Community Discovery in Dynamic Networks: A Survey

GIULIO ROSSETTI, Institute of Information Science and Technologies (ISTI) - Italian National Research Council (CNR)
RÉMY CAZABET, Univ Lyon, Université Lyon 1, CNRS, LIRIS UMR5205, F-69622 France
& Sorbonne Universités, UPMC Univ Paris 06, CNRS, UMR 7606, LIP6, F-75005, Paris, France

Several research studies have shown that complex networks modeling real-world phenomena are characterized by striking properties: (i) they are organized according to community structure, and (ii) their structure evolves with time. Many researchers have worked on methods that can efficiently unveil substructures in complex networks, giving birth to the field of community discovery. A novel and fascinating problem started capturing researcher interest recently: the identification of evolving communities. Dynamic networks can be used to model the evolution of a system: nodes and edges are mutable, and their presence, or absence, deeply impacts the community structure that composes them.

This survey aims to present the distinctive features and challenges of dynamic community discovery and propose a classification of published approaches. As a "user manual," this work organizes state-of-the-art methodologies into a taxonomy, based on their rationale, and their specific instantiation. Given a definition of network dynamics, desired community characteristics, and analytical needs, this survey will support researchers to identify the set of approaches that best fit their needs. The proposed classification could also help researchers choose in which direction to orient their future research.

CCS Concepts: • **General and reference** → **Surveys and overviews**; • **Information systems** → **Clustering**; **Clustering and classification**; • **Mathematics of computing** → *Graph algorithms*; • **Networks** → *Network dynamics*; • **Theory of computation** → Dynamic graph algorithms;

Additional Key Words and Phrases: Dynamic networks, temporal networks, community discovery

ACM Reference format:
Giulio Rossetti and Rémy Cazabet. 2018. Community Discovery in Dynamic Networks: A Survey. *ACM Comput. Surv.* 51, 2, Article 35 (February 2018), 37 pages.
https://doi.org/10.1145/3172867

1 INTRODUCTION

Complex networks (Newman 2003) are popular tools—both theoretical and analytical—commonly used to describe and analyze interaction phenomena that occur in the real world. Social ties formation, economic transaction, face-to-face communication, and the unfolding of human mobility

This work was partially supported by grant O18062-44430 of the French program PIA "Usages, services et contenus innovants" and partially funded by the European Community's H2020 Program under the funding scheme "INFRAIA-1-2014-2015: Research Infrastructures" grant agreement 654024, *SoBigData: Social Mining and Big Data Ecosystem* (http://www.sobigdata.eu).

Authors' addresses: G. Rossetti, KDD Lab. - Istituto ISTI - Area della Ricerca CNR di Pisa - Via G. Moruzzi 1, 56124 Pisa, Italy; email: giulio.rossetti@isti.cnr.it; R. Cazabet, Batiment Blaise Pascal, 12 - 14 Rue de la Physique, Campus de la Doua, 69100 Villeurbanne, France; email: remy.cazabet@gmail.com.

https://doi.org/10.1145/3172867

are a few examples of observable events that are often investigated using instruments borrowed from graph theory.

One main issue to address while studying this type of real-world events lies in the identification of meaningful substructures hidden within the overall complex system. How to partition a complex network into *communities* is a widely studied problem (Fortunato 2010; Coscia et al. 2011): several works, from a broad set of disciplines, proposed different community definitions and approaches able to retrieve such structures automatically. The magnitude of works that cover this fascinating topic highlights its ubiquity: communities can capture groups of strongly connected online documents, individuals sharing a social environment as well as products being frequently purchased together.

Most of the scientific literature dedicated to the community discovery (CD) problem starts from a very stringent assumption: real-world phenomena can be modeled with static networks—that is, mathematical objects *frozen* in time. Unfortunately, such a simplified scenario seldom fits the evolving nature of the world we live in. Indeed, one of the key features that characterize interaction phenomena is that they naturally unfold through time: social interactions evolve, phone calls have a limited duration, and human mobility changes as time goes by. The temporal dimension conveys highly valuable information that the analyst needs to exploit to better understand the reality that he or she observes. The need to include such knowledge in the tools offered by graph theory has been one of the linchpins in an entirely new field of investigation that emerged in the past decade: dynamic network analysis. Within this new field, well-known problems defined in static settings were reformulated, and time-aware approaches able to solve them were proposed.

Among them, dynamic community discovery (DCD) represents one of the most challenging problems. As time goes by, nodes and edges can join and leave a dynamic network, producing relevant perturbations on its topology: communities are the basic bricks of that complex structure and as such are profoundly affected by local changes. Whereas classical CD algorithms aim to identify hidden substructures, dynamic approaches are devoted to the tracking of such local topologies and of their mutations, a goal that can be fulfilled pursuing different strategies and modeling choices.

In this survey, we have thus chosen to design a taxonomy of DCD algorithms not by considering the technical solution they use to find them but rather the strategy they use to identify time-aware meaningful network substructures. Our aim is to support researchers and analysts in choosing the approach, or at least the class of approaches, that better fits their needs and data. For each of the proposed classes of DCD algorithms, we discuss both advantages and drawbacks. Moreover, we provide a brief description of those methods that implement their rationale.

To provide a comprehensive view of this field of research, we also discuss issues related to the evaluation of DCD algorithms, which requires modifications compared to the static case; moreover, we summarize other works on DCD, such as dynamic community visualization methods, and analytical applications to real-world problems.

The survey is organized as follows. In Section 2, we discuss the general concepts behind dynamic network analysis and the mathematical models used to describe, in graph-theoretical terms, temporal evolving phenomena. In Section 3, we formalize the CD problem on dynamic networks and discuss community life cycle and instability. In Section 4, we introduce our DCD taxonomy describing the peculiarity, pros, and cons of each class of approaches. A detailed description of the methods is then made available in the appendix. In Section 5 we introduce and discuss two classes of techniques often used to compare and characterize communities identified by DCD algorithms, namely internal and external quality evaluation. In Section 6, we review some real-world scenarios that have been studied through the lenses of dynamic community mining and address the problem of visualizing analytical results. Section 7 concludes the survey, providing insights into DCD-related open issues.

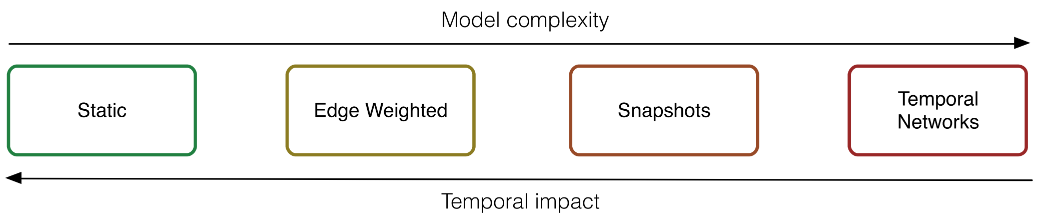

Fig. 1. Network representations. Moving from a *static* graph to *interaction* ones, the model complexity increases while the grain of the temporal information available decreases (Rossetti 2015).

2 DYNAMIC NETWORKS

Since its beginning, complex network analysis has been approached by researchers through the definition of very specific mining problems. Among them, CD (Fortunato 2010; Coscia et al. 2011), link-based object ranking (Getoor and Diehl 2005; Duhan et al. 2009), and frequent pattern mining (Milo et al. 2002; Alon 2007) are examples of analytical tasks originally defined on networks frozen in time.

The absence of the time dimension comes from historical reasons: the graph theory ancestry of the field and the scarcity of existing dynamic data at the time the field of complex networks emerged.

With the explosion of human-generated data, often collected by sociotechnical platforms, more and more datasets having rich temporal information that can be studied as dynamic networks becomes available. Many works tried to transpose well-known graph mining problems from static networks to temporal networks: temporal motifs mining (Kovanen et al. 2011), diffusion (Lee et al. 2012), link prediction (Tabourier et al. 2016), and so forth.

To better address the central topic of this survey, in Section 2.1 we introduce the most used formalisms to represent dynamic networks, and in Section 2.2 we discuss one of the leading issues arising when dealing with temporally annotated networks: network memory.

2.1 Network Models

Time plays a crucial role in shaping network topologies. One of the most important issue to address to reason on time-evolving networks is thus related to the mathematical formalism used to describe them. Figure 1 shows four different conceptual solutions that gradually introduce the temporal dimension in the network modeling process. In one respect, we find the complete *atemporal* scenario (i.e., a single network snapshot capturing a static glimpse of a dynamic phenomenon): network dynamics can be gradually introduced by adopting labels that weights nodes and edges, thus capturing an *aggregate* network. Following this approach, weights models the number of occurrence of a given node/edge in a predetermined observation window: with this little increment in the model complexity, several time-dependent analyses neglected before become possible (i.e., tie strength estimation). Aggregation strategies, however, suffer a severe limitation: they do not capture dynamics. For this reason, several works model dynamic phenomena with temporally ordered series of network *snapshots*. This simple modeling choice allows one to efficiently keep track of the perturbations occurring on the network topology. However, while the expressivity of the model is increased, the analytical complexity increases as well. When dealing with network snapshots to perform time-aware mining tasks, two issues need to be addressed: (i) how to keep track of multiple stages of the network life and (ii) how to harmonize the analytical results obtained in a snapshot with the outcome of subsequent ones.

