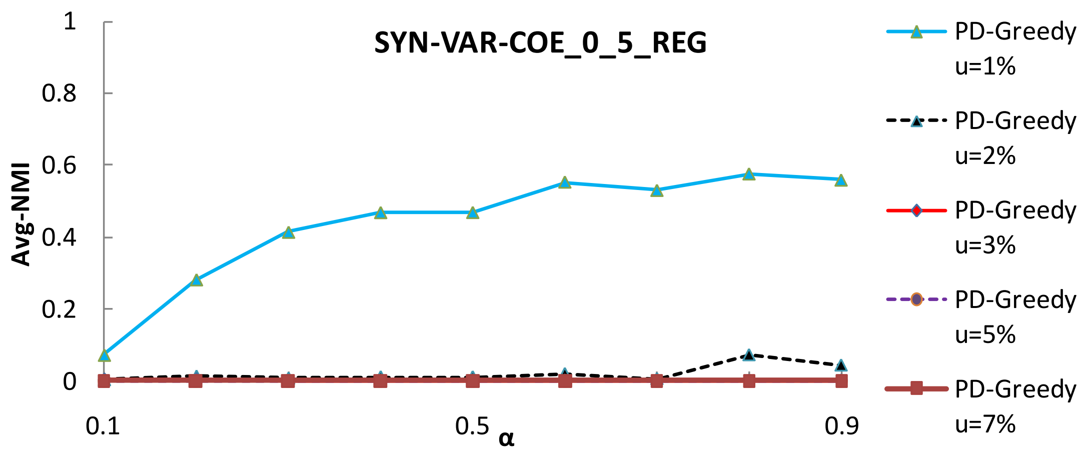


Figure 7-5 Parameters of PD-Greedy: α , μ on dataset SYN-VAR-COD_0_5_REG

Fig 7-4 and 7-5 are the experimental results of PD-Greedy on different datasets. Fig 7-4 and Fig 7-5 show the same phenomenon the same as Fig 7-2. i.e.: the Avg-NMI grows with α while the parameter α is below 0.5; the trend of Avg-NMI changed fuzzily while the parameter α is above 0.5. While the noise level is higher, Fig 7-5 shows that the Avg-NMI is higher using lower parameter μ . We do not show the experimental result of SYN-VAR-COE_0_3_RAN and SYN-VAR-COE_0_5_RAN due to the same property as shown in Fig 7-4 and 7-5.

In summary, the parameter α should be setting within the range between 0.6~0.9 and the parameter μ should be setting within the range between 1%~3% for producing better Avg-NMI.

7.2.3 Parameters of EPC-SHRINK: observation eyeshot, decay weight functions

EPC-SHRINK uses the clustering algorithm SHRINK [12] and has two parameters, observation window and four weight functions. EPC-SHRINK use equal weight function (EQL) and other three normalized decay weight functions, linear decay (LIN), exponential decay (EXP) and wave decay (WAVE). While we set the observation eyeshot wr increasing, the interaction data of overlap time points would increase, the change of relationship strength of relationship graph would be small then the change of community partition is guaranteed to be small. However, increasing the wr is not always better for all situations and we use Fig 7-6 to illustrate the circumstance.

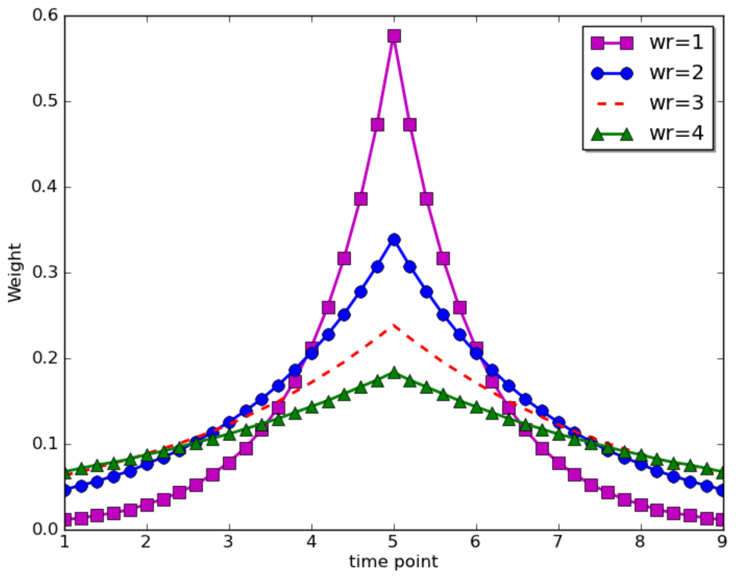


Figure 7-6 The EXP weight function under different observation eyeshot wr

Fig 7-6 shows the curves of EXP weight function based on different wr. That the curve is getting smooth with the increasing wr shows the decay property of EXP being averaged out with the increasing wr. So the range of wr should be fixed and the setting of wr depends on the property of network.

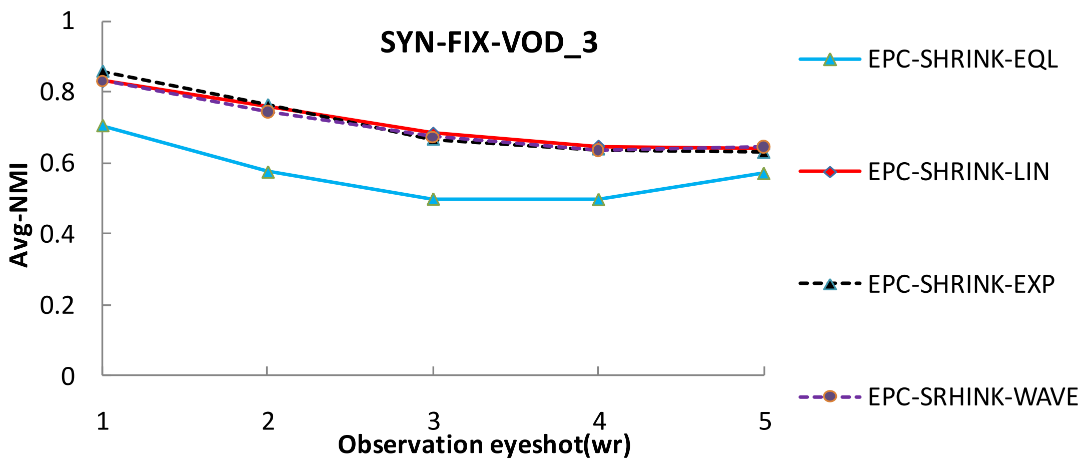


Figure 7-7 Parameters of EPC-SHRINK: observation eyeshot, decay weight functions on dataset SYN-FIX-VOD_3

Fig 7-7 shows the experimental result of EPC-SHRINK on dataset SYN-FIX-VOD_3. The vertical axis is the average NMI and the horizontal axis is the different values of observation eyeshot (wr). The results revealed that equal weight function (EQL) is significantly inferior to other three normalized decay weight functions, LIN; EXP and WAVE. The LIN; EXP and WAVE, have the same behavior and stable accuracy. Besides, the observation eyeshot is moderately negative related to Avg-NMI.

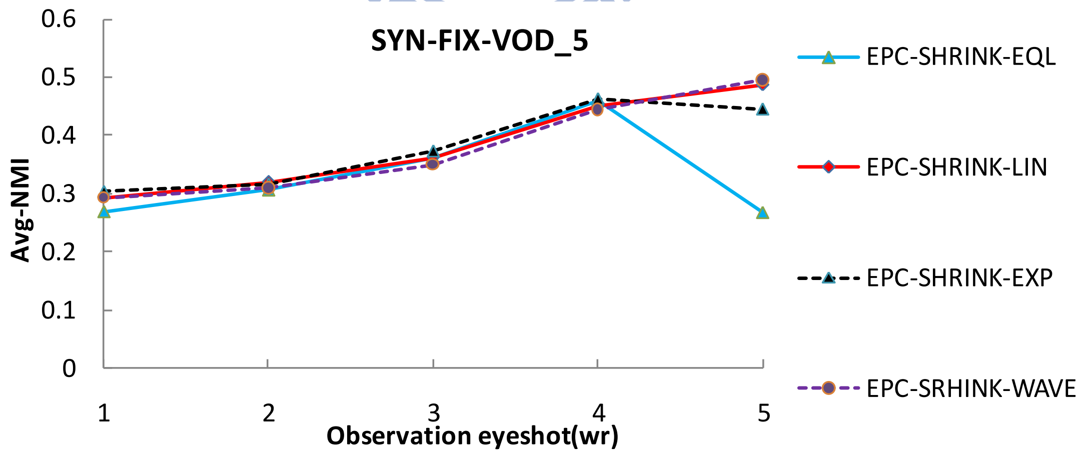


Figure 7-8 Parameters of EPC-SHRINK: observation eyeshot, decay weight functions on dataset SYN-FIX-VOD_5

Fig 7-8 shows the experimental result of EPC-SHRINK on dataset SYN-FIX-VOD_5. The vertical axis and horizontal axis are the same as Fig 7-7. On the opposite, the observation eyeshot and Avg-NMI have been shown to be positively correlated with each other. The experiments summarized indicate no strong relationship between Avg-NMI and observation eyeshot. The results of other datasets show the same property as Fig 7-7 and 7-8 so we do not display the experimental result of other datasets here.

