Chapter 10
Challenges in Community Discovery on Temporal Networks

Remy Cazabet and Giulio Rossetti

Abstract Community discovery is one of the most studied problems in network science. In recent years, many works have focused on discovering communities in temporal networks, thus identifying dynamic communities. Interestingly, dynamic communities are not mere sequences of static ones; new challenges arise from their dynamic nature. Despite the large number of algorithms introduced in the literature, some of these challenges have been overlooked or little studied until recently. In this chapter, we will discuss some of these challenges and recent propositions to tackle them. We will, among other topics, discuss of community events in gradually evolving networks, on the notion of identity through change and the ship of Theseus paradox, on dynamic communities in different types of networks including link streams, on the smoothness of dynamic communities, and on the different types of complexity of algorithms for their discovery. We will also list available tools and libraries adapted to work with this problem.

Keywords Temporal networks · Community detection

10.1 Introduction

The modular nature of networks is one of the most studied aspects of network science. In most real-world networks, a mesoscale organization exists, with nodes belonging to one or several *modules* or *clusters* (Newman 2006): think of groups in social networks (groups of friends, families, organizations, countries, etc.), or biological networks such as brain networks (Meunier et al. 2010). The term *community* is commonly used in the network science literature to describe a *set* of nodes that are grouped for *topological* reasons (e.g., they are strongly connected together and more

R. Cazabet (✉)
Univ Lyon, UCBL, CNRS, LIRIS UMR 5205, F-69621 Lyon, France
e-mail: remy.cazabet@gmail.com

G. Rossetti
Knowledge Discovery and Data Mining Lab, ISTI-CNR, Pisa, Italy

weakly connected to the rest of the network. Other topological criteria exist, such as having a high internal clustering, similar connection patterns, etc. See Sect. 10.3 for more on this topic). The literature on the topic is large and diverse, not only on the topic of automatic community discovery but also on community evaluation, analysis, or even generation of networks with realistic community structure. In the last ten years, many works have focused on adapting those problems to temporal networks (Rossetti and Cazabet 2018). In this chapter, we present an overview of the active topics of research on dynamic communities. For each of these topics, when relevant, we highlight some current challenges.

The chapter is organized into five parts. In the first one, we discuss the definition of dynamic *clusters* in temporal networks, and how to represent them. In the second section, we concentrate on the specificity of dynamic communities, in particular focusing on *smoothness*, *identity* and *algorithmic complexity*. Section 10.4 focuses on the differences between communities in different types of dynamic networks such as link streams or snapshot sequences. In Sect. 10.5, we discuss the evaluation of dynamic communities, using internal and external evaluation –requiring appropriate synthetic benchmarks. Finally, in Sect. 10.6, we briefly introduce existing tools to work with dynamic communities.

10.2   Representing Dynamic Communities

The first question to answer when dealing with communities is: *what is a good community?* There is no universal consensus on this topic in the literature; thus, in this article, we adopt a definition as large as possible:

Definition 10.1 (*Community*) A (static) community in a graph $G = (V, E)$ is i) a *cluster* (i.e., a *set*) of nodes $C \subseteq V$ ii) having *relevant* topological characteristics as defined by a community detection algorithm.

The second part of this definition will be discussed in Sect. 10.3, and is concerned by the question of the *quality* of a set of nodes as a community, based on a topological criterion. On the contrary, this section discusses the transposition of the first part of this definition to temporal networks, i.e., the definition of *dynamic node clusters* themselves, independently of any quality criteria. We use the term cluster in its data analysis meaning, i.e., clusters are groups of items defined such as those items are more similar (in some sense) to each other than to those in other groups (clusters).

To define dynamic clusters, we first need to define what is a temporal network. This question will be discussed in detail in Sect. 10.4. For now, let's adopt a generic definition provided in Latapy et al. (2017), representing in an abstract way any type of temporal network:

Definition 10.2 (*Temporal Network*) A temporal network, or stream graph, is defined as $S = (T, V, W, E)$, with V a set of nodes, T a set of time instants (continuous or discrete), $W \subseteq T \times V$, and $E \subseteq T \times V \otimes V$.

10.2.1 Fixed Membership Cluster in Temporal Networks

The first possible transposition of static clusters to temporal networks is to consider memberships as fixed.

Definition 10.3 (*Fixed Membership Cluster*) A fixed membership cluster is defined on a temporal network $S = (T, V, W, E)$ as a cluster of nodes $C \subseteq V$.

In fixed membership clusters, nodes cannot change community along time. Communities identified using this definition in a temporal network are usually considered relevant when the clustering they induce would be considered relevant according to a static definition of communities (e.g., modularity) in most times t of the temporal network. Those communities are different from static ones found in the aggregated graph in that they take into account the *temporal order* of edges. Note that in some algorithms such as stochastic block models, in which communities are defined not only by sets of nodes but also by properties of relations between communities, those properties might evolve, while membership themselves stay unchanged (e.g., Matias et al. 2015). This approach can also be combined with *change point detection* to find periods of the graph with *stable* community structures (Peel and Clauset 2014).

10.2.2 Evolving-Membership Clusters in Temporal Networks

In this second transposition of the definition of cluster, nodes can change membership along time. Note that, for methods based on *crisp* communities, each node must belong to one (and only one) community at each step, while less constrained methods allow having nodes not belonging to any community (conversely, belonging to several communities), in some or all steps.

