Communities in Evolving Networks: Definitions, Detection, and Analysis Techniques

Thomas Aynaud, Eric Fleury, Jean-Loup Guillaume, and Qinna Wang

1 Introduction

A complex network models multiple interactions between components in a complex system. A complex network can be represented by a graph. A diverse range of networks have been studied, for example, the Internet [34], World Wide Web, citation networks [35], coauthorship networks [50], metabolic networks [43], and social networks whose nodes are connected by one or more specific types of relations, like friendship, workplace, and common interest [5, 9]. In the study of these networks, the topic of community structure has attracted much attention in several fields.

A network has an underlying community structure if its nodes can be "naturally" grouped into sets such that each set of nodes is densely connected internally, and these groups of nodes are called *communities*. While other definitions of community exist, not related to the topology but, for instance, to common interests, we refer here to community structure as a network structural property. For instance, in protein–protein interaction networks, communities correspond to specific functions [15]; in the World Wide Web, communities may be related to topics [22]; and in food webs, communities correspond to compartments [51]. Studies in community structure should lead to a better understanding of complex systems.

T. Aynaud (✉) • J.-L. Guillaume
UPMC, CNRS (UMR 7606), France
e-mail: taynaud@gmail.com; jean-loup.guillaume@lip6.fr

E. Fleury
ENS de Lyon (UMR CNRS – ENS de Lyon – UCB Lyon 1 – Inria 5668), France
e-mail: eric.fleury@inria.fr

Q. Wang
Inria (UMR CNRS – ENS de Lyon – UCB Lyon 1 – Inria 5668), France
e-mail: qinna.wang@inria.fr

In the past, communities in networks have been widely studied. Uncovering community methods proposed usually deal with *static* graphs. Indeed, even if the data observed span a time interval, and not only a specific snapshot, a *static* graph is extracted from the dataset by aggregating interaction behaviors on the whole time interval yielding a static graph, possibly weighted. However, an important property has been largely ignored: networks tend to evolve over time. The interaction between components in a complex system is not stable but changes over time. For example, *Facebook* and *Myspace* social network sites have grown dramatically in recent years, and many people join or post over time. There are also many other dynamic networks with millions of nodes [80, 87]. To learn about how a network evolves over time, it is important to develop methods that allow to study the dynamic aspects of the intrinsic network structure, namely, its community dynamics. In a seminal paper published in 2007, Palla et al. [70] have proposed six possible scenarios that may occur during the evolution of communities: *birth* (a new community appears), *growth* (a growing community), *merging* (merging communities), *contracting* (a shrunken community), *splitting* (a split into communities), and *death* (a community vanishes). In [13], the definition of *change point* is given to represent a significant time point when the system evolves, that is, a major change (or critical event) occurs in the graph structure during a short period.

Several problems and challenges remain while studying community evolution. A first one is that there is no well-established standard definition of the notion itself when dealing with dynamic networks. Some authors are in favor to define a community as a structure that is observable over time. Such communities can be detected by matching observable communities at different time steps. Some authors propose to define a community as a structure that evolves over time. Therefore, communities can be detected by incremental updating. Authors also propose to transform the problem of community detection in dynamic networks into a problem of community detection in static networks. Different approaches are proposed to detect communities and analyze community evolution in dynamic networks.

In this survey, we try to detail current work on community detection in dynamic networks. We review these methods and discuss their performance in analyzing networks. This survey is organized as follows: we first describe the definition of community in static networks (see Sect. 2). In Sect. 3, we present different dynamic models such as network evolution models and community evolution models. These models are the principles of algorithms designed for tracking community evolution. In Sect. 4, we describe different methods for community detection in dynamic networks, discuss how to test them, and visualize their results. Section 5 contains the summary of this survey, along with a discussion about issues raised in tracking community evolution.

2 Communities in Networks

It appears natural to model the topology structure of a complex system by a graph (one can equally use the term network). Many real-world problems (biological, social, web) can be effectively modeled by graphs where nodes represent entities of interest and edges mimic the interactions or relationships among them.

In the study of networks, such as computer networks, information networks, social networks, or biological networks, uncovering the underlying community structure is essential. Social networks often include community groups based on common location, interests, hobbies, etc. Metabolic networks have communities based on modular functions [76]. Citation networks form communities by research topics. In each context, communities in a network can be described by dense groups of nodes, with more edges inside groups than edges linking the rest of the network.

In the following, we introduce the definition of community which depends on the context. Social network analysts have devised many definitions of communities with various degrees of internal cohesion among nodes [46, 80]. Many other definitions have been introduced by computer scientists and physicists. We distinguish three main classes of definitions: local, global, and based on vertex similarity. We review the notion of community structure. We also discuss the definition of the modularity function, derived to measure the quality of a graph partition into communities.

2.1 Preliminaries, Notations, and Definitions

A graph $G = (V, E)$ consists of two sets V and E, where $V = \{v_1, v_2, \dots, v_n\}$ are the nodes (or vertices, or points) of the graph G and $E \subseteq V \times V$ are its links (or edges, or lines). The number of elements in V and E are denoted by $n = |V|$ and $m = |E|$, respectively.

In the context of graph theory, an adjacency (or connectivity) matrix $\mathbf{A}$ is often used to describe a graph G. Specifically, the adjacency matrix of a finite graph G on n vertices is the $n \times n$ matrix $\mathbf{A} = [A_{ij}]_{n\times n}$, where an entry A_{ij} of $\mathbf{A}$ is equal to 1 if the link $e_{ij} = (v_i, v_j) \in E$ exists, and zero otherwise.

A *partition* is a division of a graph into disjoint communities, such that each node belongs to a unique community. A division of a graph into overlapping (or fuzzy) communities is called a *cover*. We use $\mathcal{P} = \{\mathcal{C}_1, \dots, \mathcal{C}_{n_c}\}$ to denote a partition, which is composed of n_c communities. In $\mathcal{P}$, the community to which the node v belongs to is denoted by σ_v. By definition we have $V = \cup_1^{n_c} C_i$ and $\forall i \neq j, C_i \cap C_j = \emptyset$. We denote by $\mathcal{S} = \{S_1, \dots, S_{n_c}\}$ a cover composed of n_c communities. In $\mathcal{S}$, we may find a pair of community S_i and S_j with $i \neq j$ such that $S_i \cap S_j \neq \emptyset$.

Given a community $\mathcal{C} \subseteq V$ of a graph $G = (V, E)$, we define the internal degree $k_v^{\text{int}} = |\{e = (v, u) \mid u \in \mathcal{C}\}|$ (respectively the external degree $k_v^{\text{ext}} = |\{e = (v, u) \mid u \notin \mathcal{C}\}|$) of a node $v \in \mathcal{C}$, as the number of edges connecting v to other nodes u belonging to $\mathcal{C}$ (respectively to the rest of the graph). If $k_v^{\text{ext}} = 0$, the only neighbors of node v are within $\mathcal{C}$: assigning v to the current community $\mathcal{C}$ is likely to be a good choice. If $k_v^{\text{int}} = 0$ instead, the node is disjoint from $\mathcal{C}$ and it should be better to assign it to a different community. Classically, we note $k_v = k_v^{\text{int}} + k_v^{\text{ext}}$ the degree of node v. The internal degree k^{int} of $\mathcal{C}$ is the sum of the internal degrees of its nodes: $k^{\text{int}} = \sum_{v \in \mathcal{C}} k_v^{\text{int}}$. Likewise, the external degree k^{ext} of $\mathcal{C}$ is the sum of the external degrees of its nodes. The total degree $k_{\mathcal{C}}$ is the sum of the degrees of the nodes of $\mathcal{C}$. By definition, $k_{\mathcal{C}} = k_{\mathcal{C}}^{\text{int}} + k_{\mathcal{C}}^{\text{ext}}$.

2.2 Definitions of a Community

When studying community structure, authors often analyze structural properties of communities in the networks. The notion of communities can be formalized based on statistical properties. Furthermore, we can distinguish three types of community definition: local definition, global definition, and the definition based on node similarity. There, definitions are used to automatically detect community structure in networks.

Local Definitions

Communities are parts of the graph (group of nodes), within which the connections are dense and between which the connections are sparse. In some specific systems or applications, they can be considered as separate entities with their own autonomy, which do not depend on the whole graph. For instance, in [59], communities are defined in a very strict sense and require that all pairs of nodes are connected. In other words, this corresponds to a clique, that is, a subset of nodes such that every two vertices in the subset are connected by an edge. However, such criteria are too strict. A relaxable extended definition is *k-clique community*, which is the basis of CPM (Clique Percolation Method) [71]. A *k-clique community* is a sequence of adjacent cliques, where two k-cliques are *adjacent* if they share k-1 nodes.

Another criterion for community cohesion is the difference between the internal and external cohesion of the community. This idea is also used to define communities. For instance, Radicchi et al. [75] proposed the definitions of *strong communities* and *weak communities*. A set of nodes is a community in a strong sense if the internal degree of each node is greater than its external degree. This definition seems too strict. Its relaxable definition is the community in a weak sense: the internal degree of the community (sum of all its node internal degree) should exceed its external degree. Note that a community in a strong sense is also a weak community, while the converse is not generally true.

Global Definitions

Communities can be defined with respect to the graph as a whole. This seems to be reasonable when the community structure is exactly the division of the graph into several groups of nodes. In such a context, many global criteria are used to identify communities, which are all based on the intrinsic idea that a graph offers a community structure if its structure is far from a random graph. Random networks such as Erdös–Renyi's graphs do not display community structure. Indeed, as any pair of nodes are independently linked with the same probability, there should be no preferential wiring involving special groups of nodes. Therefore, one may define a *null model*, that is, a random graph that shares some structural properties of the original graph such as its degree distribution. The null model is the basic element in the conception of the notion of quality function named *modularity*. The modularity evaluates the quality of a graph partition into disjoint communities. The most popular modularity is proposed by Newman and Girvan [65], which compares the number of edges inside the community to the expected number of internal edges in the null model. A series of algorithms using modularity maximization heuristics [28, 66] for finding communities are proposed and developed.

Definitions Based on Node Similarity

It seems also natural to assume that communities are groups of nodes similar to each other. One can compute the similarity between each pair of nodes with respect to some reference properties. An important class of node similarity measures is based on properties of random walks on graphs, such as *commute time*. The *commute time* between a pair of nodes is the average number of steps needed for a random walker, starting at either node, to reach the other node for the first time and to come back to the starting node. Saerens et al. [79] have studied and used the commute time as a similarity measure: the larger the commute time is, the less similar the nodes are.

2.3 Modularity

One may want to measure the quality of a partition through a *quality function*, which assigns a score to each partition of a graph. In this way, partitions can be ranked based on their score given by the quality function. Partitions with high scores are *"good"*, so the one with the highest score is by definition the best.

The widest accepted quality function is the modularity introduced by Newman and Girvan [65, 68]. Let e_{ij} be the fraction of edges in the network that connect nodes in community i to those in community j, and $a_i = \sum_j e_{ij}$. The modularity measure is defined as

$$Q = \sum_i \left(e_{ii} - a_i^2\right). \tag{1}$$

This quantity measures the fraction of the within-community edges in the network minus the expected value in a network with the same community division but when connections between nodes are random. If the number of within-community edges is less than the expected number of edges in a random graph, we will get $Q = 0$. Values approaching $Q = 1$, which is the maximum, indicate networks with strong community structure. In practice, values for real networks typically fall in the range from 0.3 to 0.7. Higher values are rare.

Suppose we have a division of a network into communities. Let σ_i be the community to which node i is assigned. The fraction of the edges in the graph that fall within communities, that is, that connect nodes that both lie in the same community, is

$$\frac{\sum_{ij} A_{ij}\delta(\sigma_i,\sigma_j)}{\sum_{ij} A_{ij}} = \frac{1}{2m}\sum_{ij} A_{ij}\delta(\sigma_i,\sigma_j)$$

where the function $\delta(\sigma_i,\sigma_j)$ is 1 if $\sigma_i = \sigma_j$ and 0 otherwise. At the same time, the expected number of edges between nodes i and j, if edges are drawn at random, is $k_i k_j/2m$, where k_i and k_j are the degrees of the nodes and m is the total number of edges in the network. Thus, the modularity [64], as defined above, is given by

$$Q = \frac{1}{2m}\sum_{i\neq j}\left(A_{ij} - \frac{k_i k_j}{2m}\right)\delta(\sigma_i,\sigma_j) \tag{2}$$

Note that the modularity is always smaller than one but can be negative as well. For instance, the partition where each node represents a single community is always negative. When considering the whole graph as a single community, the modularity is zero as the two terms are equal and cancels each other out. There are also other types of modularity, some of which are motivated by specific classes of clustering problems or graphs. The goal here is not to list exhaustively all algorithms which were built upon the modularity. The interested reader should turn to Fortunato's extensive review of the field [32], which does not only cover the modularity but community detection as a whole.

Modularity has been employed as quality function in many algorithms, like some division algorithms [66] which give a trade-off between high accuracy and low complexity. In addition, modularity optimization is the most popular method for community detection. Heuristic proposed in [11] runs fast and handles very large-scale networks. Modularity also allows to assess the stability of partitions [60].

However, the applicability and reliability of modularity for the problem of graph clustering may be limited. An important issue concerning the limits of modularity is raised by Fortunato and Barthelemy [33]. The study shows that a large value for the maximum modularity does not necessarily mean that a graph has a clear community structure. In a random graph, such as the Erdös–Rényi model, the distribution of edges among the nodes is highly homogeneous. For instance, the distribution of the number of neighbors of a node, or degree, is binomial, so most nodes have equal or similar degree. The random graph is supposed to have no community structure, as the link probability between nodes is either constant or a function of the node degrees, so there is no bias a priori towards special groups of nodes. Still, random graphs may have partitions with large modularity values [42, 77]. This is due to fluctuations in the distribution of edges in the graph, which determine concentrations of links in some subsets of the graph, which then appear as communities.