In summary, the presetting of observation eyeshot has dependency on specific dataset. On selection of decay weight functions, LIN, EXP and WAVE have roughly the same Avg-NMI on all datasets generated by us.

7.2.4 Parameters of EPC-GSCAN: observation eyeshot, decay function and μ

EPC-GSCAN uses the clustering algorithm GSCAN [5]. We try to clarify the correlation between Avg-NMI and observation eyeshot (wr). Considering that the experimental results of most datasets are approximately the same. We use datasets, SYN-VAR-COE_0_3_RAN; SYN-VAR-COE_0_5_RAN, and linear decay weight function in the experiment.

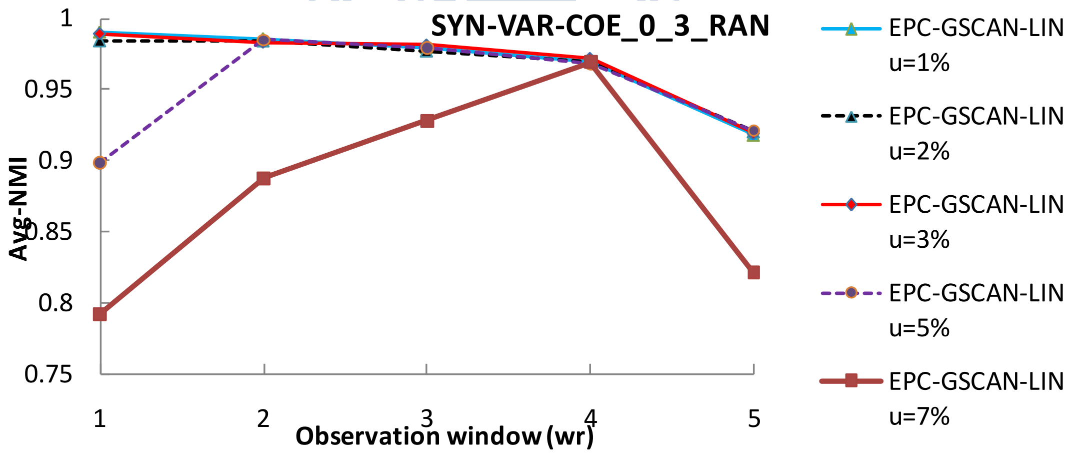


Figure 7-9 Parameters of EPC-GSCAN: observation eyeshot, μ on dataset SYN-VAR-COE_0_3_RAN

Fig 7-9 shows the experimental result of EPC-GSCAN-LIN on dataset SYN-FIX-VOD_3. The vertical axis is the average NMI and the horizontal axis is the different values of observation eyeshot (wr). The 5 curves in Fig 7-9 use different parameters μ . The result appear that using small μ and small observation eyeshot (wr) would get higher value of Avg-NMI.

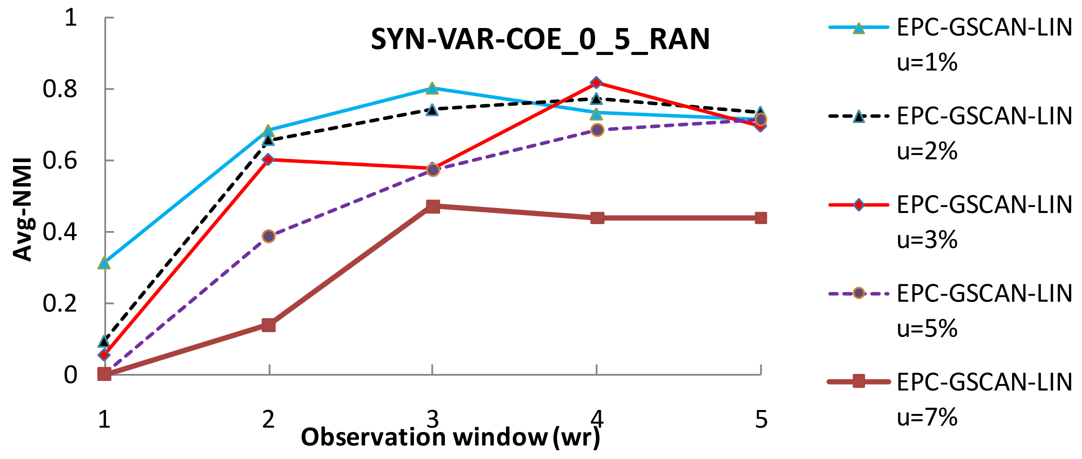


Figure 7-10 Parameters of EPC-GSCAN: observation eyeshot, μ on dataset

SYN-VAR-COE_0_5_RAN

Fig 7-10 shows the experiment of EPC-GSCAN-LIN on dataset SYN-VAR-COE_0_5_RAN. The vertical axis and the horizontal axis indicates the same information as Fig 7-9. The result appear to reject the assumption that using small μ and small observation eyeshot (wr) would get better Avg-NMI. In terms of the relationships between observation eyeshot (wr) and Avg-NMI, the results depict no correlation and the parameter presetting is data dependent.

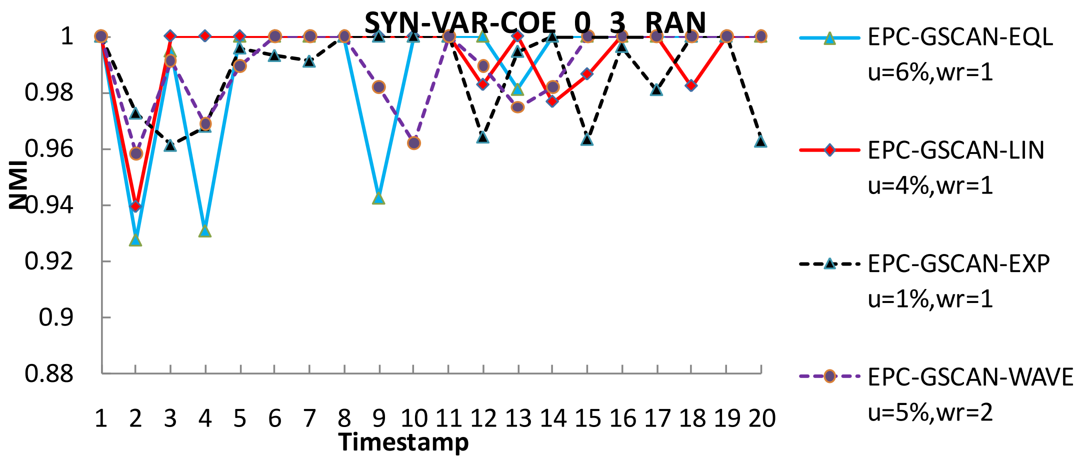


Figure 7-11 Parameters of EPC-GSCAN: decay weight functions on dataset

SYN-VAR-COE_0_3_RAN

We compare with different decay functions on dataset SYN-VAR-COE_0_3_RAN in Fig

7-11. The vertical axis is the NMI score and the horizontal axis is timestamp. Each decay function uses the optimal parameters, μ and observation eyeshot wr. The presetting parameter is shown on the right of Fig 7-11. The result shows that the equal weight function is worse than other decay weighted functions.