Definition 10.4 (*Evolving-Membership Cluster*) An evolving-membership cluster is defined on a temporal network $S = (T, V, W, E)$ as a cluster $C = \{(t, v), (t, v) \subseteq W\}$.

Dynamic communities using this type of clusters are usually considered relevant when i) the clusters it defines at each t would be considered relevant according to a static definition of communities (e.g., modularity) at each step t, and ii) the clusters it defines at time t are relatively similar to those belonging to the same dynamic cluster at $t - 1$ and $t + 1$. This is related to the notion of dynamic community *smoothness* discussed in Sect. 10.3.1.

10.2.2.1 Persistent-Labels Formalism

The usual way to implement this definition is by using what we call the *persistent labels* formalism: community identifiers—labels—are associated with some nodes

over some periods. There is, therefore, no notion of being an *ancestor/descendent* of another community: two nodes can either share a common label, and therefore be part of the same dynamic community, or not. This representation is the most widespread, used for instance in Mucha et al. (2010), Falkowski et al. (2006).

10.2.3 Evolving-Membership Clusters with Events

One of the most interesting features of dynamic communities is that they can undergo *events*. Their first formal categorization was introduced in Palla et al. (2007), which listed six of them (birth, death, growth, contraction, merge, and split). A seventh operation, continue, is sometimes added to these. In Cazabet and Amblard (2014), an eighth operation was proposed (resurgence). These events, illustrated in Fig. 10.1, are the following:

- **Birth**: The first appearance of a new community composed of any number of nodes.
- **Death**: The vanishing of a community: all nodes belonging to the vanished community lose this membership.
- **Growth**: New nodes increase the size of a community.
- **Contraction**: Some nodes are lost by a community, thus reducing its size.
- **Merge**: Two communities or more merge into a single one.
- **Split**: A community, as a consequence of node/edge vanishing, splits into two or more components.
- **Continue**: A community remains unchanged in consecutive time steps.
- **Resurgence**: A community vanishes for a period, then comes back without perturbations as if it has never stopped existing. This event can be seen as a fake death-birth pair involving the same node set over a lagged period (e.g., seasonal behaviors).

Not all operations are necessarily handled by a generic Dynamic Community Detection algorithm.

Let's consider a situation in which two communities merge at time t. Using the *persistent-labels* formalism introduced previously, this event can be represented in two ways: either both clusters disappear at time t and a new one—the result of the merge—is created, or one of the clusters becomes the merged one from time t, and the other—considered absorbed—disappear. In both cases, important information is lost. A third definition of evolving membership can be used to solve this problem:

Definition 10.5 (*Evolving-membership clusters with events*) Evolving membership cluster with events are defined on a temporal network $S = (T, V, W, E)$ as a set of *fixed-membership Cluster* defined at each time t (or as a set of *evolving-membership clusters*), and a set of community events F. Those events can involve several clusters (merge, split), or a single one (birth, death, shrink, etc.)
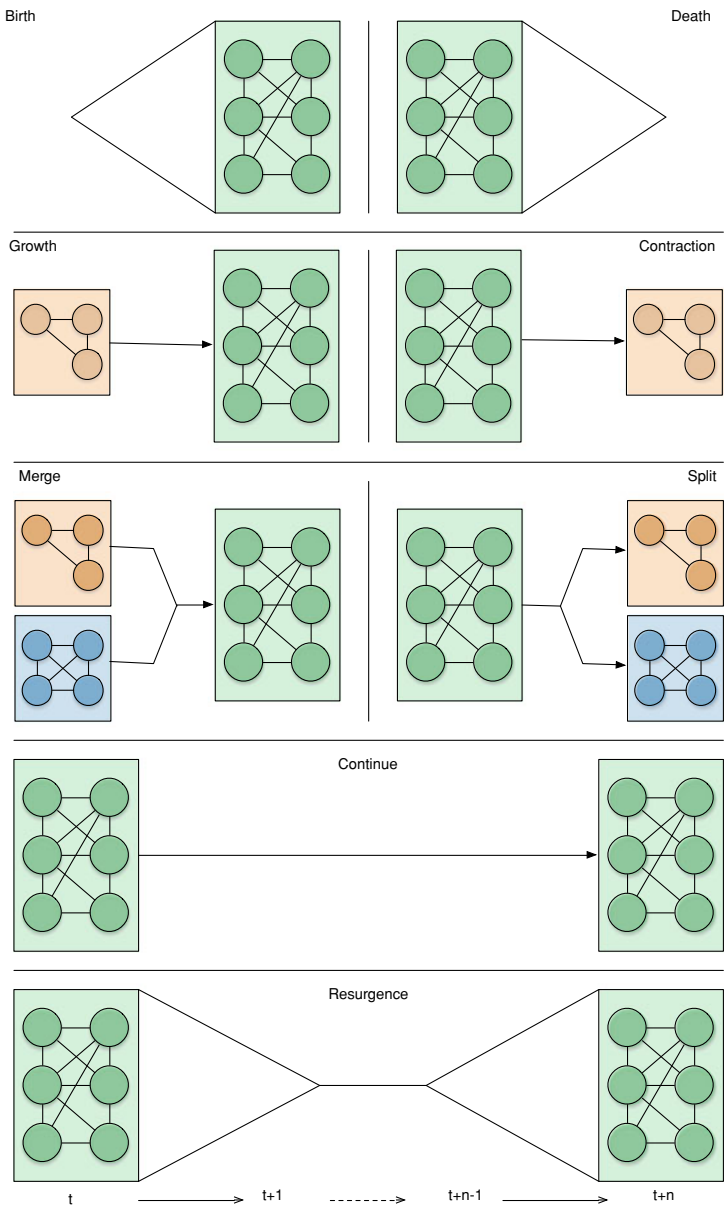
Fig. 10.1 Different types of community events