Moreover, Fortunato and Barthelemy [33] have found that modularity optimization has a resolution limit. It may prevent from detecting communities which are comparatively small with respect to the graph as a whole. Given two communities $\mathcal{A}$ and $\mathcal{B}$, with a total degree $k_{\mathcal{A}}$ and $k_{\mathcal{B}}$, respectively, and where the number of edges connecting $\mathcal{A}$ and $\mathcal{B}$ is $l_{\mathcal{AB}}$, the difference of modularity determining the merger of two communities with respect to the whole graph partition is

$$\Delta Q = \left[\frac{k_{\mathcal{A}}^{int} + k_{\mathcal{B}}^{int} + 2l_{\mathcal{AB}}}{2m} - \left(\frac{k_{\mathcal{A}} + k_{\mathcal{B}}}{2m}\right)^2\right] - \left[\frac{k_{\mathcal{A}}^{int} + k_{\mathcal{B}}^{int}}{2m} - \left(\frac{k_{\mathcal{A}}}{2m}\right)^2 - \left(\frac{k_{\mathcal{B}}}{2m}\right)^2\right]$$

If $l_{\mathcal{AB}} = 1$, that is, there is a single edge joining $\mathcal{A}$ to $\mathcal{B}$, we expect that the two communities should be separated. If $k_{\mathcal{A}}k_{\mathcal{B}}/2m^2 < \frac{1}{m}$, we have $\Delta Q_{\mathcal{AB}} > 0$. For simplicity, let us suppose that $k_{\mathcal{A}} \sim k_{\mathcal{B}} = k$, that is, that the two subgraphs have roughly the same number of edges. We conclude that when $k < \sqrt{2m}$ and the two communities $\mathcal{A}$ and $\mathcal{B}$ are connected, then the modularity is higher if they are in the same cluster [33]. So, if the partition with maximum modularity includes clusters with total degree of the order of $\mathcal{O}(\sqrt{m})$ (or smaller), one cannot know a priori whether the clusters are composed of single communities or are in fact a combination of smaller weakly interconnected communities. This resolution problem may have important impacts in practical applications.

3 Community Evolution in Dynamic Networks

In complex networks, the interactions between entities dynamically evolve over time [7]. Let us take Facebook[1] as an example: users add or delete "friends" [26]. Similarly, new forms of social contacts can be observed in phone calls, e-mail exchanges [58], or other communications on the Internet.

As mentioned previously, traditional analysis treats networks as *static* graphs, which is derived either from an aggregation of data over the whole network life (experiment measure) or from a snapshot of data at a particular time step. Although

[1] http://www.facebook.com/

this study provides meaningful results, the dynamic features are neglected. Dynamic features are however crucial in order to better understand complex networks.

During the last decade, the avalanche of data footprint provided by the traceability of many social activities represents a major scientific, economic, and social revolution. The availability of large dataset (thanks to Open Data initiative), the optimized rating of computing facilities, and the development of powerful and reliable data analysis tools have constituted a better and better machinery to explore the topological properties of several networked systems from the real world. This has allowed to study the topology of the dynamic interactions in a large variety of Big Data [19] as diverse as communication [72, 73], social [24, 67], and biological systems [12, 48].

In the following, we discuss network evolution models in Sect. 3.1. Next, we introduce the definitions and notations of community evolution in dynamic networks in Sects. 3.2 and 3.3. Moreover, we discuss how to evaluate community evolution. Evaluating community evolution depends on the definition about community evolution and the chosen similarity measures. There is a similarity measure proposed for measuring the quality of the found community structure at each time step. It is called *α-cost*. There are also many other similarity measures such as *matching metrics*. Matching metrics are used for matching communities at different time steps. We introduce the quality function: α-cost (see Sect. 3.4) and discuss matching metrics (see Sect. 3.5). In addition, we discuss the definition of community dynamics, which are based on matching metrics (see Sect. 3.6).

3.1 Network Evolution Models

When networks tend to gradually evolve over time, a first class of models considering networks as dynamical systems can be derived. A first common class of evolving network models is based on two ingredients: *growth* and *preferential attachment*. The growth hypothesis suggests that networks continuously expand through the arrival of new nodes and new links between existing nodes, while the preferential attachment hypothesis states that strongly connected nodes increase their connectivity faster than less connected nodes. In [49], the authors have measured different networks. Results show different attachment rate functions: the attachment rate in some systems depends linearly on the node degree, while the dependence of other systems follows a sublinear power law.

In order to mine dynamic properties of networks, Leskovec et al. [57] have studied a wide range of real networks from several domains. Empirical observations revealed that most networks are becoming denser over time with the increasing average degree and decreasing effective diameters.

As one may expect, the evolution of real networks is complicated. It is possibly related to community structure. For instance, Backstrom et al. [5] investigated an online network [2]. They computed the probability of joining a community as a function of internal friends (who are already in the community). The empirical studies showed that the probability of joining a community increases with the number of internal friends but is very noisy. Moreover, their studies also compare the effects between different friends in attracting new community members: the number of friends who know each other provides a stronger effect than the friends who do not know each other.

Asur et al. [3] have measured *sociability index* in testing the DBLP coauthor dataset. The *sociability index* gives high scores to nodes that are involved in interactions with different groups. Their analysis showed that the sociability index could be used to predict future co-occurrences of nodes in clusters.

Therefore, current statistical models based on growth and preferential attachment only capture one part of dynamic behaviors of real networks and fail to capture many other dynamic structural properties such as the evolution of community structure. Community evolution is important for analyzing dynamic structural properties of real networks.

3.2 Community Evolution Models

There is no accepted standard definition of community in dynamic networks. A community can be defined as a structure that is observable over time. Such communities can be detected by matching observable communities at different time steps. In [44], Hopcroft et al. have detected the partition for each snapshot graph by hierarchical clustering [45] and then matched communities at different time steps through the notion of *natural communities*. They define *natural communities* as groups of nodes having high stability against perturbations of their interactions. When analyzing citation networks, natural communities can be used to denote topics of communities. Tracking natural community evolution allows them to understand the history of topics, such as the emergence of new topics. The idea of detecting *time-independent* communities at different time steps and then matching them becomes the basis for several algorithms. They are called *two-stage approaches* (see Sect. 4.1). Each *time-independent* community is detected independent of the results at other time steps.

Another definition of community in dynamic networks is a structure that evolves over time. For instance, a community is defined as the neighborhood of a chosen node in [27]. Therefore, each community is detected by incrementally updating the neighborhood of the chosen node corresponding to the evolution of the graph structure over time. Such methods that propose to detect *time-dependent* temporal clusters are called *evolutionary clustering* (see Sect. 4.2). The principle of *evolutionary clustering* [13] is to simultaneously optimize two potentially conflicting criteria:

[2]LiveJournal (LJ)http://www.livejournal.com/

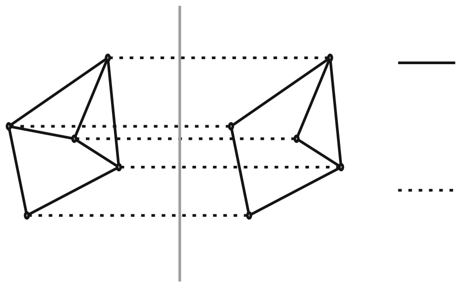

Fig. 1 An example of a coupling graph, where graphs at different time steps are connected through couplings. The real interactions between nodes are shown in *solid lines*, while the coupling interactions are denoted by *dotted lines*. The figure is gained from [47]

(i) first, the clustering at any time step should remain faithful to the current data as much as possible *(ii)* and second, the clustering should not shift dramatically from one time step to the next time step.

There are also many other models for capturing community structure in dynamic networks. The *coupling graph clustering* is a framework which detects community structure of a *coupling graph* (see Sect. 4.3). A *coupling graph* is a graph linking a sequence of graphs over several time steps by adding coupling edges between the same nodes at different time steps (see Fig. 1). Given a coupling graph, a subgraph which describes all interactions at a specific time step is called a *slice*.

3.3 Notations of Community Evolution

A *dynamic graph* $\mathcal{G}(V, \mathcal{E})$ on a finite time sequence $1 \dots \Delta$ is a sequence of graph snapshots $\{G(1), \dots, G(\Delta)\}$. There is a set $V = \{v_1, \dots, v_n\}$ of nodes. Each node $v_i \in V$ appears at least one during the dynamic graph lifetime, *that is*, $\exists t \ s.t. \ v_i \in G(t)$.

At each time step t where $1 \leq t \leq \Delta$, the corresponding snapshot $G(t)$ describes interactions between active nodes at time t, where the edges of a snapshot graph are a set of active dynamic links. $G(t)$ is partitioned into a set of *temporal clusters* $\mathcal{P}(t) = \{C_1(t), \dots, C_{n_c^t}(t)\}$, where n_c^t denotes the number of temporal clusters in $G(t)$. In some definitions of communities in dynamic networks [30,31], the number of temporal clusters may be not equal to the number of communities at the same time step t . One community $\mathcal{C}_i$ at time step t is possibly represented by a set of temporal clusters such that $\mathcal{C}_i(t) = \{C_1(t), \dots\}$.

The problem of tracking community evolution can be resolved by the identification of a set of *community evolution paths* (or *community evolution traces* [92], *dynamic communities* [39]).

Definition 1 (Community evolution path). For a given time window $[\delta_0, \delta_0 + \Delta]$, an evolution path $\text{Evol}(\mathcal{C}_i)$ is a time series of temporal clusters: $\text{Evol}(\mathcal{C}_i) := \{C_i(\delta_0), \dots, C_i(\delta_0 + \Delta)\}$ where each temporal cluster $C_i(t) \in \text{Evol}(\mathcal{C}_i), t \in [\delta_0, \delta_0 + \Delta]$ is the observation of the community $\mathcal{C}_i$.

In the definition of Wang et al. [92], the observation of the community $\mathcal{C}_i$ at time t can be the union of several temporal clusters. When a community appears for the first time, it should be a unique temporal cluster.

3.4 Quality Function: α-Cost

Reliable algorithms are supposed to provide results having a high-quality value. In the case of community detection in dynamic networks, a famous function named α-cost has been used by several algorithms [52, 83, 89] for measuring the quality of the found dynamic communities. This α-cost is motivated by the principle of clustering evolution: the community structure at each time step is the evolution of the community structure at the previous time step. Therefore, one can see the evolution as a combination of a *snapshot cost* and a *past history cost*. The parameter α controls the relative weight of recent and past history:

$$\text{cost} = \alpha\, \mathcal{CS} + (1-\alpha)\, \mathcal{CT} \tag{3}$$

where the snapshot cost $\mathcal{CS}$ measures how a community structure fits the graph interactions at time t and the past history cost $\mathcal{CT}$ qualifies how consistent the community structure is with the past history community structure at time $t-1$.

Let X represent the current community structure, Y represent the community structure at the previous time step, W denote current graph interaction, Λ be a nonnegative diagonal matrix, and $D(\bullet)$ be the function for measuring the cost such that $D(\bullet)$ computes the similarity between the network structure and the community structure and the similarity between the current community structure and the previous community structure.

In [89], authors defined $D(\bullet)$ as a KL-divergence between two objects such that $\mathcal{CS} = D(W \parallel X\Lambda X^T)$ and $\mathcal{CT} = D(Y \parallel X\Lambda)$. Given two objects A and B, $D(A \parallel B) = \sum_{ij}\left(a_{ij}\log\frac{a_{ij}}{b_{ij}} - a_{ij} + b_{ij}\right)$. Through this cost definition, the snapshot cost is high when the approximate community structure fails to fit the graph interactions at time t, while the past history cost is high when there is a dramatic change of community structure from time $t-1$ to t.

There exist also other definitions of $D(\bullet)$. In [52], two definitions of the cost are introduced: one is the distance between all pairs of objects in an agglomerative hierarchical clustering, and the other is associated with the centroid of the community in k-means clustering [10]. In k-means clustering, community memberships are measured by the membership degrees of nodes, that is, the distance between the node to the centroid of its community. Then, in the cost of the community structure of a dynamic graph, the snapshot cost is associated with the distance between the node and the centroid of its community, and the past history cost is computed by the difference between the current community centroid and the community centroid at the previous time step.

In the case of multimode networks, Tang et al. [84] have suggested the resolution by transforming the problem in multimode networks into the problem of two mode. Most of existing work concentrates on *one-mode network*. That is, there is only one type of social actors (nodes) involved in the network and the ties (interactions) between actors are all of the same type. This is common in a broad sense such as friendship network, the Internet, and phone call network. However, some applications such as web mining, collaborative filtering, and online-targeted marketing involve more than one type of actors and multiple heterogeneous interactions between different types of actors. Such a network is called *multimode network* [93].

Given an m-mode network, for each mode i, let $\mathbb{X}_i$ denote this mode of nodes, such as $\mathbb{X}_i = \{x_1^i, \ldots, x_{n_i}^i\}$, where n_i is the number of nodes for $\mathbb{X}_i$. Then, for each pair of modes, we use $\mathbf{R}_{ij}^t \subseteq \mathbb{X}_i \times \mathbb{X}_j$ to represent interactions between two modes of nodes $\mathbb{X}_i, \mathbb{X}_j$ at time t. Ideally, the interaction between nodes can be approximated by

$$\mathbf{R}_{ij}^t \approx \mathbf{C}_i^t \mathbf{A}_{ij}^t (\mathbf{C}_j^t)^T$$

where $\mathbf{C}_i^t$ is the cluster membership for $\mathbb{X}_i$ at time t and A_{ij}^t represents the group interaction. The group interaction is computed by $A_{ij}^t = (\mathbf{C}_i^t)^T \mathbf{R}_{ij}^t \mathbf{C}_j^t$. Therefore, for each temporal m-mode graph at time t, its snapshot cost $\mathcal{CS}$ can be formulated as

$$\mathcal{CS} = \sum_{1 \leqslant i < j \leqslant m} w_a^{(i,j)} D\left(\mathbf{R}_{ij}^t \parallel \mathbf{C}_i^t \mathbf{A}_{ij}^t (\mathbf{C}_j^t)^T\right)$$

and its history cost $\mathcal{CT}$ is expressed as

$$\mathcal{CT} = \sum_{1 \leqslant i \leqslant m} w_b^i D\left(\mathbf{C}_i^t \parallel \mathbf{C}_i^{t-1}\right)$$

where w_a^{ij} is an importance factor for every pair of modes i and j, and w_b^i is a relative importance factor for each mode i.

