Community Detection in Dynamic Networks

Niloufar Afsariardchi

Department of Electrical & Computer Engineering
McGill University
Montreal, Canada

December 2012

A thesis submitted to McGill University in partial fulfillment of the requirements for the degree of Master of Engineering.

Abstract

A reasonable representation of some complex systems such as social and biological systems is a network topology that allows its components and interactions among them to change over time. Understanding the time-dependence of these networks can lead to invaluable insight about characteristics and structure of time-varying networks. In this thesis, several classes of static and dynamic clustering algorithms and ideas are reviewed. A challenge arising in dynamic clustering schemes is that the detected communities are not independent over time and the identified clusters at one point of time should not dramatically deviate from the results of previous timesteps. It is especially important to reduce large short term variations and ensure that communities smoothly change over time. Here we present a novel method which is built upon a probabilistic generative Bayesian model to address the problem of identifying consistent and stable overlapping communities in dynamic networks. Synthetic and real networks are used to evaluate the performance with respect to different parameter settings, the model order selection, and the run-time of the proposed algorithm. Performance analysis indicates that the algorithm proposed in this thesis outperforms several other state-of-the-art algorithms and provides valuable insights into the evolution and underlying structure.

Sommaire

Une représentation raisonnable de certains systèmes complexes tels que les systèmes sociaux et biologiques est une topologie de réseau qui permet à ses composants et les interactions entre eux de changer au fil du temps. Comprendre la dépendance temporelle de ces réseaux, conduire à de précieux renseignements sur les caractéristiques et la structure de variables dans le temps des réseaux. Dans cette thèse, plusieurs classes d'algorithmes de clustering statiques et dynamiques et des idées sont passées en revue. Un défi se pose dans des plans de regroupement dynamiques est que les communautés détectées ne sont pas indépendants dans le temps et les grappes fondées à un moment donné du temps ne doit pas s'écarter de façon spectaculaire à partir des résultats de pas de temps précédents. Spécialement, il est de l'importance de diminuer de fortes variations à court terme et d'assurer que les communautés progressivement changer au fil du temps. Ici, nous présentons une nouvelle méthode qui repose sur un modèle bayésien génératif probabiliste pour résoudre le problème de l'identification des communautés stables et cohérentes qui se chevauchent dans les réseaux dynamiques. Réseaux synthétiques et réelles sont utilisées pour évaluer la performance par rapport à différents paramètres, la sélection pour modèle, et le moment de l'exécution de l'algorithme proposé. Analyse de la performance indique que l'algorithme proposé dans cette thèse surpasse plusieurs autres algorithmes et révèle l'aperçu inestimable d'un réseau e-mail réelle.

Acknowledgments

First and foremost, I would like to thank my supervisor, Professor Mark Coates. Without his invaluable guidance, discussions, and feedbacks this thesis would not be what it is today.

I am sincerely thankful of my beloved parents Mojgan Moradi Haghgoo and Nasser Afsari Erdchi for their indescribable efforts and encouragement throughout my life especially years of studying abroad. Without your infinite love and faith in my abilities, I would never be who I am right now.

I am blessed of experiencing the ultimate love and happiness with Mahdi Mirhoseini whose presence in my life makes my all distances bearable.

I am grateful of my kind and wonderful sister Erfaneh Afsari Ardchi for continuously supporting me from childhood up to now.

My special thanks to my grand parents, Ghodrat Alizadeh Rastan and late Houshang Moradi Haghgoo for all I have learned from them. Thank you for motivating me during my master studies. I also would like to thank my second mother Mari (Fatemeh Parkook). It was you who taught me how to count, thank you for your pure kindness.

I would also like to thank Professor Michael Rabbat for his cooperative and friendly presence in the Computer Networks Lab through my masters' studies.

I sincerely thank my fellow colleagues of the Computer Networks Lab, Rizwan, Rodrigo, Babak, Deniz, Hakon, Tao, Oscar, Santosh, Bassel, Salim, Haani, Ali, and Zhe for their support and friendship. My special thanks to Milad for taking care of my initial submission process when I was away and also to Benjamin for translating the abstract to French.

I heartily thank Fereshteh Giahi Foomani for supporting me in two years of living in Montreal especially in the first days of coming here when I needed help a lot.

I would like to express my deepest gratitude to my great friends. Samira Soltani if

we had not met, I would have been still searching for a good friend in Montreal. I hope our wonderful friendship lasts independent of time and place. Zahra Jalili your passion for exploring the life, people, and nature has inspired me forever. I am very glad of having you among my friends. Najme Kishani, I appreciate your hard-working sprite and your insatiable hunger for learning. If it was not you working with me until midnight, I would have been still writing the first page of my thesis. Thank you. Mohammad Key Manesh thank you for listening to my complains patiently especially in my final months.

Last but not least, I thank the God for inspiring me throughout my life.

Contents

List of Figures

List of Tables

List of Acronyms

NMF	Non-negative Matrix Factorization
PCA	Principal Component Analysis
ARD	Automatic Relevance Determination
MAP	Maximum A-posteriori
MCL	Markov Clustering
MDL	Minimum Description Length
CMP	Clique Percolation Method
PCM	Preservation of Cluster Membership
PCQ	Preservation of Cluster Quality
NMI	Normalized Mutual Information
KL	Kubler-Leiber distance
MCMC	Markov Chain Monte-Carlo
DNMF	Dynamic Non-negative Matrix Factorization

Chapter 1

Introduction

1.1 Motivation

A plausible representation of many complex systems of scientific interest is a network topology that consists of a set of nodes or vertices joined in pairs by links or edges portraying a number of entities (e.g., individuals, websites, genes) interconnected by a set of relationships (e.g., communication, friendship, flows, collaboration). We face the concept of a *network* on a regular basis in the variety of contexts. A myriad of examples, including communication and transportation systems, social relations [8, 9], ecological dependencies [10], biological networks [9, 11], and food web [12] have been observed and modelled as networks. Due to the growing availability of enormous volumes of accurate information about the topologies of real-life networks, especially social networks, the Internet, and the world-wide web, the development of powerful network analysis tools and algorithms continues to be a topic of much interest.

How can we infer the structure of a network? There is no universally-accepted answer to this network data analysis question. The general scheme of inference, however, strives

to *cluster* elements of network to obtain densely connected groups of vertices, with only sparser connections between the groups. This approach simplifies the data by providing a microscopic view of the network [13], eliciting structure that cannot be easily observed when viewing the original network topology. Since the goal of clustering is not precisely defined, there is much room left for choosing various mathematical criteria to optimize, and the choice in general depends on the intention behind the inference and the characteristics of network. Detecting outliers and anomalies [14], network visualization, dimensionality reduction and coding, and data network simplification are possible aims of clustering[1] that can be studied in various networks.

In many applications of clustering, networks have a dynamic nature as they are actually observed at different timesteps. In such networks, the appearance and disappearance of nodes, edges, and communities are possible. For instance, in real social networks (e.g., Facebook, Twitter, Linkedin) individuals may join new communities due to the change of their interests or an external event [13]. In biological networks such as protein-protein interaction networks [15], time-dependent multi-molecular complexes can be represented naturally as dynamic networks whose vertices are biomolecules (e.g., DNA/genes, RNA/transcripts, proteins) and whose edges represent physical interactions [16]. Understanding the time-dependence of these networks can lead to invaluable insight about the characteristics and structure of time-varying networks.

Furthermore, in real life it is possible to have overlapping communities. In biological networks, for example, several nodes can be member of more than one group. In metabolic networks, metabolites can work in association with several metabolic processes [17]. In social networks, individuals can be member of more than one group; these groups might

[1] In the literature, clusters are often called modules and communities, and similarly the clustering task is sometimes referred to *community detection*. In this work we utilize these terms interchangeably.

represent the individual's family, his/her co-workers, acquaintances, friends, etc. [18]. The groups are more appropriately modelled as non-disjoint, since they frequently have more that one common member. In this thesis I propose an effective algorithm to identify, and track over time, the overlapping clusters of dynamic networks.

1.2 Thesis Problem Statement

A challenge raised in clustering of dynamic networks that makes evolving networks clustering more difficult and different than analysis of a series of unrelated snapshots, is *continuity* of dataset at different timesteps.

We consider a dataset that consists of snapshots of networks. Each snapshot specifies the nodes and noisy observations of the weights, or similarities, associated with links between pairs of nodes. This dataset represents a dynamic network that evolves over time. If each snapshot were unrelated to the next, then static clustering algorithms could be applied at each timestep. Most dynamic networks, however, have a temporal structure, and there is *continuity* of the dataset between timesteps. In other words, although the network is changing over time, dramatic short term variations are less probable and the structure of the network, captured in two adjacent snapshots, tends to be correlated. This property is also called *temporal smoothness* in the literature [19].

Having the notion of temporal smoothness in mind, a dynamic network clustering method should strive to incorporate the clustering results from previous timesteps when attempting to infer the structure based on the observations at the current timestep. It must also address the possibility that the number of communities can change over time. It should target a balance between imposing temporal smoothness while allowing the formation of new clusters and the elimination of old clusters if this is strongly indicated by the

data. The algorithm should be capable of estimating how many communities are present at each timestep. Finanlly, we require that the algorithm is capable of detecting overlapping communities and estimating how strongly each node is associated with each community. This makes our algorithm appropriate for application domains where the network cannot be meaningfully partitioned into disjoint clusters and where the strength of the association between that data element and a particular cluster is important information [20].

In summary, our goal is to specify an efficient clustering algorithm for analysis of dynamic networks that can: (i) detect and track overlapping communities; (ii) address situations where communities appear and disappear over time; (iii) estimate the strength of association between each node and each commmunity and (iv) estimate the number of communities.

1.3 Thesis Contribution and Organization

In addition to summarizing key static clustering schemes and providing a detailed literature review on dynamic clustering, we propose a novel dynamic clustering algorithm. Our method is based on a probabilistic generative model. Utilizing this model makes our algorithm capable of detecting not only hard clusters but also soft partitions in the case that nodes are deemed to belong more than one cluster. We also used a machine learning tool, called Automatic Relevance Determination, to overcome the challenge of selecting the number of clusters in the partitioning (model order selection). The organization of this thesis is summarized below.

In Chapter 2 various important static clustering algorithms are summarized. The discussion is followed by describing the early thread of clustering techniques for dynamic networks. We point out the drawbacks and limitations of this family of techniques. After-

wards we review several new algorithms which are considered as evolutionary techniques. We mention a few other fields where evolutionary clustering is being used. At the end of Chapter 2 we summarize the essential points of community detection in dynamic networks.

In Chapter 3 the proposed algorithm is presented. Our dynamic algorithm is based on the static algorithm of Psorakis et al. [21]. The model, parameter inference, model order selection, and the complexity of our method are thoroughly described.

The performance of our algorithm with respect to different parameter settings, model order selection, and run-time is studied in various synthetic and real datasets in Chapter 4, and a comparison of its performance to other state-of-the-art algorithms is presented.

The concluding remarks and future work are summarized in Chapter 5.

Chapter 2

Literature Review

In this chapter, we discuss previous significant advances in determining the topological structure and detecting communities of networks. Since most of the techniques that have been developed for the clustering of dynamic networks are originally based on the definitions and the primary works of static clustering methods, we briefly review the most important methods of community detection in static networks and then provide a detailed literature review on dynamic algorithms.

2.1 Static Clustering

2.1.1 Graph Notations

Generally speaking, a *graph* is the mathematical representation of a network. Often the term *network* refers to a real-world instance of a graph. Sometimes determining the corresponding graph of a network and finding the appropriate set of edges and nodes is not straightforward. In this thesis our analysis involves either pre-modelled networks or instances with obvious graph models.

Let $\mathcal{G} = (V, E, w)$ be a graph where $V = \{v_1, v_2, \cdots, v_N\}$ is a set of vertices, E is a set of edges, and w is the weight associated with each edge $e \in E$. The number of nodes is denoted by $N = |V|$ and the number of edges is denoted by $M = |E|$. The weight of an edge $\{u, v\} \in E$ is written as $w(u, v)$. Note that unweighted graphs, represented by $\mathcal{G} = (V, E)$, can be seen as a special weighted case where $\forall\{u, v\} \in E : w(u, v) = 1$. For unweighted graphs, the adjacency matrix $\mathbf{A} = \{A_{ij}\} \in \{0, 1\}^{N \times N}$, is defined as:

$$A_{ij} \quad = \quad \begin{cases} 1 & \text{if } \{v_i, v_j\} \in E, \\ 0 & \text{if } \{v_i, v_j\} \notin E. \end{cases} \tag{2.1}$$

We assume that self-loops are also allowed, i.e., A_{ii} can be either 0 or 1. Weighted graphs are often characterized by a similarity matrix $\mathbf{W}$. In general the weights can assume any value, however they are usually non-negative and sometimes defined as integers. Mathematically, the matrix $\mathbf{W} \in \mathbb{R}^{N \times N}$ consists of the elements:

$$w_{ij} \quad = \quad \begin{cases} w(v_i, v_j) & \text{if } \{v_i, v_j\} \in E, \\ 0 & \text{if } \{v_i, v_j\} \notin E. \end{cases} \tag{2.2}$$

If the similarity matrix $\mathbf{W}$ of a graph is symmetric, i.e., $w_{ij} = w_{ji}$, we call the graph *undirected* and otherwise it is a *directed* graph.

The degree of a node is the number of its incident edges:

$$d(u) \quad = \quad \sum_{v} \mathbf{1}_{\{\{u,v\} \in E\}}\,, \tag{2.3}$$

where $\mathbf{1}_{\{\cdot\}}$ is the indicator function. The generalization of the degree function for weighted

graphs is a weight function, defined for a single node as:

$$w(u) \quad = \quad \sum_{v:\{u,v\}\in E} w(u,v). \tag{2.4}$$

A similar function can be defined over a set of edges:

$$w(E) \quad = \quad \sum_{\{u,v\}\in E} w(u,v). \tag{2.5}$$

2.1.2 Clustering

Hard Clustering. *Hard clustering* is the partitioning of the nodes into non-overlapping groups so that each group contains a set of densely connected vertices, with only sparser connections between the groups. Let $\mathcal{C} = \{C_1, \cdots, C_K\}$ be a partition of V, with each C_k, $k \in \{1, \cdots, K\}$, being non-empty. We call $\mathcal{C}$ a clustering of $\mathcal{G}$ with the elements C_k as the clusters. By hard clustering, we mean that each node is assigned to a single cluster as:

$$C_i \cap C_j \quad = \quad 0, \quad \forall i,j : i \neq j,\ C_i, C_j \in \mathcal{C}, \tag{2.6}$$

Given a partitioning of the nodes, we can also define two categories of edges, namely intra-cluster edges and inter-cluster edges. We refer to the set

$$E(C_k) \quad = \quad \{\{u,v\} \in E : u, v \in C_k\} \tag{2.7}$$

as the set of intra-cluster edges for a cluster C_k. Similarly the set of inter-cluster edges of the clusters C_i and C_j is

$$E(C_i, C_j) \quad = \quad \{\{u,v\} \in E : u \in C_i, v \in C_j\}. \tag{2.8}$$

Soft Clustering. *Soft clustering*, also called *fuzzy clustering*, is the method of clustering in which the set of clusters are not necessarily disjoint. In other words, we allow the clusters $\mathcal{C} = \{C_1, \cdots, C_K\}$ to share their nodes and overlap by eliminating the condition (2.6). To identify soft clusters, we employ a matrix $\mathbf{G} \in \mathbb{R}^{N \times K}$, where each element $g_{ik} \in [0, 1]$, $i = 1, ..., N$, $k = 1, ..., K$ represents the degree to which the i-th node belongs to the j-th cluster. Using $\mathbf{G}$, the partitioning $\mathcal{C}$ is defined as

$$C_k \quad = \quad \{v_i \in V : g_{ik} \neq 0\}. \tag{2.9}$$

2.1.3 Quality Indices

A crucial step of the clustering task is selecting a criterion that measures the merit of each candidate partitioning of the nodes. Quality measures are therefore used to assess how well a partitioning describe natural communities of a network. By defining a quality index, we map the space of all possible partitions into the space of real numbers $\mathbb{R}$. The central principle among all of these indices is to rate a partitioning based on the paradigm of intra-cluster density versus inter-cluster sparsity. This implies that a high quality clustering should obtain densely connected groups of vertices, with only sparser connections between the groups. Here we discuss widely used quality metrics.

Coverage [22]. *Coverage* is defined as the ratio of the number (or weight) of the edges contained within clusters to the total number (or weight):

$$Coverage(\mathcal{C}) \quad = \quad \sum_{C_k \in \mathcal{C}} \frac{w(E(C_k))}{w(E)}\,. \tag{2.10}$$

Maximization of coverage results in minimizing the number (or weight) of inter-cluster edges which is desirable because it represents stronger similarity of the nodes belonging to the same cluster. Coverage maps each clustering onto the interval $[0, 1]$. However, one drawback of coverage is that it takes its largest value of 1 in the trivial case where there is a single cluster of all nodes.

Inter-Cluster Conductance [23]. *Conductance* of a cut $(C, V \setminus C)$ (with $C \subset V$), i.e., dividing nodes into two non-empty groups, is defined as

$$\phi(C, V \setminus C) \;=\; \frac{w(E(C, V \setminus C))}{\min(\sum_{v \in C} w(v), \sum_{u \in V \setminus C} w(u))}\,. \tag{2.11}$$

The conductance measures the normalized weight of the cut. The normalization is determined by calculating the total weight of the edges within each of the two groups, and taking the minimum of these two values. The inter-cluster conductance of a clustering is

$$\phi(\mathcal{C}) \;=\; 1 - \max_{C_k \in \mathcal{C}} \frac{w(E(C_k, V \setminus C_k))}{\min(\sum_{v \in C_k} w(v), \sum_{u \in V \setminus C_k} w(u))}\,. \tag{2.12}$$

Inter-cluster conductance is based on the concept of network bottlenecks. It measures the maximum conductance over all of the cuts that define the clusters in the clustering. As this value grows large, it indicates that there are many (or highly weighted) edges between one of the clusters and the other identified clusters, relative to the number (or weight) of the intra-cluster edges. Inter-cluster conductance measures clustering quality and maps it onto the interval $[0, 1]$.

Ratio Cut [24]. The *ratio cut* of a clustering is defined as

$$Rcut(\mathcal{C}) = \sum_{C_k \in \mathcal{C}} \frac{w(E(C_k, V \setminus C_k))}{|C_k|} . \tag{2.13}$$

It indicates the strength of within-cluster edges, relative to the cluster size.

Normalized Cut [25]. The normalized cut of a partitioning is defined as the ratio of the size cut to the degree of each cluster as follows:

$$Ncut(\mathcal{C}) = \sum_{C_k \in \mathcal{C}} \frac{w(E(C_k, V \setminus C_k))}{w(E(C_k))} . \tag{2.14}$$

Minimizing the normalized cut can be interpreted as minimizing the similarity of nodes in different clusters, relative to the degree of each cluster.

Modularity [9]. The widely-used metric, *Modularity*, is related to the coverage metric but attempts to overcome the drawback that the coverage is maximized by a single cluster. Modularity is defined as the fraction of within-cluster edges (weights) minus the expected value of it in a null model where the positions of the edges were randomized [26], while preserving the node degrees. A single cluster has a modularity value of 0, because the coverage for the actual edge structure of the graph has the same value as that expected from a random graph. Modularity maps onto the interval $[-1, 1]$. The definition of modularity for unweighted graphs is:

$$Q(\mathcal{C}) = \frac{1}{2M} \sum_{u,v} \Big[w(u, v) - \frac{d(u)d(v)}{2M} \Big] \delta_{c(u)c(v)} , \tag{2.15}$$

where $\delta_{c(u)c(v)}$ is equal to one if and only if both u and v are assigned into the same community. For weighted graphs the definition changes to

$$Q(\mathcal{C}) = \sum_{C_k \in \mathcal{C}} \Big[\frac{w(E(C_k))}{w(E)} - \Big(\frac{\sum_{v \in C_k} w(v)}{2w(E)} \Big)^2 \Big] . \tag{2.16}$$

Reichardt et al. [27] generalized the modularity metric to incorporate a resolution parameter. This parameter, denoted by γ, allows us to tune the size of clusters in the optimal clustering and also aim at finding clusters at the true scale of resolution of a network which is not necessarily the same imposed scale while optimizing the traditional modularity metric [28]. The $Q_\gamma(\mathcal{C})$ is therefore computed as

$$Q_\gamma(\mathcal{C}) = \frac{1}{2M} \sum_{uv} \Big[\Big(w(u,v) - \gamma \frac{d(u)d(v)}{2M} \Big) \Big] \delta_{c(u)c(v)} , \tag{2.17}$$

More details on multi-resolution modularity quality functions are described in [28].

2.1.4 Hierarchical Clustering

Hierarchical clustering consists of a large family of early, but still widely used methods. A hierarchical clustering algorithm produces a hierarchy of several clusterings with different scales. Conventionally, hierarchical structure can be represented by a tree structure, called a dendrogram, whose leaves correspond to individual nodes and the root represents a trivial cluster with all nodes. As shown in Figure 2.1, representing the structure of a network in a dendrogram format has the advantage of explicitly illustrating the organization at all scales in a network simultaneously such that a flat partitioning can simply be founded by cutting the tree horizontally at an appropriate level of resolution.

A standard approach in hierarchical clustering is to define a metric for measuring the

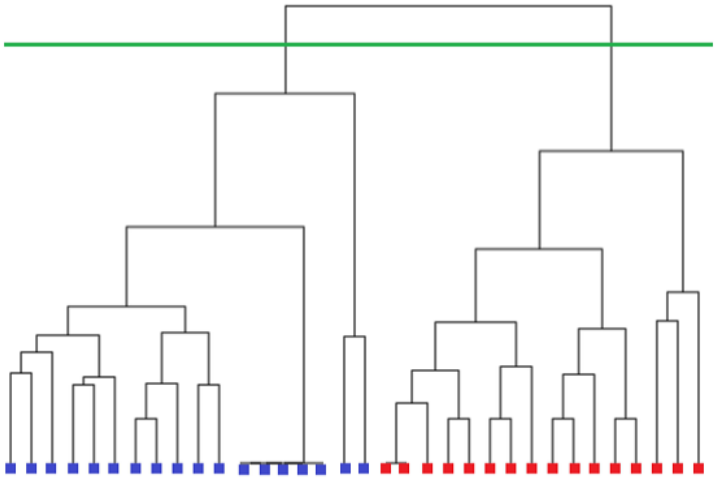


Fig. 2.1 The dendrogram reveals the results of average-linkage hierarchical clustering using the Euclidean distance metric for the karate-club network [1]. Edges indicate friendship between members of the sport club. The left and right subtrees of the green cut correspond to known groups within the club. The figure has been produced using Pajek software [2]

similarity (or dissimilarity) of two clusters. An algorithm then strives to construct the hierarchy by either continuously merging or splitting clusters at each step based on the the metric. The goal is for closely related nodes to have common ancestors that are lower in the tree than those of more distantly related nodes.

The hierarchical clustering is therefore divided into two major paradigms [29]:

1. *Agglomerative* algorithms (bottom-up)

2. *Divisive* algorithms (top-down)

In agglomerative clustering every node is initially considered as a cluster. Then we compute the (dis)similarity for each pair of node and form an $N \times N$ matrix. According to this matrix, two clusters that have the smallest intergroup dissimilarity or the largest similarity are merged together. The matrix is then updated and the merging task repeated until only a single cluster containing all nodes is left. At each step, this produces a grouping at the

next higher level with one less cluster. In traditional methods, a distance (dissimilarity) matrix (e.g., Euclidean distance, Manhattan distance, Cosine similarity) is first defined and then grouping can be carried out using three common methods: single-linkage clustering, complete-linkage clustering, and average-linkage clustering [29]. Recently, some techniques have been proposed based on the improvement of a given objective function. These objective functions can be one of those introduced in Section 2.1.3. Examples include modularity improvement update techniques [30,31] and a greedy agglomerative clustering based on the optimization of the normalized cut of a network [32].

Divisive clustering algorithms start by considering all nodes in a single cluster. They then recursively split one of the clusters into two clusters by removing a set of edges within the cluster to form two disjoint groups of nodes. In general, divisive methods have not been studied as extensively as agglomerative approaches in the clustering literature [29]. More recently, however, there has been a surge of interest around this family of techniques. One of the most influential divisive algorithms was proposed by Girvan and Newman [13]. It is based on the measure of *betweenness*, which represents the frequency of the participation of edges in a process [33]. The betweenness metric can defined in three ways a) Short-path, b) Network resistor, and c) Random walk. At each iteration an edge is removed based on the betweenness score and the betweenness of all edges is recalculated.

2.1.5 Spectral Clustering

Spectral clustering refers to the entire family of both graph and data clustering methods that are based on the spectral analysis of the similarity matrix or its variants such as the Laplacian matrix $\mathbf{L}$, the symmetric normalized Laplacian $\mathbf{L}_s$, and the unsymmetrical

normalize Laplacian $\mathbf{L}_{rw}$. These matrices are defined as follows:

$$\mathbf{L} = \mathbf{D} - \mathbf{W} \tag{2.18}$$

$$\mathbf{L}_s = I - \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}} \tag{2.19}$$

$$\mathbf{L}_{rw} = I - \mathbf{D}^{-1}\mathbf{W}\,, \tag{2.20}$$

where $\mathbf{D}$ is a diagonal matrix with the elements corresponding to row sums of the similarity matrix $\mathbf{W}$. Considering the Laplacian $\mathbf{L}$ of a graph, a fundamental observation is that the eigenvalues of the Laplacian matrix of network contain K zeros if the network has K connected components. If the network is connected, but consists of K prominent clusters, the spectrum of the unnormalized Laplacian will have one zero eigenvalue, with all others being positive. The K lowest eigenvalues will still have values close to zero, representing that there are K weakly connected subgraphs. As discussed in [34], the most common spectral clustering approaches strive to optimize the normalized cut or the ratio cut. Since finding an exact solution for the minimization of (2.13) and (2.14) is an NP-hard problem, the use of approximate techniques is inevitable. Spectral clustering formulates the minimization into trace optimization problems and then solves relaxed versions of them. The ratio cut can be written as [24]:

$$Rcut(\mathcal{C}) = \mathrm{tr}(\mathbf{X}^T\mathbf{L}\mathbf{X})\,, \tag{2.21}$$

in which $\mathbf{X} \in \mathbb{R}^{N\times K}$ is a matrix whose entries are determined, using the nodes $v_i \in V$, as:

$$x_{ik} \;=\; \begin{cases} \frac{1}{\sqrt{|C_k|}} & \text{if } v_i \in C_k, \\ 0 & \text{otherwise} \end{cases}. \tag{2.22}$$

With this construction, the columns in $\mathbf{X}$ are orthonormal to each other, so $\mathbf{X}^T\mathbf{X} = I$. Finding the optimal clustering (or binary-valued matrix $\mathbf{X}$) is very difficult, but a relaxed optimization problem can be formulated by allowing the elements of the matrix $\mathbf{X}$ to take real values. The relaxed optimization problem becomes:

$$Rcut^*(\mathcal{C}) \quad = \quad \min_{\mathbf{X}\in\mathbb{R}^{N\times K}} \operatorname{tr}(\mathbf{X}^T\mathbf{L}\mathbf{X}) \quad \text{s.t.} \quad \mathbf{X}^T\mathbf{X} = I\,. \tag{2.23}$$

The optimal solution of (2.23) consists of the eigenvectors corresponding to the K smallest eigenvalues of $\mathbf{L}$. The optimal solution $\mathbf{X}$ is then discretized to obtain the clustering result, typically by applying k-means clustering on the rows of $\mathbf{X}$.