10.2.3.1   Event-Graph Formalism

In practice, most algorithms that do detect events record them in an ad-hoc manner (e.g., the same event can be recorded as: "a split event occurred to community c_1 at time t, yielding communities c_1 and c_2" or "community c_2 was born at time t, spawn from c_1. Different representations might even be semantically different. A few works, notably (Greene et al. 2010), have used an alternative way to represent dynamic communities and events, using what we call here an *event-graph*. We define it as follows:

Definition 10.6 (*Event Graph*) An event graph is an oriented graph representing dynamic communities of the temporal network $S = (T, V, W, E)$, in which each node corresponds to a pair $\langle C, t \rangle$, with $C \subseteq V$, $t \subseteq T$, and each directed edge represents a relation of continuity between two communities, directed from the earlier to the latter.

Using this representation, some events can be characterized using nodes in/out degrees:

- **In-degree=0** represents new-born communities
- **In-degree$\geq$2** represents merge events
- **Out-degree=0** represents death events
- **Out-degree$\geq$2** represents split events

Events represented by an event graph can be much more complex than simple merge/split, since, for instance, a node-community can have multiple out-going links towards node-community having themselves multiple incoming ones.

Both representations, *event-graph* and *persistent labels*, have advantages and drawbacks. The former can represent any event or relation between different communities at different times, while the later can identify which community is *the same as* which other one in a different time.

10.2.4   Community Life-Cycle

Identified events allow to describe for each cluster the *life-cycle* of its corresponding community:

Definition 10.7 (*Community Life-Cycle*) Given a community C, its life-cycle (which univocally identifies C's complete evolution history) is composed of the directed acyclic graph (DAG) such that (i) the roots are birth events of C, and of its potential predecessors if C has been implicated in merge events; (ii) the leafs are death events, corresponding to deaths of C and of its successors, if C has been implicated in split events; and (iii) the central nodes are the remaining actions of C, its successors, and predecessors. The edges of the tree represent transitions between subsequent actions in C life.

Challenges

Usual events such as birth, merge or shrink were designed to describe a few steps of evolution in the context of snapshot graphs, but are not well suited to describe complex dynamics in networks studied at a fine temporal granularity. In real scenarios, communities are susceptible to evolve gradually. A *shrink* event might corresponds to different scenarios, such as a node switching to another community, a node leaving the system (disappearing), or the community spouting a newborn community composed of a subset of its nodes –and maybe, of other nodes. The usual representation with only labels, even with the addition of some simple events, might be too limited to represent the full range of possible community life-cycle. Defining a complete framework to represent formally complex community evolution scenarios therefore represents a challenge for researchers in the field.

10.3 Detecting Dynamic Communities

Defining what are good communities in networks is already a challenge in itself. Community discovery is often used as an umbrella term for several related problems, not sharing the same formal objective. It stems from earlier, well-defined problems, in particular, *graph partitioning*, which consists, for a graph and given properties of a partition (number and size of clusters), to find affiliations of nodes minimizing the number of inter-cluster edges. This problem is well-defined, in that its objective can be expressed unequivocally in mathematical terms, and has no trivial solution. But having to provide the number and size of communities was considered too constraining when working with real networks having unknown properties. New methods were therefore introduced, based on ideas such as the modularity (Newman and Girvan 2004), compression of random walks (Rosvall and Bergstrom 2008), stochastic block models (SBM) and minimal description length (MDL) (Peixoto 2014), intrinsic properties of communities, and so on. While some of them—e.g., modularity—are based on the same principle of keeping (exceptionally) low the number of inter-community edges, other techniques are searching for completely different things, such as methods based on the Stochastic Block Model framework, in which *blocks* are groups of nodes sharing a similar pattern of connections with nodes belonging to other groups. Furthermore, communities are often categorized in overlapping—one node can belong to several communities—and non-overlapping (crisp) clustering methods. In this chapter, we make abstraction of those differences: each algorithm has a definition of what are *good* static communities, and what we focus on are challenges introduced when going from static to dynamic ones, in particular the notions of temporal smoothness, of identity preservation, and finally the problem of scalability of existing algorithms.