Dynamic network partition, as well as aggregation, suffers from a pressing issue: to be performed, a temporal granularity—a threshold to transform the original dynamic system—needs to be fixed. Since the identification of such a threshold is not trivial—it is, indeed, context dependent and often profoundly impacts analytical results—recent works propose modeling dynamic phenomena without any aggregation, keeping all temporal details. Such studies usually decompose a dynamic network in its elementary bricks: temporally ordered, timestamped, interactions or relations. We will describe in more details such an approach, namely temporal networks modeling, in the next section.

Temporal networks, avoiding aggregation, allow for a complete and fine-grain description of network dynamics: as a drawback, this solution increases the complexity of the network model and requires the definition of novel analytical methodologies.

Different problems, as well as available data, impose different modeling choices: static networks, as well as weighted ones, are often used to identify stable patterns and describe the actual status of a network, whereas snapshots and interactions are proxies for the study of increasingly dynamic scenarios. Obviously, starting from fine-grain temporal data, it is possible to generate all of the other models by subsequent aggregations. Since we are interested in community detection approaches that deal with temporally annotated graphs, in the following we will highlight the major strengths and drawbacks of the most used models in this context: temporal networks and network snapshots. In this survey, we will not discuss static and weighted networks, as by definition, they do not allow one to make direct use of temporal information, thus reducing dynamic problems to static ones: for an in-depth overview of CD approaches defined for such scenarios, refer to Fortunato (2010), Fortunato and Hric (2016), Xie et al. (2013b), and Coscia et al. (2011).

2.1.1 Temporal Networks. Several formalisms have been proposed to represent evolving networks without loss of information: temporal networks (Holme and Saramäki 2012), time-varying graphs (Casteigts et al. 2012), interaction networks (Rossetti et al. 2015), and link streams and stream graphs (Viard et al. 2016; Latapy et al. 2017), to name the most popular. Hereafter, we use the term *temporal networks* with a general definition that encompasses all of those formalisms.

A temporal network models a dynamic structure in which both nodes and edges may appear and disappear as time goes by. More formally, we have the following definition.

Definition 2.1 (Temporal Network). A *temporal network* is a graph $G = (V, E, T)$, where V is a set of triplets of the form (v, t_s, t_e), with v a vertex of the graph and $t_s, t_e \in T$ respectively being the birth and death timestamps of the corresponding vertex (with $t_s \leq t_e$); E is a set of quadruplets (u, v, t_s, t_e), with $u, v \in V$ being vertices of the graph, and $t_s, t_e \in T$ respectively being the birth and death timestamps of the corresponding edge (with $t_s \leq t_e$).

Depending on the interaction semantics, we can deal with undirected temporal networks (TNs) or with directed temporal networks (DTNs).

The high-level definition proposed encompasses nodes and edges with or without duration (if $t_s = t_e$). Often, a strong distinction is made between those two categories. Networks without edges durations are often referred to as *contact sequences* and those with durations as *interval graphs* (Holme and Saramäki 2012). We argue that for community detection, this distinction is not relevant to characterize the nature of temporal networks. Indeed, a *contact sequence* can always be transformed into an *interval graph* by extending edges of an infinitesimal amount. Conversely, an interval graph can be discretized into a contact sequence at the desired resolution. However, such transformations will not fundamentally change the nature of the network.

Differently, a distinction must be made between two types of temporal network, *interaction networks* and *relation networks*: the former model interactions either with duration (phone calls, face-to-face communication, etc.) or not (emails, short messages, etc.) that can repeat as time goes by. Relation networks, conversely, model more stable relations (friendship, coworker, belonging to the same group, etc.). In relation networks, the state of the graph at any given time is well defined and can be studied through classical static analysis tools. Conversely, in interaction networks, the analyst needs to focus his attention on a temporal windows or on aggregations of some sort to obtain a stable dynamic graph that does not change dramatically between close time periods. This distinction is important, as many methods—particularly the ones working on edge streams—update the state of communities after each atomic network perturbation. Indeed, classical approaches do not produce interpretable results when edges are instantaneous or have very low persistence (e.g., phone calls).

Due to the lack of literature, and particularly a formal definition able to distinguish between these two types of temporal networks, in this article we have no choice but to ignore this distinction. We stress that most algorithms present in the current survey implicitly assume a relation network, save three methods (Matias and Miele 2016; Viard et al. 2016; Himmel et al. 2016), specifically designed to work with edges without duration.

2.1.2 Network Snapshots. Often, network history is partitioned into a series of snapshots, with each one of them corresponding either to the state of the network at a time t (relation network) or to the aggregation of observed interactions during a period (interaction network).

Definition 2.2 (Snapshot Network). A snapshot graph $\mathcal{G}_\tau$ is defined by an *ordered* set $\langle G_1, G_2 \ldots G_t \rangle$ where each snapshot $G_i = (V_i, E_i)$ is univocally identified by the sets of nodes V_i and edges E_i.

Depending on the network semantic, we can deal with undirected snapshot graphs (SNs) or directed snapshot graphs (DSNs). Dynamic networks represented using snapshots are similar in nature to multiplex (or multislice) networks, which are *unordered* sets of static networks. To learn more on community detection on multislice networks, readers can refer to Kivel et al. (2014). When the adaptation of methods to the dynamic case have been explored, methods will be included in the present article (e.g., see Mucha et al. (2010)).

Network snapshots can be effectively used, for instance, to model a phenomenon that generates network perturbations (almost) at regular intervals. In this scenario, context-dependent temporal windows are used to partition the network history into consecutive snapshots: time-bounded observations describing a precise, static, discretization of the network life.

The snapshot-based analysis is frequently adopted for networks with a natural temporal discretization, due to the balance that they propose between model complexity and expressivity. SNs allow application of static networks tools on evolving networks. Algorithms, as well as network measures and definitions, can be independently applied to each temporal partition without the need of novel analytical tools that explicitly manage temporal information. SNs can also be seen as a convenient modeling choice to avoid the instability problem of temporal networks, using aggregation over periods of time and sliding windows (Morini et al. 2017).

However, the strength of this model also represents its major drawback. Even if network snapshots reduce the analytical complexity and allow for task parallelization, they intrinsically imply the need for a reconciliation phase in which the independently computed, partial results are combined. The need for *two-step* (mining and reconciliation) procedures can often be the cause of the quality degradation of analytical results. Another issue related to the SN model is the identification of the optimal window size to use when generating the snapshots—a choice that can profoundly affect the outcome of the subsequent analysis.

2.1.3 Discussion: Temporal Networks Versus Snapshots. An important issue to address is the relation between snapshot networks and temporal networks, since, as we will see in Section 4, a generic DCD algorithm is usually tailored to exploit only one of such representations. The differences among TNs and SNs are tightly connected to the constraints that such models impose: the choice of employing TNs instead of SNs (or vice versa) can have significant implications on aspects related to the storage of data and the processing part of the designed analysis. In particular:

(1) If the original data already describe punctual states of the network evolution (i.e., a weekly/monthly/yearly complete crawl of a system), the snapshot model is obviously the natural one to adopt. However, if a more precise temporal information is available (i.e., an email, a friend list, or phone call dataset with timestamps of finite precision), both solutions can be considered.
(2) Since, unlike TNs, SNs are composed of aggregated data, the typical complexity of processes on an SN depends on the chosen level of aggregation and on the sizes of the produced snapshots (number of nodes/edges). For instance, methods for DCD on SNs typically need to run a community detection process on each snapshot. The cost of running a CD algorithm at $t + 1$ typically does not depend on the number of changes between SNs at t and $t + 1$. Processes running on TNs, however, typically depend mainly on the number of network modifications, not on the granularity or the total number of nodes.

2.2 Network Memory

A modeling choice that profoundly affects the analysis performed on dynamic networks is the system memory. When the analyzed data describe an interaction network (i.e., emails or phone calls) not composed of long-lasting relations, most methods require transforming it into a relation network—for instance, to assign to each edge a duration appropriate to consider it as relation network. Such transformation can yield snapshots (if synchronized) or temporal networks (if each edge duration computation is done independently).

Two main scenarios are often considered: (i) a *perfect* memory network (or accumulative growth scenario) and (ii) a *limited* memory network. In the former scenario, nodes and edges can only join the network: the dynamic network has a perfect memory, meaning that old edges/nodes cannot disappear (e.g., a citation graph). Conversely, in the latter, nodes/edges can vanish as time goes by (i.e., in a social network context, edge disappearance can be used to model the decay of interactions' social effect).

We call *time to live* (TTL) the artificially defined duration of an edge. Several strategies exist to fix this TTL:

— *Fixed-size static time window*: The TTL is equal for all network entities, and there is a finite set of possible periods whose start and end dates are defined in advance. This strategy produces snapshot networks.
— *Fixed-size sliding time window*: The TTL is equal for all network entities, but it starts independently for each one of them at the moment of their first appearance (i.e., an email graph in which each email is considered to have the same social persistence starting from the time it is made).
— *Dynamic-size time window*: The TTL is equal for all network entities at a given time but depends on the current network status (i.e., a system for which it is interesting to capture bursts of interactions with different frequency).
— *(Global/local) decay function*: The TTL of each node/edge is defined independently, usually as a function of the global/local network activity (i.e., fitting the interarrival time distribution of nodes or edges for the whole network, as well as for the individual node pair).