A similar approach is employed for optimizing the normalized cut metric in [35, 36]. In this algorithm, the only difference is that instead of $\mathbf{X}^T\mathbf{L}\mathbf{X}$, the term $\mathbf{X}^T\mathbf{L}_{rw}\mathbf{X}$ is the objective function and the optimization is solved by finding eigenvectors corresponding to the K smallest eigenvalues of $\mathbf{L}_{rw}$. Later, Ng et al. [37] proposed that the symmetrical normalized Laplacian $\mathbf{L}_s$ can be used for spectral clustering because the technique of Maila and Shi [35] does not perform well when intra-connection of clusters varies extremely among the clusters. Other spectral approaches such as Markov Clustering (MCL) and Geometric Minimum Spanning Tree Clustering (GMC) are described in detail in [38].

2.1.6 Fuzzy Clustering

The methods discussed in the previous sections aim to detect hard partitions. However, in practice nodes are often shared among communities. In this section, we briefly present common algorithms that find overlapping communities.

Clique Percolation Method [18]

This method is based on identifying k-cliques, i.e., complete graphs with k vertices. The concept of identifying cliques perfectly matches the definition of a community because cliques include well-connected subgraphs and exclude bottleneck edges. Palla et al. [18] define a k-clique community as the union of adjacent k-cliques (i.e., k-cliques that share $k-1$ nodes). By this definition, k-clique communities can share vertices and therefore represent overlapping communities. This algorithm first finds all complete subgraphs, called cliques, of the network that are not parts of larger complete subgraphs. Then a square matrix is formed, which has size equal to the number of cliques, and whose elements represent the number of vertices shared by each pair of cliques. To find the k-cliques communities all of the non-diagonal entries with value less than $k-1$ are erased from the matrix. Diagonal entries with value less than k are also erased. The separate components can now be determined from the new matrix. The drawbacks of this method include the need to choose the clique size k and the high computational cost, which depends on many factors and is difficult to estimate ahead of time.

Fitness-based Method [39–41]

Lancichinetti et al. [39] introduced the concept of *fitness* of a subgraph and used it to identify overlapping communities by maximizing it in greedy fashion. The fitness of a subgraph C is defined as

$$f(C) = \frac{d_{in}(C)}{(d_{in}(C) + d_{out}(C))^{\alpha}}, \tag{2.24}$$

where $d_{in}(C)$ and $d_{out}(C)$ are the total internal and external degrees of the nodes of C and α is a parameter to control the size of the communities. The fitness of a clustering is then

the sum of the fitness vlues for the individual

To optimize the fitness of the entire network, a random node is picked and considered as a seed cluster. This community containing a single node is then expanded by an iterative procedure, similar to the one originally presented in [42], such that a neighbour node which causes the largest improvement in the fitness is added to the community. The fitness improvement of all the nodes of the community is then recalculated and if there is a node with negative improvement, it is excluded from the community, yielding a new community. Next, both new and old communities are expanded iteratively. The iteration stops when no fitness improvement occurs while adding neighbour nodes. At this point, a node which is not yet assigned to any group (if there is any) is randomly picked and expanded until all unassigned nodes are exhausted. During the expansion, clusters are allowed to add any neighbour nodes, even if they are already members of other clusters. This allows the formation of overlapping communities. Lancichinetti et al. claim that by changing the parameter α, one can construct a hierarchy of partitions.

Later, Lee et al. [40] employed a similar algorithm without a node exclusion mechanism. Their algorithm also differs from the original one in the sense that it starts with cliques of nodes, rather than single nodes, as the seed communities. Havemann et al. [41] recently proposed a new fitness-based algorithm which is parameter free and is able to extract the hierarchy of a network at all resolution levels without the need to repeat the clustering procedure for many different values of α. Their algorithm avoids unnecessary computational expense. They also add one to the numerator of Equation (2.24) because if the original definition is employed in a clustering algorithm there is no way for a node to remain isolated and form a single-node cluster.

Non-negative Matrix Factorization Method [21, 43]

Non-negative matrix factorization (NMF), proposed in [44], is an alternative approach to principal components analysis (PCA). In NMF, however, both data and components are assumed to be non-negative. NMF was originally applied to a facial image dataset and successfully learned to represent faces with a set of basis images resembling different parts of faces. This is mainly because in NMF, unlike PCA, due to the non-negativity constraints only additive combinations are allowed. Recently, many variants of this algorithm for various applications have been proposed. Yu et al. [43] used NMF in the context of graph factorization and clustering. They strived to factorize the similarity matrix $\mathbf{W}$ into three components as

$$\mathbf{W} \approx \mathbf{X}\mathbf{\Lambda}\mathbf{X}^T, \tag{2.25}$$

where $\mathbf{X} \in \mathbb{R}_+^{N\times K}$ is a non-negative matrix with the elements x_{ik} representing the probability that an interaction in community k involves node v_i. The matrix $\mathbf{\Lambda} \in \mathbb{R}_+^{K\times K}$ is a non-negative diagonal matrix with λ_{ii} being the prior probability that the interaction in the w_{ij} is due to the k-th community. Having identified the matrices $\mathbf{X}$ and $\mathbf{\Lambda}$, one can fully characterize fuzzy clusters by assigning nodes to the communities with respect to these probabilities. The inference was performed by minimizing the KL-divergence distance between the two sides of Equation (2.25).

Recently, Psorakis et al. [21] proposed a NMF-based technique for detecting overlapping communities. Their model is based on the original model of Lee and Seung [44]. In this model, the goal is to factorize the similarity matrix $\mathbf{W}$ such that

$$\mathbf{W} \approx \mathbf{G}\mathbf{H}, \tag{2.26}$$

where matrix $\mathbf{G} \in \mathbb{R}^{N \times K}$ and each element g_{ik} represents the degree to which the i-th node belongs to the k-th community. The matrix $\mathbf{H}$ is different from $\mathbf{G}$ if the network $\mathbf{W}$ is directed. Psorakis et al. also employed a Bayesian model called *Automatic Relevance Determination* (ARD) [45] to solve the problem of model order selection. This model is thoroughly described in Section 3.6. Utilizing ARD has the advantage of finding the appropriate number of clusters in the network without the need to use modularity-based methods, which suffer from several drawbacks [46].

Other Methods

The literature on fuzzy community detection is not as comprehensive as hard clustering techniques. More recently, however, there have been some algorithms proposed to address this issue. Nepusz et al. [47] formulated the task as a constrained optimization problem and solve it by a quadratic-time algorithm. Zhang et al [48] represented a network in a Euclidean space with $k-1$ dimension and then employed the fuzzy c-means algorithm to detect fuzzy communities. Ball et al. [17] defined a generative probabilistic model and found the parameters of the model by maximum likelihood estimation. Their method differs from others in term of grouping the edges of a network instead of the nodes. By assigning the edges to the communities overlapping communities can be founded by computing the fraction of the incident edges of nodes belonging to the each community. Gregory [49] extended the label propagation technique for detecting overlapping communities. In this algorithm for each node a label is defined that propagates between neighbouring vertices so that members of each community reach a consensus on their community membership.

2.2 Dynamic Clustering

2.2.1 Overview

While there has been a surge of interest toward the exploration of the general trend and rules of evolution in social networks [50–53] and biological network [16] among scholars in recent years, the importance of identifying meaningful communities and tracking them in time has been perceived even more than before. Determining the communities of a dynamic network, however, has been shown to be a non-trivial task. Having considered the fact that clustering a single graph itself is still a challenging topic with the new insights of it being discovered everyday, it becomes predictable that investigating clusters of a dynamic network is not straightforward either.

Over the past decade, many techniques attempted to address various challenges that arise in the analysis of dynamic communities. The earliest thread of works, which calls the problem *incremental clustering*, focuses mostly on community tracking over time. The goal is to effectively trace the evolution of a *specific* community [54]. Incremental clustering techniques strive to handle the changing number of nodes and communities while finding the *best* possible sequence of communities in time. Generally speaking, each method has defined *best* in its own way by introducing a metric or set of metrics. A set of events are often defined based on the metrics to characterize the evolution of a community. These events usually determine when a community faces a transition such as birth, death, contraction, and expansion. We discuss incremental clustering techniques in Section 2.2.3.

Whereas incremental clustering techniques strive to trace a community over time, another group of algorithms focuses on the challenge of *temporal smoothness* discussed in Section 1.2. This approach to dynamic clustering is called *evolutionary clustering* in the literature. The name was introduced by Chakrabarti et al. [55], who proposed an algorithm

to identify clusters smoothly in time. The quality of temporal smoothness implies that the network structures at adjacent timesteps are likely to be correlated.

Evolutionary clustering techniques incorporate historic data or historic clusters into the processing at the current timestep so that the communities detected at each timestep do not differ significantly from the previous ones. This approach is especially useful for analysis of noisy datasets. Measurement noise may cause substantial variations in the identified communities at consecutive timesteps if one attempts to find clusters independently at each time step. By employing evolutionary clustering the short term temporal variations are diminished and clusters tend to change smoothly over time. We elaborate on various evolutionary methods in Section 2.2.4.

2.2.2 Notation

In the following we define the basic terms of dynamic networks and the clustering task in dynamic graphs. A dynamic graph $\mathcal{G} = (\mathcal{G}_0, \mathcal{G}_1, \cdots, \mathcal{G}_T)$ is a sequence of graphs, with $\mathcal{G}_t = (V_t, E_t, \mathbf{W}_t)$ being the state of dynamic graph at time step t with $N_t = |V_t|$ nodes and $M_t = |E_t|$ edges. The matrix $\mathbf{W}_t$ identifies the count weight associated with each link of the graph. In the network clustering context, $\mathbf{W}_t$ is our observation at timestep t and each element $w_{ij,t}$ represents the similarity between i-th and j-th nodes. The goal of clustering is to provide a sequence of partitions over time $\mathcal{C} = (\mathcal{C}_0, \mathcal{C}_1, \cdots, \mathcal{C}_T)$ with each member $\mathcal{C}_t = \{C_{1t}, C_{2t}, \cdots, C_{K_t t}\}$ being a set of K_t clusters at timestep t.

2.2.3 Incremental Clustering

In one of the earliest methods of incremental clustering, Toyoda and Kitsuregawa [56] conducted research into community evolution for the specific network of web communities where URLs are nodes and hyperlinks between them define their associated link weights.

They first extract the web communities using HITS-based algorithms [57] [1] and then categorize the evolution of each cluster into communities that grow, shrink, emerge, dissolve, and split over time by introducing a set of metrics.

Later, Hopcroft et al. [58] detected communities using agglomerative hierarchical clustering (Section 2.1.4). In order to match communities in time, the concept of *natural communities* is introduced; these are defined as those that appear in the majority of clusterings applied to multiple small perturbations of the original graph (the perturbations involve removing a small fraction of the core nodes). As pointed out in [33], the main drawback of the method is the hierarchical clustering. Hierarchical trees consist of numerous partitions of the network and therefore extracting meaningful communities is not a trivial task.

The algorithm proposed by Aggarwal and Yu [59] has the two steps of online data summarization and offline detection of evolving communities. The first step is responsible for efficiently storing data changes over time while respecting disk storage limits. In the offline step, the algorithm attempts to find communities which expand, contract, or remain the same size. Aggarwal and Yu construct a differential graph for each pair of timesteps, where the weight of each edge is equal to the difference between the normalized weights at each timestep. Developing or growing communities are expected to contain primarily positive edge weights in the differential graph; shrinking communities will have more negative edge weights. The clustering is performed using an iterative, heuristic approach that involves growing communities around seed nodes, trying to ensure that each community contains mostly positive edges or mostly negative edges, rather than a mixture.

Falkowski et al. [60] also used the notion of positive and negative edges in the paradigm

[1] HITS refers to link analysis algorithms for ranking web pages based on the definition of Hub and Authoritative pages. By definition, hubs contain the databases of links to Authoritative pages, whereas Authoritative pages have the authority of information [57]

of density-based incremental clustering [61] to adapt communities. Falkowski et al. claimed that a positive edge that either emerges or demonstrates intensified interaction in the most recent timestep, may lead to one of three events for its corresponding cluster: creation, absorption, or merge; whereas a negative edge can potentially result in removal, reduction, or split of the corresponding cluster. Thereby, the clusters are constantly modified over time based on the evolutionary characteristics of the network.

While the algorithms in [59, 60] attempt to place nodes with similar evolutionary characteristics in the same clusters over time, the method in [62] focuses on matching and tracking communities across time. Sun et al. proposed a parameter-free framework called GraphScope [62] to identify the unknown number of communities and break the stream of network data into a sets of consecutive graphs (graph segments) according to the notion of *change-points*. The identification of the graph segments is based on the concept of minimum description length (MDL) encoding for graph transmission. The total encoding cost of all graph segments is given by the sum of the encoding costs of each segment. Finding the optimal partition and time segmentation to minimize the total cost is a NP-hard problem, thereby a local search is used to find an approximate solution. Although GraphScope strives to obtain optimal clusters by changing the number and size of communities for each graph segment, it does not explicitly address the problem of insertion/deletion of nodes over time, which occurs quite often in real dynamic networks. Ferlez et al. [63] also proposed a similar MDL-based method to identify clusters and their evolutions. Their emphasis, however, is on the topic model for research papers and the visualization of temporal network.

Palla et al. [64] applied the Clique Percolation Method (CPM) introduced in Section 2.1.6 at each timestep. To match the communities over time, the authors claimed that two communities with the largest relative overlap in consecutive timesteps actually belong to

the same evolving community. The relative overlap for two communities C_{i,t_0} and C_{j,t_0+t} is defined as

$$a_{ij}(t) = \frac{C_{i,t_0} \cap C_{j,t_0+t}}{C_{i,t_0} \cup C_{j,t_0+t}}. \tag{2.27}$$

Palla et al. pointed out that by applying this method to joint graphs for pairs of consecutive timesteps, the resulting clique-based communities contain communities in each of the individual timesteps. In Equation (2.27), if both subgraphs C_{i,t_0} and C_{j,t_0+t} belong to the same community with different states, the function is called autocorrelation. The authors also explored the evolutionary characteristics of the identified communities such as age and community size in graph of phone calls between customers of a mobile phone company and a collaboration network between scientists. They demonstrated empirically that the life time of communities in these networks depends on the temporal features. The larger groups that frequently change in time decay faster, whereas smaller groups were found to persist longer if their membership remained unchanged. One drawback of this method is that, unlike the method in [62], it does not address how the appropriate number of clusters is determined.

An event-based incremental clustering scheme is proposed by Asur et al. [65]. In this framework, clusters are found independently at each timestep by a static clustering method and further analysis matches clusters in time and reveals their evolution. The authors introduced several indices for distinguishing two categories of events. Community events include Continue (most of the vertices of a community do not change their membership in consecutive time steps), k-Merge (two clusters are considered merged based on a predefined threshold k), k-Split (a cluster split into two based on a threshold), Form (no pair of vertices of the cluster were in the same cluster at a previous timestep) and Dissolve

(opposite of Form). The events involving nodes are Appear (node insertion), Disappear (node removal), Join (when a vertex joins a new cluster) and Leave (when a vertex leaves its previous cluster). Asur et al. also defined popularity and influence metrics to respectively capture the number of nodes joins a cluster over time and determine the influence a node has on the evolution of its corresponding cluster. This method did not provide any information on the choice of static clustering and its required characteristics. Recently, a similar method to [65] is proposed in [66]. In this method, however, a *social positioning* metric discussed in [67] is utilized to identify the events. This metric quantifies how important a node is for the entire community.

Green et al. [68] also identify clusters using a static clustering technique in time and then clusters identified at each timesteps are compared and matched to communities found at previous timesteps. To match clusters over time, a method is used similar to that employed in [64]. The relative overlap (2.27) between the current communities and the previous ones is computed and based on that a set of events such as expansion, contraction, merge, split, death, and birth is defined. In this approach, in contrast to similar methods, dead clusters are allowed to emerge again after a dormant period. A distinction between [65] and [68] is that in [68] the evolution of clusters together with its events is given by computing a single metric. This substantially reduces the complexity and running time of the algorithm. The approach of Goldberg et al. [69] is very similar to [68]. They also find static clusters at each timestep and attempt to match the clusters according to the relative overlap measure in time. Their justification for the use of relative overlap measure involves identifying a set of axioms to detect the best evolution chain of temporal communities. Like [65], neither of [68,69] comment on appropriate static clustering techniques to find communities at each timestep.

Ning et al. [70] proposed a different approach to handle community detection when tem-

poral variations change the similarity matrix. Their approach is based on the optimization of the normalized cut (2.14) by a spectral analysis technique described in Section 2.1.5. Their main contribution includes an algorithm that incrementally derives the approximate change of eigenvalues and eigenvectors and subsequently refines community assignment in time. Although this technique can handle node removal/insertion, one should have information about the number of communities beforehand.

Gorke et al. [71] extended an algorithm proposed by Flake et al. [72] for dynamic clustering. In [72] an algorithm to provide a guarantee on the quality of the clusters in terms of the expansion of the inter-cluster cuts and the intra-cluster cuts is proposed. To make it compatible with dynamic networks, Gorke et al. processed arbitrary atomic changes in the graph.

Fenn et al. [73] identified clusters at each timestep using the paradigm of Hamiltonian dynamics [27] where the modularity metric is a special case. In this formulation, a parameter is responsible for controlling the resolution of clustering. By tuning the resolution parameter one is able to construct a hierarchy of partitions. The authors optimized the Hamiltonian by a greedy modularity optimization algorithm proposed by Blondel et al. [31]. Their analysis is distinct from other incremental clustering techniques in the sense that they track the community evolution by a node-centric approach, without the need to track all communities. Their analysis involves using a static measure called *community alignment*. Defined for each node, this indicates to what extent the node is located in the center of a community. To identify the persistence of nodes over time the authors introduced a node-centric version of the relative overlap measure,

$$a^v(t) = \frac{C^v_{t_0} \cap C^v_{t_0+t}}{C^v_{t_0} \cup C^v_{t_0+t}}, \tag{2.28}$$

where $C^v_{t_0}$ and $C^v_{t_0+t}$ are communities containing node v at timesteps t_0 and $t_0 + t$.

All of the methods introduced above are basically two-step algorithms where at first step communities are detected at each timestep by a static clustering method and then their evolution and characteristics are inferred by matching communities across time. Such approaches fail to capture the temporal smoothness of the network and subsequently suffer from inappropriate temporal variations when analysing noisy datasets. These drawbacks suggest more unified approaches that seek to identify communities and their evolution at the same time. We discuss a number of such methods in the following section.

2.2.4 Evolutionary Clustering

In this section, we review the methods proposed for discovering community structures with maintaining temporal smoothness in the network. First, in Section 2.2.4, we discuss traditional algorithms that capture the continuity of the data by either a) introducing a cost function with a penalty on the temporal variations of communities; b) filtering the similarity matrices to calculate smoothed weights; or c) building a graph containing the entire data of all timesteps and performing static clustering on the resultant graph. In Section 2.2.4 we review algorithms built upon probabilistic models. At the end of this section, we provide a brief overview of fields and algorithms that are related to evolutionary network clustering.

Capturing Temporal Smoothness

Chakrabarti et al. for the first time captured temporal smoothness in their model in [55]. They incorporate previously learned communities into the model through an objective cost function that assesses both the quality of clustering and shift between partitions of two consecutive timesteps. The function to be optimized consists of two aspects: a snapshot cost responsible for measuring the quality of the clustering and a temporal cost which

measures the (dis)similarity of two clusterings. In other words, penalizing the objective function by a temporal cost results in a trade-off between the quality of clustering and temporal smoothness. Ensuring the continuity of clusters over time comes at the price of reducing the overall quality of model. The goal of of model is to minimize the whole cost function defined as,

$$\text{cost}(\mathcal{C}_t, \mathbf{W}_t) \quad = \quad sc(\mathcal{C}_t, \mathbf{W}_t) - cp.tc(\mathcal{C}_{t-1}, \mathcal{C}_t)\,, \tag{2.29}$$

where $cp > 0$ is a parameter that controls the trade-off between the snapshot cost sc and the temporal cost tc. Chakrabarti et al. define these costs in two different frameworks: hierarchical agglomerative clustering and k-means. The latter can only be used for the clustering of objects in Euclidean space (not data networks). In the hierarchical agglomerative clustering, the snapshot cost is defined as the sum of the similarity of objects that were merged to produce a internal node over all internal nodes, whereas the temporal cost is the average tree distance between all pairs of nodes within two trees. Heuristics are then introduced to optimize the cost function. In the methods outlined in [55], the key assumption is that the number of clusters is known and fixed over time. These constraints reduce the applicability of the method. Later, Chi et al. strived to address variations in the number of clusters in [19, 74]. They proposed a similar cost function with only a small change as

$$\text{cost}(\mathcal{C}_t, \mathbf{W}_t) \quad = \quad \alpha.sc(\mathcal{C}_t, \mathbf{W}_t) + \beta.tc(\mathcal{C}_{t-1}, \mathcal{C}_t)\,, \tag{2.30}$$

where $\alpha \in [0, 1]$ together with $\beta(= 1 - \alpha)$ reflects the emphasis on the snapshot cost and temporal cost respectively. Chi et al. formulated the cost function in the context of spectral

clustering. They assessed the snapshot cost using popular spectral quality indices including Normalized Cut. It can be written as

$$sc(\mathcal{C}_t, \mathbf{W}_t) = Ncut(\mathcal{C}_t) \quad (2.31)$$

$$= \mathrm{tr}(\mathbf{X}_t^T \mathbf{L}_{s,t} \mathbf{X}_t)\,, \quad (2.32)$$

which is the relaxed formulation of Normalized Cut (Section 2.1.5). Two different approaches are then proposed for determining the temporal cost: Preservation of Cluster Membership (PCM) and Preservation of Cluster Quality (PCQ). In the former the current partition is applied to historic data and the resulting cluster quality determines the temporal cost. In the latter, the partition at time t is compared to the partition at time $t-1$ and the resulting difference determines the temporal cost. In PCM, the temporal cost is computed as

$$tc(\mathcal{C}_{t-1}, \mathcal{C}_t) = \frac{1}{2} \| \mathbf{X}_t \mathbf{X}_t^T - \mathbf{X}_{t-1} \mathbf{X}_{t-1}^T \|\,, \quad (2.33)$$

where $\mathbf{X}$ is the matrix previously defined in expression (2.23). The temporal cost for PCQ is

$$tc(\mathcal{C}_{t-1}, \mathcal{C}_t) = \mathrm{tr}(\mathbf{X}_t^T \mathbf{L}_{s,t-1} \mathbf{X}_t)\,. \quad (2.34)$$

Chi et al. claimed that although their model can handle variations in the number of clusters, it still needs to derive the initial number of clusters by an auxiliary technique. Furthermore, the authors attempted to address the problem of node insertion and removal. The node removal can be easily done by removing the corresponding rows and columns of the removed nodes from the matrices of the previous timestep. Node insertion, however,

is a challenging problem. Chi et al. proposed to fill the corresponding rows and columns of the similarity matrix by the average of its elements. This approach is ad-hoc and may cause errors in the clustering results.

To address the issues discussed above, Kim and Han [75] proposed a distinct algorithm. In their proposed method, the temporal smoothness is captured at the edge level instead of the community level. The smoothed similarity between two nodes u and v is given by

$$\sigma'_t(u,v) \;=\; \alpha.[\sigma_t(u,v) - \sigma_{t-1}(u,v)] + \sigma_{t-1}(u,v)\,, \tag{2.35}$$

where $\sigma_t(u,v)$ is the *cosine similarity* metric of nodes u and v at timestep t and $\sigma'_t(u,v)$ denotes its smoothed version. The parameter α controls the emphasis on historic data. The cosine similarity for nodes u and v is defined as

$$\sigma_t(u,v) \;=\; \frac{|N_t(u) \cap N_t(v)|}{\sqrt{|N_t(u)||N_t(v)|}}\,, \tag{2.36}$$

where $N_t(u)$ is the set of the neighbouring nodes of u at time t defined by

$$N_t(u) \;=\; \{w \in V_t | \{u,w\} \in E_t\}\,. \tag{2.37}$$

At each timestep communities are derived by a density-based approach [61] using the smoothed cosine similarity for all connected nodes. To find the number of communities at each step and also tune the parameter of the density-based algorithm, ϵ_t, Kim and Han proposed a modularity optimization heuristic. The resulting clusters then need to be matched in time. To address this issue, the authors construct a T-partite graph by

connecting the nodes of consecutive timesteps based on the following function:

$$\Gamma(N_t(v_t), N_{t-1}(v_{t-1})) \quad = \quad \begin{cases} 1 & \text{if } \; v_t \in N_{t-1}(v_{t-1}) \text{ and } v_{t-1} \in N_t(v_t) \\ 0 & \text{otherwise} \end{cases}, \qquad (2.38)$$

where $v_t \in V_t$ and $v_{t-1} \in V_{t-1}$. If $\Gamma(N_t(v_t), N_{t-1}(v_{t-1}))$ is non-zero for two nodes of consecutive timesteps, a link, called a *nano-community*, would connect them. Communities are then matched based on the density of nano-communities and a set of events consisting grow, shrink, drift, form, and dissolve. The strengths of this technique are its ability to determine the number of clusters and handle variations in the number of clusters through the notion of edge level smoothness. The main drawback of this model is its rather high computational complexity and running time due to the modularity optimization, forming nano-communities, and matching process which itself includes link counting and computing normalized mutual information. Another weakness is that the model cannot support weighted graphs due to the definition of the density based clustering employed for finding communities at each timestep. Kim and Han pointed out more weaknesses of their algorithm in its later version [76]. The weaknesses addressed in [76] are: "(a) determining only stable clusters of single state (i.e a sequence of clusters connected in time), (b) obtaining few clusters (i.e., the most stable top-k clusters), (c) not being scalable considering the length of dynamic networks, and (d) using a large amount of memory depending on its parameters". In their new framework, called CHRONICLE, the density-based clustering is employed in two steps: finding communities in the graph of each timestep and the clusters of the entire t-partite network.