10.3.1 Different Approaches of Temporal Smoothness

In the process of searching for communities over an evolving topology, one of the main questions that need to be answered is: how can the stability of the identified solution be ensured? In static contexts, it has been shown that a generic algorithm executed on the same network that experienced a few topological variations—or even none in case of stochastic algorithms—might lead to different results (Aynaud and Guillaume 2010). The way Dynamic Community Discover (henceforth, DCD) algorithms take into account this problem plays a crucial role in the degree of stability of the solutions they can identify, i.e., on their *smoothness*. In Rossetti and Cazabet (2018) DCD algorithms were grouped in three main categories, depending on the degree of smoothness they aim for:

- Instant Optimal: it assumes that communities existing at time t only depend on the current state of the network at t. Matching communities found at different steps might involve looking at communities found in previous steps, or considering all steps, but communities found at t are considered optimal concerning the topology of the network at t. By definition, algorithms falling in this family are not temporally smoothed. Examples of Instant Optimal algorithms are Palla et al. (2007), Rosvall and Bergstrom (2010), Takaffoli et al. (2011), Chen et al. (2010).
- Temporal Trade-off: it assumes that communities defined at time t depend not only on the topology of the network at t but also on the past topology, past identified partitions, or both. Communities at t are therefore defined as a trade-off between an optimal solution at t and the known past. They do not depend on future topological perturbations. Conversely, from Instant Optimal approaches, the Temporal Trade-off ones are incrementally temporally smoothed. Examples of Temporal Trade-off algorithms are Görke et al. (2010), Cazabet et al. (2010), Rossetti et al. (2017), Folino and Pizzuti (2010).
- Cross-Time: algorithms of this class focus on searching communities relevant when considering the whole network evolution. Methods of this class search a single temporal partition that encompasses all the topological evolution of the observed network: communities identified at time t depend on both past and future network structures. Methods in this class produce communities that are completely temporally smoothed. Examples of Cross-Time algorithms are Aynaud and Guillaume (2011), Matias and Miele (2017), Ghasemian et al. (2016), Jdidia et al. (2007), Viard et al. (2016).

All three classes of approaches have advantages and drawbacks; none is superior to the other since they model different DCD problem definition. Nevertheless, we can observe how each one of them is more suitable for some specific use cases. For instance, if the final goal is to provide on-the-fly community detection on a network that will evolve in the future, Instant Optimal and Temporal Trade-off approaches represent the most suitable fit since they do not require to know in advance all the topological history of the analyzed network. Moreover, if the context requires working with a fine temporal granularity, therefore modeling the observed phenomena

with link streams instead of snapshots, it is suggested to avoid methods of the first class, which are usually defined to handle well defined—stable—topologies.

Temporal smoothness and partition quality often play conflicting roles. We can observe, for instance that, usually:

- Instant Optimal approaches are the best choice when the final goal is to provide communities that are as good as possible at each step of the evolution of the network;
- Cross-Time approaches are the best choice when the final goal is to provide communities that are coherent in time, particularly over the long term;
- Temporal Trade-off approaches represent a trade-off between these other two classes: they are the best choice in the case of continuous monitoring, rapidly evolving data, and in some cases, limited memory applications. However, they can be subject to "avalanche" effects due to the limited temporal information they leverage to identify communities (i.e., partitions evolve based on local temporal-optimal solutions that, on the long run may degenerate).

10.3.2 Preservation of Identity: The Ship of Theseus Paradox

The smoothness problem affects the way nodes are split into communities at each time. A different notion is the question of identity preservation along time, which arises in particular in case of a continued slow evolution of communities. It is well illustrated by the paradox of the *ship of Theseus*. It is originally an ancient thought experiment introduced by Plutarch about the identity of an object evolving through time. It can be formulated as follows.

Let's consider a famous ship, the *ship of Theseus*, composed of planks, and kept in a harbor as a historical artifact. As time passes, some planks deteriorate and need to be replaced by new ones. After a long enough period, all the original planks of the ship have been replaced. Can we consider the ship in the harbor to still be *the same* ship of Theseus? If not, at which point exactly did it ceased to be the same ship?

Another aspect of the problem arises if we add a second part to the story. Let's consider that the removed planks were stored in a warehouse, cleaned, and that a new ship, identical to the original one, is built with them. Should this ship, just built out, be considered as the *real* ship of Theseus, because it is composed of the same elements?

Let's call the original ship A, the ship that is in the harbor after all replacements B, and the reconstructed from original pieces, C. In terms of dynamic community detection, this scenario can be modeled by a slowly evolving community c_1 ($c_1 = A$), from which nodes are removed one after the others, until all of them have been replaced ($c_1 = B$). A new community c_2 appearing after that, composed of the same nodes as the original community ($c_2 = C$). See Fig. 10.2 for an illustration. A static algorithm analyzing the state of the network at every step would be able to discover that there is, at each step, a community (c_1, slowly evolving), and, at the end of the
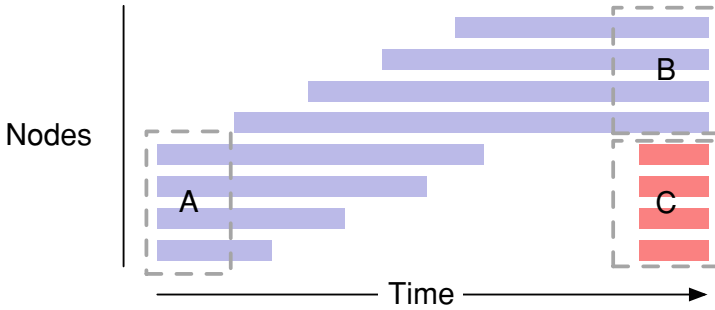
Fig. 10.2 Illustration of the ship of Theseus paradox. Each horizontal line represents a node. A same color represents nodes belonging to the same community according to a topological criterion (e.g., SBM). The community A is progressively modified until reaching state B. Community C is composed of the same nodes as the other community at its start. Which cluster (B or C) has the same identity as A? What if all details of the evolution are not known?

experiment, two communities (c_1 and c_2). But the whole point of dynamic community detection is to yield a longitudinal description, and therefore, to decide when two ships at different points in time are *the same* or not.