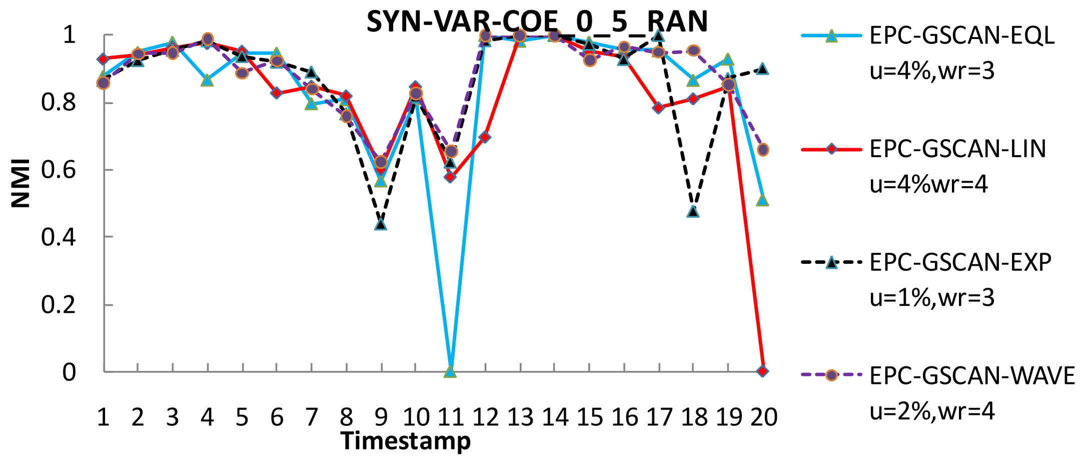


Figure 7-12 Parameters of EPC-GSCAN: decay weight functions on dataset SYN-VAR-COE_0_5_RAN

In Fig 7-12, the result shows the same property as Fig 7-11. While the noise level is increasing, using EXP or WAVE function has better Avg-NMI than using EQL or LIN.

In summary, the presetting of observation eyeshot and the parameter μ depend on specified dataset. On selection of decay weight functions, the EXP and WAVE decay weight function have better Avg-NMI than other functions.

7.3 Accuracy Comparison

EPC has two algorithm versions, EPC-SHRINK and EPC-GSCAN. We compare EPC with two algorithms, FacetNet [4] and PD-Greedy [5]. These algorithms need to preset their respective parameters as discussed above. In the figures of this section, "a" is the parameter of temporal smoothness, μ is the constraint of community size and wr is the observation eyeshot. We use the parameters of highest Avg-NMI for each algorithm.

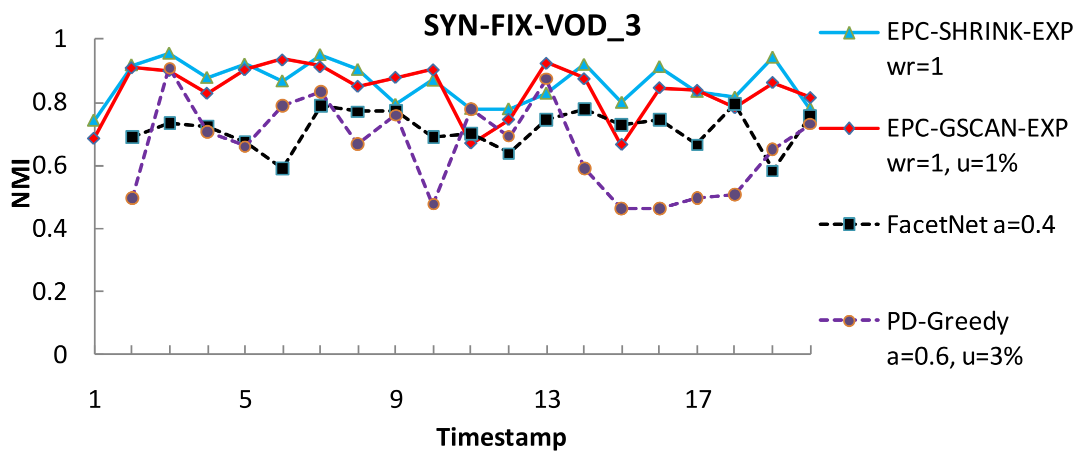


Figure 7-13 Accuracy comparison on dataset SYN-FIX-VOD_3

Fig 7-13 shows the accuracy comparison for SYN-FIX datasets. The vertical axis indicates the NMI score and the horizontal axis indicates timestamp. The solid curves of EPC-SHINK and EPC-GSCAN show higher accuracy than the dotted curves of FacetNet and PD-Greedy.

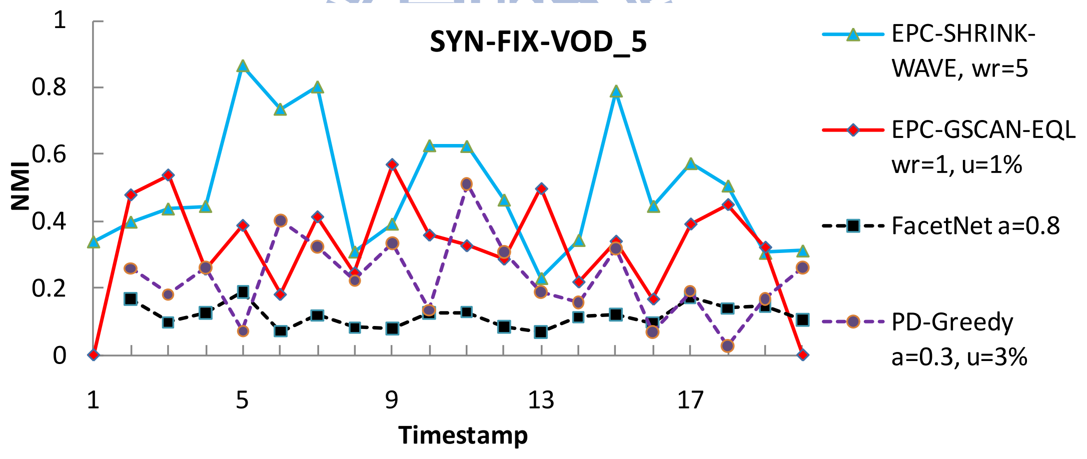


Figure 7-14 Accuracy comparison on dataset SYN-FIX-VOD_5

In Fig 7-14, the vertical axis and horizontal axis are the same as Fig 7-13. The solid curves of EPC-SHINK and EPC-GSCAN show the higher accuracy than both the FacetNet and PD-Greedy even though the noise level is higher than Fig 7-13.