Similar to the concept of nano-communities [19, 74], connecting nodes across time is the core idea of Mucha et al. [3]. They defined a *multislice network* as the generalized

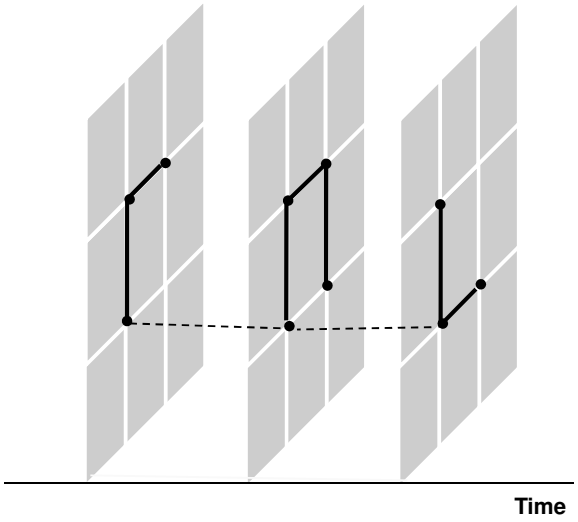


Fig. 2.2 Schematic illustration of the multislice network. Dashed line indicates how a node is connected to itself across different slices. Adapted figure from [3].

form of a static network; it is constructed by coupling multiple adjacency matrices. These additional connections may connect a node to itself in either an ordered manner (e.g., over a series of consecutive timesteps as depicted in Figure 2.2) or all-to-all way (e.g., n-partite graphs appropriate for categorical network slices). In this definition, one is even able to construct a multislice graph from a static network by coupling the same network across multiple resolutions. The main contribution of [3] is a generalization of the modularity metric (Section 2.1.3) with a resolution parameter [27] to make it applicable to multislice networks. The generalization is founded on the Laplacian dynamics formalism of [77]. Under the definition of the stability of communities in Laplacian dynamics [77], Mucha et al. derived a multislice generalization of modularity as

$$Q_{\text{multislice}} = \frac{1}{2\mu}\sum_{uvsr}\Big[\Big(A_{uvs}-\gamma_s\frac{d_s(u)d_s(v)}{2M_s}\Big)\delta_{sr}+\delta_{uv}C_{vsr}\Big]\delta_{c_s(u)c_s(v)}\,, \tag{2.39}$$

where subscripts s and r differentiate the parameters associated with two different network slices (in the case of dynamic networks they represent two timesteps) and C_{vsr} represents the coupling coefficient of node v between slices r and s. A is the adjacency matrix, $d_s(v)$ is the degree of node v and M_s is the total number of edges for slice s. The term 2μ in the denominator represents the total number of inter slice edges plus coupling coefficients, i.e., $2\mu = \sum_{vr}(d_r(v) + \sum_s C_{vsr})$. For simplicity, Mucha et al. assumed that the coupling coefficients only take binary values $\{0, \omega\}$ indicating absence or presence of inter-slice connections and $\gamma_s = \gamma$ for all slices.

The case $\gamma = 1$ can recover the classical interpretation of modularity as the total weight of intra-community edges minus its expected value if edges were allocated randomly while preserving node degrees. Mucha et al. also pointed out that for $\omega = 0$, there are no inter-slice edges and therefore the partition that has the maximum modularity can be obtained by independently optimizing the modularity at each slice. On the other hand, when ω is sufficiently large, the optimal partition is reached when the community assignment of a node remains unchanged over all slices. Thus, this generalization of modularity is able to capture the temporal smoothness when it is applied to a multislice network containing a series of graphs observed at each timestep. The authors have not suggested any algorithm for optimizing the multislice modularity and just claim that it can be achieved by the same heuristics available for the static optimization of modularity.

Xu et al. [7, 78], similar to the methods in [75, 76], focus on smoothing changes in the edges of the graph over time, instead of enforcing smoothness of the clusters themselves. They, however, consider weighted graphs and therefore their framework, called AFFECT,

is more general than those in [75, 76]. The smoothed similarity matrix is given by

$$\overline{\mathbf{W}}_t \;=\; \alpha_t \overline{\mathbf{W}}_{t-1} + (1-\alpha_t)\mathbf{W}_t\,, \tag{2.40}$$

for $t > 1$ and by $\overline{\mathbf{W}}_0 = \mathbf{W}_0$. The goal is to estimate the appropriate *forgetting factor* α_t by trying to determine to what extent short term variations are caused by noise. In the AFFECT model, the similarity matrix is treated as a realization of a non-stationary random process plus a zero-mean noise:

$$\mathbf{W}_t \;=\; \mathbf{\Psi}_t + \mathbf{N}_t\,, \tag{2.41}$$

where $\mathbf{\Psi}$ is an unknown deterministic matrix and $\mathbf{N}_t$ is a noise matrix. The goal is to derive the smoothed similarity matrix $\overline{\mathbf{W}}$ such that its difference from the true similarity matrix $\mathbf{\Psi}$ is minimized. The optimal α_t is estimated by employing shrinkage techniques [79]. After estimating the optimal α_t and subsequently finding the smoothed similarity matrix $\overline{\mathbf{W}}$, a static clustering method is applied to the smoothed similarity matrix of each timestep.

Automatic choice of the parameter which controls the emphasis on the historic dataset was not addressed in other algorithms. Xu et. al elegantly handle this issue, but the smoothing of the similarity matrix does not necessarily translate to continuity in community membership. The method can thus result in undesirable temporal variations in the clusters.

Folino and Pizzuti [80] considered the cost function used by Chi et al. (Equation (2.30)). They, however, utilized a different snapshot cost and a different temporal cost. As a snapshot cost they employed the *community score* introduced in [81]. The community score takes into account both the number of connections inside each community and the number of links between the communities. The snapshot cost for the partition $\mathcal{C}_t$ is therefore

given by

$$sc(\mathcal{C}_t) = \sum_{i=0}^{K_t} \text{score}(C_{i,t}) , \tag{2.42}$$

where

$$\text{score}(C_{i,t}) = \frac{\sum_{v \in C_{i,t}} (\mu_{v,t})^2}{|C_{i,t}|} \times |E(C_{i,t})| , \tag{2.43}$$

in which $\mu_{v,t}$ is

$$\mu_{v,t} = \frac{\sum_{u \in C_{i,t}} A_{uv,t}}{|C_{i,t}|} . \tag{2.44}$$

The parameters $A_{uv,t}$ are the elements of adjacency matrix A_t. For the temporal cost function, the Normalized Mutual Information (NMI) metric is used. Let $\mathbf{M}$ denote the confusion matrix whose element m_{ij} is the number of nodes that communities $C_{j,t-1}$ and $C_{i,t}$ share. The normalized mutual information $NMI(\mathcal{C}_{t-1}, \mathcal{C}_t)$ is then defined as

$$NMI(\mathcal{C}_{t-1}, \mathcal{C}_t) = \frac{-2 \sum_{i=1}^{K_t} \sum_{j=1}^{K_{t-1}} m_{ij} \log(m_{ij} N / m_{i\cdot} m_{\cdot j})}{\sum_{i=1}^{K_t} m_{i\cdot} \log(m_{i\cdot}/N) + \sum_{j=1}^{K_{t-1}} m_{\cdot j} \log(m_{\cdot j}/N)} , \tag{2.45}$$

where $m_{i\cdot}$ and $m_{\cdot j}$ are the sums of the elements of $\mathbf{M}$ over the i-th row and j-th column, respectively. The value of NMI varies between 0 and 1 and the higher NMI becomes, the more similar two partitions are.

By defining the snapshot and temporal costs, the problem can be formulated as a multi-objective optimization problem. Folino and Pizzuti solved it using a genetic algorithm. In order to choose the best solution among all the possible solutions that correspond to different trade-offs between the two objectives, Folino and Pizzuti used the modularity measure

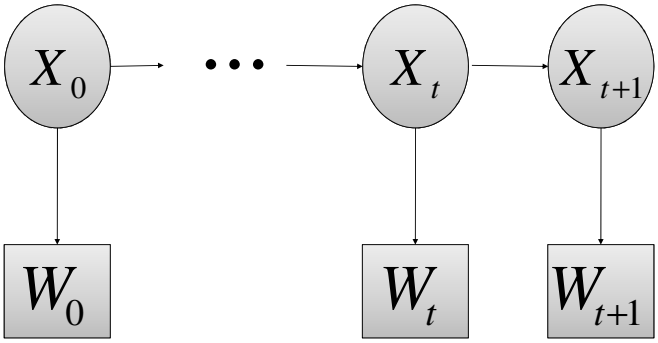


Fig. 2.3 Schematic illustration of the model of Sarkar and Moore [4]. $\mathbf{W}_t$ is the similarity matrix observed timestep t. $\mathbf{X}_t$ is an $N \times K$ matrix representing the community assignment of all nodes in a K-dimensional space. Adapted figure from [4].

(Section 2.1.3). Although this algorithm allows the number of communities to change over time, it can only be applied to unweighted networks. There are several drawbacks to using the modularity metric to select the number of clusters, chief among which is its inability to detect small clusters in large networks (a resolution limit) [46].

Model-based approaches

Sarkar and Moore [4] were among the first to explore the advantages of using probabilistic models in the analysis of dynamic networks. Their focus was social networks. Their model is based on the assumption of the temporal smoothness of the data. In the methodology of [4] the community memberships at each timestep were captured using a hidden network containing a set of variables. These variables, denoted by the matrix $\mathbf{X} \in \mathbb{R}^{N \times K}$, are responsible for mapping the community structure of a network to a K-dimensional space. As depicted in Figure 2.3, the model has two different aspects: a) an observation model that describes how the latent variables govern the observed data and b) a transition model which

is responsible for capturing the temporal smoothness by not allowing the latent variables to change dramatically between timesteps. The goal is to determine the matrix $\mathbf{X}_t$ such that it best describes the observations based on the probabilistic model. The problem is formulated as

$$\mathbf{X}_t = \arg\max_{\mathbf{X}} \mathbb{P}(\mathbf{X}|\mathbf{W}_t, \mathbf{X}_{t-1}) \tag{2.46}$$

or equivalently, after application of Bayes' rule:

$$\mathbf{X}_t = \arg\max_{\mathbf{X}} \mathbb{P}(\mathbf{W}_t|\mathbf{X})\mathbb{P}(\mathbf{X}|\mathbf{X}_{t-1})\,. \tag{2.47}$$

The first term in Equation 2.47 represents the observation model and the second term indicates the transition model. Sarkar and Moore placed probability distribution functions on two models. In the observation model, each pair of nodes is connected by a Bernoulli distribution with parameter depending on the latent variable. In contrast, in the transition model each latent matrix is assumed to be subjected to a Gaussian perturbation with zero mean and σ^2 variance. Finally the authors learned the latent variables by locally optimizing the term 2.47.

Probabilistic modelling is also used in *FacetNet* framework proposed by Lin et al. [6,82]. In the FacetNet framework, Lin et al. strived to extend the non-negative matrix factorization soft clustering technique [43] reviewed in Section 2.1.6 to dynamic networks. Similar to [4], the community participation of each node in all communities is fully characterized by a matrix $\mathbf{X}_t \in \mathbb{R}^{N_t \times K_t}$. In this technique, the goal is to factorize the similarity matrix $\mathbf{W}_t$ in the form $\mathbf{X}_t\mathbf{\Lambda}_t\mathbf{X}_t^T$ (as mentioned in Section 2.1.6) ,while maintaining the smoothness of the matrix $\mathbf{X}_t$ with respect to its previous value $\mathbf{X}_{t-1}$. The snapshot cost and temporal

cost are both defined using Kullback-Leibler (KL) divergence as

$$\text{cost}_t = \alpha sc_t + (1-\alpha)tc_t\,. \tag{2.48}$$

where

$$sc_t = D(\mathbf{W}_t \| \mathbf{X}_t \mathbf{\Lambda}_t \mathbf{X}_t^T) \tag{2.49}$$

and

$$tc_t = D(\mathbf{X}_{t-1}\mathbf{\Lambda}_{t-1} \| \mathbf{X}_t \mathbf{\Lambda}_t)\,. \tag{2.50}$$

Function D denotes the KL divergence function and can be defined between two matrices $A = \{a_{ij}\}$ and $B = \{b_{ij}\}$ as

$$D(\mathbf{A}\|\mathbf{B}) = \sum_{i,j}(a_{ij}\log\frac{a_{ij}}{b_{ij}} - a_{ij} + b_{ij}) \tag{2.51}$$

The snapshot and temporal costs are high when two inputs of KL function do not fit well. The problem is then formulated as a maximum a posteriori (MAP) estimation and the cost function is minimized by expectation-maximization (EM) optimization. The FacetNet framework is followed by additional features including node removal/insertion and identifying the number of communities by choosing the partition which results in the maximum modularity. It can also capture the interactions of communities and their evolution by two analysis tools; *Community Net* and *Evolution Net*. Community Net provides a microscopic view of the communities, showing the graph of their interactions at each timestep, whereas Evolution Net depicts how communities evolve over time. These

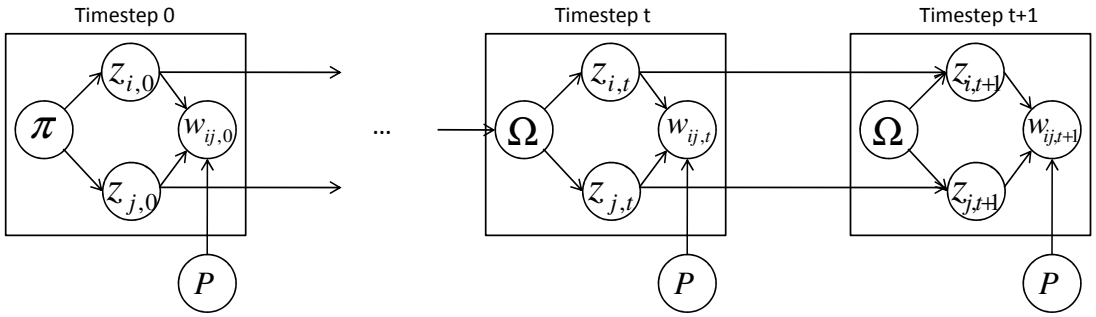


Fig. 2.4 Schematic illustration of the Dynamic Stochastic Block Model [5]. Adapted figure from [5].

two tools are simply expressed by terms depending on the latent variable.

In contrast to FacetNet [6,82], Yang et al. discover communities in the dynamic networks by finding hard community memberships for each node [5]. They extended the static stochastic block model to dynamic networks. The graphical representation of the Dynamic Stochastic Block Model (DSBM) is illustrated in Figure 2.4. The assigned community of node i is denoted by $z_{i,t} \in \{1, 2, ..., K\}$ and the community assignment matrix is represented by $\mathbf{Z}_t \in \{0, 1\}^{N \times K}$. The matrix $\Omega \in \mathbb{R}^{K \times K}$ is transition matrix, with elements indicating the probabilities that a node remains in the same community in the next timestep or transitions to a new community. The vector π of size K contains the initial probabilities π_k that a node belongs to the k-th community at timestep 0. The matrix $\mathbf{P} \in \mathbb{R}^{K \times K}$ consists of link generation probabilities; the diagonal entries are the probabilities of intra-community edges and off-diagonal entries indicate the probabilities of inter-community links. The key assumption is this model is that the community membership of node i at times t given its community at time $t-1$, is independent of all other community assignments (as shown in

Figure 2.4). Therefore the model can be written as

$$\mathbb{P}[\mathbf{W}_t, \mathbf{Z}_t | \mathbf{Z}_{t-1}, \mathbf{P}, \Omega] \;=\; \mathbb{P}[\mathbf{W}_t | \mathbf{Z}_t, \mathbf{P}] \mathbb{P}[\mathbf{Z}_t | \mathbf{Z}_{t-1}, \Omega]\,, \tag{2.52}$$

where the similarity matrix is produced by a Bernoulli distribution as

$$\mathbb{P}[\mathbf{W}_t | \mathbf{Z}_t, \mathbf{P}] \;=\; \prod_{i \sim j} \prod_{kl} \left(P_{kl}^{w_{ij,t}} (1 - P_{kl})^{1 - w_{ij,t}} \right)^{z_{ik,t} z_{jl,t}}. \tag{2.53}$$

The community assignments depend on Ω and $\mathbf{Z}_{t-1}$ as

$$\mathbb{P}[\mathbf{Z}_t | \mathbf{Z}_{t-1}, \Omega] \;=\; \prod_{i} \prod_{kl} \Omega_{kl}^{z_{ik,t-1} z_{il,t}}\,, \tag{2.54}$$

where the links are assumed to have binary weights (i.e., the graph is unweighted). Equation 2.54 holds for $t > 0$. Yang et al. employed Bayesian inference with a variational EM algorithm and Gibbs sampling to optimize the posterior probabilities. A distinctive feature of this method is the ability to discover the communities through either online or offline inference. The latter is of the importance when the entire data of all timesteps is available and one wants to infer the parameters not only based on the previous timestep but also using information from the future timesteps. The authors extended their model to handle weighted graphs and node insertion/removal, but their algorithm suffers from the requirement that the number of communities is known and remains constant over time.

The static hierarchical block model clustering method proposed in [83,84] was extended to dynamic networks by Park et al. in [16]. In the static hierarchical block model the nodes are iteratively divided into two left and right sub-groups based on a Bernoulli distribution. In this model, $r \in \mathcal{I}$ represents an internal node. The left subtree and right subtree of r are denoted by $L(r)$ and $R(r)$. The likelihood of a hierarchy D can be written by the count

of edges E_r and non-edges $\bar{E}_r$ and a probability θ_r as

$$L(D, \theta) \;=\; \prod_{r \in \mathcal{I}} \theta_r^{E_r} (1-\theta)^{\bar{E}_r} \,, \tag{2.55}$$

The hierarchy that best describes the data, together with a set θ parameters, is then inferred by MCMC sampling. Park et al. extended this technique by penalizing the hierarchy changes over time to ensure temporal smoothness of the communities. They also argue that MCMC method does not perform well for the optimization of 2.55 and instead used a variational method and mean-field approximation. In their technique, the hierarchical tree is pruned after a pre-defined height. Hence the number of communities is assumed to be known and fixed over time. Like the static model, this model also can only support unweighted graphs.

Smooth Evolution in Other Fields

Smooth evolution in the field of text stream mining has been extensively studied under the name of *dynamic evolutionary topic modelling* [85–87]. Many methods using probabilistic models have been proposed; these attempt to extract the best topic models in each timestamp while satisfying constraints of temporal smoothness. Blei and Lafferty developed in [86] a topic model wherein the parameters of multinomial distributions represent the topics. These evolve with a Gaussian noise to ensure temporal smoothness. The approximate posterior inference over the latent topics is then carried out by variational approximations based on Kalman filters and non-parametric wavelet regression [86]. Mei and Zhai [85] extracted latent patterns from documents and introduced an evolution graph of themes for temporal text mining. An Infinite dynamic topic model is proposed in [87]. In this method, Ahmed and Xing suggest a non-parametric technique using a hierarchical Dirichlet process.

There has been recent interest in the development of clustering algorithms using Dirichlet processes. Xu et al. discover the communities of data points distributed in a Euclidean space by proposing evolutionary Dirichlet process-based models [88, 89].

Inspired by Latent Dirichlet Allocation (LDA) Model [90], some researchers attempts to avoid time-discrete approaches [91–93] and develop rather time-continuous methods to handle dynamic topic models. They argue that although time-continuous models introduce more latent variables than time-discrete ones, it makes it possible to fit models at varying time resolution. This might be helpful for analyzing datasets which have some unobserved data periods. In paper [91], Wang et al. proposed continuous-time topics over time (TOT) model. The TOT model treats the time stamps as observations of the latent topics. It assumes that the topics themselves are constant, and the time information is used just to better discover them. Unlike the TOT model, the continuous-time topics model proposed by Wang et al. [92] can analyze and infer evolving topics over continuous time. This model replaces Dynamic Topic Model [86] by its continuous generalization; Brownian motion.

In this chapter we only discuss algorithms that exist for the clustering of dynamic *homogeneous networks*. Such networks can only model one element type and are unable to model networks containing multiple types of objects. Communities in *heterogeneous network* are allowed to have several types of objects. An example is a DataBase systems and Logic Programming (DBLP) network wherein clusters are research domains, which consist of researchers, keywords and venues [94,95]. In [94], a multi-mode network is defined as a network with multiple types of nodes and various types of interactions between them. The authors modelled the interaction between actors as a type of block model and then utilized matrix approximation to minimize the distance between the model and the actual multi-mode network. The model captures continuity of communities by a regularization term. Sun et al. [95] also proposed a Dirichlet process mixture model for clustering in

heterogeneous networks with a focus on star-shaped networks. The FacetNet framework explained in Section 2.2.4 was extended to social networks with several types of contents and various modes of actions such as uploading, tagging, and leaving comments [96]. Lin et al. in [96] represented all these types of content and actions in an extended notion of a graph, called a *multi-relational hypergraph* or a *metagraph*. Data in metagraphs are modelled by multiple tensors and similar to FacetNet, a tensor factorization algorithm is introduced in [96] to identify communities.

2.3 Summary

In this chapter we first reviewed the primary schemes of static clustering. We introduced the most widely-adopted clustering quality objectives. Some optimization approaches such as spectral clustering analysis and hierarchical clustering were briefly explained. We also dedicated a section to fuzzy (soft) clustering algorithms where clique-based methods, fitness-based techniques, algorithms involving matrix factorization and some other recent advances were discussed.

We then described incremental clustering techniques, one of the earliest families of methods for community detection in dynamic networks. As pointed out, these methods perform static clustering at each timestep, then apply a community matching technique over time rather than analysing community evolution in a unified framework. They do not consider the continuity of datasets over time and therefore their resulting communities are unstable and subject to unexplainable short term variations. Moreover, when applied to noisy datasets, they are not only unable to mitigate the effect of noise, but also magnify it by identifying communities with high temporal variations.

We also provided an overview of evolutionary clustering algorithms. We showed how

these methods ensure the continuity of communities over time. The early methods in this category mostly optimize a cost function that includes a regularization term that penalizes the temporal variations of communities [19, 55, 80]. Building a multislice graph and then applying static clustering using a generalized quality function is an alternative approach [3]. There are also some algorithms that introduce temporal smoothness by smoothing the weights, rather than the communities [75, 78]. Recently, probabilistic modelling of the dynamic structures has become popular. In these methods communities are described by latent variables which are determined by parameter estimation.

There are various challenges and constraints in the evolutionary methods. It is important to know the characteristics, limits and drawbacks of these algorithms and carefully select the most suitable one depending on the network of interest. For example some of the techniques cannot handle node insertion/removal, and others assume that there is a known or fixed number of communities over time in the network. All of them, except for the AFFECT algorithm [78], require at least one parameter to be set beforehand. Choosing a suitable value can be challenging when we do not have much information about the dataset. Also, there are algorithms that cannot handle weighted graphs. None of the algorithms discussed above, expect for FacetNet [6], are able to detect soft (overlapping) clusters.

Chapter 3

Dynamic Network Clustering using Non-negative Matrix Factorization

3.1 Overview

In this chapter, we propose an evolutionary framework for clustering dynamic networks and graphs using non-negative matrix factorization. We introduce a probabilistic generative model called Dynamic Non-negative Matrix Factorization (DNMF) capable of detecting overlapping communities in dynamic weighted directed graphs as well as undirected ones. This technique efficiently incorporates historic clustering results from the previous timestep into the current timestep and therefore can be considered as an evolutionary technique. Moreover, in Section 3.6 we use Automatic Relevance Determination [21, 45] to estimate effective number of communities in the case that we do not have any information about it beforehand. In Section 3.7 we propose a procedure to automatically adjust the historic data preference parameter introduced in Section 3.5.

Non-negative Matrix Factorization (NMF) is a widely used technique for extracting

components of non-negative data. It satisfies two features of clustering simultaneously: finding highly connected parts and detecting overlapping communities. In this procedure, the goal is to factorize the network data into non-negative components (i.e., clusters). This goal is achieved by considering a probability distribution depending on the describing parameters of components over the network data. Then we strive to maximize the log-posterior of these parameters given the observations and infer the unknown clustering parameters. Although we have not been able to develop any guarantees concerning the accuracy, the algorithm yields excellent performance on a variety of examples, as illustrated in Chapter 4.

3.2 Preliminaries and Notation

In the following we clarify the basic terms in the context of dynamic networks and the clustering task in dynamic graphs. A dynamic graph $\mathcal{G} = (\mathcal{G}_0, \mathcal{G}_1, ..., \mathcal{G}_T)$ is a sequence of graphs, with $\mathcal{G}_t = (V_t, E_t, \mathbf{W}_t)$ being the state of dynamic graph at time step t with $N_t = |V_t|$ nodes. $\mathbf{W}_t$ is a matrix that identifies the count weight associated with each link of the graph. In the network clustering context, $\mathbf{W}_t$ is our observation at timestep t and each element w_{ij} represents the similarity between i-th and j-th nodes. We assume that $w_{ij} > 0$ except in the case that there is no interaction between nodes i and j, in which case $w_{ij} = 0$. Self loops are also allowed. Hence, the network is represented by its similarity matrix $\mathbf{W} \in \mathbb{Z}^{N \times N}$.

In the case of soft clustering, the goal is to identify a matrix $\mathbf{G}_t \in \mathbb{R}^{N \times K}$, where each element $g_{ik,t} \in [0, 1],\ i = 1, ..., N,\ k = 1, ..., K$ represents the degree to which the i-th node belongs to the k-th community at timestep t. From the soft clustering definition, we can simply form a hard partitioning of nodes by assigning each node to the cluster with

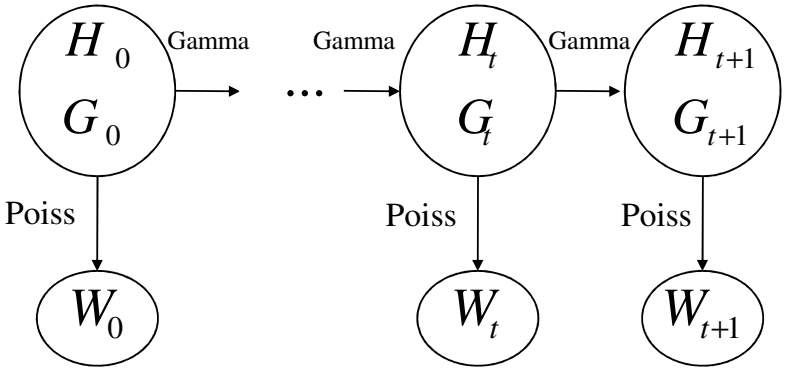


Fig. 3.1 Schematic illustration of the model. $\mathbf{W}_t$s are similarity matrices observed at each timestep. $\mathbf{H}_t$ and $\mathbf{G}_t$ are two non-negative matrices resulting from factorization of $\widehat{\mathbf{W}}_t$. Poiss is an abbreviation for the Poisson Distribution. Each arrow points to the random variables derived from a probability distribution with the parameters taken from the other side of the arrow.

the maximum g_{ij} among $j = 1, ..., K$. Mathematically, nodes are grouped into K clusters $\mathcal{C}_t : V \to \{C_{1,t}, C_{2,t}, ..., C_{K,t}\}$ where $C_{j,t}$ is defined by

$$C_{j,t} \quad = \quad \{i | g_{ij,t} = \max_k g_{ik,t}\} .$$

In Sections 3.3, 3.4, and 3.5 we assume there is a fixed number of communities K in the network and later in Section 3.6 we address the model selection problem using Automatic Relevance Determination. Note that the similarity matrix $\mathbf{W}_t$ can be symmetric or asymmetric. If $\mathbf{W}_t$ is asymmetric, then the graph is directed, i.e. the rate that node i interacts individual j is not the same as vice versa. Hence a single matrix $\mathbf{G}_t$ is not enough to fully characterize soft clusters for directed graphs, one requires another matrix $\mathbf{H}_t \in \mathbb{R}^{K \times N}$ to be able to model directed links.