3.5 Matching Metrics

A matching metric is a similarity function, which measures how similar two communities are. It is often used in two-stage approaches (see Sect. 4.1) to connect similar communities. One can measure the similarity between two temporal clusters at different time steps and naturally obtain how one community evolves from one time step to the following time steps.

Hopcroft et al. [44] defined a *match* function. Let C and C' be two clusters; their match value is written as follows:

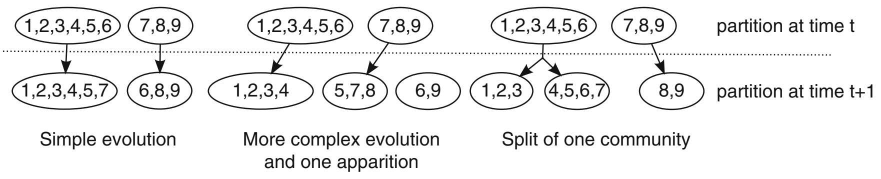

Fig. 2 Examples of community evolution in a time period $[t, t+1]$. We match clusters at time t to the clusters at time $t+1$. Given a cluster at time t, it remains stable if it is matched to a unique cluster at time $t+1$; it splits if it is matched to more than one cluster at time $t+1$. In addition, one new cluster appears at time $t+1$, if no cluster at time t is matched to it. The figure is obtained from [4]

$$\text{match}(C, C') = \min\left(\frac{|C \cap C'|}{|C|}, \frac{|C \cap C'|}{|C'|}\right) \tag{4}$$

The definition ensures that a high matching value (close to 1) occurs when two clusters have many common nodes and are roughly of the same size. The best match value for C at time t is the highest match(C, C') value for any cluster C' at time t (Fig. 2).

Palla et al. [70] defined *relative overlap*, which is a Jaccard index. The relative overlap value between two communities X and Y is written as follows:

$$J(X, Y) = \frac{|X \cap Y|}{|X \cup Y|} \tag{5}$$

By definition, the cluster $C(t+1)$ at time $t+1$ is matched to the cluster $C(t)$ which has the largest overlap at time t.

Another bipartite mapping metric is *dynamic Jaccard's index*, whose definition is

$$JacD'(X, Y) = \frac{J(X, Y)}{|t - t'|} \tag{6}$$

where $|t - t'|$ represents the time interval duration between communities X and Y. It allows a temporal cluster to be matched to an older one ($|t - t'| > 1$) which may have disappeared during several time steps.

Two communities are matched if they share the highest matching value. The matching metric is a natural resolution to connect temporal clusters over time. So, it is often used in two-stage methods [44, 70, 86]. Its other advantage is to characterize community dynamics. However, there is no standard definition of matching metric. In Hopcroft et al. [44]'s *match* function (4), the minimum size of communities is important for the comparison. Instead, the size of the union of communities is essential in the relative overlap (5). Furthermore, a minimum intersection size threshold needs to be set, that is, the minimum number of common nodes shared by the matching communities.

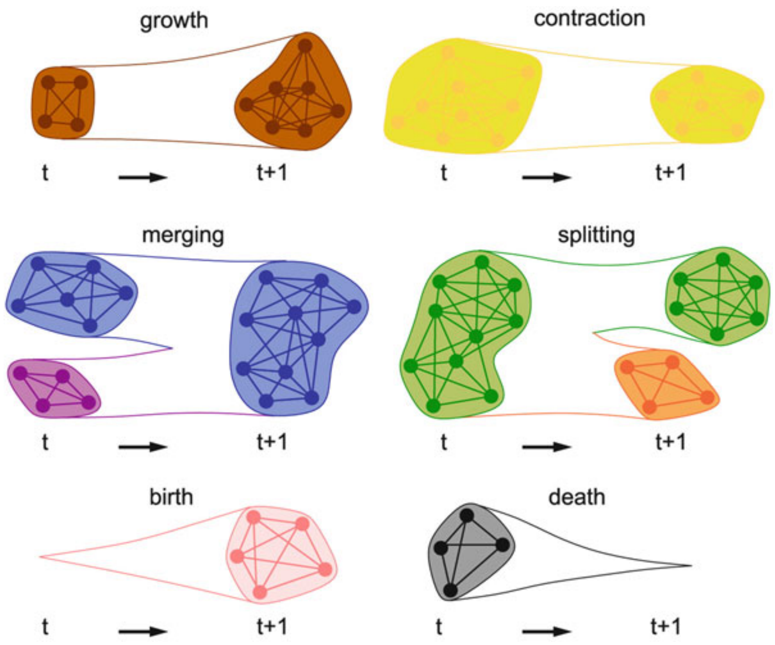

Fig. 3 Possible scenarios in the evolution of communities. The figure is gained from [70]

3.6 Definition of Community Dynamics

When we track community evolution, one problem is to characterize community dynamics. How a community changes over time? Chakrabarti et al. [13] proposed the definition of *change point* to describe a significant change in community structure. In the following, we describe them in details. Moreover, Palla et al. have introduced the main phenomena occurring during the lifetime of a community (see Fig. 3): birth, growth, contraction, merging, splitting, and death.

Change Point

Chakrabarti et al. [13] have detected *change points*, which represent the significant time points when the system evolves, that is, a major change (or critical event) occurs in the graph structure during a short period. The approach called GraphScope [13] applied the MDL (Minimum Description Length) principle [40] to compute the encoding cost of assigning nodes into communities. A *segment* presents a sequence of graphs without any change in its community structure. So the graphs of each segment are characterized by the same partition with the lowest encoding cost. If the cost for encoding a graph into the existing segment is higher than the cost for encoding the graph into a new segment, a significant change of community structure occurs. The change point offers one important benefit of detecting community evolution using information theory.

Community Changes

We show six community changes in Fig. 3, which are used to describe the main events occurring in dynamic graphs. In order to identify them, matching metrics are often used such as the definition given in [3].

Definition 2. Let $G(t)$ and $G(t+1)$ be two snapshots of $\mathcal{G}$ at two consecutive time steps. Let the cluster $C_i(t)$ and $C_i(t+1)$ denote the observations of the community $\mathcal{C}_i$ at time step t and $t+1$, respectively.

Continue: $C_i(t+1)$ is the continuation of $C_i(t)$ if $C_i(t+1)$ is the same as $C_i(t)$:

$$C_i(t) = C_i(t+1)$$

κ−Merge: Two clusters $C_i(t)$ and $C_j(t)$ merge into $C_i(t+1)$ if $C_i(t+1)$ contains at least $\kappa\%$ of nodes belonging to the union of $C_i(t)$ and $C_j(t)$ and the renewal of $C_i(t)$ and $C_j(t)$ is at least 50%:

$$\frac{|(C_i(t) \cup C_j(t)) \cap C_i(t+1)|}{\max(|C_i(t) \cap C_j(t)|, |C_i(t+1)|)} > \kappa$$
$$|C_i(t) \cap C_i(t+1)| > |C_i(t)|/2$$
$$|C_j(t) \cap C_i(t+1)| > |C_j(t)|/2$$

κ−Split: $C_i(t)$ is split into $C_i(t+1)$ and $C_j(t+1)$ if $\kappa\%$ of nodes belonging to $C_i(t)$ are in two different clusters at time $t+1$, such as

$$\frac{|(C_i(t+1) \cup C_j(t+1)) \cap C_i(t)|}{|\max(|C_i(t+1) \cap C_j(t+1)|, |C_i(t)|)} > \kappa$$
$$|C_i(t) \cap C_i(t+1)| > |C_i(t+1)|/2$$
$$|C_i(t) \cap C_j(t+1)| > |C_j(t+1)|/2$$

Emerge: A new cluster $C_i(t+1)$ emerges at time $t+1$ if none of the nodes in the cluster $C_i(t+1)$ are grouped together at time t, that is, $\nexists\ C_i(t)$, such that $|C_i(t) \cap C_i(t+1)| > 1$.

Disappear: $C_i(t)$ disappears if none of the nodes in the cluster $C_i(t)$ are grouped at time $t+1$, that is,$\nexists\ C_i(t+1)$, such that $|C_i(t) \cap C_i(t+1)| > 1$.

This definition has several limits. First, the definition of one continuation is so strict that almost all communities do not have any continuation at the next time step. Second, the value of κ needs to be set to determine when a community is merged or when a community is splited. Varying κ may lead to different results. Finally, the definition of emerging community or disappearing community has weakness. Some clusters may be generated only by the fluctuation of degree distribution. These artificial clusters will not share a strong common interest. For the disappearance, the process may be too slow: a community may lose its core nodes but still have node attached to it. In this case, the observed community does not share a strong common interest anymore. It is difficult to determine whether a community exists.

Another definition of community dynamics is based on community predecessor/successor relationship [91].

Definition 3 (Community predecessor and successor). Given a temporal cluster $C_i(t)$ at time t, if the temporal cluster $C_j(t-1)$ has the maximum overlap size among all temporal clusters at time $t-1$, we define that $C_j(t-1)$ is the predecessor of $C_i(t)$. If the temporal cluster $C_k(t+1)$ has the maximum overlap size among all temporal clusters at time $t+1$, we define that $C_k(t+1)$ is the successor of $C_i(t)$.

In the following, given a pair of temporal clusters (X, Y), $X \rightarrow Y$ is used to denote that Y is Xs successor and $X \leftarrow Y$ represents that X is Ys predecessor.

The relationship between one community and its successor (or its predecessor) may be asymmetrical. That is, for one community and its successor, this community may not be the predecessor of its successor. Similarly, for one community and its predecessor, it is possible that the community is not the successor of its predecessor. This asymmetrical property is used to characterize community dynamics.

Definition 4. Let $G(t)$ and $G(t+1)$ be snapshots of $\mathcal{G}$ at two consecutive time steps with the temporal partition $\mathcal{P}(t)$ and $\mathcal{P}(t+1)$ denoting the community structure of $\mathcal{G}$ at time step t and $t+1$, respectively.

Survive. $C_j(t+1)$ is the continuation of $C_i(t)$, if and only if $C_i(t)$ is the predecessor of $C_j(t+1)$ and $C_j(t+1)$ is the successor of $C_i(t)$, such that

$$C_i(t) \leftarrow C_j(t+1) \wedge C_i(t) \rightarrow C_j(t+1)$$

This relationship is denoted by $C_i(t) \leftrightarrows C_j(t+1)$. When a community survives from t to $t+1$, it is a growing community if it has an increasing number of community members; otherwise, it is a shrinking community.

Emerge. $C_j(t+1)$ is a creation if and only if $C_j(t+1)$ has no predecessor such that

$$\nexists C_i(t) \in \mathcal{P}(t) | \big(C_i(t) \rightarrow C_j(t+1)\big)$$

Merge. $C_j(t+1)$ is a fusion if and only if $C_j(t+1)$ is the successors of several clusters at time step t such that

$$\exists \{C_i(t), C_k(t)\} \subseteq \mathcal{P}(t) | \big(C_i(t) \rightarrow C_j(t+1) \wedge C_k(t) \rightarrow C_j(t+1)\big)$$

where $i \neq k$.

Split. $C_j(t+1)$ is a split if and only if $C_j(t+1)$ is not the successor of its predecessor such that

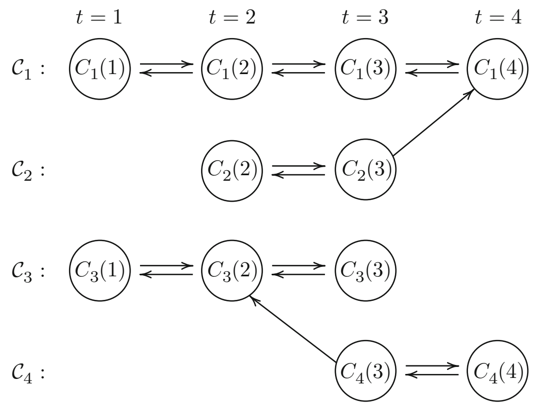

Fig. 4 Diagrams of four communities observed during four time steps, featuring continuation, creation, disappearance, fusion, and split

$$C_i(t) \leftarrow C_j(t+1) \wedge C_i(t) \nrightarrow C_j(t+1)$$

Disappear. A community disappears at time $t + 1$ if and only if its observation $C_i(t)$ at time step t has no successor such that

$$\nexists C_j(t+1) \in \mathcal{P}(t+1) | \big(C_i(t) \nrightarrow C_j(t+1)\big)$$

Diagrams in Fig. 4 show several cases illustrating community dynamics which can be featured by continuation, creation, disappearance, fusion, and split. For better understanding of community evolution, their evolution paths (Def. 1) will be shown. For each community $\mathcal{C}$, its evolution path is $\mathrm{Evol}(\mathcal{C}) := \{C(1), \ldots, C(\Delta)\}$, where each element $C(i)$ $(1 \leq i \leq \Delta)$ represents its observation at time step $t = i$.