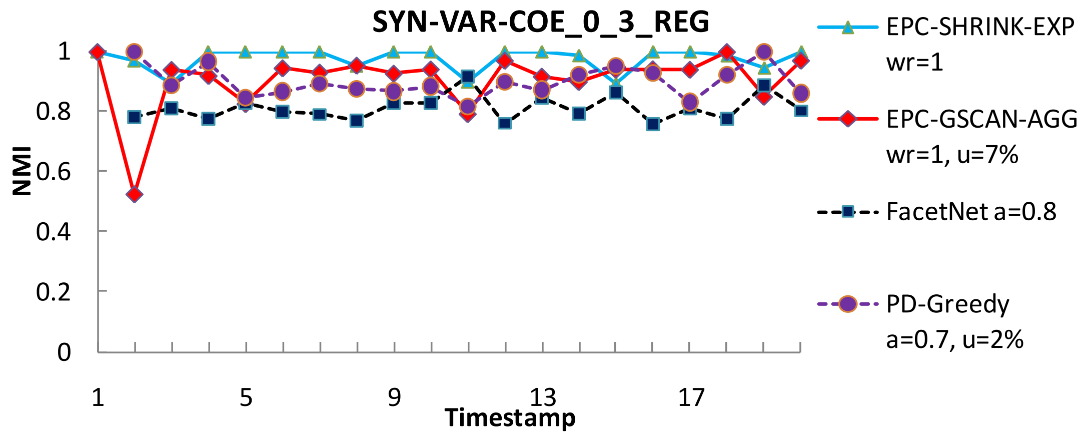


Figure 7-15(a) Accuracy comparison on dataset SYN-VAR-COE_0_3_REG

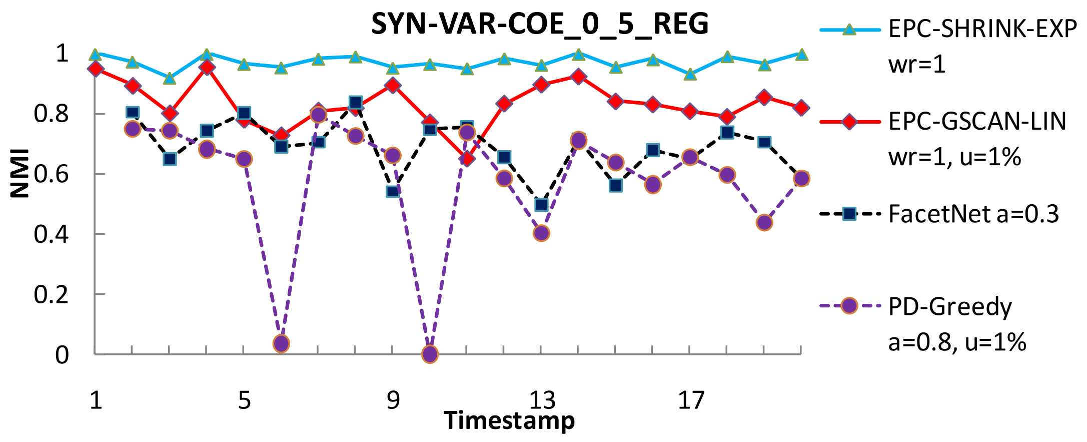


Figure 7-15(b) Accuracy comparison on dataset SYN-VAR-COE_0_5_REG

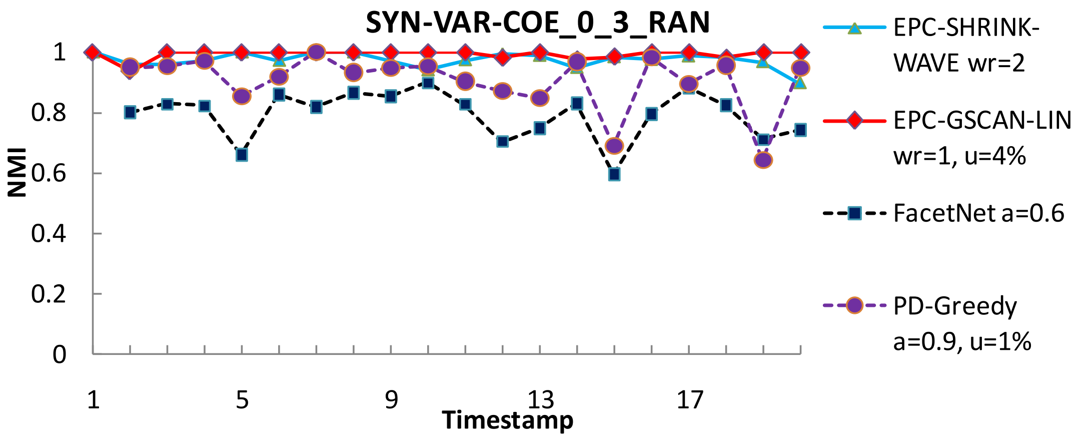


Figure 7-15(c) Accuracy comparison on dataset SYN-VAR-COE_0_3_RAN

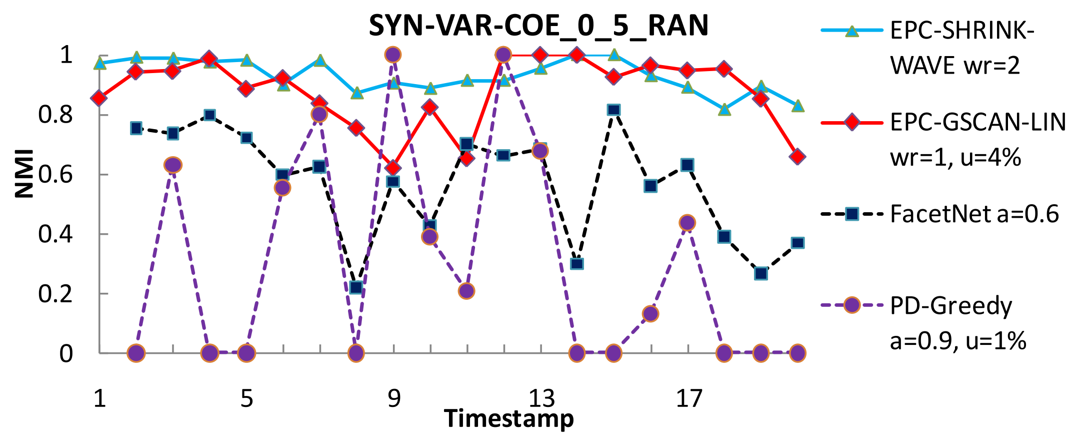


Figure 7-15(d) Accuracy comparison on dataset SYN-VAR-COE_0_5_RAN

Fig 7-15 shows the accuracy for SYN-VAR datasets. Fig 7-15 shows that the accuracies of EPC-SHRINK and EPC-GSCAN are more stable and better than FacetNet and PD-Greedy whatever the noise level is high or low in the datasets SYN-VAR.

In general, the accuracy of EPC-SHRINK is better than EPC-GSCAN and it appears that the clustering quality of SHRINK is better than the clustering quality of GSCAN. The accuracy of EPC-GSCAN is better than PD-Greedy. The experimental results validate that the concept of Relationship graph adopted outperforms the concept of temporal smoothness in dynamic networks over time.

7.4 Smoothness Comparison

For the smoothness property of community partitioning, we further assume that the normalized mutual information (NMI) of every two consecutive timestamps is higher and the smoothness quality is better. We compare the smoothness among EPC-SHRINK, EPC-GSCAN, FacetNet and PD-Greedy using the same parameters in section 7.3.

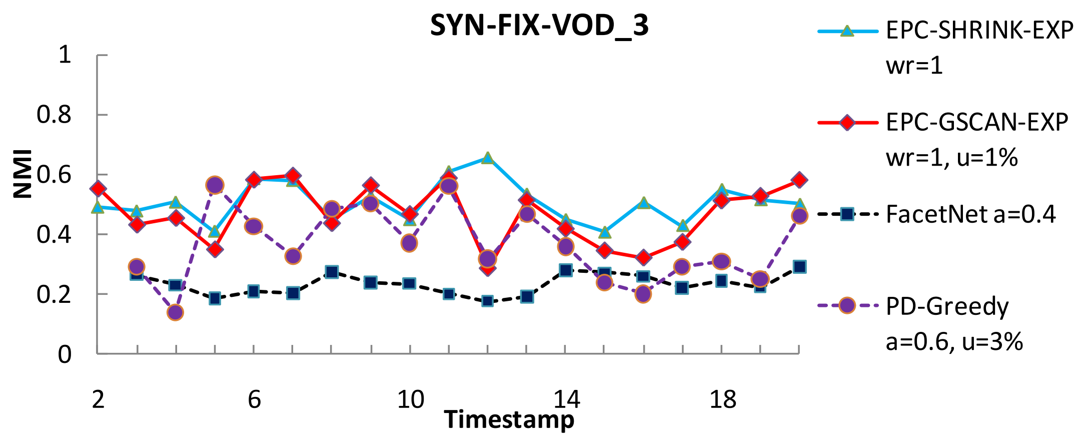


Figure 7-16 Smoothness comparison on dataset SYN-FIX-VOD_3

Fig 7-16 shows the smoothness quality of EPC-SHRINK, EPC-GSCAN, FacetNet and PD-Greedy. The vertical axis is the NMI score and the horizontal axis is timestamp. The solid curves of EPC-SHRINK and EPC-GSCAN have higher quality of smoothness than the dotted curves of FacetNet and PD-Greedy.