3.3 Block Model

Consider the first order Markov generative model depicted in Figure 3.1. Our goal is to assign nodes to communities using latent variables with values representing to what extent each node is a member of hidden communities of the network. By considering an unobserved hidden network with similarity matrix $\widehat{\mathbf{W}}_t$, we can describe our model by $\hat{w}_{ij,t}$, the expected number of interactions that occurs between i and j. The hidden model is the product of two non-negative matrices $\mathbf{G}_t \in \mathbb{R}^{N\times K}$ and $\mathbf{H}_t \in \mathbb{R}^{K\times N}$. For the observation model we assume that w_{ij} is drawn from a Poisson distribution, as follows

$$\mathbb{P}[\mathbf{w}_{ij,t} = w_{ij,t}] \;=\; \frac{e^{-\hat{w}_{ij,t}}}{w_{ij,t}!}\hat{w}_{ij,t}^{w_{ij,t}}\,, \tag{3.1}$$

where

$$\hat{w}_{ij,t} \;=\; \sum_{k=1}^{K} g_{ik,t}h_{kj,t}\,. \tag{3.2}$$

Equation (3.1) implies that

$$\mathbb{E}[w_{ij,t}] \;=\; \hat{w}_{ij,t}\,. \tag{3.3}$$

If we scale the elements of matrices $\mathbf{G}$ and $\mathbf{H}$ such that $\sum_i g_{ik} = 1$ and $\sum_j h_{kj} = 1$, we can interpret g_{ik} as the probability of participation of node i in community k and h_{ki} as the probability of including node i in activities of k-th community. For undirected networks g_{ik} and h_{ki} are equal. Therefore these matrices fully describe the overlapping structure of the network. Adopting a Poisson distribution on the static part of our model has several interpretations: 1) it expresses the probability of a given number of interactions occurring

in a fixed interval of time given the average interaction rate $\widehat{\mathbf{W}}_t$ which we strive to find. 2) As we show in Section 3.4, maximizing the aposteriori probability of Poisson distribution is equivalent to minimizing the KL-divergence function and therefore has a information theoretic interpretation.

The dynamic part of the model consists of blocks within the matrices $\mathbf{G}$ and $\mathbf{H}$ being updated at each timestep with respect to the current observation and communities identified at previous timesteps. This is what evolutionary clustering is supposed to perform: learning the model through the notion of continuity over time. The new latent variable takes into account not only the changes in the observed similarity matrix but also previous communities. As proposed in [6], we adopt a Gamma distribution for the transition model as follows

$$\mathbb{P}[g_{ik,t}] = \frac{\mu^{\gamma_{ik,t}}}{\Gamma(\gamma_{ik,t})} g_{ik,t}^{\gamma_{ik,t}-1} e^{-\mu g_{ik,t}}, \tag{3.4}$$

where

$$\gamma_{ik,t} = \mu g_{ik,t} + 1 \tag{3.5}$$

and similarly for $\mathbf{H}_t$

$$\mathbb{P}[h_{ki,t}] = \frac{\mu^{\gamma'_{ki,t}}}{\Gamma(\gamma'_{ki,t})} h_{ki,t}^{\gamma'_{ki,t}-1} e^{-\mu h_{ki,t}}, \tag{3.6}$$

where

$$\gamma'_{ki,t} = \mu h_{ki,t} + 1\,. \tag{3.7}$$

The parameter μ in Equations (3.4) and (3.6) is responsible for the amount of emphasis on the historic data. For $\mu \longrightarrow \infty$ the emphasis on the historic data is maximized and for $\mu = 0$ only the current data is taken into consideration.

One of the advantages of adopting the Gamma distribution to describe the dynamic part of the model is that it is a conjugate prior of the Poisson distribution. This makes our formulation easier. Furthermore, in Section 3.4 we show that adopting the Gamma distribution has an information theoretic interpretation.

Having defined all distributions and their relationships shown in 3.1, we can now formulate the problem for inference. We assumed that our observation matrix $\mathbf{W}_t$ is a random matrix of variables derived from a hidden network that smoothly changes over time. In Section 3.4, our goal is to find the factorization of this hidden network $\mathbf{G}_t$ and $\mathbf{H}_t$ using maximum a-posteriori (MAP) estimation.

3.4 MAP Estimation

In this section, we formulate the NMF problem by striving to identify the maximum a posteriori (MAP) estimate at each timestep. At timestep $t = 0$, as depicted in Figure 3.1, there is no prior clustering result and therefore we have the static clustering problem which can be summarized as the need to identify suitable $\mathbf{G}_0$ and $\mathbf{H}_0$. By inferring $\mathbf{G}_0$ and $\mathbf{H}_0$, the clustering parameters of the subsequent timesteps are drawn only with respect to the clustering results of the previous timestep, not future ones. Therefore, our algorithm is able to iteratively identify clusters as soon as new network data arrives without depending on the future observations or clusters. This is a reason why DNMF is considered as an *on-line* dynamic network clustering technique. For inferring $\mathbf{G}_t$ and $\mathbf{H}_t$ at timestep $t > 0$,

we aim to maximize the posterior as follows

$$\mathbf{G}_t^*, \mathbf{H}_t^* = arg \max_{\mathbf{G}_t, \mathbf{H}_t} \mathbb{P}[\mathbf{G}_t, \mathbf{H}_t | \mathbf{G}_{t-1}, \mathbf{H}_{t-1}, \mathbf{W}_t] \,. \tag{3.8}$$

Using the chain rule and Bayes' theorem, we define the log posterior function as

$$\log L = \log \mathbb{P}[\mathbf{W}_t | \mathbf{G}_t, \mathbf{H}_t] + \log \mathbb{P}[\mathbf{G}_t | \mathbf{G}_{t-1}] + \log \mathbb{P}[\mathbf{H}_t | \mathbf{H}_{t-1}] \,, \tag{3.9}$$

where the first term is log likelihood of our data, derived from the probability $\mathbb{P}[\mathbf{W}_t|\mathbf{G}_t, \mathbf{H}_t] = \mathbb{P}[\mathbf{W}_t|\widehat{\mathbf{W}_t}]$ of observing every interaction w_{ij_t} given a Poisson parameter of $\hat{w}_{ij,t}$. Hence we derive

$$\log \mathbb{P}[\mathbf{W}_t | \widehat{\mathbf{W}}_t] = \sum_{ij} \log \frac{e^{-\hat{w}_{ij,t}}}{w_{ij,t}!} \hat{w}_{ij,t}^{w_{ij,t}} \tag{3.10}$$

$$= \sum_{ij} w_{ij,t} \log \hat{w}_{ij,t} - \hat{w}_{ij,t} + c \,, \tag{3.11}$$

where c is constant. Note that $w_{ij,t}$ is what we observed and hence is considered as constant. It does not have any influence on maximization. The maximization of the Equation (3.11) corresponds to minimization of the Kullback-Leiber (KL) divergence

$$D_{KL}(\mathbf{W}_t \| \widehat{\mathbf{W}}_t) = \sum_{ij} w_{ij,t} \log \left(\frac{w_{ij,t}}{\hat{w}_{ij,t}} \right) - w_{ij,t} + \hat{w}_{ij,t} \tag{3.12}$$

$$= \sum_{ij} -w_{ij,t} \log \hat{w}_{ik,t} + \hat{w}_{ij,t} + c \,. \tag{3.13}$$

The KL-divergence $D(A\|B)$ also indicates the information gain if we use the precise distribution A instead of the approximate model of B. In this context, the higher information gain suggests a larger error introduced by using the approximate model $\widehat{\mathbf{W}}_t$. We want

this error to be minimized.

We expand the second and third terms of Equation (3.9) as

$$\log \mathbb{P}[\mathbf{G}_t \| \mathbf{G}_{t-1}] = \sum_k \log \left(\frac{\mu^{\gamma_{ik,t}}}{\Gamma(\gamma_{ik,t})} g_{ik,t}^{\gamma_{ik,t}-1} e^{-\mu g_{ik,t}} \right) \tag{3.14}$$

$$= \mu \sum_{ik} (g_{ik,t-1} \log g_{ik,t} - g_{ik,t}) + c \tag{3.15}$$

$$\log \mathbb{P}[\mathbf{H}_t | \mathbf{H}_{t-1}] = \sum_k \log \left(\frac{\mu^{\gamma'_{ki,t}}}{\Gamma(\gamma'_{ki,t})} h_{ki,t}^{\gamma'_{ki,t}-1} e^{-\mu h_{ki,t}} \right) \tag{3.16}$$

$$= \mu \sum_{ki} (h_{ki,t-1} \log h_{ki,t} - h_{ki,t}) + c\,, \tag{3.17}$$

where $\gamma_{ik,t}$ and $\gamma'_{ki,t}$ are the parameters of Gamma distribution, defined in Equations (3.5) and (3.7) previously. The Gamma functions in the denominator of the equations only depend on the parameters of timestep $t-1$ which already have been inferred and hence they are considered constant at time t. Note that Equations (3.14) and (3.16) hold for $t > 0$.

The maximization of the Equations (3.15) and (3.17) with respect to the matrices $\mathbf{G}_t$ and $\mathbf{H}_t$ corresponds to minimizing the Kullback-Leiber (KL) divergence

$$D_{KL}(\mathbf{G}_{t-1} | \mathbf{G}_t) = \sum_{ik} g_{ik,t-1} \log \left(\frac{g_{ik,t-1}}{g_{ik,t}} \right) - g_{ik,t-1} + g_{ik,t} \tag{3.18}$$

$$= \sum_{ik} -g_{ik,t-1} \log g_{ik,t} + g_{ik,t} + c\,. \tag{3.19}$$

The same equations can be written for $\mathbf{H}_t$. Since we consider $\mathbf{G}_{t-1}$ as constant at timestep t, the Equation (3.18) can be written as (3.19) which is the negative of (3.15). This implies that when we attempt to optimize the log posterior function, we are actually considering KL-divergence objective function.

Putting Equations (3.11), (3.15), and (3.17) together, the log posterior function is equal

to

$$\begin{aligned}\log L &= \sum_{ij}(w_{ij,t}\log \hat{w}_{ij,t} - \hat{w}_{ij,t}) + \mu\sum_{ik}(g_{ik,t-1}\log g_{ik,t} - g_{ik,t}) \\ &\quad +\mu\sum_{ki}(h_{ki,t-1}\log h_{ki,t} - h_{ki,t}) + c\end{aligned} \tag{3.20}$$

for $t > 0$. At $t = 0$ only the first term of Equation (3.20) appears in the log posterior function.

3.5 Parameter Inference

In this section, we solve the optimization problem defined by (3.20) using coordinate descent update rules to infer $\mathbf{G}$ and $\mathbf{H}$. This is the same procedure used in [44]. We first find the gradient of Equation (3.20) with respect to $\mathbf{G}$ and $\mathbf{H}$:

$$\nabla_{\mathbf{G}}\log L = \Big(\frac{\mathbf{W}-\mathbf{GH}}{\mathbf{GH}}\Big)\mathbf{H^T} + \mu\frac{\mathbf{G_{t-1}}-\mathbf{G}}{\mathbf{G}} \tag{3.21}$$

$$\nabla_{\mathbf{H}}\log L = \mathbf{G^T}\Big(\frac{\mathbf{W}-\mathbf{GH}}{\mathbf{GH}}\Big) + \mu\frac{\mathbf{H_{t-1}}-\mathbf{H}}{\mathbf{H}}\,. \tag{3.22}$$

Here $\frac{X}{Y}$ represents element by element division. In multiplicative coordinate descent we simply update elements of $\mathbf{G}$ and $\mathbf{H}$ by multiplying its current value with the ratio of the positive to negative parts of the gradient of log posterior function

$$\mathbf{G} \leftarrow \mathbf{G}.\frac{[\nabla_{\mathbf{G}}\log L]_+}{[\nabla_{\mathbf{G}}\log L]_-} \tag{3.23}$$

$$\mathbf{H} \leftarrow \mathbf{H}.\frac{[\nabla_{\mathbf{H}}\log L]_+}{[\nabla_{\mathbf{H}}\log L]_-}\,, \tag{3.24}$$

where $X.Y$ is element by element multiplication. As discussed in [97,98], the update rule in (3.23) maintains the non-negativity of $\mathbf{G}$. More precisely, $\mathbf{G}$ increases when $[\nabla_{\mathbf{G}} \log L]_+ > [\nabla_{\mathbf{G}} \log L]_-$ which occurs when $\nabla_{\mathbf{G}} \log L > 0$ and similarly decreases when $[\nabla_{\mathbf{G}} \log L]_+ < [\nabla_{\mathbf{G}} \log L]_-$ i.e. when $\nabla_{\mathbf{G}} \log L > 0$. There are two stationary solution points for this multiplicative update rule: one is $[\nabla_{\mathbf{G}} \log L]_+ = [\nabla_{\mathbf{G}} \log L]_-$ which is the same condition for obtaining local optima i.e. $\nabla_{\mathbf{G}} \log L = 0$ as in the conventional additive updates. The other point is $\mathbf{G} \to 0$; this solution yields sparsity in $\mathbf{G}$ that is desirable for many applications including our application. As we discuss later in Section 3.6, eliminating the weak and irrelevant elements of $\mathbf{G}$ and $\mathbf{H}$ helps us to identify communities that are required by the structure of network.

In Equations (3.23) and (3.24), if we replace μ by $\frac{1-\alpha}{\alpha}$, then α is a parameter between 0 to 1 which controls the amount of emphasis on historic data, for $\alpha = 0$ we only consider historic data. For $\alpha = 1$ only data at the current snapshot. Replacing μ by $\frac{1-\alpha}{\alpha}$ and rewriting Equations (3.23) and (3.24) using (3.21) and (3.22), we derive the following update rules

$$\mathbf{G} \leftarrow \frac{\alpha\Big[\mathbf{G}.\big(\frac{\mathbf{W}}{\mathbf{GH}}\mathbf{H}^T\big)\Big] + (1-\alpha)\mathbf{G}_{t-1}}{\alpha\mathbf{H^T} + (\mathbf{1}-\alpha)\mathbf{1_{N\times K}}} \tag{3.25}$$

$$\mathbf{H} \leftarrow \frac{\alpha\Big[\mathbf{H}.\big(\mathbf{G}^T\frac{\mathbf{W}}{\mathbf{GH}}\big)\Big] + (1-\alpha)\mathbf{H}_{t-1}}{\alpha\mathbf{G^T} + (\mathbf{1}-\alpha)\mathbf{1_{K\times N}}}\,. \tag{3.26}$$

The overall algorithm is summarized in Algorithm 1. We can either stop iterations after a pre-defined number n_{iter} or when the absolute sum of differences between the results of two consecutive iterations is less than a small constant ϵ. Here we stop iterations after n_{iter} iterations as in Algorithm 1. Note that the update rules constrain the columns of $\mathbf{G}$ and the rows of $\mathbf{H}$ to sum to unity.

Algorithm 1 Community Detection in Dynamic Networks

INPUT: Matrices of observation W_t, $\forall t$, constant α, and number of clusters K
OUTPUT: $\mathbf{G_t}$ and $\mathbf{H_t}$, $\forall t$

Initialize $\mathbf{G}$ and $\mathbf{H}$ randomly.
for $i = 1$ to n_{iter} **do**
$\mathbf{H} \leftarrow \left(\frac{\mathbf{H}}{\mathbf{G}^T}\right).\left[\mathbf{G}^T(\frac{\mathbf{W}}{\mathbf{GH}})\right]$
Scale h_{kj}s such that $\sum_j h_{kj} = 1$.
$\mathbf{G} \leftarrow \left(\frac{\mathbf{G}}{\mathbf{H}^T}\right).\left[(\frac{\mathbf{W}}{\mathbf{GH}})\mathbf{H}^T\right]$
Scale g_{ik}s such that $\sum_i g_{ik} = 1$
end for
$\mathbf{G_0} \leftarrow \mathbf{G}$ and $\mathbf{H_0} \leftarrow \mathbf{H}$
for $t = 1$ to T **do**
Initialize $\mathbf{G}$ and $\mathbf{H}$ randomly.
for $i = 1$ to n_{iter} **do**
$\mathbf{H} \leftarrow \frac{\alpha\left[\mathbf{H}.\left(\mathbf{G}^T\frac{\mathbf{W}}{\mathbf{GH}}\right)\right]+(1-\alpha)\mathbf{H}_{t-1}}{\alpha\mathbf{G^T}+(\mathbf{1}-\alpha)\mathbf{1_{K\times N}}}$
Scale h_{kj}s such that $\sum_j h_{kj} = 1$.
$\mathbf{G} \leftarrow \frac{\alpha\left[\mathbf{G}.\left(\frac{\mathbf{W}}{\mathbf{GH}}\mathbf{H}^T\right)\right]+(1-\alpha)\mathbf{G}_{t-1}}{\alpha\mathbf{H^T}+(\mathbf{1}-\alpha)\mathbf{1_{N\times K}}}$
Scale g_{ik}s such that $\sum_i g_{ik} = 1$
end for
$\mathbf{G_t} \leftarrow \mathbf{G}$ and $\mathbf{H_t} \leftarrow \mathbf{H}$
end for
return $\mathbf{G_t}$ and $\mathbf{H_t}$, $\forall t$

3.6 Model Order Selection

In this section, we describe how a technique proposed in [45] called *Automatic Relevance Determination* (ARD) can help us with the model order selection problem. This problem appears in real networks that we usually do not have any precise knowledge of about the effective number of communities K. We are interested to encode our network datasets by the $\mathbf{G}_t$ and $\mathbf{H}_t$ matrices. However, this encoding should determine the number of clusters effectively in order to extract meaningful communities. Tan et al. in [45] addressed this problem by formulating a Bayesian approach to obtain the effective dimensionality K_{eff}.

In previous clustering techniques for dynamic and static networks, a common approach to model order selection was to apply the clustering technique several times, varying the number of clusters. The number of clusters was identified by choosing the clustering that achieved the best performance in terms of a pre-defined metric. In [21], however, ARD was effectively used for static networks. Here we use the same technique for handling unknown number of clusters and also managing changing number of clusters over time.

Let $\boldsymbol{\beta}_t = \{\beta_{1,t}, .., \beta_{K,t}\}$ be parameters of a prior probability distribution placed on columns of $\mathbf{G}_t$ and rows of $\mathbf{H}_t$. Placing an appropriate distribution allow us to obtain a sparse yet accurate encoding of the data. $\boldsymbol{\beta}_t$ parameters are learnt from data in a Bayesian approach: as presented in [21] and illustrated in Figure 3.2, we place half-normal priors over each column of $\mathbf{G}_t$ and row of $\mathbf{H}_t$. Thus

$$\mathbb{P}[\mathbf{G}_t|\mathbf{G}_{t-1}, \boldsymbol{\beta}_t] = \text{Gamma}(\mu\mathbf{G}_{t-1} + I, \mu^{-1})\mathcal{HN}(0, \boldsymbol{\beta}_t^{-1}) \tag{3.27}$$

$$\mathbb{P}[\mathbf{H}_t|\mathbf{H}_{t-1}, \boldsymbol{\beta}_t] = \text{Gamma}(\mu\mathbf{H}_{t-1} + I, \mu^{-1})\mathcal{HN}(0, \boldsymbol{\beta}_t^{-1})\,, \tag{3.28}$$

where

$$\mathcal{HN}(x|0, \sigma^2) \;=\; \frac{1}{\sigma}\sqrt{\frac{2}{\pi}}\exp(\frac{x^2}{2\sigma^2})\,. \tag{3.29}$$

Rewriting Equations (3.27) and (3.28) using half-normal pdf, we obtain log priors as

$$\log \mathbb{P}[\mathbf{G}_t|\mathbf{G}_{t-1}, \boldsymbol{\beta}_t] \;=\; \sum_{ik} -g_{ik,t-1}\log g_{ik,t} + g_{ik,t} - \frac{1}{2}\beta_{k,t}g_{ik,t}^2 - \frac{N}{2}\log\beta_{k,t} + c \tag{3.30}$$

$$\log \mathbb{P}[\mathbf{H}_t|\mathbf{H}_{t-1}, \boldsymbol{\beta}_t] \;=\; \sum_{ki} -h_{ki,t-1}\log h_{ki,t} + h_{ki,t} - \frac{1}{2}\beta_{k,t}h_{ki,t}^2 - \frac{N}{2}\log\beta_{k,t} + c\,. \tag{3.31}$$

In this setting we start with a large K (the maximum K is the number of nodes N in the

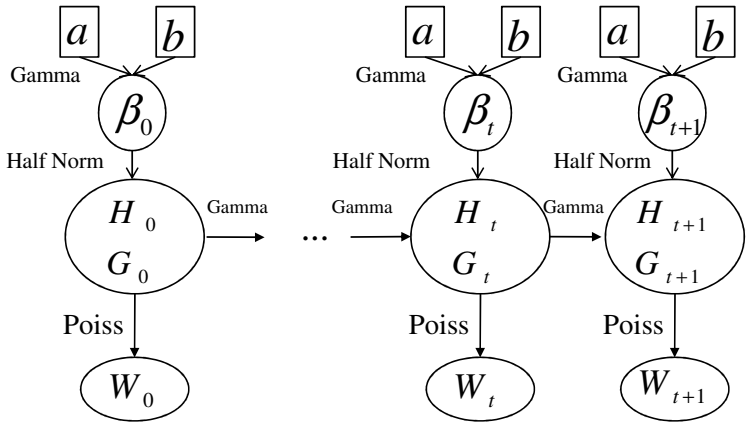


Fig. 3.2 Schematic illustration of the model with ARD. $\boldsymbol{\beta}_t$ is a prior Half-Normal probability distribution placed on columns of $\mathbf{G}_t$ and rows of $\mathbf{H}_t$. $\boldsymbol{\beta}_t$ itself is drawn from a Gamma distribution with parameters a and b.

network) and the role of each $\beta_{k,t}$ is to indicate to what extent community k contributes in the representation of the network at timestep t. If we do not use ARD in our model while starting from a large number of communities K, we will finally end up with a large number of non-empty yet very weak communities. What ARD does here is to efficiently penalize the objective log posterior function and make irrelevant columns of $\mathbf{G}_t$ and rows of $\mathbf{H}_t$ close to zero. By thresholding, we can obtain a sparse representation of the network with effective number of clusters K_{eff}. As $\beta_{k,t}$ becomes larger, the more irrelevant the k-th community becomes and the corresponding columns of $\mathbf{G}_t$ and rows of $\mathbf{H}_t$ approach zero. More precisely, large values of $\beta_{k,t}$ indicate that the expected values of the elements of the corresponding $\mathbf{G}_t$ and $\mathbf{H}_t$ are close to zero. This suggests that the k-th cluster is not a genuine cluster and should be eliminated from the representation. The $\beta_{k,t}$ parameters themselves are drawn from from a Gamma distribution with fixed hyperparameters a and

b. The Gamma distribution is a conjugate prior of the half-normal distribution. Therefore

$$\mathbb{P}[\beta_{k,t}|a,b] \;=\; \frac{b^a}{\Gamma(a)}\beta_{k,t}^{a-1}\exp(-\beta_{k,t}b) \tag{3.32}$$

and the log prior is

$$\log\mathbb{P}[\boldsymbol{\beta}_t] \;=\; \sum_k (a-1)\log\beta_{k,t} - b\beta_{k,t} + c\,. \tag{3.33}$$

The posterior we want to optimize can be written as

$$\begin{aligned}\log L \;=\;& \log\mathbb{P}[\mathbf{W}_t|\mathbf{G}_t,\mathbf{H}_t] + \log\mathbb{P}[\mathbf{G}_t|\mathbf{G}_{t-1},\boldsymbol{\beta}_t] + \log\mathbb{P}[\mathbf{H}_t|\mathbf{H}_{t-1},\boldsymbol{\beta}_t] \\ &+ \log\mathbb{P}[\boldsymbol{\beta}_t] + c\,,\end{aligned} \tag{3.34}$$

where the first three terms are what we had before in Equation (3.9) and the others are added in the ARD framework. By expanding the terms in (3.34) using Equations (3.30), (3.31), and (3.33), the overall log posterior function can be written as

$$\begin{aligned}\log L \;=\;& \sum_{ij} w_{ij,t}\log\hat{w}_{ij,t} - \hat{w}_{ij,t} + \sum_{ik}(\mu g_{ik,t-1}-1)\log g_{ik,t} + \sum_{ki}(\mu h_{ki,t-1}-1)\log h_{ki,t} \\ &- \sum_k \Big[\frac{1}{2}\Big(\sum_{ij} g_{ik,t}^2 + h_{kj,t}^2 + 2b\Big)\beta_{k,t} + (N-(a-1))\log\beta_{k,t}\Big]\,,\end{aligned} \tag{3.35}$$

for $t>0$. The parameter inference is similar to Section 3.5; we take the gradient of (3.35) with respect to $\mathbf{G}_t$, $\mathbf{H}_t$, and $\boldsymbol{\beta}_t$ and then follow the update rules (3.23) and (3.24). The overall algorithm is shown in Algorithm 2.