This problem has barely been considered explicitly in the literature. However, each algorithm has implicitly to make a choice between which ship is the *true* ship of Theseus. For instance, methods that are based on a successive match of communities, such as Greene et al. (2010), consider that *A* and *B* are the same boats, but not *A* and *C*. On the contrary, a method that matches similar clusters without the constraint of being consecutive, such as Falkowski et al. (2006), consider that *C* is more likely than *B* to be the same ship than *A*. Finally, methods such as Mucha et al. (2010) allow to set what is the influence of time on similarity, and therefore, to choose between those two extreme solutions.

Challenges

The question of identity preservation in dynamic communities has been little discussed and experimented in the literature. For the sake of simplicity, most proposed methods use a mechanism of iterative matching or update of communities and therefore ignore the similarity between ships *A* and *C*. However, this situation is probably very common in real networks, for instance, when confronted with seasonal or other cyclical patterns, where groups can disband and reform later. Developing new methods aware of the choice made in terms of identity preservation is, therefore, a challenge for the community.

10.3.3 Scalability and Computational Complexity

Early methods for community detection in static graphs had high computational complexity (e.g., Girvan and Newman 2002), thus were not scalable to large graphs. One part of the success of methods such as louvain (Blondel et al. 2008) or infomap (Rosvall and Bergstrom 2008) is that they can handle networks of thousands of nodes and millions of edges.

Dynamic graphs represent a new challenge in terms of complexity. Among existing algorithms, we can distinguish different categories

- Those whose complexity depends on the *average size of the graph*
- Those whose complexity depends on the *number of graph changes*.

Let's consider the example of a (large) graph composed of n nodes and m edges at time t, and which is evolving at the speed of k changes every step, for s steps. Algorithms in the first category, such as *identify & match* methods, needs to first compute communities at every step, thus their complexity is proportional to $s\mathcal{O}^{CD}(n, m) + (s - 1)\mathcal{O}^{\rightarrow}(n)$ with $\mathcal{O}^{CD}(n, m)$ the complexity of the algorithm used at each step, and $\mathcal{O}^{\rightarrow}(n)$ the complexity of the matching process for communities found on the n nodes.

Conversely, the complexity of an algorithm that update communities at each step such as Cazabet et al. (2010) is roughly proportional (after the initial detection) to $s\mathcal{O}^{+=}(k)$, with $\mathcal{O}^{+=}(k)$ the complexity of updating the community structure according to k changes. As a consequence, the first category is more efficient in situations where k is large, and n/m are small, while the second is more efficient when n/m are large and k small. The complexity is not necessarily imposed by the adopted definition of community. For instance, algorithms proposed in Palla et al. (2007) and Boudebza et al. (2018) yields rigorously the same dynamic communities, but they belong respectively to the first and second categories, as studied in Boudebza et al. (2018).

Another aspect to consider is parallelization. Although the computation of $\mathcal{O}^{CD}$ on many steps might seem expensive, this task can straightforwardly be processed in parallel. On the contrary, methods involving smoothing, or updating the structure in order, cannot be parallelized, as they need to know the communities at time t to compute communities at time $t + 1$. One must, therefore, consider the properties of a temporal network to know which method will or will not be computationally efficient on it.

Challenges

The complexity of DCD algorithms has barely been explored and represents an important challenge to consider in future works. It is important to note that when dynamic networks are considered at a fine temporal resolution as in link streams, the number of edges (interactions) can be much larger than the number of nodes. For

instance, in the SocioPatterns Primary School dataset (Stehlé et al. 2011), more than 77 000 interactions are observed in a period spanning two days, despite having only 242 nodes. Algorithms developed for static algorithms use the sparsity of networks to improve their efficiency, but such an approach might be less rewarding in temporal networks. Analyzing the complexities of existing algorithms and developing new ones adapted to fine temporal resolution is, therefore, a challenge for researchers of the field.

10.4 Handling Different Types of Temporal Networks

Temporal networks can be modeled in different ways. Among the most common framework, we can cite:

- **Snapshot sequences**, in which the dynamic is represented as an ordered series of graphs
- **Interval graphs** (or series of change) (Holme and Saramäki 2012), in which intervals of time are associated with edges, and sometimes nodes
- **Link streams** (Latapy et al. 2017), in which edges are associated with a finite set of transient interaction times.

Each DCD algorithm is designed to work on a particular type of network representation. For instance, *Identify & Match* approaches consists of first identifying communities in each snapshot, and then matching similar communities across snapshots. Such a method is therefore designed to work (only) with snapshot sequences. However, as it has been done in several articles, datasets can be transformed from one representation to the other, for instance by aggregating link streams into snapshots (e.g., Mucha et al. 2010), or into interval graphs (e.g., Cazabet et al. 2012); thus the *representation* of the dynamic graph does not necessarily limit our capacity to use a particular algorithm on a particular dataset.