In the example illustrated by the Fig. 4, four communities have the evolution paths:

- $\mathrm{Evol}(\mathcal{C}_1) := \{C_1(1), C_1(2), C_1(3), C_1(4)\}$
- $\mathrm{Evol}(\mathcal{C}_2) := \{C_2(2), C_2(3)\}$
- $\mathrm{Evol}(\mathcal{C}_3) := \{C_3(1), C_3(2), C_3(3)\}$
- $\mathrm{Evol}(\mathcal{C}_4) := \{C_4(3), C_4(4)\}$

Nearly all types of community changes are observed:

- Community $\mathcal{C}_2$ is created at $t = 2$ as it has no predecessor at $t = 1$.
- Community $\mathcal{C}_3$ disappears at $t = 4$ as it has no successor at $t = 4$.
- Community $\mathcal{C}_2$ is merged into $\mathcal{C}_1$ at $t = 4$ since its successor at $t = 4$ is $C_1(4)$ whose predecessor is not $C_2(3)$.
- Community $\mathcal{C}_4$ is split from $\mathcal{C}_2$ since $t = 2$ as its predecessor at $t = 2$ is $C_3(2)$ whose successor is not $C_4(3)$.

$$\begin{tikzcd}
 & t=1 & t=2 & t=3 & t=4 \\
\mathcal{C}_1 : & & & \boxed{C_1(3)} \arrow[r, shift left] \arrow[dl] & \boxed{C_1(4)} \arrow[l, shift left] \\
\mathcal{C}_2 : & \boxed{C_2(1)} \arrow[r, shift left] & \boxed{C_2(2)} \arrow[l, shift left] \arrow[dr] & & \\
\mathcal{C}_3 : & \boxed{C_3(1)} \arrow[r, shift left] & \boxed{C_3(2)} \arrow[l, shift left] \arrow[r, shift left] & \boxed{C_3(3)} \arrow[l, shift left] \arrow[r, shift left] & \boxed{C_3(4)} \arrow[l, shift left]
\end{tikzcd}$$

Fig. 5 Diagram of four clusters observed over 4 time steps, featuring fusion and split community events

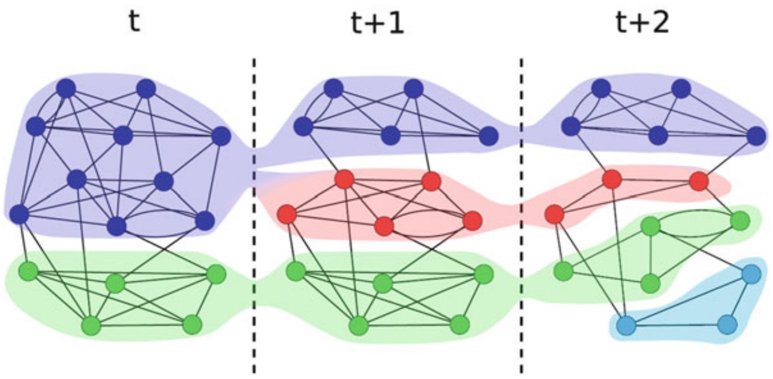

Fig. 6 Examples of community evolution over three snapshot graphs by matching temporal clusters to dynamic communities. We observe 4 dynamic communities, indicated by colors: $\mathcal{C}_1$ in *dark blue*, $\mathcal{C}_2$ in *red*, $\mathcal{C}_3$ in *green*, and $\mathcal{C}_4$ in *light blue*. During their evolution, we observe the community $\mathcal{C}_1$ is split into $\mathcal{C}_1$ and $\mathcal{C}_2$ between t and $t+1$

Community $\mathcal{C}_1$ is observable during all the observation window (only four time steps on this toy example). At time step $t = 4$, community $\mathcal{C}_2$ joins it. This community fusion event seems to be more an event related to $\mathcal{C}_2$ rather than to $\mathcal{C}_1$.

A more complex diagram is displayed in Fig. 5. We observe the changes of communities from time step $t = 2$ to $t = 3$. At time step $t = 3$, community $\mathcal{C}_2$ partially merges with $\mathcal{C}_3$, while its split $C_1(3)$ starts a new community $\mathcal{C}_1$.

There are also other types of definitions [16,37,39]. For example, Chen et al. [16] characterize community dynamics by tracking community core evolution. Greene et al. [39] use the definition of dynamic communities described above but require that if several dynamic communities share the same temporal cluster at time t, then these dynamic communities should merge.

In Fig. 6, we have shown examples of community evolution. There are four dynamic communities over the total three time steps, whose evolution paths are expressed as following:

$$\begin{aligned}
\mathrm{Evol}(\mathcal{C}_1) &\leftarrow \{C_1(t), C_1(t+1), C_1(t+2)\}\\
\mathrm{Evol}(\mathcal{C}_2) &\leftarrow \{C_2(t+1), C_2(t+2)\}\\
\mathrm{Evol}(\mathcal{C}_3) &\leftarrow \{C_3(t), C_3(t+1), C_3(t+2)\}\\
\mathrm{Evol}(\mathcal{C}_4) &\leftarrow \{C_4(t+2)\}
\end{aligned}$$

Through these evolution paths, we observe two new communities appearing during network evolution: the community $\mathcal{C}_2$ is the branch of $\mathcal{C}_1$ and the community $\mathcal{C}_2$ emerges at time $t = 2$.

This is an example to illustrate the relationship between community dynamics and community evolution paths. We conclude that the problem of identifying and characterizing community dynamics can be revealed by community evolution paths, whereas the problem of tracking community evolution in dynamic networks can be reformulated as a problem of constructing community evolution paths across one or more time steps.

4 Tracking Community Evolution

Tracking community evolution is a key problem for which many algorithms have been proposed and developed. In the following, we will review them in details. Being able to benchmark proposed heuristics appears to be an important challenge too. Indeed, once an algorithm is designed, it is mandatory to test its performance. A natural solution is to design benchmark graphs (see Sect. 4.4). Benchmark graphs should offer an a priori known community structure. When an algorithm is reliable and efficient, it has to perform well when applied to benchmark graphs. Another important challenge is to design adapted tools for visualizing and representing how community structure evolves over time (see Sect. 4.5).

4.1 Two-Stage Approaches

The basic idea of two-stage approaches is to detect temporal clusters at each time step and then establish relationships between clusters for tracking community evolution over time. Figure 6 illustrates the result of applying a two-stage approach to a dynamic network across three time steps. In a first phase, clusters at each time step are detected: at time t, there are two clusters, then there are three clusters at time $t + 1$ and four clusters at time $t + 2$. In a second phase, the relationship between clusters at different time steps is established, which is shown by colors. Through the above results, we learn how the community structure of this graph evolves from the time step t to the time step $t + 2$. For the first phase, we apply a graph clustering algorithm [36]. For the second phase, we can use a matching metric (see Sect. 3.5). However, it may lead to noisy results where some nodes often change their community memberships. Therefore, many advanced resolutions are proposed to resolve this general matching problem.

Core-Based Methods

If a partition is significant, it will be recovered even if the structure of the graph is modified, as long as the modification is not too extensive. Instead, if a partition is not significant, we may observe that a minimal perturbation of the graph is enough to disrupt its group memberships. Since most stochastic community detection are searching for local optima due to computational costs, the detection results can be different simply if the ordering of the nodes is modified, without any modification of the topology. This is often referred as the consistency problem [53, 54, 81]. A *significant cluster*, that is, a significant group of nodes, is often defined as a *community core*. We can reduce noisy results by matching community cores. This is the main principle of core-based methods. The matching metric (see Sect. 3.5) is often applied. Two temporal clusters are matched if their community cores share the highest similarity value.

Hopcroft et al. have proposed the concept of natural communities, which are significant clusters that have high stability against modification of graph structure. Given a temporal graph, by applying 5% of perturbations, a set of modified graphs are produced, each of which has 95% of core nodes. Each natural community is identified by the partitions corresponding to these modified graphs, which has the best match value with clusters in those partitions.

Rosvall et al. [78] used a bootstrap method [25] to detect significance of clusters. The bootstrap method assesses the accuracy of an estimate by resampling from the empirical distribution of observations. Each graph can be resampled by assigning to each edge a weight taken from a Poisson distribution with mean equal to the original edge weight. A graph clustering method is applied to the original graph and the samples. For each community in the original graph's partition, they define its largest subset of nodes that are classified in the same community in at least 95% of all bootstrap samples, as the significant cluster.

In some methods, core nodes are identified through their roles within their communities. Given a community, there are core nodes and peripheral nodes. Guimerá and Amaral [41] have classified community members into different roles according to intra- and inter-module connection patterns. With respect to core node identification, Wang et al. [92] defined core nodes, where each core node v satisfies $\sum_{u \in \text{neighbours}} (k_v - k_u) > 0$. In [8], k-cores nodes [1] are detected with a threshold k where k-core decomposition is used for filtering out peripheral nodes.

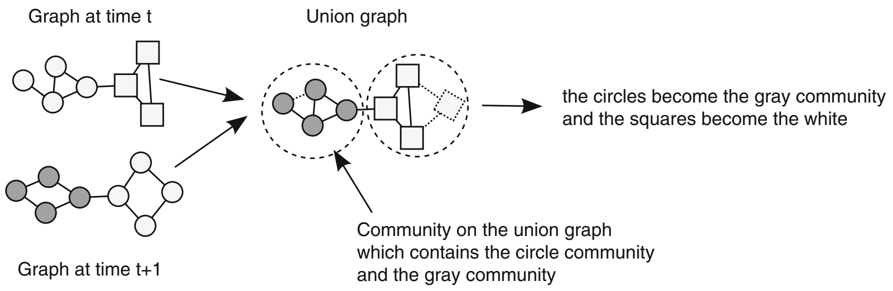
Graph at time t
Union graph
the circles become the gray community
and the squares become the white
Graph at time t+1
Community on the union graph
which contains the circle community
and the gray community

Fig. 7 An example of a union graph which is constructed by jointing two graphs at time t and $t + 1$. The figure is obtained from [70]

Although core-based approach can smooth variances caused by peripheral nodes, its results still suffer from some limits such as the parameters used in matching metrics. In additional, if we only track evolution of community cores, there is a risk of missing important structural changes which are related to peripheral nodes.

Union-Graph-Based Methods

Another important early work [70] for detecting community evolution is related to the *union graph*. Each union graph merges two graphs (union of their links) present at contiguous time steps. Let $G(t, t + 1)$ denote the union graph resulting from the union of two graphs at time t and $t + 1$. We have $E_{t,t+1} = E_t \cup E_{t+1}$. Figure 7 gives an example of a union graph. Any community present at t or $t + 1$ is contained in exactly one community in the union graph. Thus, communities in the union graph provide a natural connection between communities at t and $t + 1$. If a community in the joined graph contains a single community from t and a single community from $t + 1$, then they are matched. If the joined group contains more than one community from both time steps, the communities are matched in decreasing order of their relative node overlap (5). The technique is validated by applying it to two social systems: a graph of phone calls between customers of a mobile phone company over one year and a collaboration network between scientists spanning a period of 142 months.

The union graph smooths the changes between every pair of consecutive time steps. This property can reduce fluctuations caused by noisy data. In addition, the union graph allows us to directly determine the links between temporal clusters at consecutive time steps. It simplifies the problem of tracking community evolution.

The main disadvantage of this technique is that the CPM algorithm used only detects communities in certain contexts, that is, CPM algorithm fails to detect community structure of networks with few cliques. In addition, some parameters are used to determine how community changes due to the application of similarity metric.

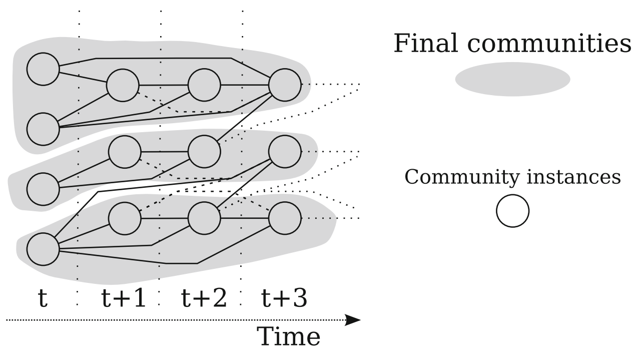

Fig. 8 An example of dynamic networks with community instances (nodes) and final communities (*in grey*). At each time step, we may match several community instances to the same temporal community

Algorithm 4 Hierarchical edge betweenness clustering

```
Input: $G = (V, E)$
Output: A dendrogram
  repeat
    Compute edge betweenness for all edges
    Remove edge with highest betweenness
  until no more edges in graph
  Return a dendrogram // The dendrogram is produced from a top down approach:
  the network is split into different communities with successive removals of links.
  The leaves of the dendrogram are individual nodes.
```

Survival-Graph-Based Methods

Given a dynamic graph, its *community survival graph* is constructed by representing community instances as nodes which are linked via edges based on their similarity. One can divide this community survival graph into final communities. Each final community groups a set of temporal clusters and spans several time steps as shown in Fig. 8.

The first approach associated with survival graph is proposed by Falkowski et al. [30, 31]: first cluster each temporal graph to find community instances at each time step, then construct a community survival graph, and finally cluster the community survival graph to find final communities by using a hierarchical edge betweenness clustering [36].

To construct a community survival graph, a time window is set to compare the similarity between community instances and connect the similar community instances with edges. In another words, this time window size is the largest time distance between every pair of connected community instances in a community survival graph. The applied hierarchical edge betweenness clustering (see Algorithm 4) contains an iteration, which eliminates edges to separate subgraphs. In Falkowski et al.'s method, a parameter k is applied to determine the number of iterations. The connected subgraphs retained after k iterations correspond to the final communities. A connected subgraph consists of similar community instances.

Chi et al. [17] have detected final communities through a soft clustering [2], after detecting community instances [66, 82, 94] at each time step. At a time step i, the graph interaction is denoted by $\mathbf{A}^i \subseteq V \times V$ with l_i *basis subgraphs* $\mathcal{B}^i = [\mathbf{B}^i_1, \ldots, \mathbf{B}^i_{l_i}]$. Each *basis subgraph* describes interactions between nodes within a community instance. Across a time window $[1, \ldots, \Delta]$, graph interactions can be denoted by a three-dimensional tensor: $\mathbb{A} = [\mathbf{A}^1, \ldots, \mathbf{A}^\Delta] \in \mathcal{R}^{n \times n \times \Delta}$. For the total $N_c = \sum_{i=1}^{\Delta} l_i$ *basis subgraphs*, another three-dimensional tensor is defined: $\mathbb{B} = [\mathbf{B}^1_1, \ldots, \mathbf{B}^\Delta_{l_\Delta}] \in \mathcal{R}^{n \times n \times N_c}$.