The assumptions made on the persistence of network entities (e.g., the strategy identified to fix their TTL) play a crucial role in the results provided by time-aware mining algorithms.

3 DYNAMIC COMMUNITY DISCOVERY

As discussed in the previous section, the temporal dimension profoundly impacts the analytical tasks that are commonly performed on complex networks. Among them, a problem that has always received high attention from the scientific community is CD. Networks built on real-world data are complex objects often composed of several substructures hidden in an apparent chaos: CD aims to decompose them in meaningful subtopologies that better describe local phenomena. Even though the problem itself is intuitively clear, its formulation is particularly ill posed: given a specific network, different partitions can exist, each of them capturing different but valuable information. In the past decades, we have witnessed the proliferation of an extensive literature on such subject, each work proposing its approach optimizing the partition for a different quality function (modularity, density, conductance, etc.), thus making the comparison between different partitions challenging. The widespread different community definitions lead to a renewed awareness: the perfect CD algorithm does not exist. Conversely, there exists a set of algorithms that perform better on a specific declination of the general problem.

In this section, we formally define the CD problem on dynamic graphs (Section 3.1) and describe how its solutions can be used to study the community life cycle (Section 3.1.1). Moreover, we discuss two relevant topics related to this complex problem: community instability and temporal smoothing (Section 3.2).

3.1 Problem Definition

CD is a typical problem in complex network analysis. One of the main reasons behind its complexity is undoubtedly related to its ill posedness. Indeed, several community definitions have been proposed so far. Classic works intuitively describe communities as sets of nodes closer among them than with the rest of the network, whereas others, looking at the same problem from another angle, only define such topologies as dense network subgraphs. To maintain a more general perspective, in this survey we will adopt a metadefinition—borrowed from Coscia et al. (2011)—to create an underlying concept able to generalize to all variants found in the literature (verbatim from Coscia et al. (2011)).

Definition 3.1 (Community). A community in a complex network is a set of entities that share some closely correlated sets of actions with the other entities of the community. We consider a direct connection as a particular and very important kind of action.

Given the high-level nature of the adopted problem definition, one could prefer the term *node clustering* over community detection. Due to the lack of consensus in the literature, we choose to keep the term *communities* that is specific to the field of networks mining and analysis.

Once we have a fixed definition of community, we can formally define the specific task of identifying and tracking mesoscale structures in a dynamic network scenario. Even in this case, we adopt a generic definition that does not make any assumption about the nature of communities and that correspond to any dynamic cluster of nodes.

Definition 3.2 (Dynamic Community Discovery). Given a dynamic network DG, a dynamic community DC is defined as a set of distinct (node, periods) pairs: $DC = \{(v_1, P_1), (v_2, P_2), \ldots, (v_n, P_n)\}$, with $P_n = ((t_{s0}, t_{e0}), (t_{s1}, t_{e1}) \ldots (t_{sN}, t_{eN}))$, with $t_{s*} \leq t_{e*}$. DCD aims to identify the set C of all dynamic communities in DG. The partitions described by C can be neat as well as overlapping.

Within this general definition fall approaches with different goals. In particular, two main analytical aims can be identified: (i) enable for an efficient identification of optimal partitions for each instant of the evolution and (ii) build evolutive chains describing the life cycle of communities.

3.1.1 Community Life Cycle. The persistence along time of communities subjected to progressive changes is an important problem to tackle. As illustrated by the famous paradox of the ship of Theseus, deciding if an element composed of several entities at a given instant is the same or not as another one composed of some—or even none—of such entities at a later point in time is necessarily arbitrary and cannot be answered unambiguously.

Most works focused on community tracking agree on the set of simple actions that involve entities of a dynamic network: node/edge appearance and vanishing. Indeed, such local and atomic operations can generate perturbations of the network topology able to affect the results produced by CD algorithms. As a consequence of this set of actions, given a community C observed at different moments in time, it is mandatory to characterize the transformations it undergoes. A first, formal categorization of the transformations that involve communities were 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 operations, illustrated in Figure 2, 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 rejected by a community, thus reducing its size.
— *Merge*: Two communities or more merge into a single one.
— *Split*: A community, as consequence of node/edge vanishing, splits into two or more components.
— *Continue*: A community remains unchanged.
— *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 time period (e.g., seasonal behaviors).

Not all operations are necessarily handled by a generic DCD algorithm. Even though the abstract semantics of such transformations are clear, different works propose to handle them differently. Among them, merge, split, and resurgence are often defined with the support of ad hoc similarity and thresholding functions, as well as community transformation strategies. For instance, when two communities are chosen to be merged, several strategies can be followed to handle the merge action (Cazabet and Amblard 2014):

— *Absorption*: After the merge, one community disappears while the other persists. In this scenario, defining a policy to decide which community will cease to exist is an open issue. Among possible choices, we can cite the following: (i) remove the oldest/youngest one, (ii) remove the biggest/smallest one, and (iii) remove the community with the highest percentage of nodes included in the other.
— *Replacement*: After the merge action, affected communities vanish and a wholly new one forms. This solution, although straightforward, causes sudden disruptions on the continuity of the evolution of communities.

Such a complex scenario symmetrically applies to the specific implementation of all remaining community actions.

Despite the peculiarities introduced by each approach when defining topology transformations, the identified events allow to describe for each community its so-called life cycle (an example is shown in Figure 3).

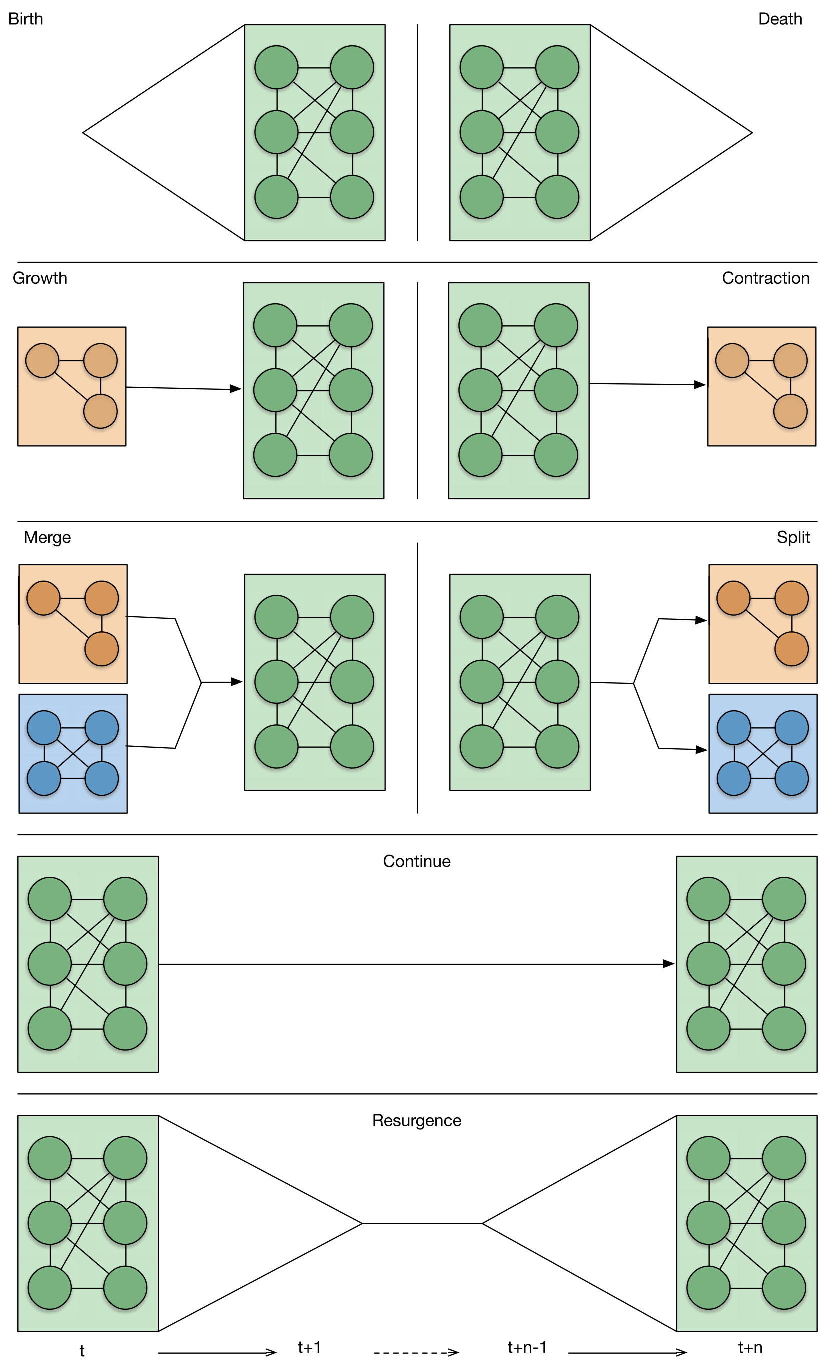

Fig. 2. Community events. This toy example captures the eight events that regulates dynamic community life. The first row shows birth and death, the second row shows growth and contraction, the third row shows merge and split, the fourth row shows continue, and the last row shows resurgence.