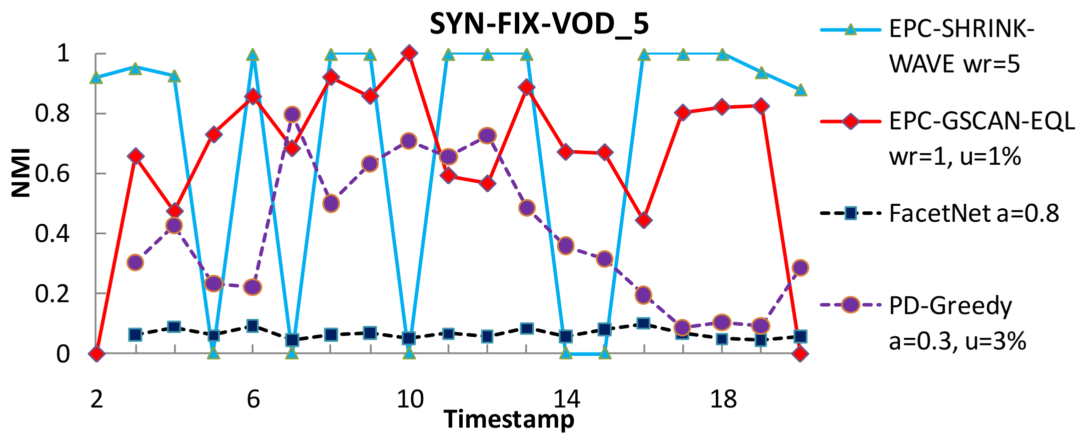


Figure 7-17 Smoothness comparison on dataset SYN-FIX-VOD_5

Fig 7-17 shows the smoothness quality on dataset SYN-FIX-VOD_5. The change of the curve of SHRINK is very violent due to higher noise level of dataset.

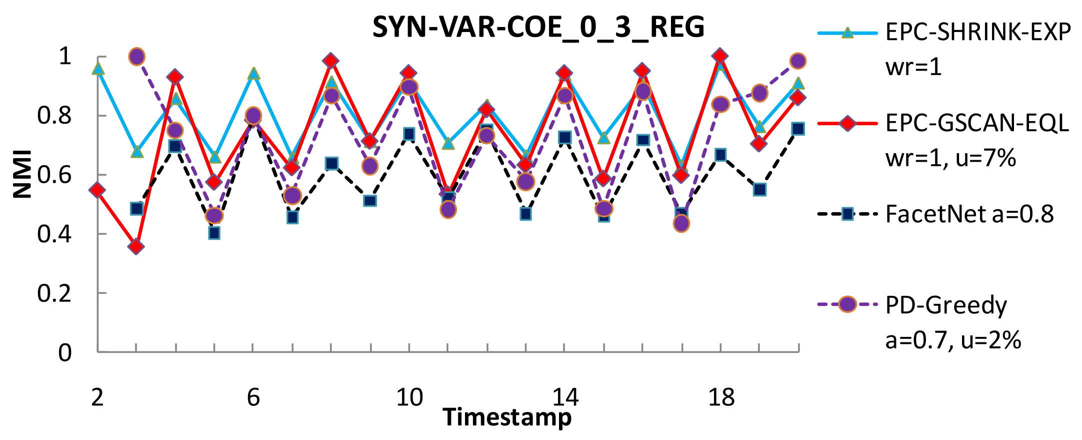


Figure 7-18(a) Smoothness comparison on dataset SYN-FIX-COE_0_3_REG

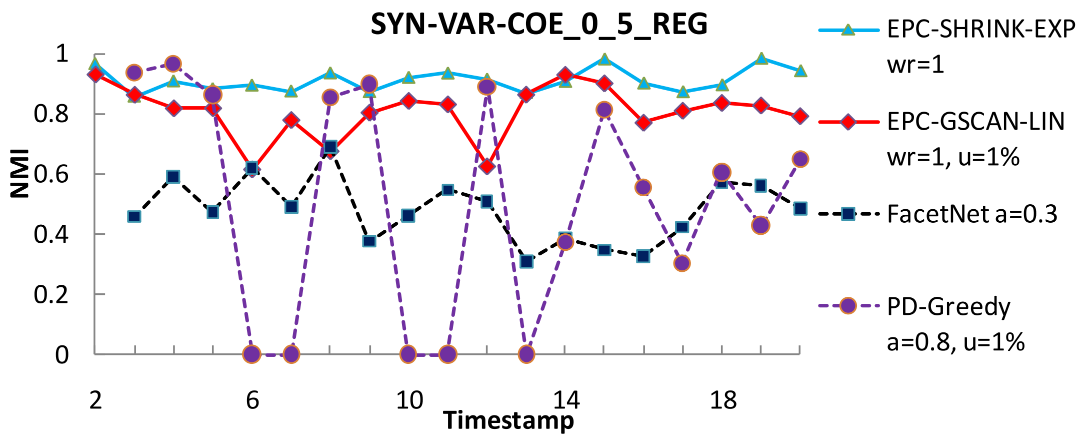


Figure 7-18(b) Smoothness comparison on dataset SYN-FIX-COE_0_5_REG

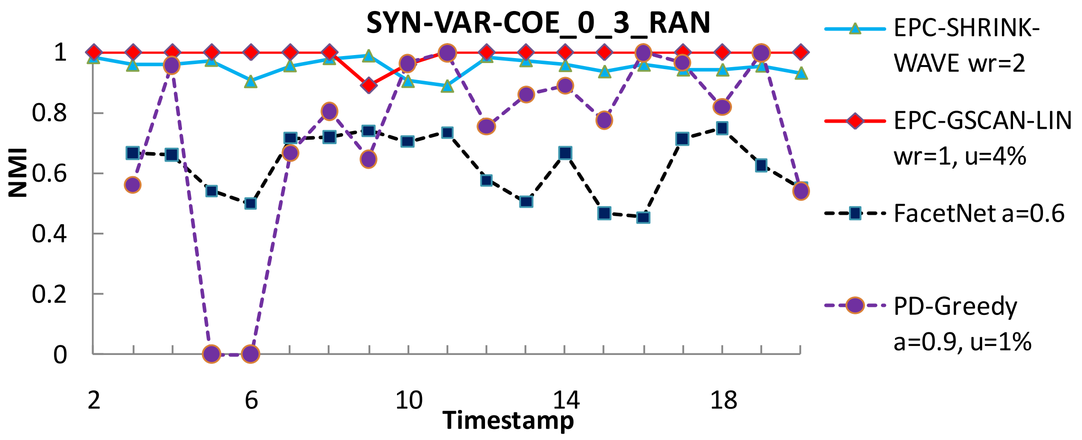


Figure 7-18(c) Smoothness comparison on dataset SYN-FIX-COE_0_3_RAN

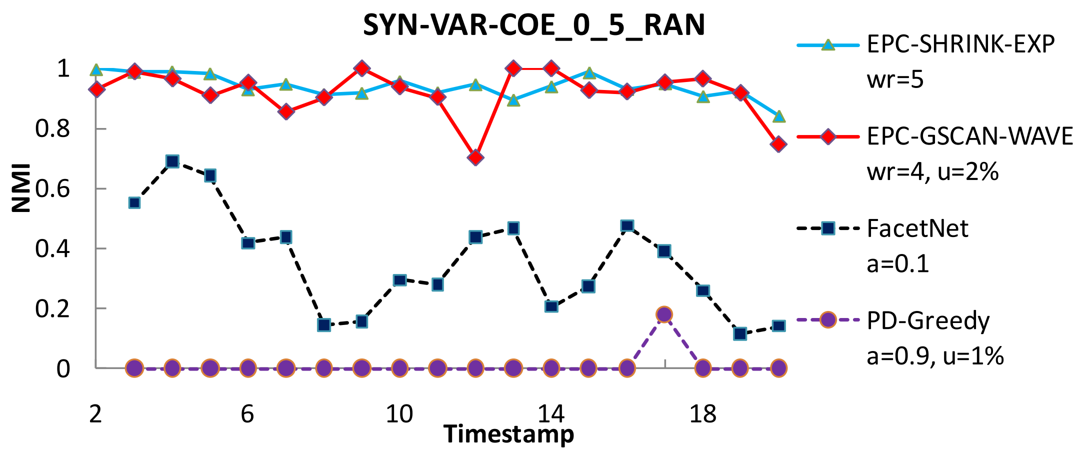


Figure 7-18(d) Smoothness comparison on dataset SYN-FIX-COE_0_5_RAN

Fig 7-18 shows the accuracy for SYN-VAR datasets. Fig 7-18 shows that the smoothness quality of EPC-SHRINK and EPC-GSCAN are more stable and better than both FacetNet and PD-Greedy whether the noise level is high or low in the datasets SYN-VAR.

In summary, the smoothness quality of EPC-SHRINK is better than EPC-GSCAN. The smoothness quality of EPC-GSCAN is better than PD-Greedy.