Then, the final communities are obtained by minimizing the objective function: $D(\mathbb{A} \parallel \mathbb{B}\mathbf{U}\mathbf{V}^T)$. The matrices $\mathbf{U} = [u_{kj}]_{N_c \times n_c}$ and $\mathbf{V} = [v_{ij}]_{\Delta \times n_c}$ are the solution of the optimization problem. For each dynamic community j, u_{kj} is a vector of weights on kth basis subgraph. At each time step i, v_{ij} is a community intensity for the jth final community.

In this method, the size of time windows and basis subgraphs are an issue. A good size value of time windows allows us to group small community instances into a final community, if these small community instances have high frequency grouped together. The size of basis subgraphs is related to insignificant subgraphs (e.g., a subgraph with only a couple of nodes), as insignificant subgraphs are removed for the computation. The larger size threshold of basis subgraphs is, the less iterations are used for computing $\mathbf{U}$ as less number of N_c. Therefore, the computation time can be optimized by increasing the size threshold of basis subgraphs.

For the number of communities n_c, they try different values to compare the reconstruction error and then choose one that is reasonably small and, at the same time, explains data reasonably well.

In [86], authors use a similar approach which tracks community evolution by connecting community instances, but they use another notion of final community. A quality function called *node cost* is defined to determine the community membership for each node over time. This function is the sum of two costs: the cost of one node to keep its community membership and the cost of one node to change its community membership. Therefore, final community detection is transformed into the problem of optimizing this function. Optimizing this function is shown to be an NP-complete problem. Another solution with an approximate factor is proposed in [85]. In their proposed node cost function, the importance of different costs is predefined. Giving a high importance to cost of a node, to keep its community membership, makes node membership stable for a long time duration. Giving a high importance to the cost of a node, to change its community member, makes node membership to fit to current snapshot structure.

Survival-graph-based method gives results about how dynamic communities evolve over time directly. It simplifies the problem of tracking community evolution. Compared to other two-stage approaches, which track community evolution by identifying observations at each time step, this technique is more practical. However, some issues arise: How to choose the *time window size*? How to choose *the number* *of clustering iterations* in [31]? How to choose *the size threshold of basis subgraphs* and *the number of final communities* in [17]? And how to choose *the importance value* in [86]?

Conclusion

Methods presented above are two-stage-like approaches:

1. Clusters are detected at each time step independently of the results at any other time step.
2. Relationships between clusters at different time steps are inferred successively.

Such natural process often produces significant variations between partitions that are close in time, especially when the datasets are noisy. Since the first phase is independent of the past history, smooth transitions are impossible. Such approach may produce artifacts if the data are noisy and variations between partitions may also be generated by the community detection algorithm itself. Such artifacts yield to artificial community dynamics rather than the real graph evolution. For each graph, let $\mathcal{O}(P)$ denote the partition detection time and $\mathcal{O}(M)$ represent the computation time for the matching problem. The total time complexity of a two-stage approach on a time window of length $\mathcal{T}$ is in $\mathcal{O}((P + M)\mathcal{T})$.

4.2 Evolutionary Clustering

An evolutionary clustering approach follows a principle of detecting community structure based on the current graph topology information at a given time t and on the community structure at previous time steps. The quality function used for dynamic community structure is α-cost (see Eq. (3)). By assuming that a good community structure has a high α-cost value, many optimization methods are proposed and are applied to real dynamic networks. For instance, Lin et al. [88, 89] used a probabilistic model to capture community evolution by maximizing α-cost. On one hand, proposed frameworks called *community model* usually search the optimal community structure for modeling the sequence of graphs by encompassing interactions of the whole graphs. On the other hand, *incremental/online algorithms* only consider interaction changes such as link insertion or link deletion which also make sense in detecting structural changes. In the following, we will review these evolutionary clustering methods.

Community Model

Community evolution can be modeled by a sequence of graphs based on a probabilistic model, which assumes that:

1. The interactions of the graph at each time step follow a certain distribution.
2. The community structure follows a certain distribution that is determined by the community structure at the previous time step.

The first attempt has been done by Lin et al. [88, 89] through α-cost function optimization. Let $\mathbf{W}^t$ denote a graph structure at time t and $\mathbf{X}^t\mathbf{\Lambda}^t$ represent its community structure. By defining $\mathbf{Z}^t = \mathbf{X}^t\mathbf{\Lambda}^t(\mathbf{X}^t)^T$, the authors have devised an α-cost (3):

$$\text{cost} = \alpha\, D(W^t \parallel \mathbf{Z}^t) + (1-\alpha)\, D(\mathbf{Z}^{t-1} \parallel \mathbf{Z}^t) \;.$$

Consequently, they estimate $\mathbf{X}^t$ and $\mathbf{\Lambda}^t$ for optimizing the cost. The problem of community detection at each time step becomes a problem in terms of maximum a posteriori (MAP) estimation. An EM algorithm for solving the MAP problem is given in [88, 89] with a low complexity when the graph structure is sparse.

This technique enables to detect overlapping community structure and track community evolution directly. So it is a good resolution for the problem of community detection in dynamic graphs. However, determining a priori the value of α is a drawback.

Yang et al. [95] also used a dynamic stochastic block model (DSBM) for finding communities and their evolutions in a dynamic social network. In their study, they have applied a Bayesian treatment for parameter estimation that computes the posterior distributions for all the unknown parameters.

Let $\mathbf{W}^t \in \mathbb{R}^{n\times n}$ denote a graph structure at time t and $\mathbf{Z}^t \in \mathbb{R}^{n\times n_c}$ is its community structure. For each node i, it is assigned into community k with a probability π_k, such as $\Pi = [\pi_1, \ldots, \pi_{n_c}] \in \mathbb{R}^{n_c}$. For a pair of nodes i and j whose community memberships are k and l, respectively, the link connecting them is assumed to follow a Bernoulli distribution with parameter P_{kl}, such as $w_{ij}^t \sim \text{Beronulli}(\bullet \mid \text{P}_{\text{kl}})$, that is, $\mathbf{W}^t \sim \Pr(\mathbf{W}^t \mid \mathbf{P}, \mathbf{Z}^t)$, where $\mathbf{P} = [P_{kl}]_{n_c\times n_c}$. For a community matrix $\mathbf{Z}^{t-1}$, a transition matrix $\mathbf{B} \in \mathbb{R}^{n\times n}$ is assumed to model $\mathbf{Z}^t$, such as $\mathbf{Z}^t \sim \Pr(\mathbf{Z}^t \mid \mathbf{Z}^{t-1}, \mathbf{B})$. So we write the likelihood for the DSBM model as follows:

$$\Pr(\mathbf{W}^t, \mathbf{Z}^t \mid \Pi, \mathbf{P}, \mathbf{B}) \;.$$

With the Bayesian Model, a posteriori probability $\Pr(\mathbf{Z}^t \mid \mathbf{W}^t)$ is computed with an inference algorithm.

There is no parameter in this technique. However, the authors only provide performances of the applications to networks with nearly ten time steps and a few hundred nodes. For large networks such as millions nodes and hundreds of time steps, the performance of this technique is not clear.

The community model captures community evolution by modeling the sequence of graphs. It performs well when applied to stable evolving graphs. However, it suffers from scalability problems due to an expensive matrix computation and storage cost.

Incremental/Online Algorithms

The incremental spectral clustering [69] is one of the early incremental algorithms that update matrices like the degree matrix or the Laplacian matrix according to changes of graph interactions [90]. In traditional spectral clustering, community detection is transformed into the eigenvalue problem of $L\mathbf{q} = \lambda D\mathbf{q}$, where L is the Laplacian matrix, $\mathbf{q}$ is the cluster indicator, λ is the eigenvalues, and D is the degree matrix. Using incremental computation yields to a lower computational cost than the standard spectral clustering. Incremental computation only takes into account changes; thus, the computation matrix is sparse. In addition, a tunable threshold τ is used to balance the computational cost and the accuracy. One drawback is that errors are accumulated after several steps, and when the dataset grows or changes frequently, the associated cost becomes expensive.

Modularity optimization is the most popular method for community detection. It is extended to detect community evolution, *for example*, the modularity-driven clustering proposed by Gorke et al. [38]. Their basic idea is to detect community structure by starting from a pre-clustering obtained from a standard modularity optimization heuristic. Then, they proposed and discussed heuristics based on global greedy algorithms or on local greedy algorithms. They pass a pre-clustering to the global version to adapt it to the dynamic case (dGlobal). Similarly, the local version remembers its old results: roughly speaking, the dynamic local version (dLocal) starts by letting all free (elementary) nodes reconsider their cluster. Then it lets all those (super-)nodes on higher levels reconsider their cluster, whose content has changed due to lower-level revisions. Similarly, Dinh et al. [21] proposed another method extended from community optimization.

The community detection based on node similarity such as DBSCAN [28] is also extended for detecting dynamic community evolution [27]. DBSCAN considers a community as a core node and a *neighborhood*. For each core node, its community must consist of at least η nodes within a radius distance ε. In IncrementalDBSCAN [27], each community updates its neighborhood if its community members have changed their neighbors. Similarly, DENGRAPH [29] detects community evolution according to the core nodes and their neighborhoods. Instead of a distance radius ε, a different distance function is proposed to compute core nodes and their neighborhoods.

Incremental or online method can detect dynamic communities and save time by avoiding computations on subgraphs where there is no change. However, all above approaches need predefined parameters.

Conclusion

There exist many other evolutionary clustering approaches. For instance, information theory has been used to detect community evolution in dynamic graphs. Sun et al. applied the MDL to find the minimum encoding cost to describe a time sequence of graphs and their partitions into communities. The basic principle of this method is to encode the graph topology into a compression information with the minimum cost of the description. This method enables to provide meaningful information on community evolutions. However, one drawback is the problem called *relevant variable*, which is the variance between real data and data compression. To what extent is information theory able to capture community structures? To our knowledge, we are still far from a precise definition of community, while modularity (defined by Eq. (2)) is the widest accepted quality function.

As opposed to two-stage approaches, evolutionary clustering does not encounter the matching problem. However, most methods are using parameters. Furthermore, we stress that evolutionary clustering results are generally too strongly correlated with community history which may occult structural changes.

4.3 Coupling Graph Clustering

Coupling graph clustering approach is based on a coupling graph as shown in Fig. 1. The underlying idea is once the coupling graph built (encompassing the time dimension as edges) to use an efficient standard static community detection heuristic. The first attempt is [47] where authors built a temporal graph and then used the classical community detection algorithm Walktrap [74]. The community evolution can be traced through group memberships over time.

Another method is proposed by Mucha et al. [63]. They detected dynamic communities by optimizing a modified modularity, which is motivated by α-cost (3). The modified modularity balances the contribution of community memberships to each slice and the cost for changing community memberships. The major advantage of this algorithm is to smooth community evolution. However, its results rely on the parameter α and the relative weight of coupling.

This idea of coupling graph clustering simplifies the problem of detecting community evolution. However, it introduces the problem about how to construct coupling graphs: how to add *the weight on coupling edges*? what is *the length of coupling windows* (i.e., the longest time interval between nodes connected by coupling edges)? For the length of coupling windows, we illustrate examples in Fig. 9. This figure is taken from [63], where each snapshot graph is called a *slice*. Two different lengths of coupling windows are given: (*a*) couplings between neighboring slices such that the length is two time steps and (*b*) all-to-all inter-slice couplings such that the length is the total time steps.

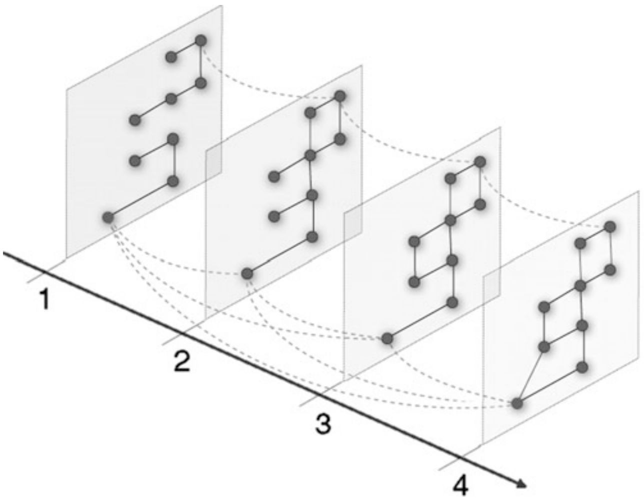

Fig. 9 Schematic of a multislice (couplings) network. Four slices $s = \{1, 2, 3, 4\}$ represented by adjacencies A_{ijs} encode intra-slice connections (*solid*). Inter-slice connections (*dashed*) are encoded by C_{jrs} , specifying coupling of node j to itself between slices r and s. For clarity, inter-slice couplings are shown for only two nodes and depict two different types of couplings: (**a**) coupling between neighboring slices, appropriate for ordered slices, and (**b**) all-to-all inter-slice coupling, appropriate for categorical slices. The figure is gained from [63]

4.4 Benchmarks

When designing a new algorithm, it is necessary to stress it through series of simple benchmark graphs, artificial or from the real world, for which the community structure is known. If the algorithm provides results agreeing with the ground truth, we may consider that the algorithm is reliable and can be used in applications. In this section, we firstly describe current benchmarks for testing dynamic community detection algorithms and secondly review measures for comparing the similarity between computed modular structure and a ground truth.