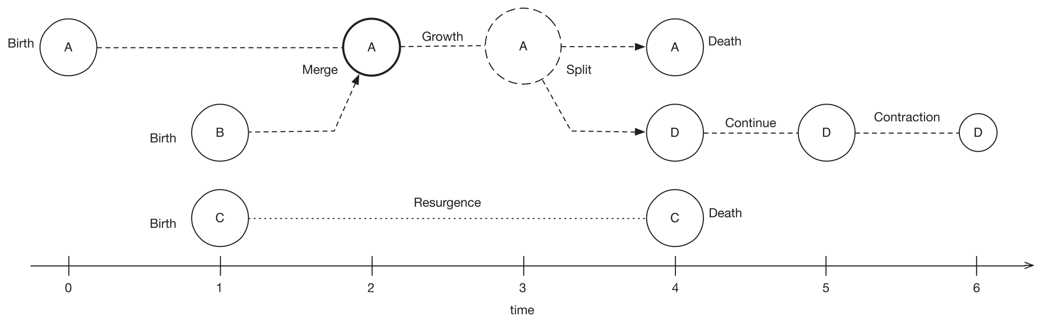

Fig. 3. Community life cycle. As time goes by, dynamic communities experience mutations that are often abstracted by adopting a predefined set of operations. This toy example highlights life cycles of communities A and C, composed of all operations that modify a community structure, namely birth, continue, growth, split, merge, contraction, resurgence, death.

Definition 3.3 (Community Life Cycle). Given a community C, its community life cycle (which univocally identifies C's complete evolutive 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.

In principle, it is possible to perform a life cycle analysis starting from the output of any generic DCD approach.

3.2 Community Instability and Temporal Smoothing

One of the main issues encountered by dynamic community detection approaches is the instability of solutions. This type of problem comes from the very nature of communities.

It is widely accepted that there is not a single valid decomposition in communities of a complex network issued from field data but, instead, several possible ones. Moreover, most algorithms generate partitionings in which a node is assigned unambiguously to one and only one community. Indeed, we are aware that such a scenario represents a simplification: nodes often belong to several communities, and the belonging to each of them is not necessarily binary (as illustrated by fuzzy partitions approaches). Another issue is that the choice between one partition and another is somewhat arbitrary: moreover, 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. To take a concrete example, if the problem of finding the solution of optimal modularity in a network is so complex, it is because many local maxima exist, whose modularity values are very close to the optimal one. Greedy approaches such as the Louvain method (Blondel et al. 2008) can find a local maximum; however, a slight variation of the initial network might switch from one local maximum to another, yielding significantly different partitions, both comparable in terms of the modularity score.

All such observations naturally lead to the general problem of identifying stable network decompositions. This is a major problem for dynamic community detection, as one cannot know if the differences observed between communities found on the network at time t and those found at $t + 1$ are due to the evolution of communities or to the algorithm's instability.

This problem was explored in Aynaud and Guillaume (2010). The authors picked three well-known static community detection algorithms (WalkTrap (Pons and Latapy 2006), Louvain (Blondel et al. 2008), and fast greedy modularity (Clauset et al. 2004)), and evaluated the consistency of their results on slight modifications of a network. To do so, they compare the modularity (Newman 2004; Newman and Girvan 2004) and normalized mutual information (NMI (Lancichinetti et al. 2008)) scores between successive results and quantify the number of transformations between two consecutive runs (where a *transformation* is the change in affiliation of a node). The results revealed the extent of the problem: on a network composed of 9,377 nodes and 24,107 edges, constructed from a coauthorship network, the removal of a single node leads to NMIs between successive results of—respectively to each algorithm—0.99, 0.9, and 0.75. The result is even more striking when looking at the number of unitary change: a single node modification respectively leads to approximately 500, 2,000, and 3,000 unitary changes in communities.

A variety of solutions has been proposed to solve, or at least mitigate, this instability problem. Their common goal is to smooth out the evolution of communities. Among them, we can identify the following main categories: smoothing by bootstrap, explicit smoothing, implicit smoothing, and global smoothing.

Smoothing by bootstrap. This technique (used in particular in Hopcroft et al. (2004) and Rosvall and Bergstrom (2010)) consists of running multiple times the same algorithm on the same static network, searching for invariants, stable parts of the communities. The stable community cores, which are less unstable than raw community partitions, can be more efficiently tracked.

Explicit smoothing. Some algorithms (e.g., Lin et al. (2009) and Folino and Pizzuti (2014)) explicitly introduce smoothing in their definition, requiring the partition found at step t to have a certain degree of similarity with the partition found at $t-1$. These methods typically introduce a parameter $\alpha \in [0, 1]$, which determines the trade-off between a solution that is optimal at t and a solution maximizing the similarity with the result at $t-1$.

Implicit smoothing. Some methods do not explicitly integrate the similarity between consecutive partitions but favor this similarity by construction. We can distinguish between two main scenarios to do so:

(1) Methods that optimize a global metric, such as the modularity, can use the communities found at the previous step as seeds for the current one (e.g., Shang et al. (2014) and Görke et al. (2010)). The probability of finding similar solutions from step to step is increased by choosing an adequate algorithm that will explore in priority good solutions around seeds.
(2) Other approaches (e.g., Cazabet et al. (2010), Xie et al. (2013a), and Rossetti et al. (2017)) do not try to optimize a global metric but locally update communities that have been directly affected by modifications between the previous and the current step. In this case, the solution at step $t-1$ is kept unchanged but for communities directly affected by changes in the network. In a slowly evolving graph, this guarantees that most of the partition stays identical between two evolution steps.

Global smoothing. Instead of smoothing communities at a given step by considering the previous one, algorithms (e.g., Aynaud and Guillaume (2011) and Mucha et al. (2010)) can search for communities that are coherent in time by examining all steps of evolution simultaneously. Such a strategy can be applied, for instance, by creating a single network from different snapshots by adding links between nodes belonging to different snapshots and then running a community detection on this network. Another way to pursue in this rationale is to optimize a global metric on all snapshots simultaneously. In both cases, the problem of instability naturally disappears, as the detection of dynamic communities is performed in a single step.

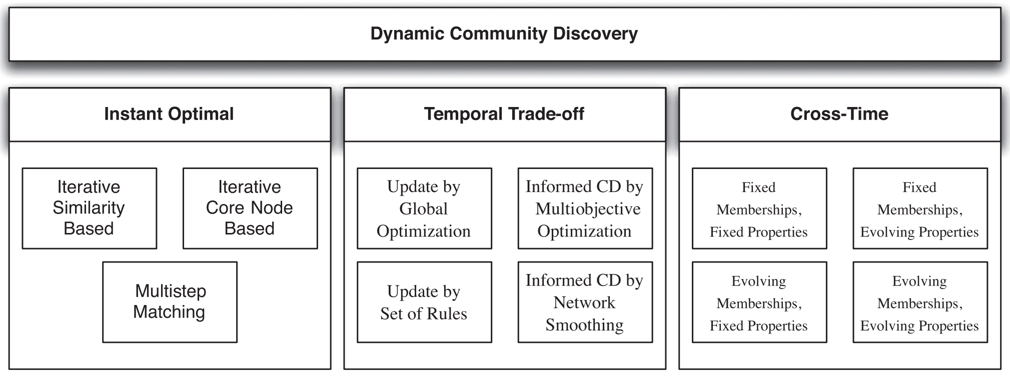
Dynamic Community Discovery
Instant Optimal
Iterative Similarity Based
Iterative Core Node Based
Multistep Matching
Temporal Trade-off
Update by Global Optimization
Informed CD by Multiobjective Optimization
Update by Set of Rules
Informed CD by Network Smoothing
Cross-Time
Fixed Memberships, Fixed Properties
Fixed Memberships, Evolving Properties
Evolving Memberships, Fixed Properties
Evolving Memberships, Evolving Properties

Fig. 4. Proposed classification for DCD algorithms. There are three high-level classes corresponding to different dynamic community definitions. Each of these classes is divided into subcategories corresponding to different manners to solve a similar problem.

When designing a DCD algorithm, the decision to adopt a specific smoothing strategy has nonnegligible impacts: indeed, it can lead to computational constraints, narrows the applicative scenarios, and impacts results that it will be able to provide. For all of those reasons, in Section 4 we describe a novel two-level taxonomy that exploits a characteristic of smoothing to discriminate among different classes of DCD approaches.

4 CLASSIFICATION

In this section, we propose a classification, summarized in Figure 4, of DCD methods.

Three previous works, to the best of our knowledge, have proposed a classification of DCD algorithms, although considering much fewer methods than the current survey. In Aynaud et al. (2013), the following classes are identified:

— *Two-stage approaches*: Methods that first detect clusters at each timestep and then match them across different timesteps
— *Evolutionary clustering*: Methods that detect communities at time t based on the topology of the graph at t and on previously found community structures
— *Coupling graphs*: Algorithms that first construct a single graph by adding edges between instances of nodes in different timesteps (coupling graph) and then run a classic CD on this graph.