7.5 Scalability

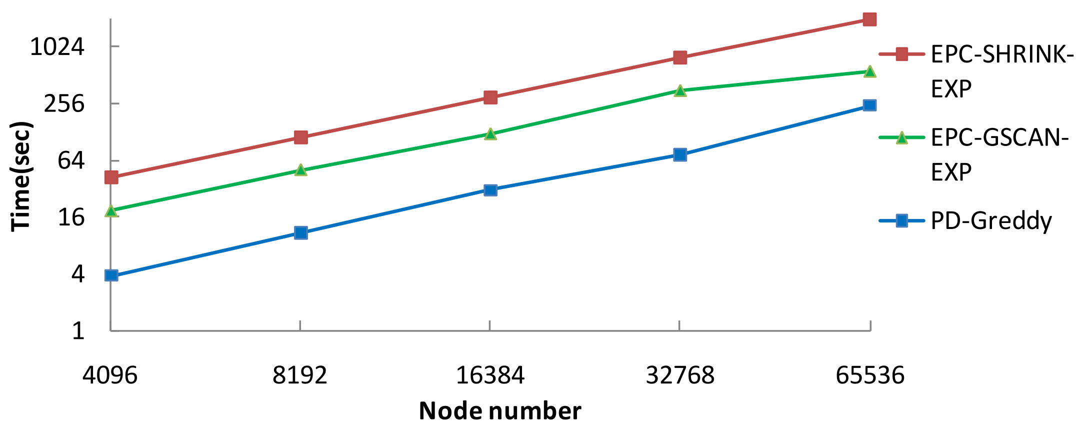


Figure 7-19: Execution time of EPC-SHRINK, EPC-GSCAN and PD-Greedy w.r.t different input size

Fig 7-19 shows the execution time EPC-SHRINK, EPC-GSCAN and PD-Greedy w.r.t different amount of nodes. The horizontal axis indicates the node number of different datasets and the node number is exponential growth in 2. The vertical axis indicates the execution time in log2 scale. Although PD-Greedy is faster than EPC-SHRINK and EPC-GSCAN, the result of PD-Greedy may be very bad and unstable. On the other hand, EPC-GSCAN has almost the same speed as PD-Greedy and good accuracy while using huge datasets.

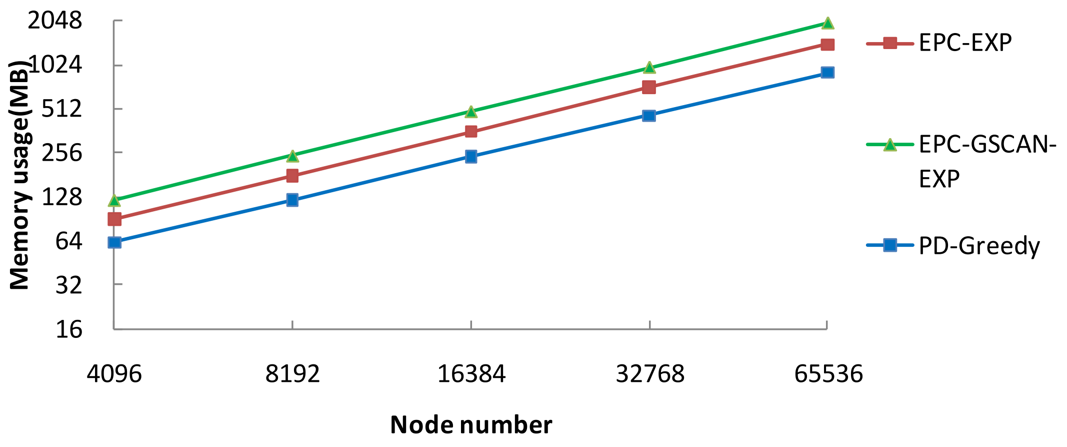


Fig 7-20: Memory usages of EPC-SHRINK, EPC-GSCAN and PD-Greedy

In Fig 7-20, the horizontal axis is the same to Fig 7-20. The vertical axis indicates the memory usage for different datasets. While the datasets is increasing, the result shows EPC-SHRINK, EPC-GSCAN and PD-Greedy are linearly scalable with the size of datasets.

7.6 Experiment on Real world dataset

We use three real datasets to evaluate our algorithm EPC: (1) Enron email [16] (2) Facebook [17] (3) DBLP data [14]. We only show the experiments of FacetNet at Enron email dataset since FacetNet only performs at small datasets experiment. In this section, we compare with PD-Greedy with all the real data experiments.

7.6.1 Enron email dataset

For the Enron email dataset, it is the email record between employees of Enron Corporation. Email address represents an individual and the edge represents email exchange. Enron email dataset contains 150 users and 24,140 messages in the company. We use a cleaner version of the dataset from March 2000 to April 2002. We split Enron dataset with the time slice unit = 5 days because an appropriate time unit is suggested for analyzing dynamic networks [10]. We arrange this data from 150 time slices, observation eyeshot wr = 2 and preset the *Minimum community similarity threshold* (ρ)=0.3.

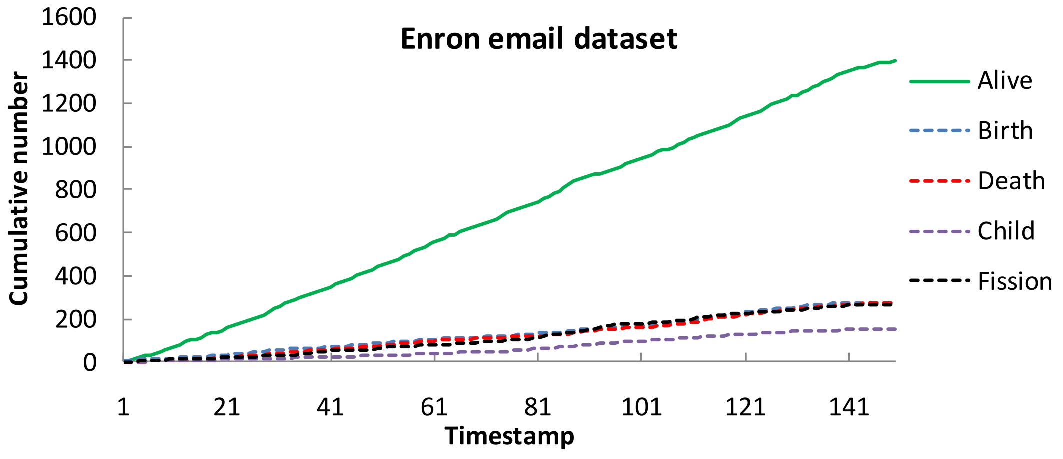


Figure 7-21 Result of Enron email dataset using EPC-SHRINK

In Fig 7-23, the horizontal axis is the time slice of Enron email dataset. The vertical axis is the cumulative number of states of community. The curve of *Alive* state is higher than other states of community and the curves of other states are the same. There is no doubt about the result because the variation of company network is relatively low than general human relationship.

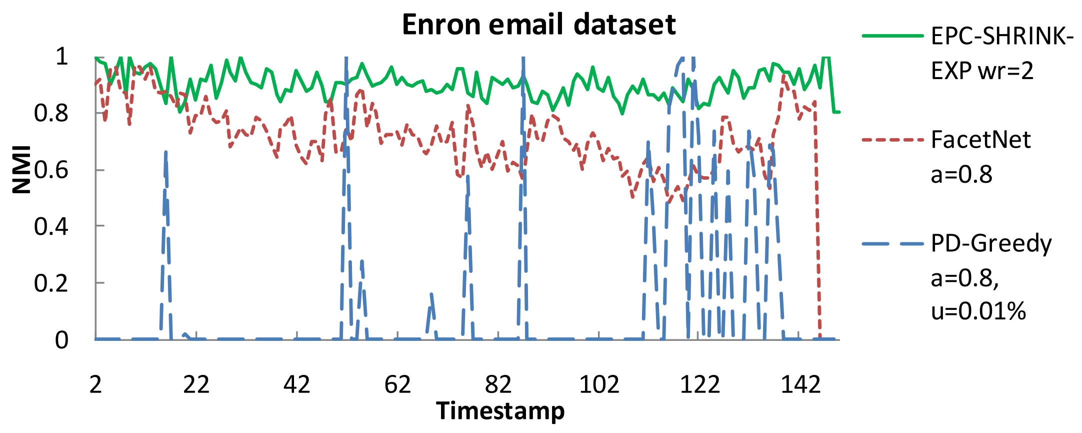


Figure 7-22 Smoothness quality result on Enron email dataset

In Fig 7-22, the horizontal axis is the time slice of Enron email dataset and the vertical axis is the cumulative number of states of community. For the smoothness quality of the EPC in Enron dataset, the curve of EPC-SHRINK shown in Fig 7-22 demonstrates the high smoothness quality. And the curves of FacetNet and PD-Greedy show that the smoothness quality is worse than EPC-SHRINK in Enron email dataset.