Algorithm 2 Community Detection in Dynamic Networks with ARD

INPUT: Matrices of observation W_t, $\forall t$, constant α, and number of clusters K
OUTPUT: $\mathbf{G_t}$ and $\mathbf{H_t}$, $\forall t$

Initialize $\mathbf{G}$ and $\mathbf{H}$ randomly.
for $i = 1$ to n_{iter} **do**
$\mathbf{H} \leftarrow \left(\frac{\mathbf{H}}{\mathbf{G}^T + diag(\beta)H}\right).\left[\mathbf{G}^T(\frac{\mathbf{W}}{\mathbf{GH}})\right]$
Scale h_{kj}s such that $\sum_j h_{kj} = 1$.
$\mathbf{G} \leftarrow \left(\frac{\mathbf{G}}{\mathbf{H}^T + Gdiag(\beta)}\right).\left[(\frac{\mathbf{W}}{\mathbf{GH}})\mathbf{H}^T\right]$
Scale g_{ik}s such that $\sum_i g_{ik} = 1$
$\beta_k \leftarrow \frac{2(N+a-1)}{(\sum_{ij} g_{ik}^2 + h_{kj}^2)+b}$
end for
$\mathbf{G_0} \leftarrow \mathbf{G}$ and $\mathbf{H_0} \leftarrow \mathbf{H}$
for $t = 1$ to T **do**
Initialize $\mathbf{G}$ and $\mathbf{H}$ randomly.
for $i = 1$ to n_{iter} **do**
$\mathbf{H} \leftarrow \frac{\alpha\left[\mathbf{H}.\left(\mathbf{G}^T\frac{\mathbf{W}}{\mathbf{GH}}\right)\right] + (1-\alpha)\mathbf{H}_{t-1}}{\alpha(\mathbf{G^T} + \mathbf{diag}(\beta)\mathbf{H}) + (\mathbf{1}-\alpha)\mathbf{1_{K\times N}}}$
Scale h_{kj}s such that $\sum_j h_{kj} = 1$.
$\mathbf{G} \leftarrow \frac{\alpha\left[\mathbf{G}.\left(\frac{\mathbf{W}}{\mathbf{GH}}\mathbf{H}^T\right)\right] + (1-\alpha)\mathbf{G}_{t-1}}{\alpha(\mathbf{H^T} + \mathbf{G}\mathbf{diag}(\beta)) + (\mathbf{1}-\alpha)\mathbf{1_{N\times K}}}$
Scale g_{ik}s such that $\sum_i g_{ik} = 1$
$\beta_k \leftarrow \frac{2(N+a-1)}{(\sum_{ij} g_{ik}^2 + h_{kj}^2)+b}$
end for
$\mathbf{G_t} \leftarrow \mathbf{G}$ and $\mathbf{H_t} \leftarrow \mathbf{H}$
end for
return $\mathbf{G_t}$ and $\mathbf{H_t}$, $\forall t$

3.7 Dynamic Setting of α

The parameter α as defined previously in Section 3.5 is responsible for controlling the amount of emphasis one would like to put on historic data or current data. For $\alpha = 0$ we only consider historic data and for $\alpha = 1$ only data at the current snapshot. However, it seems reasonable if one wants to adapt α based on short term variations observed in data over time. If the current data dramatically deviates from data in previous timestep, then

the effect of the data of the current timestep should increase and the influence of the historic dataset should be suppressed. This problem has not been addressed in most previous work except in [7] and [99]. The techniques in [7] smooths the similarity matrix with an adaptive factor α obtained from shrinkage estimation and then applies static spectral clustering on the filtered similarity matrix. However, using the smoothed similarity matrix does not always translate to continuity in community membership. Our method is similar to the data clustering technique [99] in the manner that it also has a general preference parameter α_0 and employs an exponential function to measure deviation over time. The dynamic evaluation of α is as follows

$$\alpha = \alpha_0 \exp\Big(\frac{\|\mathbf{W}_t - \mathbf{W}_{t-1}\|_F^2}{\|\mathbf{W}_{t-1}\|_F^2}\Big), \tag{3.36}$$

where

$$\|\mathbf{W}_t - \mathbf{W}_{t-1}\|_F^2 = \sum_{ij}(w_{ij,t} - w_{ij,t-1})^2 \tag{3.37}$$

is the Forbenius norm of the difference between the similarity matrices at time t and time $t-1$. $\alpha_0 \in [0,1]$ is the general preference of historic data. We conducted empirical studies to assess the impact of varying the values of α_0. I report on these studies and suggest methods for choosing appropriate values for this parameter in the next chapter.

3.8 Node Insertion and Removal

A practical challenge that arises in dynamic networks is handling the appearance and disappearance of nodes over time. The node removal task at each timestep can be easily done by removing the corresponding row and column of the inactive node from matrices

$\mathbf{G}_{t-1}$ and $\mathbf{H}_{t-1}$. Node insertion also can be managed in a similar way but we need to note that since the matrices $\mathbf{G}_{t-1}$ and $\mathbf{H}_{t-1}$ do not have the same size as $\mathbf{G}_t$ and $\mathbf{H}_t$, a row and column consisting of zeros must be inserted in $\mathbf{G}_{t-1}$ and $\mathbf{H}_{t-1}$ in an appropriate place. The place of added row and column should be determined with respect to the label of appearing node.

3.9 Complexity of DNMF

The computational load is dominated by the matrix multiplication $\mathbf{GH}$ that appears in the denominator of our update rule in both Algorithm 1 and Algorithm 2. The computational complexity of this multiplication is $\mathcal{O}(N^2K)$. However, this complexity is significantly reduced in practice due to the sparseness of the similarity matrix W: we do not need to calculate the dot product $\sum_k g_{ik}h_{kj}$ if the corresponding element of the similarity matrix $w_{ij} = 0$. Moreover, when working with undirected networks, we can take advantage of $\mathbf{GH}$ being symmetric and reduce the number of operations by half. Therefore the complexity can be mitigated if we carefully implement the algorithm. Updating $\mathbf{G}_t$ and $\mathbf{H}_t$ for n_{iter} and at all timesteps T requires the overall computational complexity $\mathcal{O}(n_{iter}N^2KT)$.

3.10 Summary

In this chapter we presented an efficient algorithm for clustering of dynamic networks. We employed a probabilistic dynamic model to obtain clusters not only based on the data available at the current timestep but also by considering the cluster parameters of previous timestep as transition parameters. The algorithm targets optimization of the log posterior of the observation in order to find the parameters of a factorized matrix and attempts to solve the optimization by an iterative multiplicative coordinate ascent procedure.

For selecting an appropriate number of clusters the algorithm employs Automatic Relevance Determination [45]. ARD places a prior distribution on the parameters used ub the non-negative matrix factorization, so that the algorithms favours a sparse representation of the data. This eliminates unnecessary clusters, leaving only those required to accurately capture the network structure. This procedure is very useful when no prior knowledge about the number of clusters is available. Another version of our algorithm without ARD can be employed when we do know number of communities beforehand.

Our algorithm is capable of detecting overlapping communities and performs well in directed networks as well as undirected ones in many practical settings. Furthermore, our algorithm is able to change how much emphasis is places on historic data.

Chapter 4

Experimental Results

4.1 Synthetic Network

In this section we analyse the performance on synthetic dynamic graphs, for which the ground truth is known. We used two synthetic datasets: **SynFix** and **SynVar**. The former mainly simulates changing community membership and the latter models changes in the number of communities. The performance of DNMF is studied for different parameter settings to examine how these affect the behaviour of the algorithm and whether the algorithm is extremely sensitive to their values. We also analyse the DNMF runtime and performance in comparison to other state-of-the-art algorithms.

4.1.1 Network Description

SynFix: Capturing membership variations

We use the synthetic network proposed by Lin et al. [6]. It is a dynamic version of the Girvan-Newman network [13]. The main goal of this synthetic dataset is to model node membership changes. It contains N nodes, divided into four communities of equal size,

and generates data for T consecutive time steps. We call this synthetic dataset **SynFix** since the number of communities does not change over time. At each timestep from 2 to T, dynamics are introduced such that from each community x members are randomly selected to leave their original community and to join randomly the other three communities. Edges are added randomly at each timestep with a higher probability p_{in} for within-community edges and a lower probability p_{out} for between-community edges. The two parameters p_{in} and p_{out} can be converted into two other parameters z_{in} and z_{out} as follows,

$$\mathbb{E}[z] = \mathbb{E}[z_{in} + z_{out}] \tag{4.1}$$

$$\bar{z} = \bar{z}_{in} + \bar{z}_{out}\,, \tag{4.2}$$

where z is the degree of each node, z_{in} is the degree of within-community edges and z_{out} is the degree of between-community edges. Both z_{in} and z_{out} have a binomial distribution,

$$z_{in} \sim B(p_{in}, |C| - 1) \tag{4.3}$$

$$z_{out} \sim B(p_{out}, 3\ |C|)\,, \tag{4.4}$$

where $|C|$ is the size of communities. For each binomial distribution we know $\mathbb{E}[z] = np$ and therefore

$$\bar{z} = p_{in}(|C| - 1) + p_{out}(3\ |C|)\,. \tag{4.5}$$

.

In this dataset, we assume the average degree for the nodes is set to constant ($\bar{z} = z$).

Hence the parameters p_{in} and p_{out} can be written using the parameters z_{in} and z_{out} as

$$p_{out} = \frac{\bar{z}_{out}}{3|C|} \tag{4.6}$$

$$p_{in} = \frac{z - \bar{z}_{out}}{|C| - 1} \tag{4.7}$$

As a result, a single parameter $\bar{z}_{out}$, which represents the mean number of edges from a node to nodes in other communities, is enough to describe the model. As $\bar{z}_{out}$ increases, more connections are created between nodes in different clusters and the clusters become difficult to distinguish.

SynVar: Capturing the number of communities variations

SynVar is a similar synthetic dataset to **SynFix** with $\bar{z}$ set to a constant and a parameter $\bar{z}_{out}$ that controls the density of edges between clusters. However, at each timestep dynamics are introduced such that the number of communities increases by one. Depending on the size of communities at the current timestep a fraction of nodes from each community is randomly chosen to leave their original community and form a new one such that the size of all clusters remains approximately the same. As a result, at each timestep the communities are related to the previous communities and hence the temporal smoothness is ensured in the **SynVar** networks.

4.1.2 Assessing the Distance of Clustering

Many measures have been introduced to assess the similarity between two partitions of a graph. These metrics receive two partitions of a network as input and map them into the interval $[0, 1]$. For most metrics, if the output is zero, two partitions are completely different; if the output is equal to one, the two partitions are exactly the same. These

metrics are in general based on cluster overlap, pair-counting, or entropy. They are also helpful when the ground truth is known and one is interested to measure how close the identified clusters are to the ground truth. The most widely-adopted metrics include the Rand Index [100], the Jaccard Index [101], and Normalized Mutual Information [102].

In this chapter we use Normalized Mutual Information (NMI) as a distance measure. As previously mentioned in Equation (2.45), the NMI for two partitions $\mathcal{C}$ and $\mathcal{C}'$ of a network is defined as

$$NMI(\mathcal{C},\mathcal{C}') = \frac{-2\sum_{i=1}^{K}\sum_{j=1}^{K'} m_{ij}\log(m_{ij}N/m_{i\cdot}m_{\cdot j})}{\sum_{i=1}^{K} m_{i\cdot}\log(m_{i\cdot}/N)+\sum_{j=1}^{K'} m_{\cdot j}\log(m_{\cdot j}/N)}, \tag{4.8}$$

where $M=\{m_{ij}\}$ is the number of nodes that communities C_j and C'_i share. Let $m_{i\cdot}$ and $m_{\cdot j}$ denote the sum of the elements of the confusion matrix $\mathbf{M}$ over the i-th row and j-th column respectively. The value of NMI varies between 0 and 1 and the higher the NMI becomes, the more similar the two partitions are.

4.1.3 The Effect of Changing a and b

In this section the goal is to determine how changing the value of parameters a and b (introduced in Section 3.6) affects the performance of our method. We applied DNMF [1] on **SynFix** datasets with the parameters shown in Table 4.1.

To clearly reveal the role of parameters a and b, we set the other parameters of DNMF to constant values. These parameters are shown in Table 4.2. Since the parameters a and b are responsible for controlling the effect of ARD in our technique and the role of ARD is to effectively determine the number of clusters, we set $K=8$ which is twice the actual number of cluster. The other parameters are just set to their typical values as thoroughly studied

[1] The MATLAB implemented code is available at
`http://networks.ece.mcgill.ca/sites/default/files/DNMF_0.zip`

Table 4.1 Parameters of **SynFix** for testing the effect of a and b

Parameters and Description	Value
T (Number of timesteps)	10
x (Number of nodes that change membership within each community)	3
N (Size of network)	128
$\bar{z}$ (Expected degree of each node)	16
$\bar{z}_{out}$ (Expected between community degree of each node)	$2, 3, 4$

in the following sections. Note that the notation $a : b : c$ used in Table 4.2 represents that all values in the range $[a, c]$ with step size b are tested in our experiment.

Table 4.2 Parameters of DNMF for testing the effect of a and b

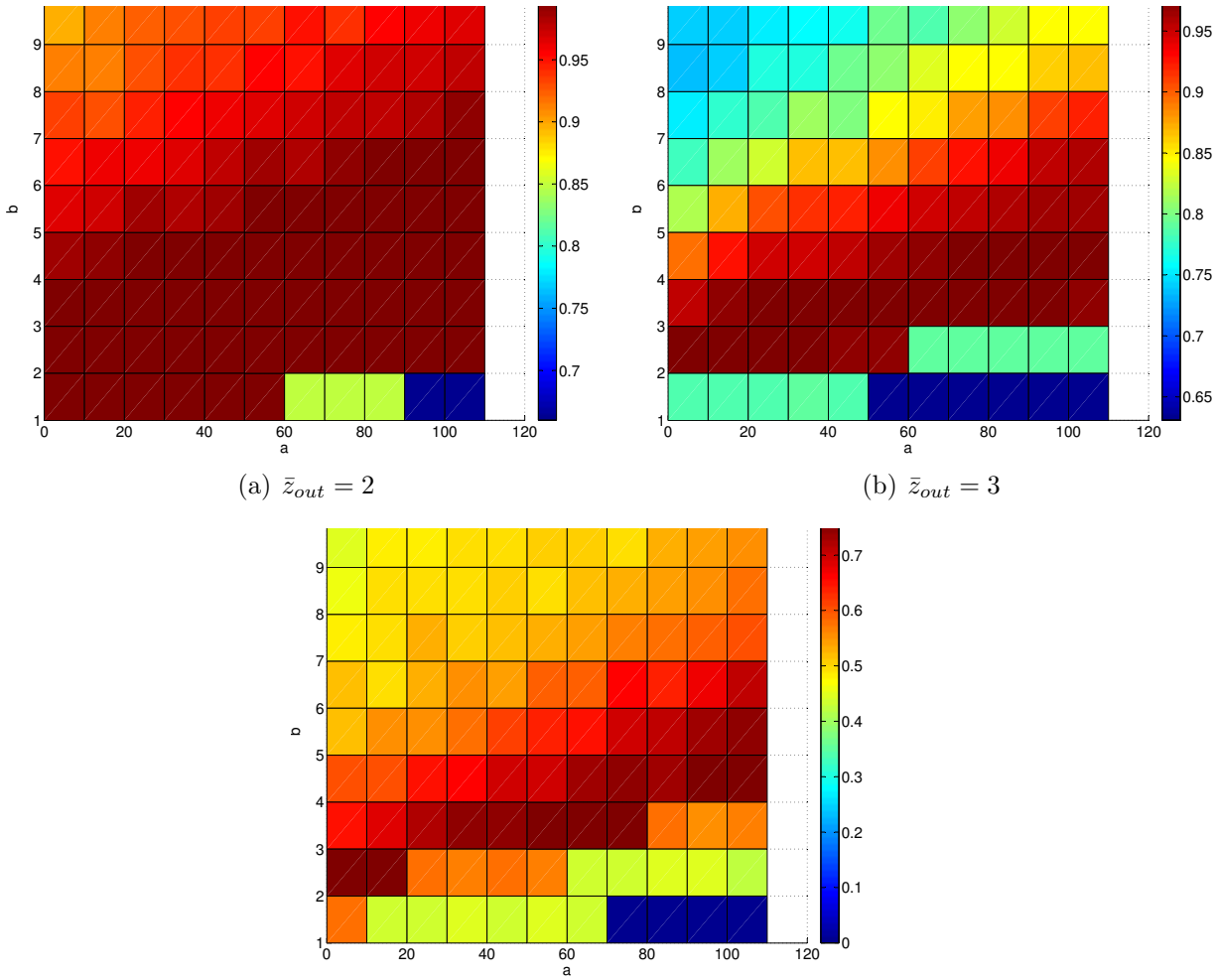
Parameters and Description	Value
K (Initial number of communities)	8
n_{iter} (Number of iteration used in Algorithm 2)	1000
α_0 (Initial preference on historic data introduced in Section 3.7)	0.3
a (Parameter of ARD introduced in Section 3.6)	$0 : 10 : 110$
b (Parameter of ARD introduced in Section 3.6)	$1 : 1 : 10$

We ran DNMF with parameters shown in Table 4.2 on **SynFix** detaset with three cases $\bar{z}_{out} = 2$, $\bar{z}_{out} = 3$, and $\bar{z}_{out} = 4$. At each timestep hard partitions are extracted based on the description mentioned in Section 3.2. The performance of DNMF was measured using NMI for different values of a and b. The results shown in Figure 4.1 indicate the average NMI over 10 timesteps. It is shown that by increasing $\bar{z}_{out}$ and as distinguishing the cluster becomes harder, the range of a and b that results in reasonable performance becomes tighter. For example for the case $\bar{z}_{out} = 2$ where community detection is fairly easy, for a very wide range of a and b values, clusters are detected without mistake. However, for the dataset with $\bar{z}_{out} = 4$, the NMI measure is sensitive to the choice of a and b. Moreover, the result for all cases revealed that as either parameter a or b increases the other parameter must decrease to ensure that DNMF performs as expected. These results are predictable

considering how a and b control the $\boldsymbol{\beta}$ parameters as shown in Algorithm 2:

$$\beta_k \leftarrow \frac{2(N+a-1)}{(\sum_{ij} g_{ik}^2 + h_{kj}^2) + b}\,. \tag{4.9}$$

In Equation (4.9), the parameter a is in the numerator, whereas the parameter b is in the



(a) $\bar{z}_{out} = 2$

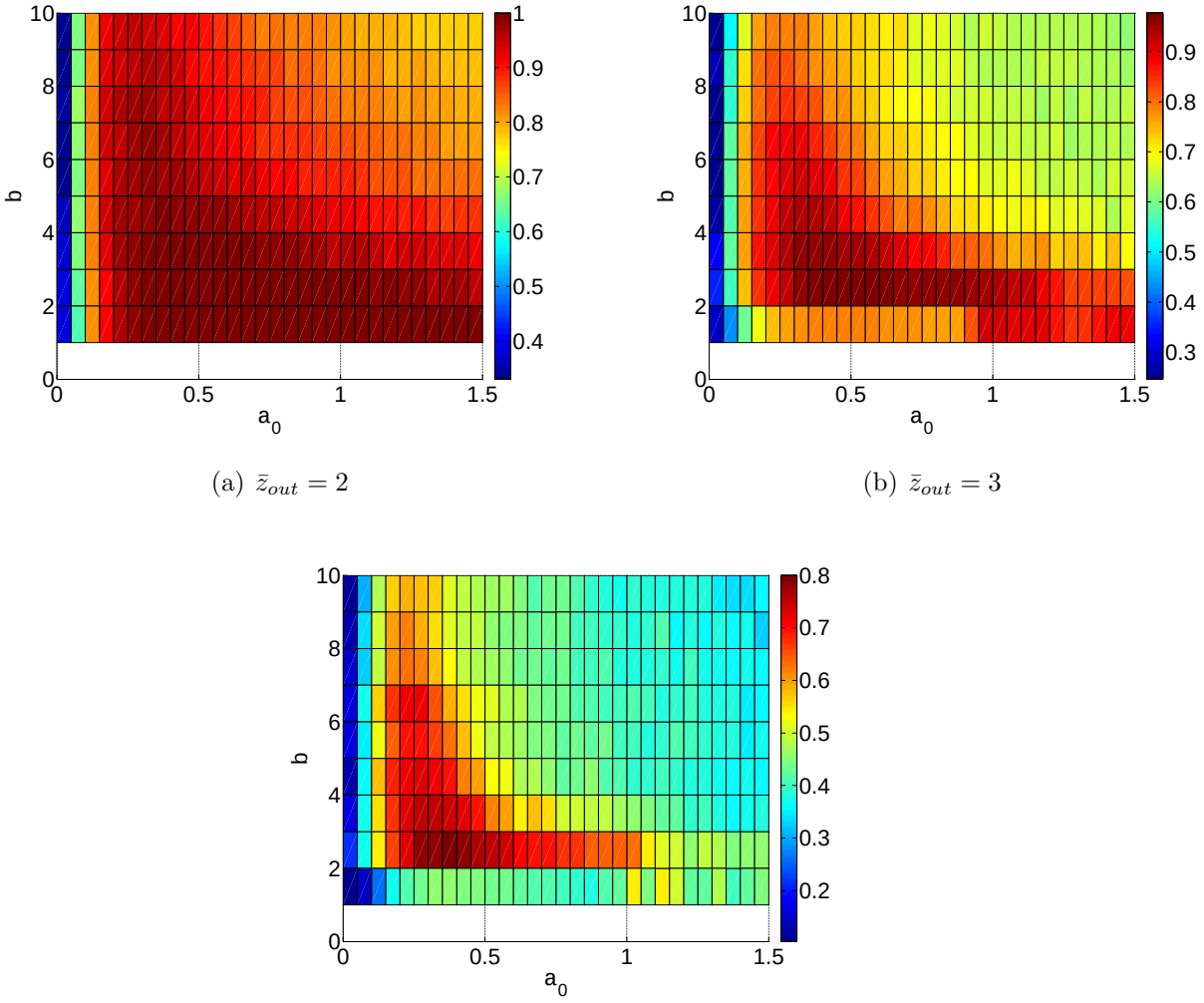
(b) $\bar{z}_{out} = 3$

(c) $\bar{z}_{out} = 4$

Fig. 4.1 Average NMI over 10 timesteps as a function of a and b

denominator. Therefore by tuning either a or b, one is able to control the range of β_k. For

this reason in all remaining experiments we consider the parameter a as constant and equal to 5.



(a) $\bar{z}_{out} = 2$

(b) $\bar{z}_{out} = 3$

(c) $\bar{z}_{out} = 4$

Fig. 4.2 Average NMI over 10 timesteps for different values of α_0 and b

Table 4.3 Parameters of DNMF for testing the effect of α_0 and b

Parameters and Description	Value
K (Initial number of communities)	8
n_{iter} (Number of iteration used in Algorithm 2)	1000
α_0 (Initial preference on historic data introduced in Section 3.7)	0:0.05:1.55
a (Parameter of ARD introduced in Section 3.6)	5
b (Parameter of ARD introduced in Section 3.6)	$1:1:10$

Table 4.4 Average NMI for $b = 2$ and different values for α_0

α_0	$\bar{z}_{out} = 2$	$\bar{z}_{out} = 3$	$\bar{z}_{out} = 4$
0.10	0.982	0.930	0.5444
0.15	0.992	0.960	0.6599
0.2	0.997	0.965	0.7289
0.25	0.997	0.974	0.7785
0.3	**1**	0.972	0.7871
0.35	**1**	**0.977**	**0.8002**
0.4	**1**	0.970	0.7831

4.1.4 The Effect of Changing α_0 and b

In this section the goal is to study the effect of the parameter α_0 and b on the NMI of SynFix datasets. The specifications of **SynFix** datasets are the same as Table 4.1. In Figure 4.2 the average NMI over 10 timesteps is shown for different values of α_0 and b for various $\bar{z}_{out}$. As depicted in Figure 4.2, the sensitivity of results to the parameters α_0 and β increases as $\bar{z}_{out}$ increases from 2 to 4. However, we can say that all three dataset have maximum NMI when $b = 2$. The sets of parameters that results in high NMI are shown in Table 4.4. From Table 4.4, it shows that for these datasets the maximum NMI occurs the interval of $\alpha_0 \in [0.3, 0.4]$.

4.1.5 The Effect of the Number of Iterations

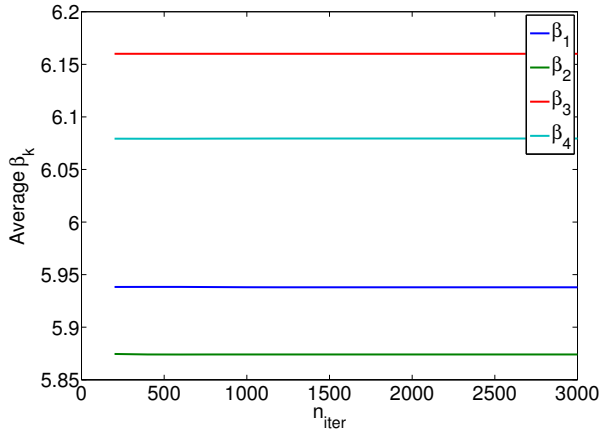
We applied DNMF with parameter shown in Table 4.5 on SynFix datasets. We plotted β_k, $k = 1, \cdots, K$ for various numbers of iterations. As illustrated in Figure 4.3, for all the cases of $\bar{z}_{out}$, β_ks become stable after 1000 iterations. Also, it is notable that the more distinguishable the clusters become (e.g., for $\bar{z}_{out} = 2$), the sooner β_ks reach stability.