Two-stage approaches are then further divided into *core-based*, *union graph–based*, and *survival graph–based* methods, corresponding to different solutions to solve the problem of graph matching.

In Hartmann et al. (2016), two high-level categories are identified: *online* and *offline* methods. The survey focuses only on online algorithms, defined as methods that explicitly exploit information about the graph structure and the community structure of previous timesteps, as opposed to offline ones that also use information from following timesteps. Online methods are divided into the following:

— *Temporal smoothness* methods that run a CD process from scratch at each step
— *Dynamic update* methods that do not run a process from scratch but instead update the methods found in previous timesteps.

These first two classifications are complementary and somewhat overlapping: coupling graph methods are offline, but some two-stage approaches can also be offline, as they match communities across all timesteps. Two-stage approaches are a special case of temporal smoothness without memory, whereas evolutionary clustering methods encompass both temporal smoothness and dynamic update methods.

Finally, in Masuda and Lambiotte (2016), the authors identify two different typologies of CD approaches in temporal networks:

- Approaches focused on determining *community evolution* that, with certain statistical accuracy, aim to detect when and how reorganizations of communities take place. In this family fall both the two-stage and model-based approaches.
- *Single community structure* methods whose goal is to find a single partition of the dynamic graph by considering time as a third dimension of the data, added to the classical two expressed by the adjacency matrix. Among them fall the *tensor factorization* approaches like the one introduced in Gauvin et al. (2014), where a single partition is identified and the strength of each community per temporal snapshot is computed.

Some methods we found in the literature did not belong to any of these categories or could be associated with several of them.

We decided to shape our classification as a two-level one, building it upon concepts shared by existing classifications.

In our taxonomy, the higher level identifies three different definitions of what are dynamic communities, without assumptions on the techniques used to find them. These high-level classes are then refined into subcategories, which correspond to different techniques used to find communities corresponding to this definition.

There are three classes of approaches at the higher level of classification:

- The first one (*Instant Optimal CD*) considers that communities existing at 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 with respect to the topology of the network at t. Approaches falling in this class are nontemporally smoothed.
- In the second class (*Temporal Trade-off CD*), communities defined at an instant t depend not only on the topology of the network at that time but also on the past evolutions of the topology, past partitions found, or both. Communities at t are therefore defined as a trade-off between an optimal solution at t and known past. They do not depend on future modification, an important point for on-the-fly CD. Conversely, from the approaches falling in the previous class, Temporal Trade-off ones are incrementally temporally smoothed.
- In the third class (*Cross-Time CD*), the focus shifts from searching communities relevant at a particular time to searching communities relevant when considering the whole network evolution. Methods of this class search a single partition directly for all timesteps. Communities found at t depend on both past and future evolutions. Methods in this class produce communities that are completely temporally smoothed.

For each of these classes, we define subcategories corresponding to different techniques used to solve a related problem. In the following, we address each one of them, discussing their advantages and drawbacks. As we will highlight in the following, each class—due to its definition—is likely to group together approaches designed to work on either network snapshots (SNs) or temporal networks (TNs).

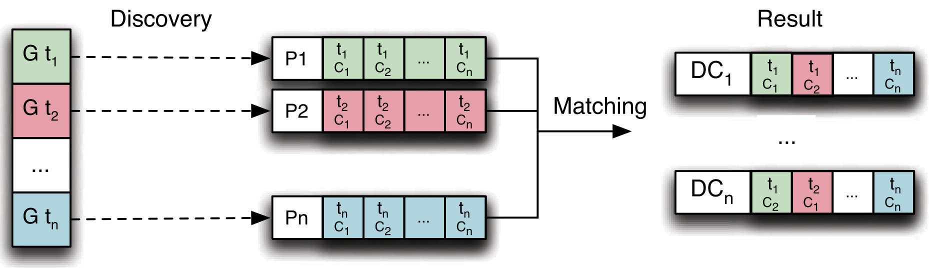

Fig. 5. Instant Optimal. The identification of dynamic communities is conceptually decomposed in two subproblems: (i) independent extraction of the network partition for each timestep and (ii) matching of communities belonging to distinct network observations. G, graph; P, partition; DC, dynamic community; C, community.

Please note that to keep our classification concise, we do not include the description of each method here. We provide a short description of methods included in each category in the appendix.

4.1 Instant Optimal Communities Discovery

This first class of approaches is derived directly from the application of static CD methods to the dynamic case. A succession of steps is used to model network evolution and for each of them is identified an optimal partition, as shown in Figure 5. Dynamic communities are defined from these optimal partitions by specifying relations that connect topologies found in different, possibly consecutive, instants.

A typical Instant Optimal technique has been identified as the two-stage approach in the literature (Aynaud et al. 2013). It can be described as a two-step process:

(1) *Identify*: Detect static communities on each step of evolution.
(2) *Match*: Align the communities found at step t with the ones found at step $t-1$, for each step.

It can be noted, however, that two-stage approaches and Instant Optimal communities do not encompass the same methods: indeed, two-stage techniques can be used with communities smoothed out using previous communities, and therefore not Instant Optimal. Yet some methods find Instant Optimal communities but match these communities considering all partitions simultaneously, instead of using an iterative process. In the following, we discuss the advantages and drawbacks of this class of algorithms and the dynamic network models to which it can be applied.

Advantages. The major benefit of adopting this approach is that it is built directly on top of existing works on static community detection. Since communities are Instant Optimal, usual DC algorithms can be used at each step. The matching process can also be derived from existing literature, as set matching is an extensively studied problem. Another advantage of this method is that it easily enables parallelization. Community detection can be a time-consuming process on large networks, and since the communities at each step are found independently, it is possible to run the detection at each step in parallel.

Drawbacks. The main limit of this approach is the instability of community detection algorithms (see Section 3.2 for more details on instability). Since the same algorithm can find different communities on the same network, it is hazardous to distinguish between changes due to the evolution of the community structure and changes due to the instability of algorithms. Recent approaches propose solutions to circumvent this weakness by studying the most stable part of communities, called *community cores* (Rosvall and Bergstrom 2010) or by considering all partitions simultaneously instead of matching only with earlier ones (Goldberg et al. 2011).

Network models. Due to its definition, algorithms that fall in this class can be used only with network snapshots, not with temporal networks. As the community detection needs to be performed entirely from scratch at each evolution step, the whole network needs to be considered, not only the changes between one step and another.

Subcategories. The first step of the process, identifying communities relevant at each timestep, can usually be done with any CD algorithm. However, the methods used to match communities found at different steps differ. We identified three subclasses: (i) *iterative similarity-based*, (ii) *iterative core-based*, and (iii) *multistep* matching.

4.1.1 Iterative Similarity-Based Approaches. In similarity-based approaches, a quality function is used to quantify the similarity between communities in different snapshots (e.g., Jaccard among node sets). Communities in adjacent time-steps having the highest similarity are considered part of the same dynamic community. Particular cases can be handled, such as a lower threshold of similarity, or the handling of 1 to n or n to n matchings, thus defining complex operations such as split and merge.

Methods in this subcategory: (Hopcroft et al. 2004; Bourqui et al. 2009; Palla et al. 2007; Greene et al. 2010; Rosvall and Bergstrom 2010; Takaffoli et al. 2011; Bóta et al. 2011; Bródka et al. 2013; Dhouioui and Akaichi 2014; İlhan and Öğüdücü 2015).

Refer to Appendix A.1 for a methods description.

4.1.2 Iterative Core Nodes–Based Approaches. Methods in this subcategory identify one or several special node(s) for each community—for instance those with the highest value of a centrality measure—called *core nodes*. Two communities in adjacent time-steps containing the same core node can subsequently be identified as part of the same dynamic community. It is thus possible to identify splits or merge operations by using several core nodes for a single community.

Methods in this subcategory: Wang et al. (2008) and Chen et al. (2010).

Refer to Appendix A.2 for methods description.

4.1.3 Multistep Matching. Methods in this subcategory do not match communities only between adjacent timesteps. Conversely, they try to match them even in snapshots potentially far apart in the network evolution. The matching itself can be done, as in other subcategories, using a similarity measure or core nodes. The removal of the adjacency constraint allows to identify resurgence operations—that is, a community existing at t and $t + n$ but not between $t + 1$ and $t + (n - 1)$. It can, however, increase the complexity of the matching algorithm, and on-the-fly detection becomes inconsistent, as past affiliations can change.

Methods in this subcategory: Falkowski et al. (2006), Falkowski and Spiliopoulou (2007), Goldberg et al. (2011), and Morini et al. (2017).

Refer to Appendix A.3 for a methods description.