We think however, that one aspect of the problem, related to representation, has not yet been considered in the literature. Methods working with snapshots and with interval graphs make the implicit assumptions that *the graph any point in time is well defined*, i.e., that each snapshot—or the graph defined by all nodes and edges present at any time *t*—is not null, has a well-defined community structure, and is somewhat similar to neighboring snapshots. Said differently, those methods expect *progressively evolving* graphs. To the best of our knowledge, this question has not been studied in the literature. A practitioner creating a snapshot sequence from a link stream using a too short sliding window (e.g., a window of one hour in a dataset of email exchanges) might obtain a well-formed dynamic graph on which an *Identify & Match* method can be applied, but the results would be inconsistent, as the community structure would not persist at such scales. The same dataset analyzed using longer sliding windows might provide insightful results. The problem is particularly pregnant for interval graphs, that can represent real situations of very different nature. For instance, an interval graph could represent relations (friend/follower relation in

social networks) as well as interactions (phone calls, face-to-face interactions, etc.).
It is clear that both networks should not be processed in the same way.

Challenges

A challenge in the field will be to define the conditions of applicability of different
methods better, and theoretical grounds to define when a network needs transforma-
tion to become suitable to be analyzed by a given method.

10.5 Evaluation of Dynamic Communities

We have seen in previous sections that several approaches and methods exist to
discover communities in temporal networks. In this section, we first discuss the eval-
uation of community quality. This process often requires the generation of dynamic
networks with community structures, the topic of the second part of the section.

10.5.1 Evaluation Methods and Scores

As already discussed, there is not a single, universal definition of what is a good
community and, consequently, no unique and universal way to evaluate their quality.
Nevertheless, for static communities, many functions have been proposed, to evaluate
them either (i) intrinsically (internal evaluation), by means of quality functions, (e.g.,
Modularity, Conductance, etc.) and (ii) Relatively to a reference partition (external
evaluation), using a similarity function (e.g., NMI, aNMI, etc.). Both approaches
have pros and cons that have been thoroughly discussed in the literature (Peel et al.
2017; Yang and Leskovec 2015). Few works have been done to extend those functions
to the dynamic case.

10.5.1.1 Internal Evaluation

In most works, static quality functions are optimized at each step, often adding a
trade-off of similarity with temporally adjacent partitions to improve community
smoothness (see Sect. 10.3.1). Some works are based on a longitudinal adaptation of
the modularity (Mucha et al. 2010; Aynaud and Guillaume 2010), but they require to
create a new graph with added inter-snapshot edges, and therefore cannot be used to
evaluate algorithms based on different principles. Works based on Stochastic Block
Model (Matias and Miele 2017; Yang et al. 2009) also optimize a custom longitudinal
quality function.

10.5.1.2 External Evaluation

Articles doing external evaluation requires to have a reference partition. Since few annotated datasets exist, a synthetic generator is used (see Sect. 10.5.2. The comparison often uses the average of a static measure (e.g., NMI) computed at each temporal step (Bazzi et al. 2016), eventually weighted to take into account the evolution of network properties (Rossetti 2017). A notable exception is found in Granell et al. (2015), where windowed versions of similarity functions (Jaccard, NMI, NVI) are introduced, by computing their contingency table on two successive snapshots at the same time.

Challenges

The evaluation of the quality of dynamic communities, both internally and externally, certainly represents a challenge for future works in dynamic community detection. Methods directly adapted from the static case do not consider the specificity of dynamic communities, in particular, the problems of smoothness and community events. This question is of utmost importance, since, despite the large variety of methods already proposed, their performances on real networks besides the ones they have been designed to work on is still mostly unknown.

10.5.2 Generating Dynamic Graphs with Communities

Complex network modeling studies gave birth to a new field of research: synthetic network generators. Generators allow scientists to evaluate their algorithms on synthetic data whose characteristics resemble the ones that can be observed in real-world networks. The main reason behind the adoption of network generators while analyzing the performance of a dynamic community detection (DCD) algorithm is the ability to produce benchmark datasets that enable (i) Controlled environment testing, e.g., in term on network size, dynamics, structural properties, etc., and (ii) comparison with a planted ground-truth.

Two families of network generators have been described to provide benchmarks for DCD algorithms: generators that produce static graphs-partitions and generators that describe dynamic graphs-partitions. Static graphs are used to evaluate the quality of the detection at a single time t, and cannot inform about the smoothness of communities. The most known are the GN benchmark (Girvan and Newman 2002), the LFR benchmark (Lancichinetti and Fortunato 2009) and planted partitions according to the stochastic block model.

Several methods have been proposed to generate dynamic networks with communities. The network can be composed of a sequence of snapshots, as in Bazzi et al. (2016), in which, at each step, the community structure (based on an SBM) drifts according to a user-defined inter-layer dependency. Another approach consists in

having an initial partition yielded by a static algorithm (LFR in Greene et al. 2010, GN in Lin et al. (2008)), and to make it evolves randomly (Greene et al. 2010) or until reaching an objective network with a different community structure (Lin et al. 2008).