7.6.2 Facebook dataset

For the Facebook, there are two records, friend list and wall posts. We use the wall posts as our interaction data. Facebook dataset contains 46,952 users and 876,255 wall posts in the dataset. We also use a cleaner version of the dataset from January 2006 to January 2009. We aggregate these data using the same time slice unit (5) as Enron email dataset to form 220 time points, observation eyeshot wr = 2 and the same *Minimum community similarity threshold* (ρ)=0.3.

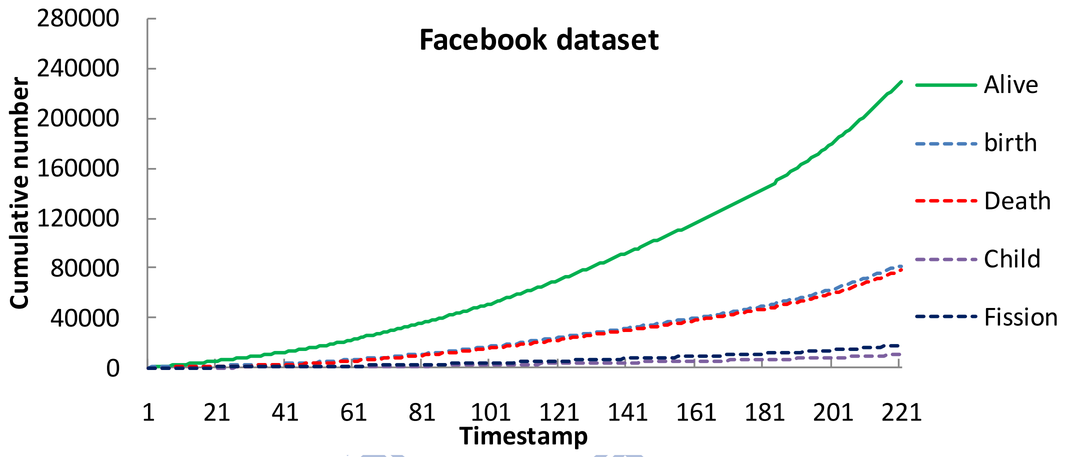


Figure 7-23 Result of Facebook dataset using EPC-SHRINK

Fig 7-23 shows the result of Facebook dataset. The respective unit scales of horizontal and vertical axis are the same as Fig 7-22. The number of Alive state is always higher shows that most communities are alive and stable in real world. The cumulative numbers of Birth states and Dead state is higher than that of Fig 7-23 demonstrates that the real dynamic networks could change quickly. Child and Fission shows that the friend relationships are cultivated quickly at recent time. The cumulative number of merge and split operations indicate that these operations are not common in dynamic social networks.

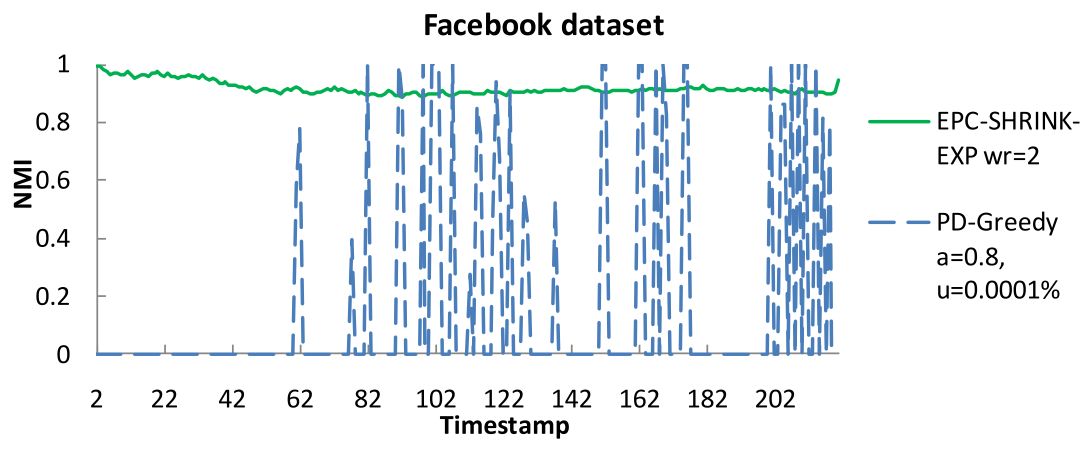


Figure 7-24 Smoothness quality result on Facebook email dataset

For the smoothness quality in Facebook dataset, the curve of EPC-SHRINK shown in Fig 7-24 expresses that the smoothness quality is still high.

7.6.3 DBLP dataset

The DBLP dataset presents the interaction information of co-authorship. We take out the related dataset from 2000 to 2009 which contains 769,137 authors and 1,068,239 records. A node represents an author and an edge between nodes represents the co-authorship between authors. We use a year as a time unit due to the characteristics of interaction data. We preset the observation eyeshot wr = 1 and the same *Minimum community similarity threshold* (ρ)=0.3.

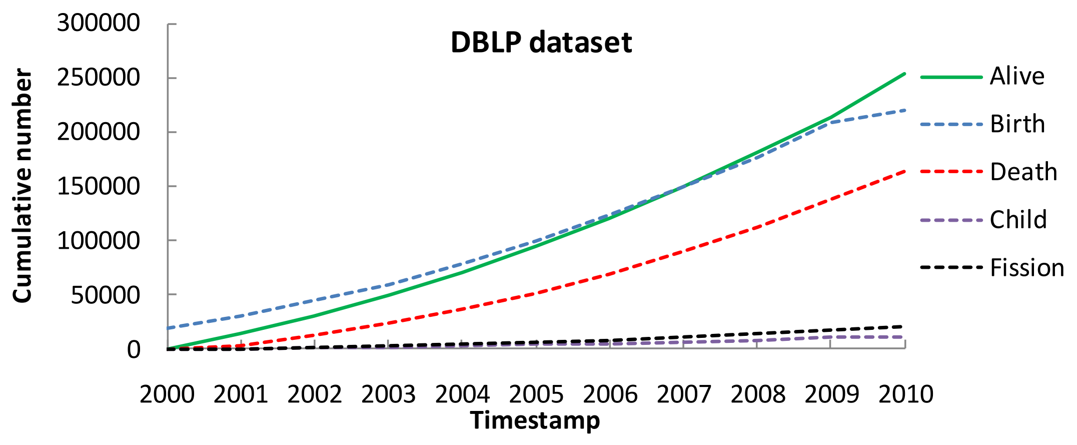


Figure 7-25 Result of DBLP dataset using EPC-SHRINK

Fig 7-25 shows the result of DBLP dataset. The cumulative number of Alive being higher than other states depicts that most communities are alive over time. The cumulative numbers of Child and Fission are still lower than that of other states depicts the same phenomenon as the Enron dataset and Facebook.

However, the aggregate number of Birth increases fast shows an obvious fact. For example: a researcher could be a Master or a Ph. D. who co-worked with the professor and later he graduated from school. Nevertheless, he chooses another job rather than a researcher so the phenomenon displays the growth rates of Birth and Death states are higher than Enron email and Facebook dataset.

Especially in Facebook and in DBLP datasets, the cumulative number of Birth and Death are higher than in Enron dataset. This phenomenon shows the networks of FacetNet and Co-authorship change fast. The highly cumulative number of Alive also shows the result that EPC is more smoothy.