4.2 Temporal Trade-Off Communities Discovery

Algorithms belonging to the Temporal Trade-off class process the evolution of the network iteratively. Moreover, unlike Instant Optimal approaches, they take into account the network and the communities found in the previous step—or n-previous steps—to identify communities in the current one. DCD algorithms falling in this category can be described by an iterative process:

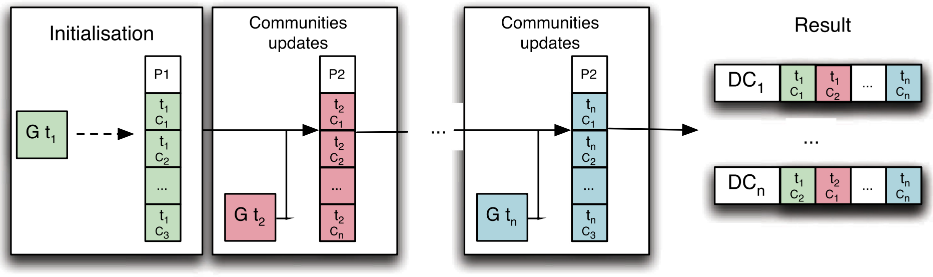

Fig. 6. Temporal Trade-off. After a *bootstrap* phase, communities are found iteratively at each step by exploiting information gathered from previous networks statuses and partitions. G, graph; P, partition; DC, dynamic community; C, community.

(1) *Initialization*: Ffind communities for the initial state of the network.
(2) *Update*: For each incoming step, find communities at step t using graph at t and past information.

As for the Instant Optimal class, this template, exemplified in Figure 6, can be implemented by several typologies of approaches. In the following, we discuss its advantages and drawbacks, and the dynamic network models to which it can be applied.

Advantages. Temporal Trade-off approaches allow the ability to cope with the instability problem that affects Instant-optimal ones (see Section 3.2) while not diverging much from the usual community detection definition (a partition is searched for at each step). Most methods of this class take advantage of the existing community structure at step $t-1$ to speed up the community detection at t.

Drawbacks. In the general case, it is not possible to easily parallelize the community detection, as each step needs the communities found at the previous ones as input. Another potential issue is the long-term coherence of dynamic communities. Since at each iteration the identified partition depends on previous ones, Temporal Trade-off approaches are subject to the risk of an avalanche effect: communities can experience substantial drifts compared to what a static algorithm would find on the static network corresponding to the state of the system at a given time.

Network models. This algorithmic schema has been used with both dynamic networks representations introduced in Section 2.1. When using temporal networks, only the incremental modifications of the network are considered to make the communities evolve, usually using local rules. Conversely, when using network snapshots, a process is usually run on the whole graph, as for a static algorithm, but taking into account past information.

Subcategories. The Temporal Trade-off template can be satisfied by very different algorithmic approaches. Among them, we identified four subcategories.

The first two, namely *update by global optimization* and *update by a set of rules*, consist in updating the partition found at the previous step, using global or local approaches.

The last two, namely *informed CD by multiobjective optimization* and *informed CD network smoothing*, run a community detection from scratch for each snapshot while considering information from previous steps.

4.2.1 Update by Global Optimization. A popular method for static CD is to optimize a global quality function, such as modularity or conductance.

This process is typically done using a heuristic (gradient descent, simulated annealing, etc.) that iteratively merges and/or splits communities, and/or moves nodes from a community to another until a local or global maximum is reached.

Methods in this subcategory use the partition existing at t as a seed to initialize a global optimization process at $t + 1$. The heuristic used can be the same as in the static case or a different one. A typical example of a method in this category (Aynaud and Guillaume 2010) consists of using the Louvain algorithm: in its original definition, at initialization, each node is in its community. The *partition update* version of it consists of initializing Louvain with communities found in the previous step.

Methods in this subcategory: Miller and Eliassi-Rad (2009), Görke et al. (2010), Aynaud and Guillaume (2010), Bansal et al. (2011), Shang et al. (2014), and Alvari et al. (2014).

Refer to Appendix B.1 for a methods description.

4.2.2 Update by a Set of Rules. Methods in this subcategory consider the list of network changes (edge/node apparitions, vanishing) that occurred between the previous step and the current one, and define a list of rules that determine how networks changes lead to communities update. Methods that follow such a rationale can vary significantly from one to the other.

Methods in this subcategory: Falkowski et al. (2008), Nguyen et al. (2011a), Cazabet et al. (2010), Nguyen et al. (2011b), Cazabet and Amblard (2011), Agarwal et al. (2012), Duan et al. (2012), Görke et al. (2012), Ma and Huang (2013), Xie et al. (2013a), Lee et al. (2014), Zakrzewska and Bader (2015), and Rossetti et al. (2017).

Refer to Appendix B.2 for a methods description.

4.2.3 Informed CD by Multiobjective Optimization. When addressing community detection on snapshot graphs, two different aspects need to be evaluated for each snapshot: partition quality and temporal partition coherence. The multiobjective optimization subclass groups together algorithms that try to balance both of them at the same time so that a partition identified at time t represents the natural evolution of the one identified at time $t - 1$. This approach optimizes a quality function of the following form:

$$c = \alpha CS + (1 - \alpha)CT, \tag{1}$$

where CS is the cost associated with the current snapshot (i.e., how well the community structure fits the graph at time t) and CT is the smoothness with respect to the past history (i.e., how different the actual community structure is with respect to the one at time $t - 1$) and $\alpha \in [0, 1]$ is a correction factor.

An instantiation of this type of optimization schema could be defined using *modularity* (Newman and Girvan 2004) as CS and NMI (Lancichinetti et al. 2008) as CT as done in Folino and Pizzuti (2010).

Methods in this subcategory: Zhou et al. (2007), Tang et al. (2008), Lin et al. (2008, 2009), Yang et al. (2009), Folino and Pizzuti (2010), Sun et al. (2010), Gong et al. (2012), Kawadia and Sreenivasan (2012), Crane and Dempsey (2015), and Görke et al. (2013).

Refer to Appendix B.3 for a methods description.

4.2.4 Informed CD by Network Smoothing. Methods in this subcategory search for communities at t by running a CD algorithm, not on the graph as it is at t but on a version of it that is smoothed according to the past evolution of the network, such as by adding weights to keep track of edges' age. In a later step, communities are usually matched between snapshots, as in typical two-stage approaches. Contrary to previous subcategories, it is not the previous communities that are used to take the past into account but the previous state of the network.

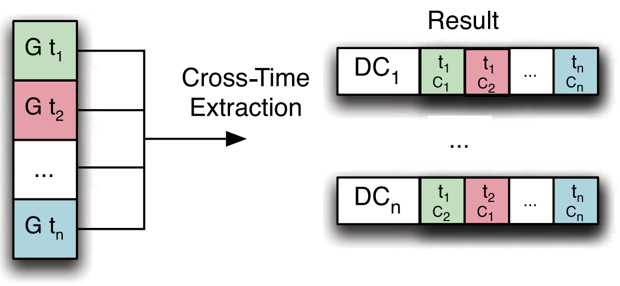

Fig. 7. Cross-Time. Communities are extracted by leveraging, all at once, the information gathered by the complete temporally annotated history of the analyzed network. G, graph; P, partition; DC, dynamic community; C, community.

Methods in this subcategory: Kim and Han (2009), Guo et al. (2014), and Xu et al. (2013a). Refer to Appendix B.4 for methods description.

4.3 Cross-Time Communities Discovery

Approaches belonging to the Cross-Time class do not consider independently the different steps of the network evolution. On the contrary, DCD is done in a single process, considering—at the same time—all states of the network. A visual representation of this class of approaches is shown in Figure 7.

An example of algorithmic procedures that fall in this class is the following:

(1) *Network transformation*: From a sequence of snapshots, a single network is created in which each metanode corresponds to the presence of a node at a given snapshot. Two kinds of edges are defined on this network: (i) the usual relationship between nodes at the same timestep and (ii) additional edges that link nodes belonging to different, adjacent timesteps.
(2) *Community detection*: A static community detection algorithm is run on this transversal network: identified communities contain nodes belonging to different timesteps, interpretable as dynamic communities.

Advantages. This class of algorithms does not suffer from problems of instability and community drift that affect previous ones.

In theory, these methods have more potential to deal with local anomalies and slow evolution running through multiple steps, which iterative processes might not be able to perceive, due to their narrower focus.

Drawbacks. Cross-Time approaches, unlike previous ones, are not based on the usual principle of a unique partition being associated with each step of the graph evolution process. It therefore requires developing novel techniques and making new assumptions on the nature of dynamic communities. As a result, most methods suffer from constraints such as a fixed number of communities and absence of operations such as merge or split.

Another drawback is that such approaches are not able to handle on-the-fly/real-time community detection. Indeed, since their computation needs a complete knowledge of all network history, in the presence of a new evolutive step the partition extraction needs to be performed from scratch.

Network models. Both network snapshots and temporal networks can be used by Cross-Time approaches.

Subcategories. All algorithms in the Cross-Time class do not have the same constraints on the nature of searched communities. We can classify them into four categories: (i) *fixed memberships, fixed properties*; (ii) *fixed memberships, evolving properties*; (iii) *evolving memberships, fixed properties*; and (iiii) *evolving memberships, evolving properties.*

4.3.1 Fixed Memberships, Fixed Properties. Methods in this subcategory do not allow nodes to switch communities, nor communities to appear or disappear. They also assume that communities stay the same throughout the studied period. As a consequence, they search for the best partition, on average, over a period of time. In most cases, they improve the solution by slicing the evolution in several time periods, each of them considered homogeneous, and separated by dramatic changes in the structure of the network, a problem related to change point detection (Peel and Clauset 2015).