Finally, another class of methods generates slowly evolving networks whose changes are driven by community events—merge, split, etc.—that can be tuned with parameters such as the probability of event occurrences. One of these methods is RDyn Rossetti (2017), whose communities are based on a similar principle than LFR. Another method has been proposed in Sengupta et al. Sengupta et al. (2017), which has the particularity of generating overlapping community structures.

Challenges

As we have seen, various methods already exist to generate dynamic graphs with slowly evolving communities. They have different properties, such as community events, stable edges, or overlapping communities. Active challenges are still open in this domain, among them (i) The generation of link streams with community structures, (ii) The empirical comparison of various DCD methods on those benchmarks, and (iii) An assessment on the realism of communities generated with such benchmarks, compared with how empirical dynamic communities behave.

10.6 Libraries and Standard Formats to Work with Dynamic Communities

In recent years, many tools and software have been developed to manipulate and process network data. Many of those tools have implemented community detection algorithms. Among the best known, we can cite networkx (Hagberg et al. 2008), iGraph (Csardi and Nepusz 2006) and snap (Leskovec and Sosič 2016), which propose a wide variety of network analysis tools, among them community detection algorithms, and related quality functions and scores. Some libraries are even designed specifically for community detection such as CDlib.[1] However, none of them can deal with dynamic networks. Very recently, a few libraries have been introduced to work with dynamic networks, such as tacoma[2] and pathpy (Scholtes 2017) but do not include community detection algorithms.

Furthermore, no standard format has yet emerged to represent dynamic communities and their evolution, which is particularly a problem to compare solutions yielded by different methods. This lack of common tools and standard representation certainly represents an obstacle, and a challenge to overcome for the DCD research community.

---

[1] https://github.com/GiulioRossetti/cdlib/tree/master/cdlib.

[2] https://github.com/benmaier/tacoma.

10.7 Conclusion

In this chapter, we have introduced the theoretical aspects of dynamic community detection and highlighted some of the most interesting challenges in the field. Among them, we think that a better formalism to represent the evolution of dynamic clusters and their events, in particular in the context of gradually evolving communities, would facilitate the comparison and the evaluation of communities and detection methods. The scalability of existing approaches is also a concern, again, in the context of *link streams* or other temporal networks studied at fine temporal scales. Finally, a recently introduced technique *graph embedding*, has attracted a lot of attention in various domains. Applications exist to temporal networks, although no work has focused on the dynamic community detection problem yet, to the best of our knowledge. Using this new technique to propose scalable methods could be another challenge worthy of investigation.

References

T. Aynaud, J.L. Guillaume, Static community detection algorithms for evolving networks, in *Proceedings of the 8th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt)* (IEEE, 2010), pp. 513–519

T. Aynaud, J.L. Guillaume, Multi-step community detection and hierarchical time segmentation in evolving networks, in *Proceedings of the 5th SNA-KDD Workshop* (2011)

M. Bazzi, L.G. Jeub, A. Arenas, S.D. Howison, M.A. Porter, Generative benchmark models for mesoscale structure in multilayer networks (2016). arXiv:1608.06196

V.D. Blondel, J.L. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks. J. Stat. Mech. Theory Exp. **2008**(10), 10,008 (2008)

S. Boudebza, R. Cazabet, F. Azouaou, O. Nouali, Olcpm: an online framework for detecting overlapping communities in dynamic social networks. Comput. Commun. **123**, 36–51 (2018)

R. Cazabet, F. Amblard, Dynamic community detection, in *Encyclopedia of Social Network Analysis and Mining* (Springer, 2014), pp. 404–414

R. Cazabet, F. Amblard, C. Hanachi, Detection of overlapping communities in dynamical social networks, in *2010 IEEE Second International Conference on Social Computing* (IEEE, 2010), pp. 309–314

R. Cazabet, H. Takeda, M. Hamasaki, F. Amblard, Using dynamic community detection to identify trends in user-generated content. Soc. Netw. Anal. Mining **2**(4), 361–371 (2012)

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* (IEEE, 2010), pp. 318–327

G. Csardi, T. Nepusz, The igraph software package for complex network research. Int. J. Complex Syst. **1695** (2006). http://igraph.org

T. Falkowski, J. Bartelheimer, M. Spiliopoulou, Mining and visualizing the evolution of subgroups in social networks, in *IEEE/WIC/ACM International Conference on Web Intelligence (WI)* (IEEE, 2006), pp. 52–58

F. Folino, C. Pizzuti, Multiobjective evolutionary community detection for dynamic networks, in *GECCO* (2010), pp. 535–536

A. Ghasemian, P. Zhang, A. Clauset, C. Moore, L. Peel, Detectability thresholds and optimal algorithms for community structure in dynamic networks. Phys. Rev. X **6**(3), 031,005 (2016)

M. Girvan, M.E. Newman, Community structure in social and biological networks. Proc. Natl. Acad. Sci. **99**(12), 7821–7826 (2002)

R. Görke, P. Maillard, C. Staudt, D. Wagner, Modularity-driven clustering of dynamic graphs, in *International Symposium on Experimental Algorithms* (Springer, 2010), pp. 436–448

C. Granell, R.K. Darst, A. Arenas, S. Fortunato, S. Gómez, Benchmark model to assess community structure in evolving networks. Phys. Rev. E **92**(1), 012,805 (2015)