Table 4.5 Parameters of DNMF for testing the effect of n_{iter}

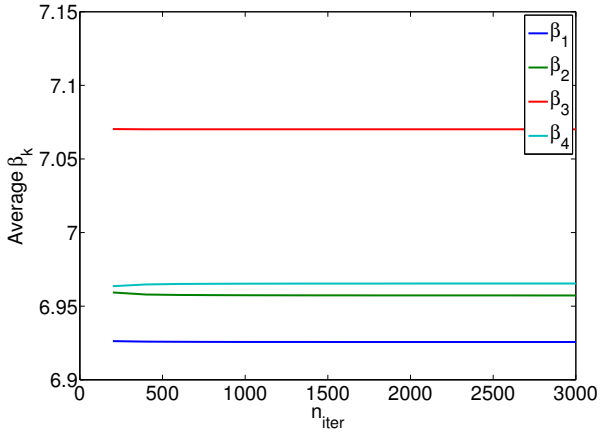
Parameters and Description	Value
K (Initial number of communities)	4
n_{iter} (Number of iteration used in Algorithm 2)	200:200:3000
α_0 (Initial preference on historic data introduced in Section 3.7)	0.3
a (Parameter of ARD introduced in Section 3.6)	5
b (Parameter of ARD introduced in Section 3.6)	2

4.1.6 The Effect of Changing the Dynamic features of Dataset

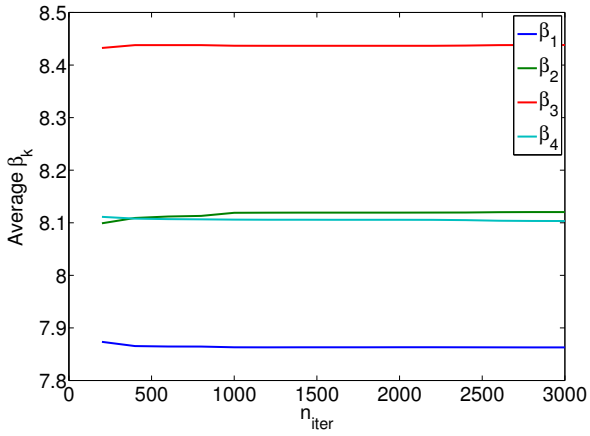
In this section we want to investigate how our algorithm responds to dynamic variations in terms of the parameter α_0. To capture this effect we vary the number of nodes that change community membership over time (the parameter x of the **SynFix** Dataset) and apply DNMF with parameters in Table 4.6 on **SynFix** datasets. We then plot the average NMI over all timesteps and 25 independent runs of DNMF as a function of the parameter α_0. As shown in Figure 4.4, not only does the average NMI decrease as more nodes change their membership, but also the α_0 that gives the best perfomance in terms of NMI also changes with the parameter x. In Figure 4.4, the standard deviation over 25 independent runs of DNMF is also indicated. In table 4.7, the best α_0 for each pair of $\bar{z}_{out}$ and x is shown. As expected from theory, when the degree of changes increases (i.e, increasing x), the parameter α_0 that reflects the general preference on the historic data must also



(a) $\bar{z}_{out} = 2$



(b) $\bar{z}_{out} = 3$



(c) $\bar{z}_{out} = 4$

Fig. 4.3 Average β_k over 10 timesteps per n_{iter}

increase to maintain the best performance. This is because by increasing α_0, DNMF puts more emphasis on the current data rather than the historic data. In this experiment the network experiences more intense short term variations by increasing x and hence one should increase the preference parameter α_0 to get better results.

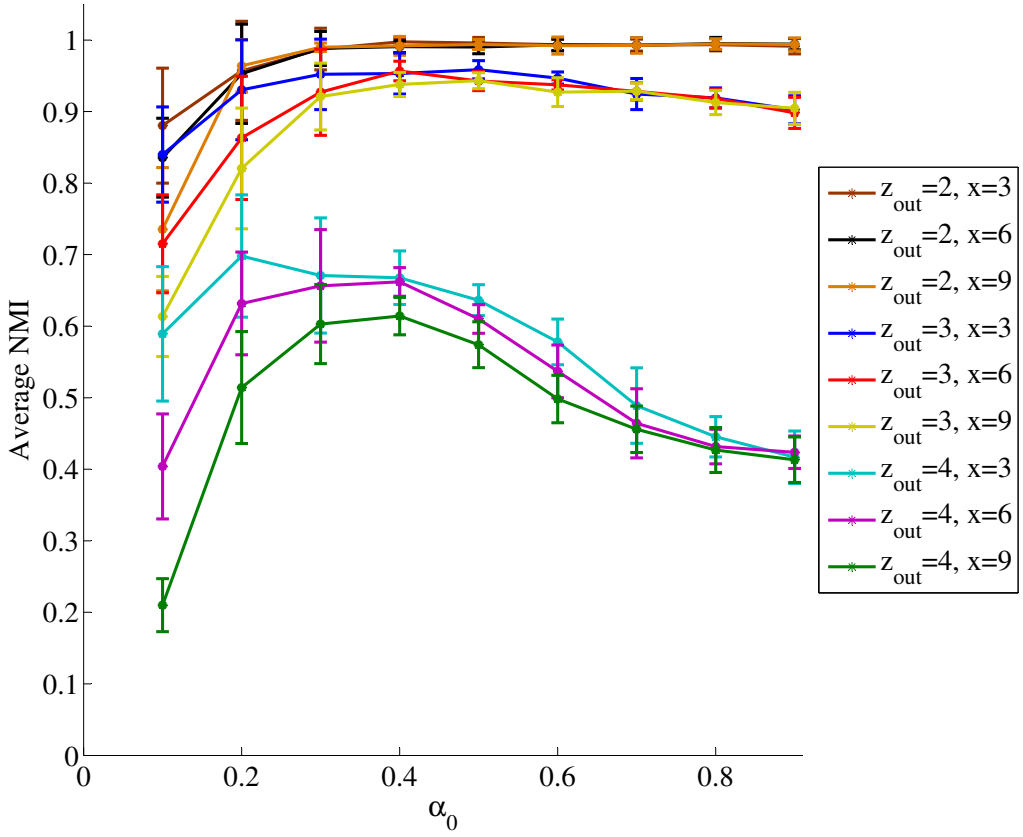


Fig. 4.4 Average NMI over 10 timesteps and 25 independent runs of DNMF per α_0 for different $\bar{z}_{out}$ and x

Table 4.6 Parameters of DNMF in Section 4.1.6

Parameters and Description	Value
K (Initial number of communities)	4
n_{iter} (Number of iteration used in Algorithm 2)	100
α_0 (Initial preference on historic data introduced in Section 3.7)	0.1:0.1:0.9
a (Parameter of ARD introduced in Section 3.6)	5
b (Parameter of ARD introduced in Section 3.6)	2

Table 4.7 Parameter α_0 that maximizes average NMI for different datasets

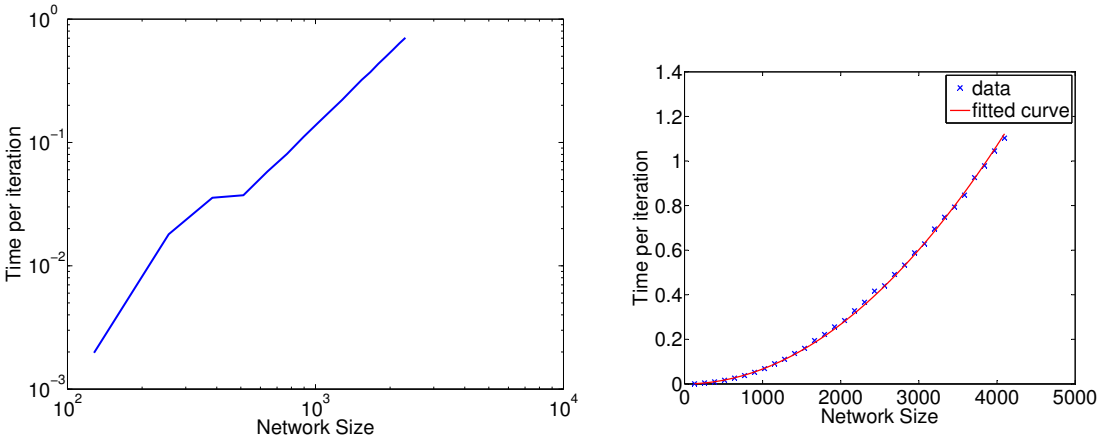
$\bar{z}_{out}$	$x = 3$	$x = 6$	$x = 9$
2	0.4	0.7	0.8
3	0.4	0.4	0.5
4	0.2	0.4	0.4

4.1.7 Runtime

As mentioned in Section 3.9, the complexity of DNMF is in the order of the network size squared. In this section, we repeat the experiments over networks of various size. Experiments were performed on an Intel 2.40 GHz Core 2 Quad CPU with 8 GB of RAM and Windows 7 Professional Operating system and the algorithm was implemented in MATLAB. In this experiment, we ran DNMF on the **SynFix** dataset for $T = 2$, $z_{out} = 3$, $K = 4$, $n_{iter} = 1000$, and various network size. In Figure 4.5, we showed the average running time per iteration and timestep. Note that the data points in Figure 4.5 represent the average amount of time that each iteration takes to be completed. This implies that for DNMF algorithm with $n_{iter} = 1000$, the total running time of each timestep will be multiplied by 1000. In Figure 4.5(a), we depicted our curve in a log log format and observed that the slop of the curve is approximately equal to 2. In Figure 4.5(b) we fitted a quadratic polynomial on data points. In both plots, it is shown that our running time matches the theory and increases with the second degree of the network size.

4.1.8 Effectiveness of ARD

In Section 3.6 we claimed that the use of ARD in our simple DNMF model helps us with the problem of model order selection. In this section, our goal is to explore the effectiveness of ARD in our synthetic networks. In our experiment we noticed that the set of $\boldsymbol{\beta}_t$ parameters identified at each timestep effectively indicates the true number of



(a) Depicted in a log log plot

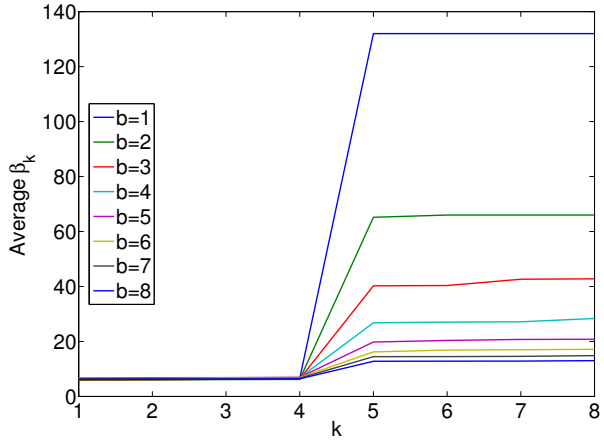
(b) A quadratic polynomial is fitted on the data points

Fig. 4.5 Average running time per iteration (sec)

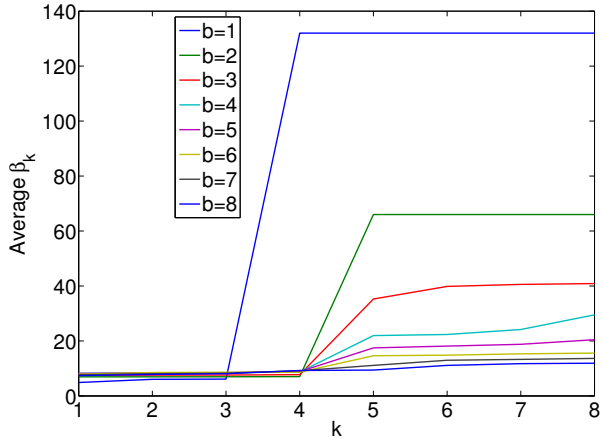
Table 4.8 Parameter setting of DNMF for testing the effectiveness of ARD

Parameters and Description	Value
K (Initial number of communities)	8
n_{iter} (Number of iteration used in Algorithm 2)	1000
α_0 (Initial preference on historic data introduced in Section 3.7)	0.3
a (Parameter of ARD introduced in Section 3.6)	5
b (Parameter of ARD introduced in Section 3.6)	1:1:8

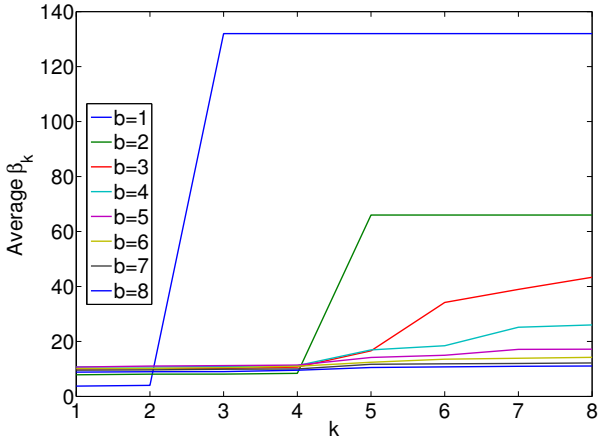
communities. As discussed in Section 3.6, when a $\beta_{k,t}$ becomes large enough, the k-th community is no longer a genuine cluster and should be eliminated. Therefore when we run the simulations with a constraint on the number of clusters K, we expect to see a gap in the sorted average β_k between those real clusters and the unnecessary ones. We applied DNMF on **SynFix** datasets with parameters shown in Table 4.1. The parameters of DNMF are as stated in Table 4.8. Note that we assumed that the number of clusters can be at most $K = 8$. We computed the average of $\beta_{k,t}$s over all timesteps and sorted the results. In Figure 4.6, the sorted values of all the β_k values are plotted. Figure 4.6



(a) $\bar{z}_{out} = 2$



(b) $\bar{z}_{out} = 3$



(c) $\bar{z}_{out} = 4$

Fig. 4.6 Sorted average β_k over 10 timesteps for various b

illustrates that for the case $\bar{z}_{out} = 2$ DNMF easily captured the gap at the true number of clusters $K = 4$ for all values of b. In the cases where the community detection is more difficult such as when $\bar{z}_{out} = 3$ and $\bar{z}_{out} = 4$ although the gap at $K = 4$ is detectable, a careful examination of the results of various b is needed. As we can observe in Figure 4.6, when we dramatically decrease b (e.g., $b = 1$ for $\bar{z}_{out} = 3$ and $\bar{z}_{out} = 4$) we force our model to have large β and subsequently the identified number of clusters is less than the actual case and hence a gap occurs at the wrong place. However, by checking the results for larger values of b, it becomes obvious that the gap does not indicate the true order of our model since there is not a similar gap for any larger values of b. The gap occurring at $K = 4$ is detectable for a wide range of b. We also need to note that the performance of DNMF in terms of NMI increases for the larger gaps at the actual number of clusters as shown in Figure 4.7.

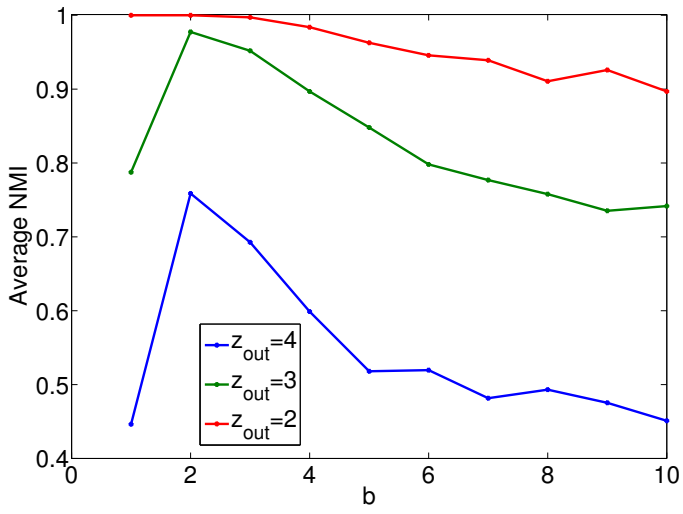


Fig. 4.7 Average NMI per b

In another experiment, we applied DNMF on the **SynVar** Network. As described in Section 4.1.1 in this synthetic network the number of clusters increases by one at each timestep (illustrated in Figure 4.8). The more precise specifications of the SynVar network

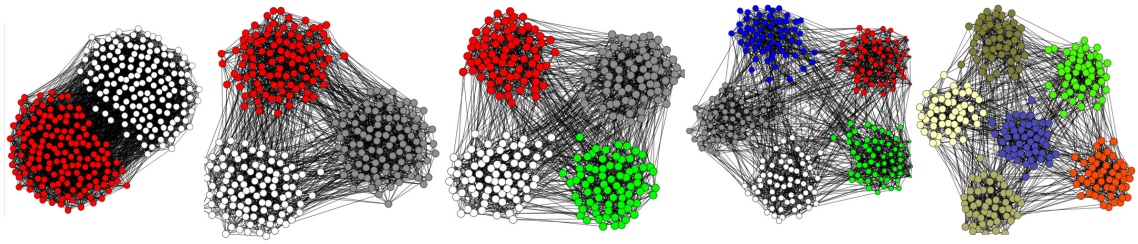


Fig. 4.8 Schematic of **SynVar** network

tested in this experiment are shown in Table 4.9. The parameter settings of DNMF were similar to Table 4.8 but with fixed $b = 2$. To show the effectiveness of ARD, we also tested DNMF without ARD (as described in Algorithm 1) on the same network. The results in terms of the number of identified clusters and NMI are shown in Table 4.10. By comparing the results it becomes clear that the use of ARD substantially improves our model in terms of both model order selection and accuracy.

Table 4.9 Parameters of **SynVar**

Parameters and Description	Value
T (Number of timesteps)	5
N (Size of network)	300
$\bar{z}$ (Expected degree of each node)	16
$\bar{z}_{out}$ (Expected between community degree of each node)	3

4.1.9 Performance of DNMF

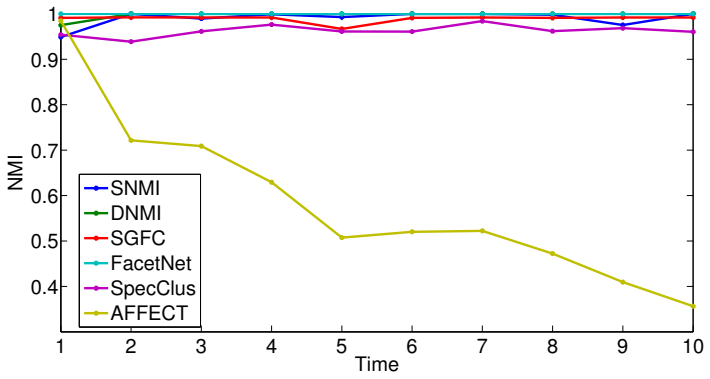
In this section we compare the performance of our dynamic method, dynamic non-negative matrix factorization (DNMF), with two other state-of-the-art algorithms, FacetNet proposed by Lin et al. [6] and AFFECT proposed by Xu et al. [7]. In addition to these algorithms, we also ran simulation using the static version of AFFECT which is static

Table 4.10 **SynVar** dataset results (Number of clusters K/NMI)

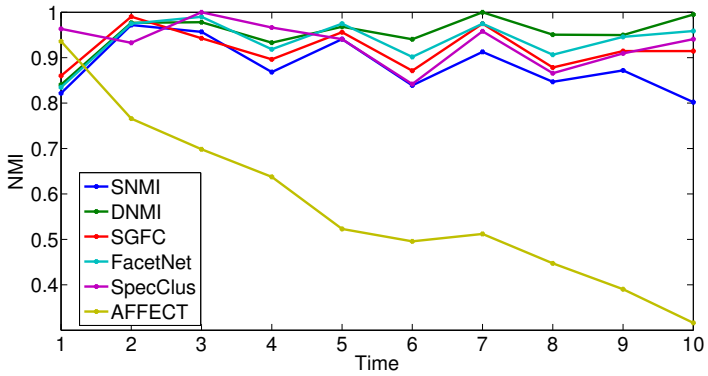
Timestep	1	2	3	4	5
True K	2	3	4	5	6
DNMF with ARD	2/1.000	4/0.895	4/1.000	5/1.000	6/1.000
DNMF without ARD	2/1.000	8/0.688	8/0.795	8/0.888	8/0.9397

spectral clustering (SpecClus) [36], static non-negative matrix factorization (SNMF) [21], and static graph factorization (SGFC) [43]. To have a fair comparison we consider the best parameter settings for all methods. We ran these algorithms 25 times on **SynFix** datasets with various $\bar{z}_{out}$. In Figure 4.9, the performance in terms of avergae NMI (over 25 runs) is depicted over 10 timesteps. We can observe that in the cases $z_{out} = 3, 4$, our method (DNMF) highly outperforms the other baseline algorithms. We can also conclude that for DNMF and FacetNet, the evolutionary version of methods gives better NMI compared to the static version of them, which shows that exploiting temporal smoothness does improve the performance of clustering methods. We computed the average NMI over 10 timesteps and 25 runs for all algorithms. The results are shown in Table 4.11 and again it illustrates that DNMF outperforms other algorithms for $z_{out} = 3$ and $z_{out} = 4$ and slightly lower than FacetNet for $z_{out} = 2$. In Table 4.11, the standard deviation of all methods is computed over 25 independent runs. This table shows despite the good results of DNMF in terms of the average NMI, with a slight difference the identified clusters are more volatile than those obtained by FacetNet. This is due to the random initialization of the matrices $\mathbf{G}$ and $\mathbf{H}$ and therefore the possibility of becoming trapped in a local minima rather than the global minimum of the aposteriori likelihood function.

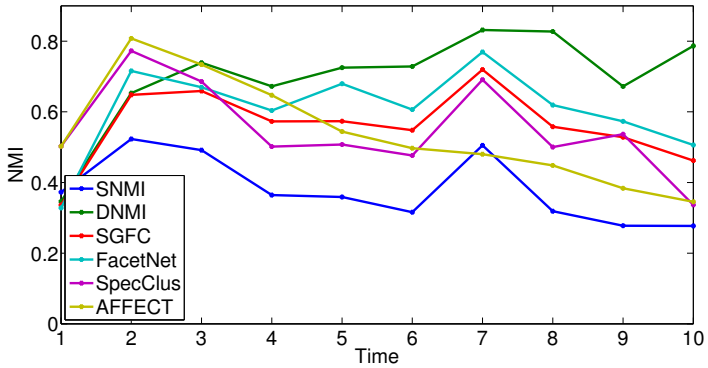
The high values of variance for the case of $z_{out} = 4$ in Table 4.11, especially for the DNMF algorithm, brings up some concerns about the statistical significance of results. Therefore we need to validate the statistical significance of the results using Lilliefors and



(a) $\bar{z}_{out} = 2$



(b) $\bar{z}_{out} = 3$



(c) $\bar{z}_{out} = 4$

Fig. 4.9 NMI over time. Clusters are founded by various clustering algorithms including non-negative matrix factorization (NMF), FacetNet [6], and AFFECT [7] in both dynamic and static cases. SpecClus is stand for Spectral Clustering which is the static version of AFFECT. Similarly SGFC represents the static version of FacetNet.

Table 4.11 Mean/Standard Deviation of NMI over time and 25 independent runs of different algorithms

z_{out}	2	3	4
SNMF	0.9904/0.0125	0.8832/0.0304	0.3805/0.0272
DNMF	0.9975/0.0059	0.9532/0.0288	0.6981/0.0854
SGFC	0.9893/0.0411	0.9199/0.023	0.5606/0.0272
FacetNet	1.0000/0	0.9380/0.0162	0.6070/0.0191
SpecClus	0.9628/0.022	0.9319/0.0187	0.5511/0.0107
AFFECT	0.5832/0.0084	0.5722/0.0117	0.5390/0.0073

analysis of variance (ANOVA) tests. Further information on the parameters and assumptions of these tests can be found in Appendix A. Here we perform Lilliefors and ANOVA tests for the case of $z_{out} = 4$ over three main competitors in Table 4.11: DNMF, FacetNet, and SGFC. For performing ANOVA test, we first require to check if our results pass Lilliefors test. Lilliefors test indicates whether the null hypothesis that data come from a normal distribution family, against the alternative that it does not come from a normal distribution (H_1). Having a sample set drawn from a normal distribution is the main assumption of ANOVA test. In ANOVA test, we investigate whether the differences in the average values of NMI metric for different algorithms are due to chance (H_0: null hypothesis) or actual differences in the means H_1. In Table 4.12, the results of Lilliefors test is shown for three DNMF, FacetNet, and SGFC algorithms. As it is shown, for all three algorithms, the Lilliefors statistics are less than the critical value at significance level $\alpha = 0.01$ and sample size $N = 25$. In other words, they all come from a normally distributed population. Now that the main assumption of ANOVA test is satisfied, we can perform the ANOVA test. The test gives us the result $F = 43.65$, which is considerably higher than the critical value of $F_{critical} = 7.1942$ at significance level $\alpha = 0.01$, sample size $N = 25$, and number of algorithms under the test $n = 3$. We observe that the test fails to validate the null hypothesis and therefore we conclude that the differences in the mean NMI values are

Table 4.12 Comparing Lilliefors statistics for NMI metrics for DNMF, FacetNet, and SGFC algorithms (the case of $z_{out} = 4$) with the critical value.

Algorithm	DNMF	FacetNet	SGFC
Lilliefors statistics	0.2012	0.1217	0.1921
Critical Value	0.2014	0.2014	0.2014
Null Hypothesis Rejected	No	No	No

not due to chance.

4.2 Real Network

4.2.1 Network Description

To test our algorithm in a real world application, we applied DNMF on the network of email contacts in the Department of Informatics at Karlsruhe Institute of Technology (KIT)[2] [103]. This dataset contains information about the sender, receiver, and also the chairs supervising both sender and receiver of emails sent in the department of informatics at KIT. The data is collected over a period of more than three years. In this experiment we construct a dynamic graph over one year with a timespan of one month such that our dynamic graph contains 12 timesteps, starting from 2006-09-13 to 2007-09-13. We count the number of emails transferred between each pair of nodes within one month and process the counts to obtain a weighted graph. As claimed in [104], the range of original counts has some outliers. Therefore to reduce the effect of these variations we normalize the weights and map the into the interval $[0, 1]$ as

$$w_{ij,t}^{no} = \frac{\log(w_{ij,t})}{\log(w_t^{\max})}. \tag{4.10}$$

[2]This dataset is available at `http://i11www.iti.uni-karlsruhe.de/en/projects/spp1307/emaildata`

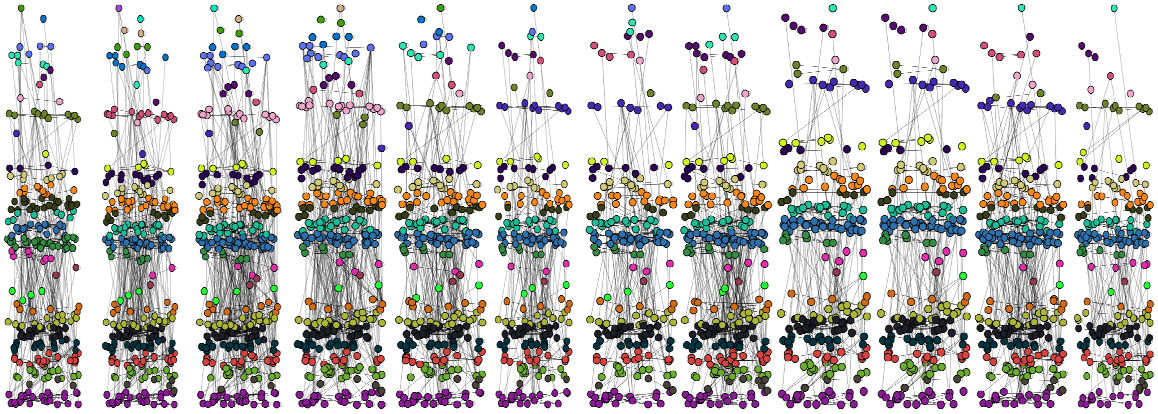


Fig. 4.10 Visualization of the dynamic email network of KIT. Nodes are colored based on their associated chairs (reference clusters). The location of nodes are preserved over time. The nodes of each cluster are positioned near to each other on vertical slaps.