Methods in this subcategory: Sun et al. (2007), Duan et al. (2009), and Aynaud and Guillaume (2011).

Refer to Appendix C.1 for a methods description.

4.3.2 Fixed Memberships, Evolving Properties. Methods in this subcategory are aware that communities are not homogeneous along time. For instance, nodes in a community can interact more actively during some recurrent time periods (hours, days, weeks, etc.), or their activity might increase or decrease along time. To each community—whose membership cannot change—they assign a temporal profile corresponding to the evolution of its activity.

Methods in this subcategory: Gauvin et al. (2014) and Matias and Miele (2016).

Refer to Appendix C.2 for a methods description.

4.3.3 Evolving Memberships, Fixed Properties. Methods in this subcategory allow node memberships to change along time—for instance, nodes can switch between communities. However, because they use stochastic block model (SBM) approaches, for which the coevolution of memberships and properties of communities is not possible (see Matias et al. (2015)), the number of communities and their density is fixed for the whole period of analysis.

Methods in this subcategory: Yang et al. (2009, 2011), Ishiguro et al. (2010), Herlau et al. (2013), Xu and Hero (2014), Matias et al. (2015), and Ghasemian et al. (2016).

Refer to Appendix C.3 for a methods description.

4.3.4 Evolving Memberships, Evolving Properties. Methods in this category do not impose constraints on how dynamic communities can evolve: nodes can switch between them, communities can appear or disappear, and their density can change. Two main approaches are currently present in this category:

— In Jdidia et al. (2007) and Mucha et al. (2010), edges are added between nodes in different snapshots. Static community detection algorithms are consequently run on this transtemporal network
— In Viard et al. (2016) and Himmel et al. (2016), the authors search for persistent cliques of a minimal size in link streams.

Methods in this subcategory: Jdidia et al. (2007), Mucha et al. (2010), Viard et al. (2016), and Himmel et al. (2016).

Refer to Appendix C.4 for a methods description.

4.4 Discussion

As we have seen in this section, all three classes of approaches have advantages and drawbacks; none is superior to the other. Nevertheless, we have observed how each one of them is more suitable for some 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, approaches that fall in the first two classes represent the most suitable fit. If the analytical context requires working with a fine temporal granularity, therefore modeling the observed phenomena with temporal networks, it is strongly suggested to avoid methods of the first class, which can only deal with snapshots.

The first layer of our taxonomy can thus be used to provide guidance and recommendations on which approach (or class of approaches) to select given a specific problem formulation. For instance, we can observe the following:

—How 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
—How 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
—How 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.

The second layer of classification groups together approaches that share the same rationale. This further classification is useful, in our opinion, to those researchers who need to frame their method within a specific literature. In identifying the particular family to which a DCD approach belongs, it is valuable to understand which are its real competitors and, doing so, to better organize comparative analysis, reducing the bias introduced by the slightly different problems addressed (as well as community definitions adopted) by alternative methods.

5 EVALUATION

So far, we have introduced our taxonomy and framed the DCD algorithms we identified within it. In this section, we will address a very sensitive theme related to any data mining task: evaluation. Indeed, finding a reliable way to evaluate partition quality is a significant issue to address while approaching CD. One of the main issues of the CD lies in the absence of a unique definition of *community*: each scholar follows its strategy to compare the results obtained by its algorithm with those produced by other state-of-the-art methods, each one defining specific characteristics that a proper network partition should express.

Often, the comparison is made among methods that have similar characteristics: algorithms that optimize the same quality function (modularity, conductance, density, etc.) and produce overlapping/crisp/hierarchical network partitions, which are designed to work on directed/undirected graphs, and so on. To shed some lights on the most popular evaluation strategies adopted in the literature, in the following we categorize them and discuss testing environments used to evaluate community partitions. In particular, in Section 5.1, we describe how synthetic network generators are often used to design controlled experiments: there we introduce two classes of benchmarks built upon them: static and dynamic ones. In Section 5.2, we discuss two families of approaches designed to assess community quality: internal and external evaluation.

5.1 Synthetic Network Generators

Several network properties can be used to characterize real-world phenomena: network modeling aims to replicate them, thus allowing for the generation of synthetic datasets that, at least to some extent, can be used as analytical proxies. The general aim of network modeling is to capture some essential properties lying behind real-world phenomena and replicate them while generating synthetic data, imposing only a few simple constraints.

Several models of increasing complexity and realism were proposed during the 20th century, the most famous being the random graphs of Rényi and Erdös (1959), the small-world networks of Watts and Strogatz (1998), the preferencial attachment model of Barabási and Albert (1999), Forest Fire by Leskovec et al. (2005), and the community-affiliation graph (Yang and Leskovec 2014). For a more detailed presentation of the history of network models, refer to Rossetti (2015).

Complex network modeling studies generated 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 CD algorithm is the ability to produce benchmark datasets that enable the following:

- *Controlled environment testing*: Network generators allow fine tuning of network characteristics, such as network size and density. Such flexibility enables an extensive algorithm evaluation on networks that have different characteristics but are generated to follow similar topologies. Generators make it possible, given a CD algorithm, to evaluate:
  - *Stability*: The performance of a CD approach can be evaluated on a high number of network instances with similar properties to provide an estimate of the algorithmic stability.
  - *Scalability*: Synthetic graphs can be used to test the actual scalability of an algorithm while increasing the network size.
- *Ground-truth testing*: Some network generators provide as a by-product a ground-truth partition of the generated network. Such a partition can be used to evaluate the one provided by the tested algorithm.

Two families of network generators have been described to provide benchmarks for CD algorithms: generators that produce *static* graphs-partitions and generators that describes *dynamic* graphs-partitions. Surprisingly, the former are often used to evaluate DCD algorithms: this happens because they are more widespread than the latter and allows comparison with static algorithms. Of course, they can only be used to evaluate the quality of the detection at time t and not the smoothness of communities.

Static benchmarks. The classic and most famous static benchmarks are the Girvan-Newman (GN, introduced by Girvan and Newman (2002)) and the LFR (introduced by Lancichinetti et al. (2008)). The GN benchmark is built upon the so-called planted l-partition model described in Condon and Karp (2001), Brandes et al. (2003), and Gaertler et al. (2007). This generator takes as input a given ground-truth partition (generated as as equally sized Erdös-Reńyi random graphs (Rényi and Erdos 1959)) and two parameters respectively identifying the probabilities of intra- and interclusters links. Planted l-partition models (also known as ad hoc models) were proposed to produce graphs that have different characteristics: overlapping community structures (Sawardecker et al. 2009; Danon et al. 2006), as well as weighted (Fan et al. 2007) and bipartite (Guimerà et al. 2007) graphs, were modeled to better mimic real network characteristics. A method to cope with the limitation of the GN benchmark (Poisson degree distribution, equal community sizes), and to fill the gap among Erdös-Rényi graphs and real ones, was introduced in Lancichinetti et al. (2008). The networks generated by this model have both node degrees and community sizes following a power law distribution. LFR has also been generalized to handle weighted and directed graphs, and to generate overlapping communities (Lancichinetti and Fortunato 2009).

Dynamic benchmarks. A few works have proposed extensions of GN and LFR that introduce topology dynamics as well as entirely new dynamic benchmarks.

In Lin et al. (2008), a variant of the GN model is introduced to evaluate the FaceNet framework. There, the authors introduce network dynamics by generating different graph snapshots and interpolating from a snapshot to its subsequent by (i) randomly selecting a fixed number of nodes from each community, (ii) removing them, and (iii) allocating them to different communities.

In Greene et al. (2010), a dynamic benchmark built upon LFR is introduced: starting from a static synthetic graph with ground-truth communities, such benchmark randomly permuted 20% of the node memberships to mimic node-community migrations.

In Granell et al. (2015), a new model based on SBMs is proposed: a graph is split into q subgraphs where nodes belonging to the same subgraph are connected with probability p_{in}, whereas edges connecting subgraphs have probability p_{out}, and then an ad hoc procedure makes the network evolve, generating community events. In Bazzi et al. (2016), a generic approach is proposed to generate both multilayer and temporal networks with a community structure. The method adopts a two-step process: (i) generating a multilayer partition satisfying user-defined parameters and (ii) sampling edges consistent with the partition. The underlying model of edges and partition is the degree-corrected SBM. An interlayer dependency tensor allows one to define how strongly the community structure in one layer depends on other layers. In the case of dynamic networks, the layer corresponding to time t typically depends only on the network at time $t-1$. The model does not support community events but provides an implementation in MATLAB.[1]

Finally, in Rossetti (2017), RDyn, a new benchmark specifically tailored for the DCD problem, is introduced. RDyn[2] generates dynamic network topologies (both in SN and TN format) and temporally evolving ground-truth communities.

5.2 Methodologies

Community partitions can be evaluated pursuing both external and internal quality analysis strategies. The former methodologies assume the existence of an external ground truth that needs to be retrieved or a specific partition quality score to optimize. The latter focus on the inspection and description of topologies identified, and on the evaluation of the approach's complexity and scalability.