Computer-Generated Graphs

Computer-generated graphs try to build random graphs that have natural partitions. The simplest model of this form is for the graph bisection problem. This is the problem of partitioning the vertices of a graph into two equal-sized sets while minimizing the number of edges bridging the sets. To create an instance of the planted bisection problem, we first choose a partition of the vertices into equal-sized sets V_1 and V_2. We then choose probabilities $p_{\text{in}} > p_{\text{out}}$ and place edges between vertices with the following probabilities: the expected number of edges crossing between V_1 and V_2 will be $p_{\text{out}}|V_1||V_2|$. If p_{in} is sufficiently larger than p_{out}, then every other bisection will have more crossing edges. There have been many analyses of the generalization of planted partition models to more than 2 partitions [18, 61]. The number of subgraphs is equal to the number of predefined communities, and nodes within the same community are connected with a probability of p_{in} and connect to the rest with a probability of p_{out}. In addition, each subgraph is modeled by an Erdös–Rényi's model, which assigns equal probability to all graph edges. The model is motivated by the idea that vertices (or general items) belong to certain categories and that vertices in the same categories are more likely to be connected. Such models also arise in the analysis of clustering algorithms. However, it is not clear that these models represent practice very well.

Lin et al. [88] have proposed a computer-generated benchmarks for testing their evolutionary clustering framework called FacetNet. They use the model of Newman [68] similar to the previous model as a basis (4 clusters of 32 nodes). They generate different graphs for each time steps. In each time step, dynamic is introduced as follows: from each community, they randomly select 3 members to leave their original community and to join randomly the other three communities. Edges are added randomly with a higher probability p_{in} for within-community edges and a lower probability p_{out} for between-community edges. The average degree for nodes is set to 16.

Another similar benchmark is proposed in [23]. To introduce change points, a sequence of graphs are separated into *segments*. Each *segment* is a sequence of graphs sharing the same community structure. The average degree of nodes and the internal and external connection probability are fixed. The edge weights are integers randomly chosen from 1 to 10 for intra-community edges and from 1 to 6 for inter-community edges.

All benchmarks for dynamic community detection extended from the planted partition model, used by Newman et al. have two main drawbacks: (a) all nodes have the same expected degree and (b) all communities have equal size. These features are unrealistic, as complex networks are known to be characterized by heterogeneous distributions of degree and community sizes.

Greene and Doyle [39] proposed a set of benchmarks based on Lancichinetti and Fortunato's technique [56]. Lancichinetti and Fortunato assumed that the distributions of degree and community size are power laws, with exponents τ_1 and τ_2, respectively. Each node shares a fraction $1 - \mu$ of its edges with the other nodes of its community and a fraction μ with the rest of the graph; μ is a mixing parameter in range of [0, 1]. Greene and Doyle contracted four different synthetic networks for four different event types, covering $15,000$ nodes over 5 time steps. In each of the four synthetic datasets, 20% of node memberships were randomly permuted at each step to simulate the natural movement of users between communities over time. Subsequently, community dynamic events were added as follows:

Intermittent communities: At each time step, 10% of communities are unobserved from time $t = 2$ onwards.

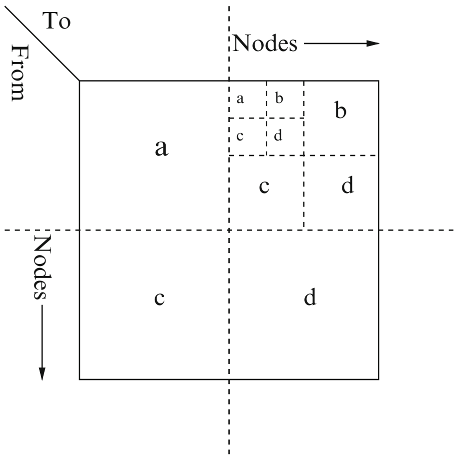

Fig. 10 The R-MAT model. The figure is gained from [14]

Expansion and contraction: At each time step, 40 randomly selected communities expand or contract by 25% of their previous size.

Birth and death: At each time step, 40 additional communities are created by removing nodes from other existing communities and randomly removing 40 existing communities.

Merging and splitting: At each time step, 40 temporal clusters of communities split, together with 40 cases where two existing communities were merged.

Chen et al. [16] constructed benchmark graphs using GTgraph [6] based on a recursive matrix graph model (R-MAT) [14]. The R-MAT model follows the preferential attachment idea (growing model where new nodes prefer to connect to existing nodes with higher degrees). In order to build a graph, the R-MAT recursively subdivides the adjacency matrix into four equal-sized partitions and assigns edges within these partitions with unequal probabilities:

1. Starting with an empty adjacency matrix, which represents a subgraph for edge assignment.
2. Assign edges into the matrix with probabilities a, b, c, d, respectively (see Fig. 10).

The chosen partition is again subdivided into four smaller partitions, and the above procedure is repeated until the chosen partition is composed of a simple cell such as a single node. In Chen et al.'s method, they define some nodes as graph-dependent nodes. These graph-dependent nodes play the role of core nodes and are used to identify communities. The community dynamics can be revealed by the community member changes, where these communities are mapped through graph-dependent nodes.

The main drawback of above computation-generated benchmarks is that the evolution of a dynamic network corresponds to a fixed probability. We may expect that in real networks, communities may experience heterogeneous changes such as bursty node insertion probability, node deletion probability, link insertion probability, or link deletion probability.

Real Networks

Real networks are also used to show performances of algorithms, such as Karate, Football, Dolphins, and Neural. When dealing with real data, the main issue is generally the ground truth or a fine and precise expertise on the datasets. Real networks are released by Newman and can be downloaded from http://www-personal.umich.edu/~mejn/netdata/.

Mucha et al. [63] performed simultaneous community detection across multiple resolutions (scales) in the well-known Zachary Karate Club network, which encoded the friendships between 34 members of a 1970s university karate club [96]. Keeping the same unweighted adjacency matrix across *slices* (each *slice* represents a graph at a time step), the resolution associated to each slice is dictated by a specified sequence of γ_Δ parameters, such as $\gamma_\Delta = \{0.25, 0.5, 0.75, \dots, 4\}$. In other words, given a sequence of slices $\mathcal{A}_{ij\Delta} = \{A_{ij}(1), \dots, A_{ij}(\Delta)\}$, these slices share the same unweighted adjacency matrix such as $\forall\ t_r, t_s\ , A_{ij}(t_r) = A_{ij}(t_s)$. Figure 11 depicts the community assignments obtained for coupling strengths $\omega = \{0, 0.1, 1\}$ between each neighboring pair of the 16 ordered slices. These results simultaneously probe all scales, including the partition of the Karate Club into four communities at the default resolution of modularity. Additionally, nodes that have an especially strong tendency to break off from larger communities are identified.

The previous definition for building benchmark graphs does not change interactions between nodes. Community structure changes observed are caused by tuning the resolution (scale) of the networks. Therefore, we cannot use it to test the reliability of community dynamic detecting algorithms. Its other drawback is that the algorithm should use the same resolution parameter; otherwise, it fails to test the performance of the algorithm in smoothing community evolution.

Comparing Partitions

To measure the similarity between the built-in modular structure of a benchmark and the one delivered by an algorithm, several similarity measurements are possible. The most used similarity measurement is the *normalized mutual information*, which is based on information theory [20]. The idea is that, if two community structures are similar to each other, only little information is used to infer one community structure by given the other one.

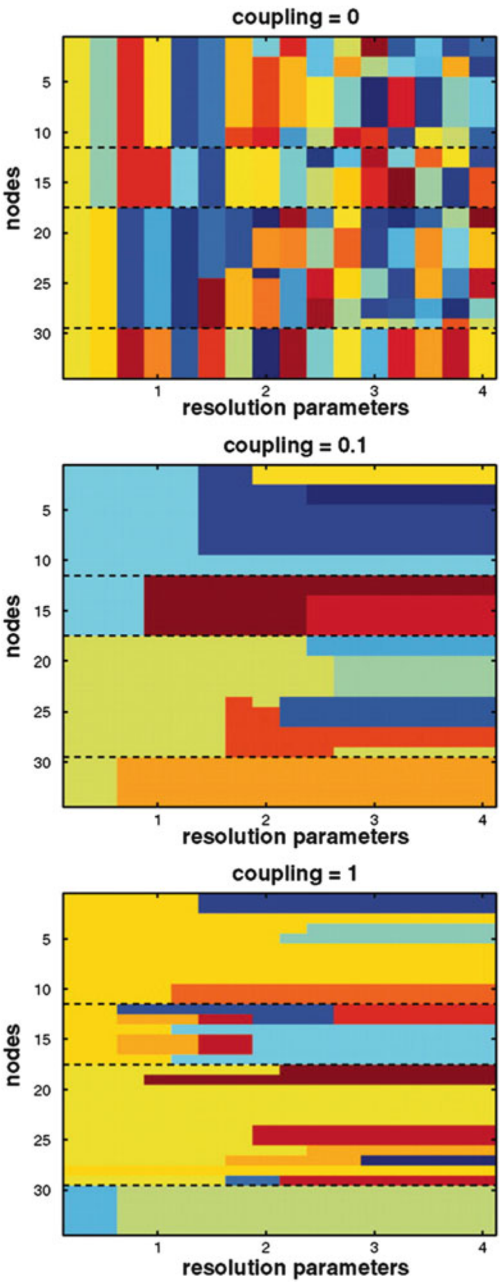

Fig. 11 Multislice community detection of the Zachary Karate Club network [96] across multiple resolutions. Colors depict community assignments of the 34 nodes in each of the 16 slices (with resolution parameters $\gamma_\Delta = \{0.25, 0.5, \ldots, 4\}$), for $\omega = 0$ (*top*), $\omega = 0.1$ (*middle*), and $\omega = 1$ (*bottom*). Dashed lines bound the communities obtained using Newman–Girvan modularity [68]. The figure is gained from [63]

The normalized mutual information is based on the *mutual information*. The mutual information for two random variables X, Y is denoted by $\mathrm{I}(X, Y)$ and is defined as

$$\mathrm{I}(X,Y) = \sum_x \sum_y P(x,y) \log \frac{P(x,y)}{P(x)P(y)}$$

where $P(x)$ indicates the probability that $X = x$ (similarly for $P(y)$) and $P(x, y)$ is the joint probability of X and Y, that is, $P(x, y) = P(X = x, Y = y)$. Actually, $\mathrm{I}(X, Y) = \mathrm{H}(X) - \mathrm{H}(X|Y)$, where $\mathrm{H}(X)$ is the Shannon entropy of X and $\mathrm{H}(X|Y)$ is the entropy of X conditional on Y.

Danon et al. [20] defined the normalized mutual information (NMI) for comparing the similarity between two partitions: $\mathcal{P}_x$ and $\mathcal{P}_y$. Let n_x and n_y denote the number of communities in the partition $\mathcal{P}_x$ and $\mathcal{P}_y$, respectively. The normalized mutual information is defined as

$$\mathrm{NMI} = \frac{2\mathrm{I}(\mathcal{P}_x, \mathcal{P}_y)}{\mathrm{H}(\mathcal{P}_x) + \mathrm{H}(\mathcal{P}_y)} \; . \tag{7}$$

Let

$$\mathrm{I}(\mathcal{P}_x, \mathcal{P}_y) = \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} P(\mathcal{C}_i, \mathcal{C}'_j) \log \frac{P(\mathcal{C}_i, \mathcal{C}'_j)}{P(\mathcal{C}_i) P(\mathcal{C}'_j)}$$

$$\mathrm{H}(\mathcal{P}_x) = -\sum_{i=1}^{n_x} P(\mathcal{C}_i) \log P(\mathcal{C}_i)$$

where n_x and n_y denote the number of communities in two partitions $\mathcal{P}_x$ and $\mathcal{P}_y$, respectively, $P(\mathcal{C}_i) = \frac{|\mathcal{C}_i|}{n}$ and $P(\mathcal{C}_i, \mathcal{C}'_j) = \frac{|\mathcal{C}_i \cap \mathcal{C}'_j|}{n}$.

Danon et al.'s normalized mutual information can be directly written in

$$\mathrm{NMI} = \frac{-2 \sum \sum N_{ij} \log \frac{N_{ij} N}{N_{i\bullet} N_{\bullet j}}}{\sum_{i=1}^{n_x} N_{i\bullet} \log \frac{N_{i\bullet}}{N} + \sum_{j=1}^{n_y} N_{\bullet j} \log \frac{N_{\bullet j}}{N}} \; , \tag{8}$$

where N_{ij} represents the size of overlaps in communities i and community j, $N_{i\bullet}$ is the sum of ith row in matrix N_{ij}, and $N_{\bullet j}$ is the sum of jth column. The normalized mutual information is equal to 1 if the partitions are identical, whereas it has an expected value of 0 if the partitions are independent.

This normalized mutual information is extended for comparing covers in [55]. The normalized mutual information for covers $\mathcal{S}_x$ and $\mathcal{S}_y$ is denoted by $\mathrm{N}(\mathcal{S}_x|\mathcal{S}_y)$ and is defined as

$$\mathrm{N}(\mathcal{S}_x|\mathcal{S}_y) = 1 - \frac{1}{2}\left[\mathrm{H}(\mathcal{S}_x|\mathcal{S}_y)_{\mathrm{norm}} + \mathrm{H}(\mathcal{S}_y|\mathcal{S}_x)_{\mathrm{norm}}\right] \tag{9}$$

where the normalized conditional entropy of $\mathrm{H}(\mathcal{S}_x|\mathcal{S}_y)_{\text{norm}}$ (similarly to $\mathrm{H}(\mathcal{S}_y|\mathcal{S}_x)_{\text{norm}}$) of the cover $\mathcal{S}_x$ with respect to $\mathcal{S}_y$ is defined as

$$\mathrm{H}(\mathcal{S}_x|\mathcal{S}_y)_{\text{norm}} = \frac{1}{n_x}\sum_{i=1}^{n_x}\frac{\mathrm{H}(S_i|\mathcal{S}_y)}{\mathrm{H}(S_i)} \text{ , where } S_i \in \mathcal{S}_x \text{ , } n_x = |\mathcal{S}_x|$$

The conditional entropy of S_i with respect to all the components of $\mathcal{S}_y$ is defined by

$$\mathrm{H}(S_i|\mathcal{S}_y) = \min_{S'_j \in \mathcal{S}_y} \mathrm{H}(S_i|S'_j) \tag{10}$$

where $\mathrm{H}(S_i|S'_j)$ denotes the conditional entropy of a community S_i by given a community S'_j.