D. Greene, D. Doyle, P. Cunningham, Tracking the evolution of communities in dynamic social networks, in *International Conference on Advances in Social Networks Analysis and Mining (ASONAM)* (IEEE, 2010), pp. 176–183

A. Hagberg, P. Swart, D.S. Chult, Exploring network structure, dynamics, and function using networkx. Technical Report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States) (2008)

P. Holme, J. Saramäki, Temporal networks. Phys. Rep. **519**(3), 97–125 (2012)

M.B. Jdidia, C. Robardet, E. Fleury, Communities detection and analysis of their dynamics in collaborative networks, in *2007 2nd International Conference on Digital Information Management*, vol. 2 (IEEE, 2007), pp. 744–749

A. Lancichinetti, S. Fortunato, Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. Phys. Rev. E **80**(1), 016,118 (2009)

M. Latapy, T. Viard, C. Magnien, Stream graphs and link streams for the modeling of interactions over time (2017). CoRR arXiv.org/abs/1710.04073

J. Leskovec, R. Sosič, Snap: a general-purpose network analysis and graph-mining library. ACM Trans. Intell. Syst. Technol. (TIST) **8**(1), 1 (2016)

Y.R. Lin, Y. Chi, S. Zhu, H. Sundaram, B.L. Tseng, Facetnet: a framework for analyzing communities and their evolutions in dynamic networks, in *Proceedings of the 17th International Conference on World Wide Web (WWW)* (ACM, 2008), pp. 685–694

C. Matias, V. Miele, Statistical clustering of temporal networks through a dynamic stochastic block model. J. R. Stat. Soc. Ser. B (Stat. Methodol.) **79**(4), 1119–1141 (2017)

C. Matias, T. Rebafka, F. Villers, Estimation and clustering in a semiparametric poisson process stochastic block model for longitudinal networks (2015)

D. Meunier, R. Lambiotte, E.T. Bullmore, Modular and hierarchically modular organization of brain networks. Front. Neurosci. **4**, 200 (2010)

P.J. Mucha, T. Richardson, K. Macon, M.A. Porter, J.P. Onnela, Community structure in time-dependent, multiscale, and multiplex networks. Science **328**(5980), 876–878 (2010)

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

M.E. Newman, M. Girvan, Finding and evaluating community structure in networks. Phys. Rev. E **69**(2), 026,113 (2004)

G. Palla, A.L. Barabási, T. Vicsek, Quantifying social group evolution. Nature **446**(7136), 664–667 (2007)

L. Peel, A. Clauset, Detecting change points in the large-scale structure of evolving networks (2014). CoRR arXiv.org/abs/1403.0989

L. Peel, D.B. Larremore, A. Clauset, The ground truth about metadata and community detection in networks. Sci. Adv. **3**(5), e1602,548 (2017)

T.P. Peixoto, Hierarchical block structures and high-resolution model selection in large networks. Phys. Rev. X **4**(1), 011,047 (2014)

G. Rossetti, Rdyn: graph benchmark handling community dynamics. J. Complex Netw. (2017). https://doi.org/10.1093/comnet/cnx016

G. Rossetti, R. Cazabet, Community discovery in dynamic networks: a survey. ACM Comput. Surveys (CSUR) **51**(2), 35 (2018)

G. Rossetti, L. Pappalardo, D. Pedreschi, F. Giannotti, Tiles: an online algorithm for community discovery in dynamic social networks. Mach. Learn. **106**(8), 1213–1241 (2017)

M. Rosvall, C.T. Bergstrom, Maps of random walks on complex networks reveal community structure. Proc. Natl. Acad. Sci. **105**(4), 1118–1123 (2008)

M. Rosvall, C.T. Bergstrom, Mapping change in large networks. PloS one **5**(1), e8694 (2010)

I. Scholtes, When is a network a network?: Multi-order graphical model selection in pathways and temporal networks, in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (ACM, 2017), pp. 1037–1046

N. Sengupta, M. Hamann, D. Wagner, Benchmark generator for dynamic overlapping communities in networks, in *2017 IEEE International Conference on Data Mining (ICDM)* (IEEE, 2017), pp. 415–424

J. Stehlé, N. Voirin, A. Barrat, C. Cattuto, L. Isella, J. Pinton, M. Quaggiotto, W. Van den Broeck, C. Régis, B. Lina, P. Vanhems, High-resolution measurements of face-to-face contact patterns in a primary school. PLOS ONE **6**(8), e23,176 (2011). https://doi.org/10.1371/journal.pone.0023176

M. Takaffoli, F. Sangi, J. Fagnan, O.R. Zaïane, Modec-modeling and detecting evolutions of communities, in *5th International Conference on Weblogs and Social Media (ICWSM)* (AAAI, 2011), pp. 30–41

T. Viard, M. Latapy, C. Magnien, Computing maximal cliques in link streams. Theoret. Comput. Sci. **609**, 245–252 (2016)

J. Yang, J. Leskovec, Defining and evaluating network communities based on ground-truth. Knowl. Inf. Syst. **42**(1), 181–213 (2015)

T. Yang, Y. Chi, S. Zhu, Y. Gong, R. Jin, A bayesian approach toward finding communities and their evolutions in dynamic social networks, in *Proceedings of the International Conference on Data Mining* (SIAM, 2009), pp. 990–1001