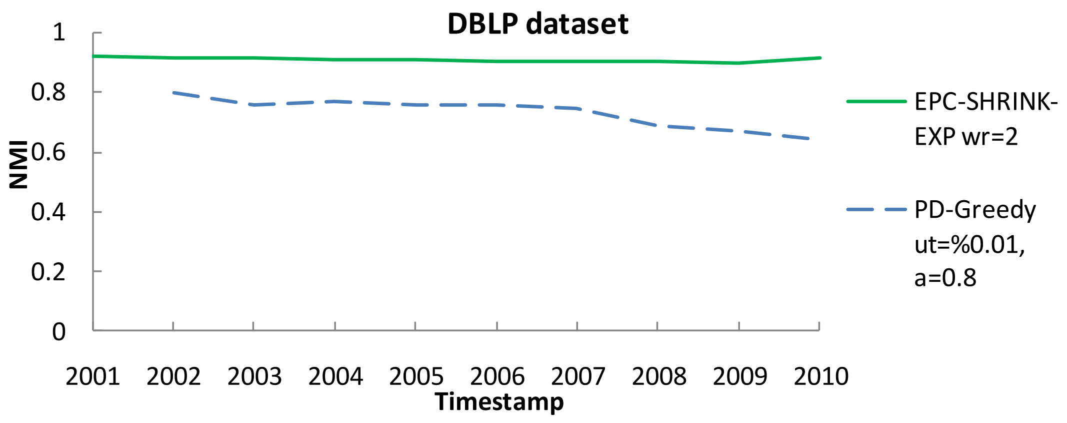


Figure 7-26 Smoothness property of EPC when running with DBLP dataset

For the smoothness quality in DBLP dataset, the curve of EPC expresses higher smoothness quality than PD-Greedy.

Chapter 8
Conclusion and Future work

8.1 Conclusion

Although a large number of studies have been made on community detection in networks, little is known about the property and feature of dynamic community. We propose the algorithm EPC which provides a novel approach of data smoothness to explore the evolution of community. The proposed *Relationship extraction strategy* not only considers the historical data but also the oncoming data. We also propose a mapping method of community partition over time called *Community Pedigree Mapping* which shows the state of community and displays the life circle of community.

In synthetic data experiment, our algorithm EPC provides a scalable way to solve the problem of mining dynamic community. There are two versions of the EPC-algorithm, EPC-SHRINK and EPC-GSCAN. The experiment demonstrates that EPC-SHRINK and EPC-GSCAN have higher accuracy than previous algorithms such as FacetNet[3] and PD-Greedy [5]. For the smoothness quality of community partitioning, the experiment shows the community partition of EPC is more smoothing than FacetNet and PD-Greedy. For the scalability of EPC, the experiment shows EPC-GSCAN is linearly scalable and almost as fast as PD-Greedy. In general, the accuracy and smoothness quality of EPC-SHRINK is better than EPC-GSCAN. It appears that the clustering quality of SHRINK is better than the clustering quality of GSCAN. The accuracy of EPC-GSCAN is better than PD-Greedy, which validates that the concept of Relationship graph over time outperforms the concept of temporal smoothness which using the same clustering algorithm.

We also apply EPC on real datasets to Enron email, Facebook and DBLP dataset. In all datasets, the cumulative number of Alive state being higher than other states indicates that the variation of real dynamic communities is quite low in social networks. The growth rates of Birth and Death in DBLP dataset and Facebook dataset are higher than Enron email dataset, indicating that new communities appear more quickly in common relationship than in company. This phenomenon supports the argument that the real world such as FacetNet and Co-authorship change fast. The high cumulative number of Alive also verifies the result of EPC provides the property of smoothness over time.

8.2 Future work

Although EPC has acceptable time complexity and higher accuracy, there still are some related works worth further investigation. (1) So far there are no convinced theories about how to measure the quality of dynamic community in dynamic social network even we have discovered the community. (2) The proposed EPC presets the observation eyeshot wr and assumes each relationship between individuals should use the same decay weighting function. In real world the decays of relationship between individuals might be different. Moreover, the observation eyeshot could be very dependent on different individuals at different timestamps. (3) In real world, there are many kinds of interaction between individuals. How to correctly determine and present the relationship strength converting various kinds of interactions is still a challenging problem.

Bibliography

[1] D. Chakrabarti, R. Kumar, and A. Tomkins. Evolutionary clustering. Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp.554-560, 2006

[2] Y. Chi, X. Song, D. Zhou, K. Hino, and B. L. Tseng. Evolutionary spectral clustering by incorporating temporal smoothness. Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 153-162, 2007.

[3] 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. Proceedings of the 17th International Conference on World Wide Web, pp. 685-694, 2008.

[4] L. Tang, H. Liu, J. Zhang, and Z. Nazeri. Community evolution in dynamic multi-mode networks. Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 677-685, 2008.

[5] M. S. Kim and J. Han. A Particle-and-Density Based Evolutionary Clustering Method for Dynamic Networks. Proceedings of the 35th International Conference on Very Large Data Bases, pp. 622-633, 2009.

[6] G. Palla, A.-La´szlo´ Baraba´si and T. Vicsek. Quantifying social group evolution. Nature 2007, vol 446, pp.664-667, 2007.

[7] J. Sun, C. Faloutsos, S. Papadimitriou and P. S. Yu: GraphScope: parameter-free mining of large time-evolving graphs. Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp.687-696, 2007.

[8] C. Tantipathananandh, T. Y. Berger-Wolf and D. Kempe. A framework for community identification in dynamic social networks. Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp.717-726, 2007.

[9] S. Wasserman and K. Faust. Social Network Analysis: Method sand Applications. Cambridge University Press, 1994.

[10] R. Sulo, T. Berger-Wolf and R. Grossman. Meaningful selection of temporal resolution for dynamic networks. Proceedings of the 8th Workshop on Mining and Learning with Graphs, pp.127-136, 2010.

[11] http://en.wikipedia.org/wiki/Exponential_decay#Mean_lifetime

[12] J. Huang, H. Sun, J. Han, H. Deng, Y. Sun and Y. Liu: SHRINK: a structural clustering algorithm for detecting hierarchical communities in networks. Proceedings of the 19th ACM Conference on Information and Knowledge Management, pp.219-228 , 2010.

[13] S. Asur, S. Parthasarathy and D. Ucar, An Event-based Framework for Characterizing the Evolutionary Behavior of Interaction Graphs. ACM Transactions on Knowledge Discovery from Data, vol 3, no 4, Article 16, 2009.

[14] http://dblp.uni-trier.de/xml/

[15] M.E.J. Newman and M. Girvan, Finding and evaluating community structure in networks, Physical Review, 2004, E 69, 026113, 2004.

[16] http://www.cs.cmu.edu/~enron/

[17] http://socialnetworks.mpi-sws.org/

[18] A. Clauset, M. E. J. Newman, and C. Moore. Finding community structure in very large networks. Physical Review, E 70, 066111, 2004.

[19] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment, vol 2008,issue 10, 2008.

[20] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, pp. 226~231, 1996.

[21] X. Xu, N. Yuruk, Z. Feng, and T. Schweiger. SCAN: a structural clustering algorithm for networks. Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 824-833, 2007.

[22] J.-G. Lee, J. Han and K.-Y. Whang. Trajectory clustering: A partition-and-group framework. Proceedings of the 2007 ACM SIGMOD international conference on Management of data, pp. 593-604, 2007.

[23] Q. Zhao, S. S. Bhowmick, X. Zheng and K. Yi. Characterizing and Predicting Community Members from Evolutionary and Heterogeneous Networks. Proceedings of the 17th ACM Conference on Information and Knowledge Management, pp. 309-318, 2008.

[24] J. Shi and J. Malik. Normalized cuts and image segmentation. Conference on Computer Vision and Pattern Recognition, pp.731-737, 1997.

[25] Z. Feng, X. Xu, N. Yuruk, and T. A. J. Schweiger. A novel similarity-based modularity function for graph partitioning. Data Warehousing and Knowledge Discovery, 9th International Conference, pp. 385–396, 2007.

[26] http://www.public.asu.edu/~ylin56/research.html

[27] A. Lancichinetti and S. Fortunato. Community detection algorithms: A comparative analysis. Physical Review, E 80, 056117, 2009.

[28] C. Tantipathananandh, T. Y. Berger-Wolf. Constant-factor approximation algorithms for identifying dynamic communities. Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp.827-836, 2009.

[29] S. Fortunato. Community detection in graphs. Physics Reporters, vol 486, issue 3-5, pp. 75-174, 2010.

[30]K. Yu, S. Yu and V. Tresp. Soft clustering on graphs. Proceedings of the nineteenth Annual Conference in Neural Information Processing Systems, 2005.

[31] http://en.wikipedia.org/wiki/APX