This email dataset provides us with invaluable information on the chairs associated the sender and receiver of emails. Since the group of people who work under the same chair tend to communicate more frequently, this information can reveal a potential reference clustering of the network. However, we need to note that this reference clustering is based on only one social factor and can thus provide some intuition about the possible partitioning of nodes. This is not necessarily the best partitioning for revealing the structure of the email network. In Figure 4.10, the email network is visualized over time. The positions of nodes are preserved over time and node colors represent the reference clustering. Note that in this Figure, edges with weight less than 0.2 are pruned for clarity. The network size and reference clustering size are shown in Table 4.13 for each timestep.

4.2.2 Measurements

In order to reduce the noise of the network and possibly make the structure of network closer to the reference clustering, we pruned the edges of the email network with weights below the pruning fraction (varying $p = 0 : 0.1 : 0.5$) similar to an experiment in [104]. We applied DNMF with various parameter settings, changing K (initial number of clusters), α_0 (general preference constant), and b (parameter of ARD) to analyse how these parameters effect the NMI measured between the identified clusters and the reference clustering.

Remark. Since DNMF has two main parameters that must be set before running and we test them on all pruned network cases ($p = 0 : 0.1 : 0.5$), we ran simulations for all possible combinations of parameters shown in Table 4.15. We only show the results for a selection of these combinations, varying one of the three parameters while keeping the other one fixed at values that lead to good performance on average.

Choice of K. We set the number of cluster to the sequence $K = [31, 31, 31, 29, 26, 23]$, associated with the datasets $p = [0, 0.1, 0.2, 0.3, 0.4, 0.5]$ respectively. In this email graph,

Table 4.13 Email Graph Characteristics

Time	Number of Nodes	Number of Chairs
1	321	30
2	344	31
3	364	30
4	417	30
5	341	29
6	309	28
7	314	28
8	322	28
9	321	27
10	317	27
11	311	28
12	293	26

Table 4.14 Email Graph Characteristics for various p (Network size/number of clusters)

Time	$p = 0.1$	$p = 0.2$	$p = 0.3$	$p = 0.4$	$p = 0.5$
1	321 / 30	270 / 28	231 / 27	160 / 24	98 / 21
2	344 / 31	313 / 31	274 / 29	187 / 25	127 / 23
3	364 / 30	323 / 30	242 / 27	169 / 26	102 / 23
4	417 / 30	318 / 29	209 / 26	128 / 25	63 / 20
5	341 / 29	271 / 28	197 / 25	102 / 22	49 / 14
6	309 / 28	252 / 27	185 / 23	142 / 22	84 / 17
7	314 / 28	225 / 26	165 / 23	82 / 19	46 / 14
8	322 / 28	287 / 26	215 / 23	138 / 20	76 / 16
9	321 / 27	241 / 25	159 / 23	82 / 17	37 / 12
10	317 / 27	276 / 26	203 / 24	158 / 20	92 / 19
11	311 / 28	277 / 25	196 / 23	154 / 20	79 / 17
12	293 / 26	240 / 25	125 / 18	72 / 18	46 / 15

Table 4.15 Parameters of DNMF in Section 4.2.2

Parameters and Description	Value
n_{iter} (Number of iteration used in Algorithm 2)	1000
α_0 (Initial preference on historic data introduced in Section 3.7)	0.1:0.1:0.9
a (Parameter of ARD introduced in Section 3.6)	5
b (Parameter of ARD introduced in Section 3.6)	1:1:9

we have an indication of the maximum number of clusters as it is shown in Table 4.14. Therefore, we simply choose the maximum number of active chairs over time from Table 4.14 as the initial K. We ran DNMF with $\alpha_0 = 0.7$ and $b = 5$ on the email dataset for 25 times. The average identified number of clusters over 25 runs is depicted in Table 4.16. The difference between the number of chairs shown in Table 4.14 and the founded number of clusters illustrated in Table 4.16 is due to the fact that ARD tends to eliminate clusters which do not have strong support for their existence in the data. As depicted in Figure 4.10, there are several reference clusters with only one or two nodes; those clusters are the most probable clusters to be absorbed by other dominant ones.

Table 4.16 Average identified number of clusters over 25 runs

Time	$p = 0$	$p = 0.1$	$p = 0.2$	$p = 0.3$	$p = 0.4$	$p = 0.5$
1	25.84	26.2	27.56	27.16	25.12	21.84
2	22.64	22.56	23.52	23.32	22.8	20.16
3	21.12	21.2	21.52	21.6	21.04	19.4
4	22.6	22.96	22.24	21.52	21.64	20.24
5	19.32	18.96	19.8	20.52	20.48	18.68
6	18.8	18.32	18.72	19.72	20.08	19.08
7	18.48	18.24	18.64	19.08	17.84	17.76
8	18.72	18.56	19.72	18.88	20.24	19.36
9	18.2	17.96	18.84	19.08	18.4	15.8
10	17.96	18	18.64	18.84	20.16	18.52
11	18.56	18.12	19.28	18.64	19.12	19.12
12	18.72	18.44	18.36	18.92	18.12	18.88

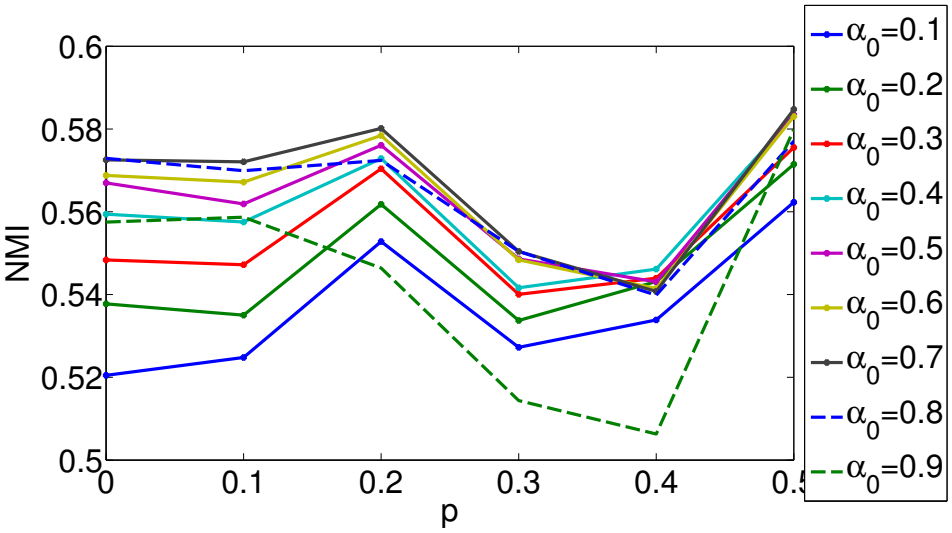


Fig. 4.11 Average NMI for different value of p and α_0

Effect of α_0. In Figure 4.11, the average NMI over time and 25 independent runs is illustrated for different values of p and α_0. DNMF was applied with $b = 5$ and $K = [31, 31, 31, 29, 26, 23]$ for $p = 0 : 0.1 : 0.5$. The figure shows that $\alpha_0 = 0.7$ results in the best performance for the cases where $p = 0, 0.1, 0.2, 0.3, 0.5$. The more extreme choices of 0.1 and 0.9 do lead to lower NMI, but the algorithm performs similarly for the intermediate

values.

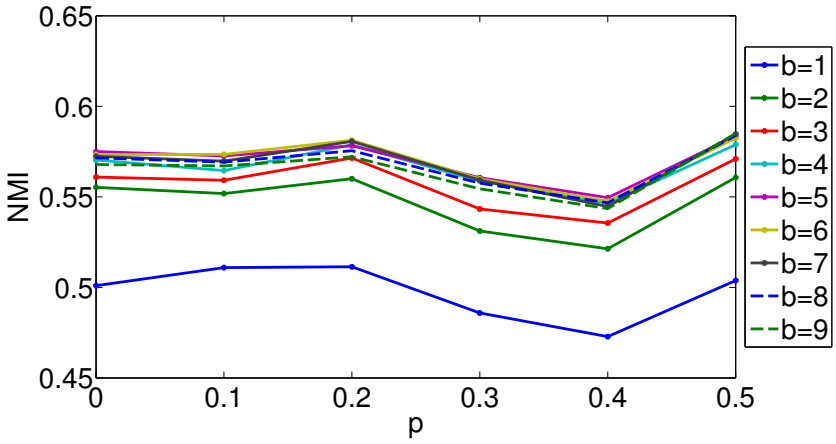


Fig. 4.12 Average NMI for different value of p and b

Effect of b. Figure 4.12 depicts the performance of DNMF in terms of average NMI over time and 25 runs as we vary p and b. The figure shows the case when $\alpha_0 = 0.7$ and $K = [31, 31, 31, 29, 26, 23]$ for the datasets $p = 0 : 0.1 : 0.5$. Figure 4.12 reveals that when $b = 1$ the average NMI of DNMF is substantially lower than the other values of b. The results have a peak at $b = 5$ and then the performance starts dropping with very low rate as b grows to 9.

We applied DNMF on the email dataset with the parameter setting described above. The network along with the identified clusters is visualized in Figure 4.13. The network is shown for the case $p = 0.2$ where we observed a local peak in the average NMI of the network. The positions of nodes are preserved over time and node colors represent identified clusters. We match the cluster over time such that a similar color is assigned to two clusters in adjacent timesteps with the most number of nodes in common. Those nodes who are in the same reference cluster are positioned on vertical lines. Although there are some nodes that are not assigned to their reference clusters as can be noticed in Figure 4.13, we can observe that most of the nodes that are positioned on vertical lines have the same color.

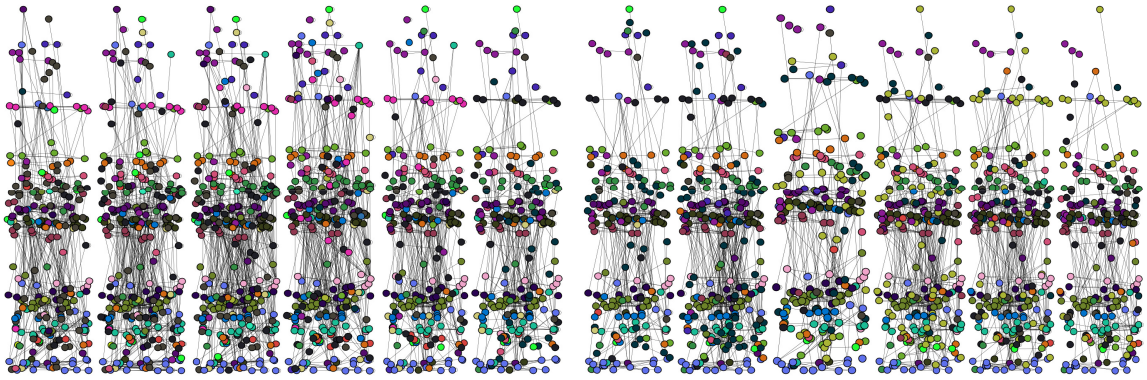


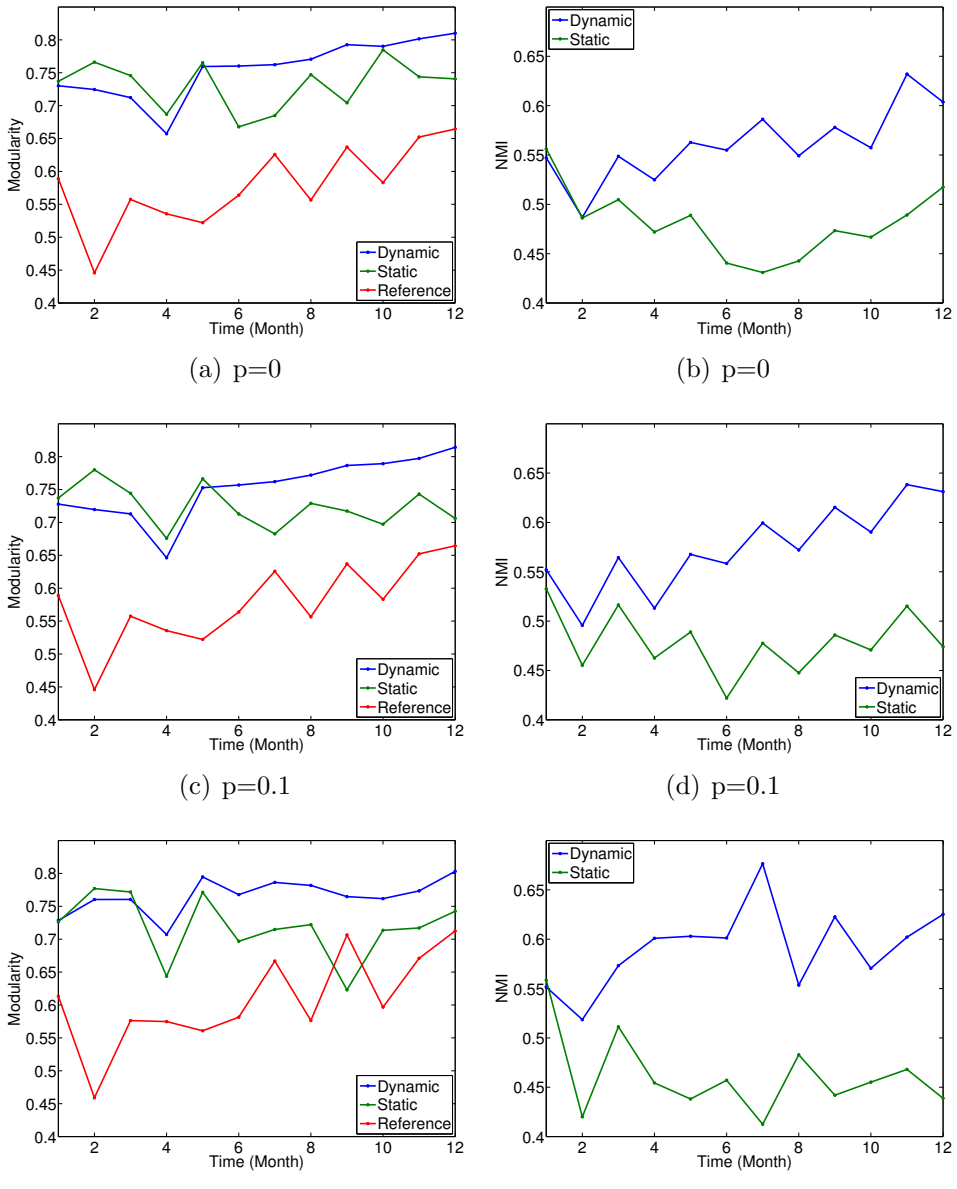
Fig. 4.13 Visualization of the dynamic email network of KIT over time. Nodes are colored based on the DNMF partitioning. The location of nodes are preserved over time. Nodes with similar reference cluster are positioned near to each other on vertical lines.

This shows that the clusters identified DNMF are strongly related to the clusters in the reference partitioning of nodes. As mentioned above, this reference clustering is derived based on only one social factor, and it cannot be considered as a ground truth.

Performance over time. In Figures 4.14 and 4.15, the modularity and NMI results of applying DNMF on the email network are shown with respect to time . As introduced in Section 2.1.3, modularity is an index representing the quality of clustering. For all the cases of p, the modularity index is computed for three groupings: 1) identified clusters of DNMF with the best parameter setting; 2) static version of DNMF ($\alpha = 0$); and 3) reference partitioning. We also illustrate the NMI (measured between the reference partitioning and clusters identified by the dynamic and static version of our algorithm) over time for the various values of p. As depicted in Figures 4.14 and 4.15, for all values of p, except for $p = 0.5$, the dynamic version of our algorithm significantly outperforms the static version on average.

Figures 4.14 and 4.15 also reveal that as more edges of the network are pruned and the parameter p is increased the reference clustering quality in terms of modularity is also increased. When we prune the network heavily, only the dominant edges remain, and most of these are within reference clusters (email exchanges between people with the same chair). The lower volume email exchanges should not be disregarded, however, and they indicate that several individuals have important work and social relationships that arise from other factors.

Comparing to other algorithms. Similar to Section 4.1.9, we compared the performance of our dynamic method, here on the real email dataset. In this section, dynamic non-negative matrix factorization (DNMF) was compared to two other state-of-the-art algorithms, FacetNet [6] and AFFECT [7]. We also considered the static version of AFFECT which is static spectral clustering (SpecClus) [36], static non-negative matrix factorization (SNMF) [21], and static graph factorization (SGFC) [43] for comparison. All algorithms are applied with their best parameter setting with respect to the reference clustering. We ran these algorithms 25 times on datasets with various p. In Table 4.17, the average and standard deviation of NMI over 25 runs and 12 timesteps are shown for all algorithms. We can observe that in all cases, our method (DNMF) outperforms AFFECT and gives very competitive results compared to FacetNet. In terms of standard deviation, however, DNMF is more volatile than AFFECT and FacetNet. Moreover, we can conclude that in all cases, the evolutionary version of methods gives better NMI compared to the static version of them, which shows that exploiting temporal smoothness does improve the performance of clustering methods.

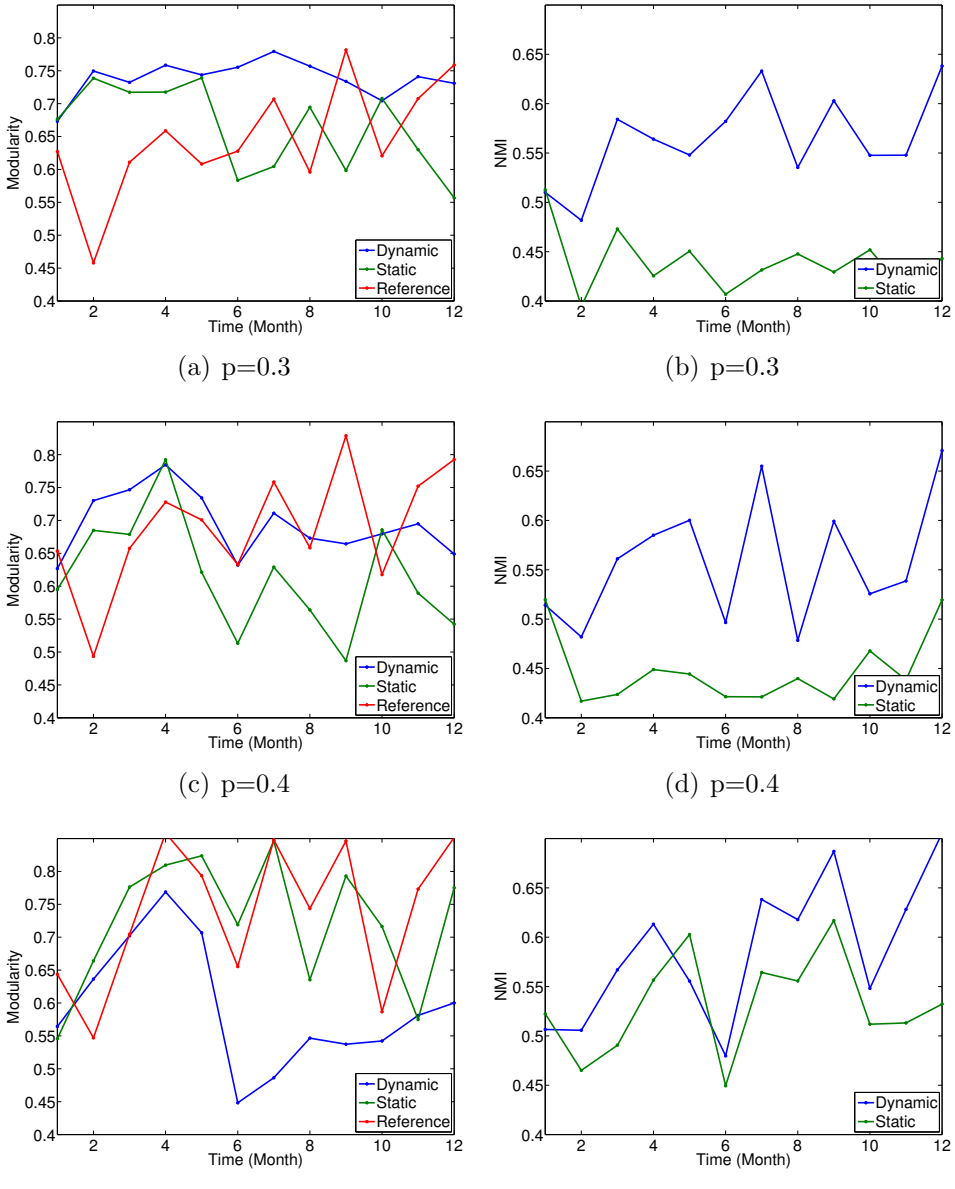


(a) p=0 (b) p=0

(c) p=0.1 (d) p=0.1

(e) p=0.2 (f) p=0.2

Fig. 4.14 NMI and Modularity over time for the KIT dynamic email network for pruning parameters $p = 0$, 0.1,and 0.2. Results are shown for static clustering using non-negative matrix factorization, DNMF, and the reference clustering based on the chairs.



(a) p=0.3 (b) p=0.3

(c) p=0.4 (d) p=0.4

(e) p=0.5 (f) p=0.5

Fig. 4.15 NMI and Modularity over time for the KIT dynamic email network for pruning parameters $p = 0.3$, 0.4,and 0.5. Results are shown for static clustering using non-negative matrix factorization, DNMF, and the reference clustering based on the chairs.

Table 4.17 Performance on Email Graph. Average NMI over 12 timesteps and 25 runs for various p (Average NMI/Average Standard Deviation)

p	SNMF	DNMF	SGFC	FacetNet	SpecClus	AFFECT
0	0.4838/0.0042	0.5726/0.0075	0.4772/0.131	0.5809/0.0066	0.5245/0.0053	0.5411/0.0047
0.1	0.4828/0.0036	0.5721/0.0094	0.4695/0.133	0.5621/0.0047	0.544/0.0053	0.5392/0.0041
0.2	0.4661/0.0057	0.5802/0.0079	0.397/0.1468	0.5895/0.0066	0.5287/0.006	0.5754/0.003
0.3	0.4458/0.005	0.5504/0.0084	0.328/0.1111	0.5434/0.007	0.5168/0.0102	0.5383/0.0047
0.4	0.4547/0.005	0.5408/0.0075	0.3218/0.1733	0.5531/0.1195	0.5016/0.0111	0.5224/0.0063
0.5	0.5212/0.0053	0.5847/0.0078	0.3327/0.1302	0.5817/0.1752	0.5409/0.014	0.5844/0.0128

4.3 Guidelines on DNMF Parameter Setting

In Sections 4.1 and 4.2 we ran several experiments to study the effect of the parameters of DNMF on the performance. In this section, the goal is to determine the characteristics of a reasonable parameter setting and provide some guidelines on identifying the parameter setting based on the characteristics of the network.

The parameter α_0 indicates the general preference on the historic dataset. Setting α_0 to high values implies having more emphasis on the data of the current timestep rather than the historic dataset. In the most of temporal networks, the extreme values of α (i.e., $\alpha = 0$ and 1) do not result in meaningful clustering. In all of our experiments, the effectiveness of the evolutionary clustering (i.e., $\alpha < 1$) was exploited. Moreover, as pointed out in Section 4.1.6, for networks with slow temporal variations, the low values of α_0 lead to better results, however for highly varying networks the high values of α_0 result in good performance. Therefore we can tune the parameter α_0 with respect the expected degree of network variations.

The initial number of clusters K is another parameter of DNMF. When the number of clusters is known, the use of ARD is unnecessary, and hence we no longer need to set the

parameters a and b. In our implemented code[3], there is an option to run simple DNMF without ARD as described in Algorithm 1. When the number of clusters is unknown, K can be replaced with the expected maximum number of clusters, and if we do not have any estimation on the number of clusters, the number of nodes can be used instead. In this case the parameters of ARD may play an important role by reducing the clusters to a reasonable number.

In Section 4.1.5, we found out that the rate of convergence is high, especially when the clusters are easily distinguished. Hence the number of iterations can be set to the minimum of 100 when running time is a constraint. Otherwise one can increase the number of iterations to 1000 to ensure the convergence with low error. An alternative approach to running with the fixed number of iteration can be introducing a threshold on how much $\mathbf{G}_t$ and $\mathbf{H}_t$ change between successive iteration. This approach allows the convergence of networks with major changes.

ARD has two parameter: a and b. In Section 4.1.3, we investigated the effect of varying the parameters a and b at the same time and figured out that varying a and b have opposite effect on the performance. We therefore concluded that by fixing one of these parameters to a constant, like $a = 5$, and varying the other one (parameter b) the functionality of ARD will be completely controllable. As mentioned above, the choice of parameter b is crucial when the number of clusters is not known. The parameter b reduces and adjusts the number of clusters. The low values of b (e.g., $b < 3$) substantially reduce the number of clusters. In contrast, a large value of b adjusts the number of clusters to a larger number below K. When the clusters are fairly easily distinguished, ARD will not be very sensitive to the choice of b and can find the true number of communities for a wide range of b. On the

[3]The MATLAB implemented code is available at
`http://networks.ece.mcgill.ca/sites/default/files/DNMF_0.zip`

other hand when noise increases and clusters become more connected, ARD shows more sensitivity to the parameter b. In this case one can determine the number of clusters by watching the β parameters. A gap that exists in these parameters for a wide range of b can potentially indicates the number of clusters. For example, when we notice that a few β parameters are substantially lower than the others and this effect is also detectable when we run DNMF with other values of b, the number of low βs can be a potential candidate for the number of clusters.

4.4 Summary

In this chapter, we examined the performance of our algorithm, Dynamic Non-negative matrix Factorization (DNMF). We chose the Normalized Mutual Information measure to identify the distance of the identified partitioning from the ground truth or reference partitioning. We applied our algorithm on several synthetic networks which were designed to capture the dynamic features of networks such as node membership changes and variations in the number of clusters. We applied our algorithm on these datasets and explored the impact of changing the intensity of the dynamic changes. We also investigated how different parameter settings affect the performance of DNMF in terms of NMI and determined the setting which results in the best performance for analysing the synthetic networks. In our experiments, we showed how ARD helps us to find the true number of clusters. We computed the runtime of each iteration for various network size and observed that it matches the theoretical complexity of DNMF.

In another experiment, we observed that our algorithm outperforms other state-of-art methods such as FacetNet [6], AFFECT [78], and Spectral Clustering [19] in terms of NMI. We also noticed that our dynamic version of our algorithm significantly outperforms its

static counterpart, indicating the importance of exploiting temporal smoothness. Finally, we analysed the performance of DNMF in a real network. We studied the case of the dynamic email network of KIT, for which we had information on the chair of each node. This enabled us to form a reference clustering and assess whether the partitioning identified by DNMF was strongly related to the reference.

Chapter 5

Conclusion and Future Work

5.1 Summary

The work conducted in this thesis includes a literature review on clustering algorithms, focusing on clustering in dynamic networks, and the proposal of a novel technique to be employed in dynamic networks for community detection.

In the literature review chapter, we introduced the main static clustering approaches and then provided a detailed review of dynamic clustering methods. As we pointed out, there has been a surge of interest in model-based clustering approaches recently [4–6,16,78].