As Eq. (10) only counts the minimum $\mathrm{H}(S_i|S'_j)$, this extended normalized mutual information suffers from the following problem: some communities sharing few common nodes may not be taken into account. Moreover, this normalized mutual information is not ideal: given two covers $\mathcal{S}_x, \mathcal{S}_y$, if only one community of $\mathcal{S}_x$ is divided into several small ones in $\mathcal{S}_y$, while all the other communities stay identical, the normalized mutual information is low because some communities have very low conditional entropy.

The main drawback of the above similarity measurements is that they are proposed for static graphs, and they do not consider the community dynamics. Therefore, we propose to measure the similarity between the found community structure and the ground truth of dynamic graphs by counting the similarity between every pair of communities' evolution paths. We can write NMI (7) by setting

$$P(\mathcal{C}_i) = \frac{\sum_{t=1}^{\Delta}|C_i(t)|}{\sum_{t=1}^{\Delta}n(t)}$$

$$P(\mathcal{C}_i, \mathcal{C}_j) = \frac{\sum_{t=1}^{\Delta}|C_i(t) \cap C_j(t)|}{\sum_{t=1}^{\Delta}n(t)}$$

where $n(t)$ represents the nodes assigned to the partition in time t and $C_i(t)$ represents the observation of community $\mathcal{C}_i$ at time t (similarly for $C_j(t)$).

4.5 Visualizing Dynamics in Communities

In early work, several tools such as SoNIA [62] and TeCFlow visualize dynamic networks by creating graph movies, where nodes move as a function of changes in relations. However, these tools fail to indicate a changing behavior of community memberships and community dynamics. In [63], matrix is used (Similar as Fig. 11), whose element represents the community membership of a node at a time step. Each

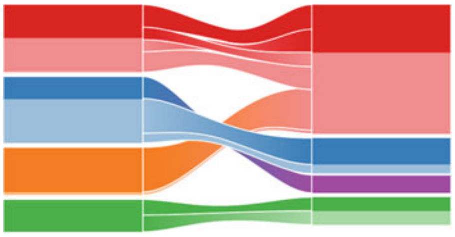

Fig. 12 *Example of mapping between communities.* In the bottom networks, the *darker colors* represent nodes that are clustered together in at least 95% of the 1, 000 bootstrap networks. The alluvial diagram highlights and summarizes the structural changes between the time 1 and time 2 significance clusters. The height of each block represents the volume of flow through the cluster. The clusters are ordered from bottom to top by their size, with mutually nonsignificant clusters placed together and separated by a third of the standard spacing. The *orange module* merges with the *red module*, but the nodes are not clustered together in 95% of the bootstrap networks. The *blue module* splits, but the significant nodes in the *blue and purple modules* are clustered together in more than 5% of the bootstrap networks. The figure is obtained from [78]

node occupies a column. Colors are used to depict communities. We can observe how a node changes its community membership through the color change in the corresponding column. The drawback is that we do not directly observe how one community emerges, merges, splits, or disappears.

An example of a graph with dynamic communities is depicted in Fig. 12. The evolution path of a dynamic community is depicted by a diagram occupying a column. Each diagram represents a community as a block and shows relationships between preceding and succeeding clusters through horizontally connected stream fields. This result is obtained by the algorithm of bootstrap [25] in [78]. It enables to show community dynamics. For example, we observe the orange module merges with the red module in Fig. 12. In addition, in this case, we are also able to observe the significance of clusters, which is shown by dark color.

Another visualization tool illustrates community evolution through *lineage diagrams* [91]. Each lineage represents a separate evolutionary path and occupies a column. Each cluster is shown by a circle whose size is proportional to its number of nodes. A lineage tie is added between two clusters if they share a successor or predecessor relationship. Therefore, if a circle has a link to another column, it indicates a community change. For example, in Fig. 13, we observe a violet cluster having a link with blue color which connects it and a blue cluster. The link color is given by the predecessor if there exists a predecessor relationship; otherwise, the color is given by the successor. Therefore, we easily identify community dynamics. In this case, we say that the blue cluster is a split of the violet community. Moreover, the blue cluster has a link with green color which links it to a green cluster. We say that the green community merges into the blue community.

Although the layout displaying the community evolution in a movie seems suitable to analyze the dynamic activities of networks and communities, it is less suitable to analyze community dynamics. Displaying the community dynamics is

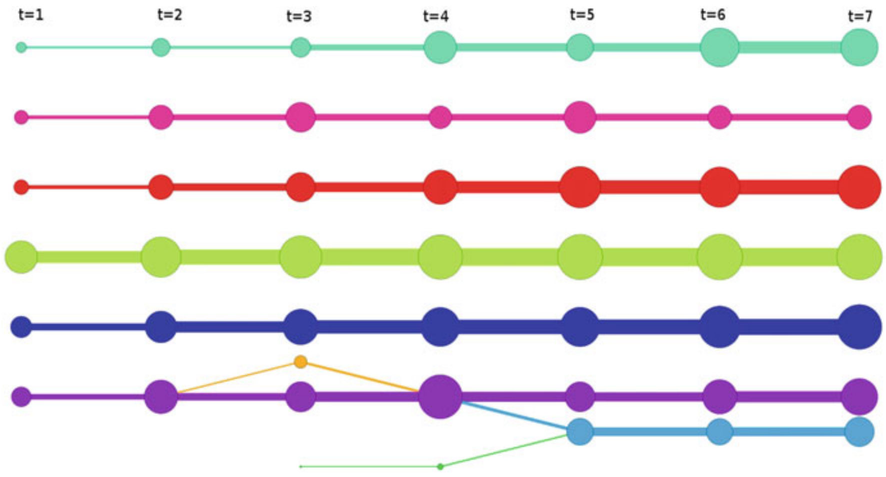

Fig. 13 Example of lineage diagrams. Between $t = 2$ and $t = 3$, an orange cluster is split from the violet community. At $t = 3$, a new green cluster is emerged. Between $t = 3$ and $t = 4$, the orange cluster is merged into the violet community. Between $t = 4$ and $t = 5$, a blue cluster is split from the violet cluster and it is merged with the green community simultaneously

very meaningful to understand the "history" of community structure in networks. There are several visualization tools which provide an overview of community dynamics. Their main drawback is that the roles and dynamic behaviors of nodes are unobservable for analysis. For example, how overlapping nodes evolve is still a problem in the visualization. Therefore, many effects need to make particularly in visualizing roles and behaviors of nodes in dynamic aspect.

5 Conclusion

In this chapter, we have reviewed current research about community detection in dynamic networks. From our exposition, we have seen a great number of clustering techniques. Two-stage algorithms are nature and simple, but the matching problem is hard to resolve. The results also suffer from temporal variance and/or algorithm-generated influence. Techniques based on the principle of evolutionary clustering also have many limits such as the tunable parameter value for community model. However, a method without faults will never be found. Thus, it is better to find a general method that gives good results for network analysis. Moreover, a number of important open issues are left in current research, such as benchmark graphs, quality function, and overlapping community evolution.

Various algorithms have been proposed and developed. Which algorithm has the best performance? To resolve this problem, we need reliable benchmarks to test algorithms. We have reviewed benchmarks [16, 23, 39]. The existing computer-generated benchmarks correspond to a fixed probability. Oppositely, communities in real networks may evolve with heterogeneous changes such as bursty node insertion probability, node deletion probability, link insertion probability, or link deletion probability. Moreover, to oppose the current real network benchmarks which are static networks, we lack real network benchmarks whose interactions change over time. The benchmark graph is a crucial issue in the area of community detection in dynamic networks.

After applying algorithms to benchmarks, we need to compare their performances to find the best one. At this moment, we cannot determine since no quality function is accepted to measure the goodness of the found dynamic communities. The α-cost function (3) depends on a tunable parameter with no a priori knowledge. There is a quality function without any parameter. It is proposed in [13] by transforming the problem of community detection into a problem of message decoding within information theory. The problem is whether information theory can be used to define communities. To our knowledge, we are still far from a precise definition of communities, while modularity (2) is the widely accepted quality function. Nonetheless, information theory study represents an important research line for the future of the field: (*a*) respect community history to better understand communities and (*b*) capture significant changes which represent crucial changes in community structure of networks.

Most algorithms in the literature deal with communities without overlapping nodes. We know that real networks may be overlapped like interdisciplinary which combines two or more academic fields in science, multiple citizenship which allows a person to be a citizen of more than one state, and proteins playing the intermediate role between several functions in biology networks. The frameworks [16] are proposed to trace overlapping community evolution. However, it is under one assumption that communities are clique-like objects, which is only suitable to detect communities in some specific networks. To detect overlapping communities in different types of networks, we need another resolution which can capture the general characteristics of overlapping nodes.

Finally, the main motivation encouraging us for community detection is to mine the relationship between the algorithmic communities with the reality. Why do communities split, or merge, or disappear? What are the effects of overlapping nodes? To answer these questions, we study features behind graph evolution and hope to learn more information.

Acknowledgement This work is supported in part by the French National Research Agency contract DynGraph ANR-10-JCJC-0202.

References

[1] J.I. Alvarez-hamelin, A. Barrat, A. Vespignani, Large scale networks fingerprinting and visualization using the k-core decomposition, in *Advances in Neural Information Processing Systems 18* (MIT Press, 2006), pp. 41–50
[2] S. White, P. Smyth, A spectral clustering approach to finding communities in graphs, in *SIAM International Conference on Data Mining* (2005)
[3] S. Asur, S. Parthasarathy, D. Ucar, An event-based framework for characterizing the evolutionary behavior of interaction graphs, in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (KDD, 2007), 978-1-59593-609-7, San Jose, CA, pp. 913–921. http://doi.acm.org/10.1145/1281192.1281290
[4] T. Aynaud, J.-L. Guillaume, Long range community detection, *LAWDN - Latin-American Workshop on Dynamic Networks*, INTECIN - Facultad de Ingeniería (U.B.A.) - I.T.B.A, p. 4 (2010). http://hal.inria.fr/inria-00531750/PDF/lawdn2010_submission_5.pdf
[5] L. Backstrom, D. Huttenlocher, J. Kleinberg, X. Lan, Group formation in large social networks: membership, growth, and evolution, in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, p. 54, 2006
[6] D.A. Bader, K. Madduri, Gtgraph: A synthetic graph generator suite (2006). http://www.cse.psu.edu/~madduri/software/GTgraph/index.html
[7] Y. Bar-Yam, *Dynamic of Complex Systems*. Addison-Wesley stydies in nonlinearity. The Advanced Book Program (Westview Press, Boulder, 1997), p. 848., ISBN 0813341213, 9780813341217
[8] M. Beiró, J. Busch, Visualizing communities in dynamic networks, in *Latin American Workshop on Dynamic Networks*, vol. 1, 2010
[9] R. Bekkerman, A. Mccallum, G. Huang, Automatic Categorization of Email into Folders: Benchmark Experiments on Enron and SRI Corpora. Center for Intelligent Information Retrieval, Department of Computer Science, University of Massachusetts Amherst, Technical Report IR (2004) pp. 4–6
[10] J.C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms* (Kluwer Academic, Norwell, MA, 1981)
[11] V.D. Blondel, J.L. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks. J. Stat. Mech. Theory Experiment **2008**(10), P10008 (2008). http://stacks.iop.org/1742-5468/2008/i=10/a=P10008
[12] J. Camacho, R. Guimerá, L.A.N. Amaral, Robust patterns in food web structure. Phys. Rev. Lett. **88**(22), 228102 (2002)
[13] D. Chakrabarti, S. Papadimitriou, D.S. Modha, C. Faloutsos, Fully automatic cross-associations, in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04 (ACM, New York, NY, 2004), pp. 79–88
[14] D. Chakrabarti, Y. Zhan, C. Faloutsos, R-mat: A recursive model for graph mining, in *In SDM*, 2004
[15] J. Chen, B. Yuan, Detecting functional modules in the yeast protein-protein interaction network. Bioinformatics **22**(18), 2283–2290 (2006)
[16] Z. Chen, K.a. Wilson, Y. Jin, W. Hendrix, N.F. Samatova, Detecting and tracking community dynamics in evolutionary networks, in *2010 IEEE International Conference on Data Mining Workshops*, pp. 318–327, Dec. 2010
[17] Y. Chi, S. Zhu, X. Song, J. Tatemura, B.L. Tseng, Structural and temporal analysis of the blogosphere through community factorization, in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '07*, p. 163, 2007
[18] A. Condon, R.M. Karp, Algorithms for graph partitioning on the planted partition model. Random Struct. Algorithm **18**, 116–140 (1999)

[19] K. Crawford, Presented at Oxford Internet Institute's A Decade in Internet Time: Symposium on the Dynamics of the Internet and Society, Six provocations for big data. Social Science Research Network Working Paper Series (13 September 2011) Computer 1–17 (2011)

[20] L. Danon, A. Díaz-Guilera, J. Duch, A. Arenas, Comparing community structure identification. J. Stat. Mech. Theor Exp. **2005**(09), P09008 (2005)

[21] T. Dinh, I. Shin, N. Thai, M. Thai, A general approach for modules identification in evolving networks. Dynam. Inform. **40**(4), 83–100 (2010)

[22] Y. Dourisboure, F. Geraci, M. Pellegrini, Extraction and classification of dense communities in the web, in *Proceedings of the 16th International Conference on World Wide Web*, WWW '07 (ACM, New York, NY, 2007), pp. 461–470

[23] D. Duan, Y. Li, Y. Jin, Z. Lu, Community mining on dynamic weighted directed graphs, in *Proceeding of the 1st ACM International Workshop on Complex Networks Meet Information and Knowledge Management* (ACM, 2009), pp. 11–18