Our proposed algorithm can also be viewed as a model-based clustering technique. We wanted our model to identify clusters over time, taking into account both the observed data of the current time and the clusters identified at the previous timestep. This way the clusters do not substantially deviate over time and hence the structural continuity of the network is preserved.

In Chapter 3 we proposed a novel algorithm, called Dynamic Non-negative Matrix Factorization (DNMF), for detecting communities in dynamic networks. Our technique

can be considered as an extension of the static clustering scheme of Psorakis et al. [21]. We presented a probabilistic generative model and used the non-negative matrix factorization technique to identify the clusters and incorporate the historic network data. The dynamic and static part of the model govern how the latent variables are inspired from previous ones and how the observations are generated. The use of non-negative matrix factorization is helpful for detecting highly connected components of the network. It is worth noting that due to the nature of the defined latent variables, clusters can be obtained either as a hard or soft partitioning of nodes in both directed and undirected networks.

The next step of our algorithm, parameter inference, is formulated as a MAP estimation problem. The posterior probability of observations is written in terms of latent parameters and then optimized by a coordinate descent procedure. Although coordinate descent does not ensure that we reach the global maximum of the posterior likelihood, we show in Chapter 4 that it gives reasonable results when applying DNMF on synthetic datasets. We also derived the computational complexity of our algorithm and found that DNMF requires $\mathcal{O}(N^2)$ operations.

Our algorithm employs Automatic Relevance Determination [21, 45] to estimate the effective number of communities in the case that we do not have any information about it beforehand. Automatic Relevance Determination applies a regularization so that weak clusters tend to be absorbed by the dominant ones. This way if we start our algorithm with a large enough number of clusters, the algorithm automatically eliminates unnecessary clusters and keeps the strong (genuine) ones. This procedure has the advantage that the number of clusters can be identified in a single run of the algorithm. Modularity-based approaches, in contrast, require multiple runs of the algorithm; hierarchical clustering methods must identify a complete hierarchy of partitions.

In Chapter 4 we studied the performance of our algorithm for various parameter set-

tings. We compared its performance to several state of-the-art algorithms using synthetic networks. We also applied DNMF to an email graph and showed that our method is able to capture interesting aspects of this real network.

5.2 Future Work

Our current proposed method has some parameters to be set before execution. Selecting these parameters when we do not have any information about the network is not a trivial task. A potential extension to our algorithm involves reducing the number of parameters or proposing a procedure for parameter estimation.

The coordinate descent procedure we employed in the parameter inference part of our algorithm can become trapped in local minima. The adoption of other inference tools could help us to avoid or ameliorate this problem. We could also employ inference techniques with lower computational costs which would make DNMF more readily scalable to large networks.

In this thesis, as mentioned before, we only considered networks with one type of interaction. Another potential extension to the current work is generalizing DNMF so that it can be applied to social networks with several types of content and various modes of interaction. One promising avenue for achieving this is tensor factorization [96].

Appendix A

Statistical Significance Tests

A.1 Lillierfors Test

The Lilliefors test is an adaptation of the Kolmogorov–Smirnov (KS) test [105]. It is used to test the null hypothesis, H_0, that data x come from a normally distributed population against the hypothesis that the data does not follow a normal distribution, H_1. Unlike the KS test, the null hypothesis in Lillierfors test does not specify which normal distribution; i.e., it does not specify the expected value and variance of the distribution. The Lilliefors test statistic is the same as for the Kolmogorov-Smirnov test:

$$KS = \max_x |SCDF(x) - CDF(x)|, \tag{A.1}$$

where SCDF is the empirical cdf estimated from the sample and CDF is the normal CDF with mean and standard deviation equal to the mean and standard deviation of the sample. In practice, given a group of data sets, the hypothesis regarding the normal distributional form, H_0, is rejected if the test statistic, KS, is greater than the critical value obtained

from a table which depends on a significance level α.

A.2 One-way analysis of variance (ANOVA) Test

There are three assumptions to be satisfied in order to be able to apply ANOVA to a group of data [106]. These assumptions are as follow

1. observations should randomly and independently be chosen from various groups;
2. the observation should be normally distributed; and
3. population variances are equal for all groups.

Note that Assumption 3 can be relaxed when the sample sizes are roughly the same for each group. In ANOVA test, we want to test that the mean values of some groups of data are due to chance (H_0; null hypothesis) or actual differences in the means (H_1). We assume that we have n groups of data, each has sample size of N and want to test the null hypothesis for them. The data points are denoted by X_{ij}, $i = 1, 2, ..., n$, $j = 1, 2, ..., N$. To perform ANOVA test we first need to define within-group estimate (MS_{WG}) and between-group estimate (MS_{BG}). They are defined as:

$$MS_{WG} = \frac{\sum_{ij}(X_{ij} - \bar{X}_j)}{n(N-1)}, \tag{A.2}$$

where $\bar{X}_j = \sum_i X_{ij}/N$ and

$$MS_{BG} = \frac{\sum_i N(\bar{X}_j - \bar{X})}{(n-1)}, \tag{A.3}$$

where $\bar{X} = \sum_{ij} X_{ij}/(nN)$. The main purpose of ANOVA is that these two estimates of variance are expected to be equal if the means of different groups are actually the same

(H_0); otherwise they are expected to differ (H_1). To find out how likely it is that the two given estimates of variance would differ that much if we have equal means for all groups (null hypothesis), a parameter denoted by F is defined as

$$F = \frac{MS_{BG}}{MS_{WG}} \tag{A.4}$$

.

The parameter F is then compared to a critical value which is the number that the test statistic, F, must exceed to reject the null hypothesis. The critical value can be computed from pre-computed tables which depends on the parameters N, n, and significance level α.

References

[1] W. W. Zachary, "An information flow model for conflict and fission in small groups," *J. Anthropological Research*, vol. 33, no. 4, pp. 452–473, 1977.

[2] V. Batagelj and A. Mrvar, "Pajek - program for large network analysis," *J. Connections*, vol. 21, pp. 47–57, 1998.

[3] P. J. Mucha, T. Richardson, K. Macon, M. A. Porter, and J. Onnela, "Community structure in time-dependent, multiscale, and multiplex networks," *J. Science*, vol. 328, pp. 876–878, May. 2010.

[4] P. Sarkar and A. W. Moore, "Dynamic social network analysis using latent space models," *ACM SIGKDD Explorations Newsletter*, vol. 7, pp. 31–40, Dec. 2005.

[5] T. Yang, Y. Chi, S. Zhu, Y. Gong, and R. Jin, "Detecting communities and their evolutions in dynamic social networks–a bayesian approach," *J. Machine Learning*, vol. 82, pp. 157–189, Feb. 2011.

[6] Y. Lin, Y. Chi, S. Zhu, H. Sundaram, and B. L. Tseng, "Facetnet: a framework for analyzing communities and their evolutions in dynamic networks," in *Proc. int. conf. on World Wide Web*, (Beijing, China), pp. 685–694, 2008.

[7] K. Xu, M. Kliger, and A. Hero III, "Evolutionary spectral clustering with adaptive forgetting factor," in *Proc. IEEE Conf. on Acous. Speech and Sig. Proc.*, (Dallas, TX, USA), pp. 2174–2177, 2010.

[8] S. Wasserman and K. Faust, *Social network analysis: Methods and applications*. Cambridge University Press, 1994.

[9] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *J. Proc. of National Academy of Sciences*, vol. 99, pp. 7821–7826, Jun. 2002.

[10] J. M. Montoya, S. L. Pimm, and R. V. Solé, "Ecological networks and their fragility," *J. Nature*, vol. 442, no. 7100, pp. 259–264, 2006.

[11] H. Jeong, B. Tombor, R. Albert, Z. N. Oltvai, and A. L. Barabasi, "The large-scale organization of metabolic networks," *J. Nature*, vol. 407, pp. 651–654, 2000.

[12] A. E. Krause, K. A. Frank, D. M. Mason, R. E. Ulanowicz, and W. W. Taylor, "Compartments revealed in food-web structure," *J. Nature*, vol. 426, pp. 282–285, 2003.

[13] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *J. Physical Review E*, vol. 69, p. 026113, 2004.

[14] D. J. C. Mackay, *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, first ed., Jun. 2003.

[15] J. D. J. Han, "Evidence for dynamically organized modularity in the yeast protein-protein interaction network," *J. Nature*, vol. 430, pp. 88–93, 2004.

[16] Y. Park, C. Moore, and J. S. Bader, "Dynamic networks from hierarchical bayesian graph clustering," *J. PLoS ONE*, vol. 5, Jan. 2010.

[17] B. Ball, B. Karrer, and M. E. J. Newman, "Efficient and principled method for detecting communities in networks," *J. Physical Review E*, vol. 84, p. 036103, Sep. 2011.

[18] G. Palla, I. Derenyi, I. Farkas, and T. Vicsek, "Uncovering the overlapping community structure of complex networks in nature and society," *J. Nature*, vol. 435, pp. 814–818, Jun. 2005.

[19] Y. Chi, X. Song, D. Zhou, K. Hino, and B. Tseng, "Evolutionary spectral clustering by incorporating temporal smoothness," in *Proc. ACM SIGKDD int. conf. on Knowledge discovery and data mining*, (San Jose, CA, USA), pp. 153–162, 2007.

[20] W. Cai, S. Chen, and D. Zhang, "Fast and robust fuzzy c-means clustering algorithms incorporating local information for image segmentation," *J. Pattern Recognition*, vol. 40, pp. 825–838, Mar. 2007.

[21] I. Psorakis, S. Roberts, M. Ebden, and B. Sheldon, "Overlapping community detection using bayesian non-negative matrix factorization," *J. Physical Review E*, vol. 83, p. 066114, Jun. 2011.

[22] U. Brandes, M. Gaertler, and D. Wagner, "Experiments on graph clustering algorithms," in *Proc. of Europ. Symp. Algorithms*, (Budapest, Hungary), pp. 568–579, Springer-Verlag, Sep. 2003.

[23] U. Brandes and T. Erlebach, *Network Analysis: Methodological Foundations*. Springer, 2005.

[24] P. K. Chan, M. D. F. Schlag, and J. Y. Zien, "Spectral k-way ratio-cut partitioning and clustering," in *Proc. int. Design Automation Conf.*, (Dallas, TX, USA), pp. 749–754, 1993.

[25] S. Yu and J. Shi, "Multiclass spectral clustering," in *Proc. IEEE Int. Conf. on Computer Vision*, (Nice, France), pp. 313–319, Oct. 2003.

[26] M. E. J. Newman, "Mixing patterns in networks," *Physical Review E*, vol. 67, p. 026126, Feb. 2003.

[27] J. Reichardt and S. Bornholdt, "Statistical mechanics of community detection," *Physical Review E*, vol. 74, p. 016110, Jul. 2006.

[28] R. Lambiotte, "Multi-scale modularity in complex networks," in *Proc. Int. Symp. on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, pp. 546–553, Jun. 2010.

[29] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference and prediction*. Springer, second ed., 2009.

[30] M. E. J. Newman, "Fast algorithm for detecting community structure in networks," *Physical Review E*, vol. 69, p. 066133, Jun. 2004.

[31] V. D. Blondel, J. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *J. Statistical Mechanics: Theory and Experiment*, vol. 2008, p. P10008, Oct. 2008.

[32] S. S. Tabatabaei, M. Coates, and M. Rabbat, "Ganc: Greedy agglomerative normalized cut for graph clustering.," *J. Pattern Recognition*, vol. 45, no. 2, pp. 831–843, 2012.

[33] S. Fortunato, "Community detection in graphs," *J. Physics Reports*, vol. 486, no. 3–5, pp. 75–174, 2010.

[34] U. Luxburg, "A tutorial on spectral clustering," *J. Statistics and Computing*, vol. 17, pp. 395–416, Dec. 2007.

[35] M. Maila and J. Shi, "A random walks view of spectral segmentation," in *Workshop on Artifcial Intelligence and Statistics*, (Key West, FL, USA), Jan. 2001.

[36] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 888–905, Aug. 2000.

[37] A. Ng, M. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances in Neural Information Processing Systems*, (Vancouver, BC,Canada), pp. 849–856, Dec. 2001.

[38] M. Gaertler, *Clustering with Spectral Methods*. Universität Konstanz, Mar. 2002.

[39] A. Lancichinetti, S. Fortunato, and J. Kertész, "Detecting the overlapping and hierarchical community structure in complex networks," *New J. of Physics*, vol. 11, no. 3, p. 033015, 2009.

[40] C. Lee, F. Reid, A. McDaid, and N. Hurley, "Detecting highly overlapping community structure by greedy clique expansion," in *Proc. SNA-KDD Workshop*, (Washington, DC, USA), Jul. 2010.

[41] F. Havemann, M. Heinz, A. Struck, and G. J., "Identification of overlapping communities and their hierarchy by locally calculating community-changing resolution levels," *J. Stat. Mech.: Theory and Experiment*, vol. 2011, p. P01023, Jan. 2011.

[42] J. Baumes, M. Goldberg, and M. Magdon-ismail, "Efficient identification of overlapping communities," in *Proc. IEEE Int. Conf. on Intelligence and Security Informatics*, (Atlanta, GA, USA), pp. 27–36, 2005.

[43] K. Yu, S. Yu, and V. Tresp, "Soft clustering on graphs," in *Proc. Advances in Neural Information Processing Systems*, p. 05, 2005.

[44] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *J. Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[45] V. Y. F. Tan and C. Févotte, "Automatic relevance determination in nonnegative matrix factorization," in *Proc. Int. Workshop Sig. Proc. with Adaptive Sparse Structured Representations*, Mar. 2009.

[46] S. Fortunato and M. Barthélemy, "Resolution limit in community detection," *J. Proc. of the National Academy of Sciences*, vol. 104, no. 1, pp. 36–41, 2007.

[47] T. Nepusz, A. Petróczi, L. Négyessy, and F. Bazsó, "Fuzzy communities and the concept of bridgeness in complex networks," *Physical Review E*, vol. 77, p. 016107, Jan. 2008.

[48] S. Zhang, R. Wang, and X. Zhang, "Identification of overlapping community structure in complex networks using fuzzy -means clustering," *J. Physica A: Statistical Mechanics and its Applications*, vol. 374, no. 1, pp. 483–490, 2007.

[49] S. Gregory, "Finding overlapping communities in networks by label propagation," *New Journal of Physics*, vol. 12, no. 10, p. 103018, 2010.

[50] R. Kumar, J. Novak, and A. Tomkins, "Structure and evolution of online social networks," in *Proc. ACM SIGKDD int. conf. on Knowledge discovery and data mining*, (Philadelphia, PA, USA), pp. 611–617, Aug. 2006.

[51] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graphs over time: densification laws, shrinking diameters and possible explanations," in *Proc. ACM SIGKDD int. conf. on Knowledge discovery in data mining*, (Chicago, IL, USA), pp. 177–187, Aug. 2005.

[52] J. Leskovec, L. Backstrom, R. Kumar, and A. Tomkins, "Microscopic evolution of social networks," in *Proc. ACM SIGKDD int. conf. on Knowledge discovery and data mining*, pp. 462–470, Aug. 2008.

[53] C. Tantipathananandh, T. Berger-Wolf, and D. Kempe, "A framework for community identification in dynamic social networks," in *Proc. ACM SIGKDD int. conf. on Knowledge discovery and data mining*, (San Jose, CA, USA), pp. 717–726, Aug. 2007.

[54] M. Spiliopoulou, "Evolution in social networks: A survey.," in *Social Network Data Analytics*, pp. 149–175, Springer, 2011.

[55] D. Chakrabarti, R. Kumar, and A. Tomkins, "Evolutionary clustering," in *Proc. ACM SIGKDD int. conf. on Knowledge discovery and data mining*, (Philadelphia, PA, USA), pp. 554–560, 2006.

[56] M. Toyoda and M. Kitsuregawa, "Extracting evolution of web communities from a series of web archives," in *Proc. the ACM conf. on Hypertext and hypermedia*, (Nottingham, UK), pp. 28–37, Aug. 2003.

[57] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," *J. ACM*, vol. 46, pp. 604–632, Sep. 1999.

[58] J. Hopcroft, O. Khan, B. Kulis, and B. Selman, "Tracking evolving communities in large linked networks," *Proc. National Academy of Sciences*, vol. 101, pp. 5249–5253, Apr. 2004.

[59] C. C. Aggarwal and P. S. Yu, "Online analysis of community evolution in data streams," in *Proc. SIAM Int. Data Mining Conf.*, (Newport Beach, CA, USA), 2005.

[60] T. Falkowski, A. Barth, and M. Spiliopoulou, "Studying community dynamics with an incremental graph mining algorithm," in *Proc. Americas Conf. on Information Systems*, (Toronto, ON, Canada), Aug. 2008.

[61] M. Ester, H. Kriegel, J. Sander, M. Wimmer, and X. Xu, "Incremental clustering for mining in a data warehousing environment," in *Proc. Int. Conf. on Very Large Data Bases*, (New York City, NY, USA), pp. 323–333, Aug. 1998.

[62] J. Sun, C. Faloutsos, S. Papadimitriou, and P. S. Yu, "Graphscope: parameter-free mining of large time-evolving graphs," in *Proc. ACM SIGKDD int. conf. on*

Knowledge discovery and data mining, (San Jose, CA, USA), pp. 687–696, Aug. 2007.

[63] J. Ferlez, C. Faloutsos, J. Leskovec, D. Mladenic, and M. Grobelnik, "Monitoring network evolution using mdl," in *Proc. IEEE Int. Conf. on Data Engineering*, (Cancun, Mexico), pp. 1328–1330, Apr. 2008.

[64] G. Palla, A. Barabasi, and T. Vicsek, "Quantifying social group evolution," *J. Nature*, vol. 446, pp. 664–667, Apr. 2007.

[65] S. Asur, S. Parthasarathy, and D. Ucar, "An event-based framework for characterizing the evolutionary behavior of interaction graphs," *J. ACM Trans. Knowledge Discovery Data*, vol. 3, pp. 16:1–16:36, Dec. 2009.

[66] P. Brodka, S. Saganowski, and P. Kazienko, "Group evolution discovery in social networks," in *Proc. Int. Conf. on Advances in Social Networks Analysis and Mining*, (Kaohsiung, Taiwan), pp. 247 –253, Jul. 2011.

[67] P. Brodka, K. Musial, and P. Kazienko, "A performance of centrality calculation in social networks," in *Proc. Int. Conf. on Computational Aspects of Social Networks*, (Fontainebleau, France), pp. 24 –31, Jun. 2009.

[68] D. Greene, D. Doyle, and P. Cunningham, "Tracking the evolution of communities in dynamic social networks," in *Proc. Int. Conf. on Advances in Social Networks Analysis and Mining*, (Odense, Denmark), pp. 176–183, Aug. 2010.

[69] M. K. Goldberg, M. Magdon-Ismail, S. Nambirajan, and J. Thompson, "Tracking and predicting evolution of social communities.," in *Proc. IEEE Int. Conf. Social Computing*, (Boston, MA, USA), pp. 780–783, Oct. 2011.

[70] H. Ning, W. Xu, Y. Chi, Y. Gong, and T. S. Huang, "Incremental spectral clustering by efficiently updating the eigen-system," *J. Pattern Recognition*, vol. 43, pp. 113–127, Jan. 2010.

[71] R. Görke, T. Hartmann, and D. Wagner, "Dynamic graph clustering using minimum-cut trees," in *Proc. Int. Symp. Algorithms and Data Structures*, (Banff, AB, Canada), pp. 339–350, 2009.

[72] G. W. Flake, R. E. Tarjan, and K. Tsioutsiouliklis, "Graph clustering and minimum cut trees," *J. Internet Mathematics*, vol. 1, no. 4, pp. 385–408, 2004.

[73] D. J. Fenn, M. A. Porter, M. McDonald, S. Williams, N. F. Johnson, and N. S. Jones, "Dynamic communities in multichannel data: An application to the foreign exchange market during the 2007–2008 credit crisis," *J. Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 19, no. 3, pp. 033119+, 2009.

[74] Y. Chi, X. Song, D. Zhou, K. Hino, and B. L. Tseng, "On evolutionary spectral clustering," *J. ACM Trans. Knowledge Discovery Data*, vol. 3, pp. 17:1–17:30, Dec. 2009.

[75] M. Kim and J. Han, "Chronicle: A two-stage density-based clustering algorithm for dynamic networks," in *Proc. Int. Conf. on Discovery Science*, (Porto, Portugal), pp. 152–167, Oct. 2009.

[76] M. Kim and J. Han, "A particle-and-density based evolutionary clustering method for dynamic networks," *J. Proc. VLDB Endow.*, vol. 2, pp. 622–633, Aug. 2009.

[77] R. Lambiotte, J. C. Delvenne, and M. Barahona, "Laplacian dynamics and multiscale modular structure in networks," *http://arxiv.org/abs/0812.1770*, 2008.

[78] K. Xu, M. Kliger, and A. Hero III, "Adaptive evolutionary clustering," *http://arxiv.org/abs/1104.1990*, Apr. 2011.

[79] O. Ledoit and M. Wolf, "Improved estimation of the covariance matrix of stock returns with an application to portfolio selection," *J. Empirical Finance*, vol. 10, pp. 603–621, Dec. 2003.

[80] F. Folino and C. Pizzuti, "A multiobjective and evolutionary clustering method for dynamic networks," in *Proc. Int. Conf. on Advances in Social Networks Analysis and Mining*, pp. 256–263, 2010.

[81] C. Pizzuti, "Ga-net: A genetic algorithm for community detection in social networks," in *Proc. Int. Conf. on Parallel Problem Solving from Nature*, vol. 5199, pp. 1081–1090, 2008.

[82] Y. Lin, Y. Chi, S. Zhu, H. Sundaram, and B. L. Tseng, "Analyzing communities and their evolutions in dynamic social networks," *J. ACM Trans. Knowledge Discovery Data*, vol. 3, pp. 8:1–8:31, Apr. 2009.

[83] A. Clauset, C. Moore, and M. E. J. Newman, "Structural inference of hierarchies in networks," in *Proc. conf. on Statistical network analysis*, (Pittsburgh, PA, USA), pp. 1–13, 2007.

[84] A. Clauset, C. Moore, and M. E. J. Newman, "Hierarchical structure and the prediction of missing links in networks," *J. Nature*, vol. 453, pp. 98–101, Feb. 2008.

[85] Q. Mei and C. Zhai, "Discovering evolutionary theme patterns from text: an exploration of temporal text mining," in *Proc. ACM SIGKDD int. conf. on Knowledge discovery and data mining*, pp. 198–207, Aug. 2005.

[86] D. M. Blei and J. D. Lafferty, "Dynamic topic models," in *Proc. int. conf. on Machine learning*, (Pittsburgh, PN, USA), pp. 113–120, Jun. 2006.

[87] A. Ahmed and E. P. Xing, "Timeline: A dynamic hierarchical dirichlet process model for recovering birth/death and evolution of topics in text stream.," in *Uncertainty in Artificial Intelligence*, pp. 20–29, 2010.

[88] T. Xu, Z. Zhang, P. S. Yu, and B. Long, "Dirichlet process based evolutionary clustering," in *Proc. IEEE Int. Conf. on Data Mining*, (Pissa, Italy), pp. 648–657, Dec. 2008.

[89] T. Xu, Z. Zhang, P. S. Yu, and B. Long, "Evolutionary clustering by hierarchical dirichlet process with hidden markov state," in *Proc. IEEE Int. Conf. on Data Mining*, (Pissa, Italy), Dec. 2008.

[90] D. Blei, A. Y. Ng, and M. Jordan, "Latent dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, Mar. 2003.

[91] X. Wang and A. McCallum, "Topics over time: a non-markov continuous-time model of topical trends," in *ACM SIGKDD int. conf. on Knowledge discovery and data mining*, (Philadelphia, PA, USA), pp. 424–433, 2006.

[92] C. Wang, D. Blei, and D. Heckerman, "Continuous time dynamic topic models," in *Conf. in Uncertainty in Artificial Intelligence*, 2008.

[93] K. Miller and T. Eliassi-Rad, "Continuous time group discovery in dynamic graphs," in *Analyzing Networks and Learning with Graphs NIPS*, 2009.

[94] L. Tang, H. Liu, J. Zhang, and Z. Nazeri, "Community evolution in dynamic multi-mode networks," in *Proc. ACM SIGKDD int. conf. on Knowledge discovery and data mining*, pp. 677–685, Aug. 2008.

[95] Y. Sun, J. Tang, J. Han, M. Gupta, and B. Zhao, "Community evolution detection in dynamic heterogeneous information networks," in *Proc. Workshop on Mining and Learning with Graphs*, (Washington, DC, USA), pp. 137–146, Jul. 2010.

[96] Y. Lin, J. Sun, P. Castro, R. Konuru, H. Sundaram, and A. Kelliher, "Metafac: community discovery via relational hypergraph factorization," in *Proc. ACM SIGKDD int. conf. on Knowledge discovery and data mining*, (Paris, France), pp. 527–536, Jun. 2009.

[97] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Proc. Advances in Neural Information Processing Systems*, pp. 556–562, Apr. 2001.

[98] Z. Yang and J. Laaksonen, "Multiplicative updates for non-negative projections," *J. Neurocomputing*, vol. 71, pp. 363–373, Dec. 2007.

[99] J. Zhang, Y. Song, G. Chen, and C. Zhang, "On-line evolutionary exponential family mixture," in *Proc. int. jont conf. on Artifical intelligence*, (Pasadena, CA, USA), pp. 1610–1615, 2009.

[100] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *J. American Statistical Association*, vol. 66, pp. 846–850, Dec. 1971.

[101] P. Jaccard, "Étude comparative de la distribution florale dans une portion des Alpes et des Jura," *Bulletin del la Société Vaudoise des Sciences Naturelles*, vol. 37, pp. 547–579, 1901.

[102] L. Danon, A. Diaz-Guilera, J. Duch, and A. Arenas, "Comparing community structure identification," *J. Statistical Mechanics: Theory and Experiment*, vol. 2005, p. P09008, Sep. 2005.

[103] R. Gorke, I. Holzer, M. of Wagner, T. J. Hopp, O., and K. Scheibenberger, "Dynamic network of email communication at the department of informatics," tech. rep., Karlsruhe Institute of Technology (KIT), 2011.

[104] R. Gorke, *An algorithmic walk from static to dynamic graph clustering.* PhD thesis, Karlsruhe Institute of Technology, 2010.

[105] H. Lilliefors, "On the kolmogorov-smirnov test for normality with mean and variance unknown," *J. American Statistical Association*, vol. 62, pp. 399–402, Jun. 1967.

[106] W. T. Eadie, D. Drijard, F. E. James, M. Roos, and B. Sadoulet, *Statistical methods in experimental physics.* American Elsevier Pub. Co, 1st ed., 1971.