[24] H. Ebel, L.-I. Mielsch, S. Bornholdt, Scale-free topology of e-mail networks. Phys. Rev. E Stat. Nonlin. Soft. Matter. Phys. **66**(3 Pt 2A), 035103 (2002)

[25] B. Efron, R.J. Tibshirani, *An Introduction to the Bootstrap* (Chapman & Hall, New York, 1993)

[26] N.B. Ellison, C. Steinfield, C. Lampe, The benefits of facebook friends: Social capital and college students use of online social network sites. J. Comput. Mediat. Comm. **12**(4), 1143–1168 (2007)

[27] M. Ester, H.-P. Kriegel, J. Sander, M. Wimmer, X. Xu, Incremental clustering for mining in a data warehousing environment, in *Proceedings of 24rd International Conference on Very Large Data Bases, VLDB'98* ed. by A. Gupta, O. Shmueli, J. Widom (Morgan Kaufmann, New York City, New York, 1998), pp. 323–333

[28] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise in *Conference on Knowledge Discovery and Data Mining* (AAAI Press, 1996), pp. 226–231

[29] T. Falkowski, A. Barth, M. Spiliopoulou, Studying community dynamics with an incremental graph mining algorithm, in *Proc. of the 14 th Americas Conference on Information Systems (AMCIS 2008)*, pp. 1–11, 2008

[30] T. Falkowski, M. Spiliopoulou, Data mining for community dynamics. Kunstliche Intelligenz **3**, 23–29 (2007)

[31] T. Falkowski, M. Spiliopoulou, Users in volatile communities: Studying active participation and community evolution. Lect. Note Comput. Sci. **4511**, 47 (2007)

[32] S. Fortunato, Community detection in graphs. Phys. Rep. **486**(3–5), 75–174 (2010)

[33] S. Fortunato, M. Barthelemy, Resolution limit in community detection. Proc. Natl. Acad. Sci. USA **104**(1), 36–41 (2007)

[34] E. Gabrilovich, S. Markovitch, Computing semantic relatedness using wikipedia-based explicit semantic analysis, in *Proceedings of The Twentieth International Joint Conference for Artificial Intelligence*, pp. 1606–1611, Hyderabad, India, 2007

[35] L.C. Giles, K. Bollacker, S. Lawrence, Citeseer: an automatic citation indexing system, in *The Third ACM Conference on Digital Libraries*, (ACM Press, Pittsburgh, 1998), pp. 89–98

[36] M. Girvan, M.E.J. Newman, Community structure in social and biological networks. Proc. Natl. Acad. Sci. USA **99**, 7821–7826 (2002)

[37] P. Gongla, C.R. Rizzuto, Where did that community go? - communities of practice that disappear, in *Knowledge Networks: Innovation Through Communities of Practice*, ed. by P. Hildreth, C. Kimble (Idea Group, Hershey, PA, 2004)

[38] R. Görke, P. Maillard, C. Staudt, Modularity-driven clustering of dynamic graphs. In: *Proceedings of the 9th International Symposium on Experimental Algorithms (SEA'10)*, Lecture Notes in Computer Science, vol. 6049 (Springer, 2010), pp. 436–448

[39] D. Greene, D. Doyle, P. Cunningham, Tracking the evolution of communities in dynamic social networks, in *2010 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2010)*, 9–11 August 2010, Odense, Denmark, N. Memon, R. Alhajj. The Institute of Electrical and Electronics Engineers, Inc. (IEEE, 2010), pp. 9–11. ISBN: 978-1-4244-7787-6

[40] P.D. Grnwald, I.J. Myung, M.A. Pitt, *Advances in Minimum Description Length: Theory and Applications (Neural Information Processing series)* (MIT Press, 2005), 444 p. ISBN 0262072629, 9780262072625

[41] R. Guimerá, L.A.N. Amaral, Cartography of complex networks: modules and universal roles. J. Stat. Mech. **2005**(P02001), nihpa35573 (2005)

[42] R. Guimerá, M. Sales-Pardo, L.A.N. Amaral, Modularity from fluctuations in random graphs and complex networks. Phys. Rev. E Stat. Nonlin. Soft. Matter. Phys. **70**(2 Pt 2), 025101 (2004)

[43] L.H. Hartwell, J.J. Hopfield, S. Leibler, A.W. Murray, From molecular to modular cell biology. Nature **402**(6761), 47 (1999)

[44] J. Hopcroft, O. Khan, B. Kulis, B. Selman, Tracking evolving communities in large linked networks. Proc Natl Acad Sci USA **101**(1), 5249–5253

[45] A. Jain, R. Dubes, *Algorithms for Clustering Data* (Prentice-Hall, Upper Saddle River, NJ, 1988)

[46] M. James, D.R, White, Structural cohesion and embeddedness: A hierarchical concept of social groups. Am. Socio. Rev. **68**(1), 103–127 (2003)

[47] M.B. Jdidia, C. Robardet, E. Fleury, Communities detection and analysis of their dynamics in collaborative networks, in *Dynamic Communities: from Connectivity to Information Society Workshop Co-located with ICDIM'07* (IEEE, 2007), pp. 11–17. http://liris.cnrs.fr/publis/?id=3258

[48] H. Jeong, S.P. Mason, A.L. Barabási, Z.N. Oltvai, Lethality and centrality in protein networks. Nature **411**(6833), 41–42 (2001)

[49] H. Jeong, Z. Néda, A.L. Barabási, Measuring preferential attachment in evolving networks. Europhys. Lett. **61**(4), 567–572 (2003)

[50] E.M. Jin, M. Girvan, M.E. Newman, Structure of growing social networks. Phys. Rev. E Stat. Nonlin. Soft. Matter. Phys. **64**(4 Pt 2), 046132 (2001)

[51] A.E. Krause, K.A. Frank, D.M. Mason, R.E. Ulanowicz, W.W. Taylor, Compartments revealed in food-web structure. Nature **426**(6964), 282–285 (2003)

[52] R. Kumar, A. Tomkins, D. Chakrabarti, Evolutionary clustering, in *Proc. of the 12th ACM SIGKDD Conference*, 2006

[53] H. Kwak, Y. Choi, Y.-H. Eom, H. Jeong, S. Moon, Mining communities in networks: a solution for consistency and its evaluation, in *Proc. of the 9th ACM SIGCOMM Conference on Internet Measurement Conference*, pp. 301–314, 2009

[54] A. Lancichinetti, S. Fortunato, Consensus clustering in complex networks. Sci. Rep. **2**, 336 (2012)

[55] A. Lancichinetti, S. Fortunato, Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. Phys. Rev. **E80**, 016118 (2009)

[56] A. Lancichinetti, S. Fortunato, F. Radicchi, Benchmark graphs for testing community detection algorithms. Phys. Rev. **E78**, 046110 (2008)

[57] J. Leskovec, J. Kleinberg, C. Faloutsos, Graphs over time: densification laws, shrinking diameters and possible explanations, in *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining* KDD'05, Chicago, IL (ACM, New York, 2005), pp. 177–187. ISBN 1-59593-135-X, http://doi.acm.org/10.1145/1081870.1081893

[58] J. Leskovec, K.J. Lang, M. Mahoney, Empirical comparison of algorithms for network community detection, in *Proceedings of the 19th International Conference on World Wide Web*, WWW '10 (ACM, New York, NY, 2010), pp. 631–640

[59] R.D. Luce, A.D. Perry, A method of matrix analysis of group structure. Psychometrika **14**(2), 95–116 (1949)

[60] C.P. Massen, J.P.K. Doye, Thermodynamics of community structure. Arxiv preprint cond-mat/0610077 (2006)

[61] F. McSherry, Spectral partitioning of random graphs, in *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*, FOCS '01 (IEEE Computer Society, Washington, DC, 2001), p. 529

[62] J. Moody, D.A. McFarland, S. Bender-DeMoll, Dynamic network visualization: methods for meaning with longitudinal network movies. Am. J. Sociol. **110**, 1206–1241 (2005)

[63] P. Mucha, T. Richardson, K. Macon, M.A. Porter, Community structure in time-dependent, multiscale, and multiplex networks. Science **876**, 10–13 (2010)

[64] M.E.J. Newman, Analysis of weighted networks. Phys. Rev. E **70**, 056131 (2004)

[65] M.E.J. Newman, Fast algorithm for detecting community structure in networks. Phys. Rev. E Stat. Nonlin. Soft. Matter. Phys. **69**(6 Pt 2), 066133 (2004)

[66] M.E.J. Newman, Modularity and community structure in networks. Proc. Natl. Acad. Sci. USA **103**(23), 8577–8582 (2006)

[67] M.E.J. Newman, A.L. Barabási, D.J. Watts (eds.), *The Structure and Dynamics of Networks* (Princeton University Press, Princeton, 2006)

[68] M.E.J. Newman, M. Girvan, Finding and evaluating community structure in networks. Phys. Rev. E **69**(2), 26113 (2004)

[69] H. Ning, W. Xu, Y. Chi, Y. Gong, T. Huang, Incremental spectral clustering with application to monitoring of evolving blog communities, in *SIAM International Conference on Data Mining*, 2007

[70] G. Palla, A.-L. Barabasi, T. Vicsek, Quantifying social group evolution. Nature **446**, 664–667 (2007)

[71] G. Palla, I. Derenyi, I. Farkas, T. Vicsek, Uncovering the overlapping community structure of complex networks in nature and society. Nature **435**, 814 (2005)

[72] R. Pastor-Satorras, A. Vázquez, A. Vespignani, Dynamical and correlation properties of the internet. Phys. Rev. Lett. **87**(25), 258701 (2001)

[73] R. Pastor-Satorras, A. Vespignani, *Evolution and Structure of the Internet: A Statistical Physics Approach* (Cambridge University Press, New York, NY, 2004)

[74] P. Pons, M. Latapy, Computing communities in large networks using random walks. J. Graph Algorithm Appl. **10**, 191–218 (2006)

[75] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, D. Parisi, Defining and identifying communities in networks. Proc. Natl. Acad. Sci. USA **101**(9), 2658–2663 (2004)

[76] E. Ravasz, A.L. Somera, D.A. Mongru, Z.N. Oltvai, A.L. Barabási, Hierarchical organization of modularity in metabolic networks. Science **297**(5586), 1551–1555 (2002)

[77] J. Reichardt, S. Bornholdt, Statistical mechanics of community detection. Phys. Rev. E **74**(1), 016110 (2006)

[78] M. Rosvall, C.T. Bergstrom, Mapping change in large networks. PLoS ONE **5**(1), e8694 (2010). doi:10.1371/journal.pone.0008694

[79] M. Saerens, F. Fouss, L. Yen, P. Dupont, The principal components analysis of a graph, its relationships to spectral clustering, in *Proceedings of the 15th European Conference on Machine Learning (ECML 2004)*. Lecture Notes in Artificial Intelligence (Springer, 2004), pp. 371–383

[80] J. Scott, *Social Network Analysis: A Handbook*, 2nd edn. (SAGE Publications, 2000), 240 p.

[81] M. Seifi, J.-L. Guillaume, I. Junier, J.-B. Rouquier, S. Iskrov, Stable community cores in complex networks, in *Proceedings of the 3rd Workshop on Complex Networks (CompleNet 2012)*, 2012

[82] J. Shi, J. Malik, Normalized cuts and image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **22**, 888–905 (1997)

[83] J. Sun, C. Faloutsos, S. Papadimitriou, P. Yu, Graphscope: parameter-free mining of large time-evolving graphs, in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (ACM, New York, NY, 2007), pp. 687–696

[84] L. Tang, H. Liu, J. Zhang, Z. Nazeri, Community evolution in dynamic multi-mode networks, in *International Conference on Knowledge Discovery and Data Mining*, p. 8, 2008

[85] C. Tantipathananandh, T. Berger-Wolf, Constant-factor approximation algorithms for identifying dynamic communities, in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09 (ACM, New York, NY, 2009), pp. 827–836

[86] C. Tantipathananandh, T. Berger-Wolf, D. Kempe, A framework for community identification in dynamic social networks, in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '07*, p. 717, 2007
[87] M. Toyoda, M. Kitsuregawa, Extracting evolution of web communities from a series of web archives, in *Proceedings of the Fourteenth ACM Conference on Hypertext and Hypermedia* (ACM, New York, NY, 2003), pp. 28–37
[88] B.L. Tseng, Y.-R. Lin, Y. Chi, S. Zhu, H. Sundaram, Facetnet: a framework for analyzing communities and their evolutions in dynamic networks. Soc. Network 685–694 (2008)
[89] B.L. Tseng, Y.-R. Lin, Y. Chi, S. Zhu, H. Sundaram, Analyzing communities and their evolutions in dynamic social networks. ACM Trans. Knowl. Discov. Data **3**(2), 1–31 (2009)
[90] U. von Luxburg, A tutorial on spectral clustering. Stat. Comput. **17**(4), 395–416 (2007)
[91] Q. Wang, Détection de communautés recouvrantes dans des réseaux de terrain dynamiques. PhD thesis, Docteur de l'Ecole Normale Supérieure de Lyon, 2012
[92] Y. Wang, B. Wu, N. Du, Community evolution of social network: feature, algorithm and model. Sci. Tech. (60402011) (2008). arXiv:0804.4356
[93] S. Wasserman, K. Faust, *Social Network Analysis: Methods and Applications (Structural Analysis in the Social Sciences).* Structural analysis in the social sciences, vol. 8, 1st edn. (Cambridge University Press, Cambridge, 1994)
[94] S. White, P. Smyth, A spectral clustering approach to finding communities in graphs, in *SDM*, pp. 43–55, 2005
[95] T. Yang, Y. Chi, S. Zhu, Y. Gong, R. Jin, A bayesian approach toward finding communities and their evolutions in dynamic social networks, in *SIAM Conference on Data Mining (SDM)*, 2009
[96] W.W. Za chary, An information flow model for conflict and fission in small groups. J. Anthropologica **1**(33), 452–473 (1977)