External: ground-truth communities. We have seen how GN and LFR were designed to produce tunable network structures having ground-truth partitions. Moreover, several real-world datasets also come with associated community annotations.[3]

Ground-truth community structures have well-known characteristics (fixed size, density, cut ratio, etc.) and/or semantic coherence (i.e., node label homophily on a given external attribute). As such, a way to evaluate the effectiveness of CD algorithms consists of comparing the divergence between the partition they produce and the planted one. Although several criticisms were opposed to this evaluation methodology (Peel et al. 2017) (e.g., since the CD is an ill-posed problem, it is not assured that the planted partition is the optimal one for all quality functions optimized by existing algorithms), it is still the most widely spread to compare different algorithmic approaches. The common way to assess how a given partition resembles the ground-truth one is to compute the NMI (Lancichinetti et al. 2008; McDaid et al. 2011; Lancichinetti et al. 2009) a measure of similarity borrowed from information theory, defined in Rossetti et al. (2016):

$$NMI(X,Y) = \frac{H(X) + H(Y) - H(X,Y)}{(H(X) + H(Y))/2}, \tag{2}$$

where $H(X)$ is the entropy of the random variable X associated to an identified community, $H(Y)$ is the entropy of the random variable Y associated to a ground-truth one, and $H(X,Y)$ is the joint

[1] https://github.com/ MultilayerBenchmark/MultilayerBenchmark/.
[2] The RDyn code is available at https://goo.gl/WtLg4V.
[3] See https://snap.stanford.edu/data/index.html#communities.

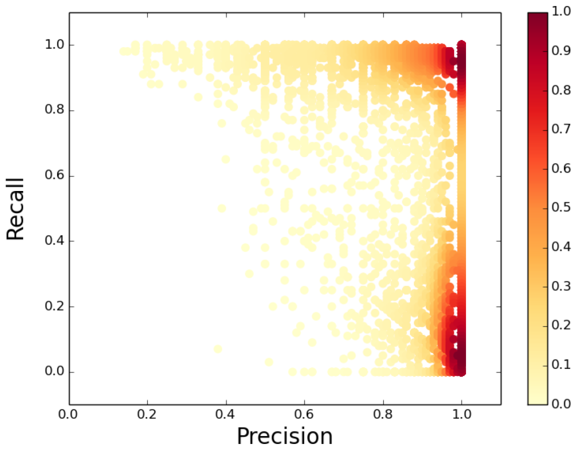

Fig. 8. F1-community measure scatter plot. Each point represents the (precision,recall) pairs for a given match: the deeper the color, the more the communities share the same values of (precision,recall) values. The optimal matching value is shown in the upper right corner (precision and recall equal to 1): communities with high precision and low recall (bottom right corner) underestimate the ground-truth ones, and communities with low precision and high recall (top left corner) overestimate the ground-truth ones.

entropy. NMI is defined in the interval [0,1] and is maximal when the compared communities are identical. One drawback of NMI is that assuming an approximate size z for the compared community sets, its computation requires $O(z^2)$ comparisons, a complexity that makes it often unusable to evaluate partitions of large-scale networks. To cope with the high computational complexity of this type of method in recent years, several approaches have been proposed (Cazabet et al. 2015; Rossetti et al. 2016, 2017). As an example, in Rossetti et al. (2016) and Rossetti et al. (2017), the F1-community score was introduced. In that work, the evaluation problem is solved as a classification task:

- Network nodes are labeled according to their ground-truth community label.
- Each community identified by the tested algorithm is matched to the ground-truth one whose label is shared by the majority of its nodes.
- Precision and recall are computed for each community by considering the actual nodes and the real ones.
- The F1-community score is computed as the average of the harmonic mean of precision and recall of the matched communities.

This approach (used in Rossetti (2017) and Rossetti et al. (2017), where a normalized version, NF1,[4] is proposed to cope with community overlap and redundancy) requires $O(z)$ to be computed (assuming an already labeled graph). Moreover, it allows graphical visualization of the performance of a given algorithm via density scatter plots. As an example, Figure 8 shows a visual representation of F1-community where the communities identified by a CD method are compared to ground-truth ones.

Although real datasets with ground-truth communities, as well as the LFR/GN benchmarks, are commonly used to perform testing of DCD approaches by considering the network at a particular instant of its evolution, it is mandatory to underline one huge deficiency: such methods do

[4]The NF1 code is available at http://goo.gl/kWIH2I.

not take into account the temporal dimension. In particular, GN and LFR were designed to allow ground-truth testing for classical static CD algorithms. For this reason, their use in conjunction with dynamic approaches raises several practical issues. How can subsequent (coherent) network snapshots be generated? Is it possible to revise them to produce interaction networks? If so, how can they reliably mimic edge/node vanishing? One of the open problems that afflicts DCD is, as of today, the evaluation of the produced communities: how can time-aware ground-truth partitions be defined (even in synthetic scenarios)? As discussed in Section 5.1, so far only a few approaches have been designed to address these latter issues (e.g., RDyn (Rossetti 2017)).

External: quality function. In absence of ground-truth communities, a common way to compare different algorithms is to rank their partitions with respect to a community quality score. The algorithm that produces the partition reaching the highest score is then, by definition, considered the best one for the analyzed network.

Modularity (Newman 2004; Newman and Girvan 2004) is probably the most widely used quality function. Values of Q approaching 1 indicate partitions with a strong community structure, whereas values close to 0 indicate that the partition does not correspond to a community structure. Modularity has been used extensively both to evaluate and define CD algorithms for static and dynamic networks. However, the legitimacy of modularity has been challenged in recent years. In Fortunato and Barthélemy (2007), the authors prove that partitions of optimal modularity do not necessarily correspond to what one expects of good communities; in particular, they introduced the problem of "resolution limit," which may prevent from detecting small communities in large networks and vice versa.

Indeed, modularity is not the only measure used to evaluate partition quality. Among the most used, we can recall the following:

—Conductance (Radicchi et al. 2004; Yang and Leskovec 2015) (i.e., the percentage of edges that cross the cluster border)
—Expansion (Radicchi et al. 2004) (i.e., the number of edges that cross the community border)
—Internal density (Radicchi et al. 2004) (i.e., the ratio of edges within the cluster with respect to all possible edges)
—Cut ratio and normalized cut (Fortunato 2010; Shi and Malik 2000) (i.e., the fraction of all possible edges leaving the cluster)
—Maximum/average ODF (Flake et al. 2000) (out-degree fraction; i.e., the maximum/average fraction of nodes' edges crossing the cluster border)
—Flake ODF (Flake et al. 2000) (i.e., the fraction of nodes involved in fewer edges within the community than outside it)
—Volume (i.e., the sum of degrees of nodes in the community)
—Edge cut (i.e., the number of edges that should be removed to destroy all paths connecting the community to the rest of the network).

All of these indicators are used to get insights on the compactness and topological consistency of communities. Studies about the relations between these quality functions can be found in Yang and Leskovec (2015) and Creusefond et al. (2016). However, evaluating solutions based on a golden quality function has a major drawback: it favors methods that are designed to maximize it. Even though this type of strategy can be used fruitfully to compare methods that explicitly optimize a specific measure, its application to approaches that search for communities with a different definition may produce misleading or inconclusive/irrelevant comparisons.

Internal. Several works overcome the issue of identifying a qualitative ranking among different algorithms by proposing a quantitative evaluation: instead of comparing to a ground truth or computing quality measures, they focus on the algorithmic complexity (Tantipathananandh et al. 2007; Bourqui et al. 2009; Rossetti et al. 2017), running time (Li et al. 2011; Rossetti et al. 2017; Gupta et al. 2012; Shao et al. 2014), scalability (Folino and Pizzuti 2014; Tan et al. 2014; Zakrzewska and Bader 2015), or analysis enabled by their approaches (i.e., identification of specific life cycle events (Palla et al. 2007; Cazabet et al. 2012; Lee et al. 2014)).

Internal evaluations, conversely from external ones, do not want to produce a direct comparison among different partitions: they assume that each algorithm is based on a different community definition, and—after describing the structures that each approach generates—they measure quantitative performances and define a context of applicability for the proposed algorithm. Indeed, internal evaluation techniques are often used as support for external one, as in Agarwal et al. (2012), Folino and Pizzuti (2014), and Rossetti et al. (2017). The need for an evaluation strategy alternative to the external ones is primarily due to the intrinsic ill posedness of the treated problem.

As we discussed, CD in dynamic networks can be declined in several ways by imposing different constraints on the network representation and its dynamics, as well as focusing the analytical goal on different objectives. This heterogeneous and fascinating scenario makes the evaluation task complex: unfortunately, so far, complete and accepted methodologies able to address it were not presented, thus leaving researchers without a golden standard to follow.

6 APPLICATIONS

Up to this point, we have categorized existing DCD approaches, as well as the strategies adopted to validate their results. In this section, we briefly discuss some related themes. In Section 6.1, the main categories of real-world dynamic network datasets are described, whereas in Section 6.2, visualization strategies for dynamic networks are introduced. Finally, in Section 6.3, examples of how DCD can be used as an analytical tool are provided.

6.1 Real-World Dynamic Networks

The proliferation of online services, as well as the continuous collection of data regarding human activities, has given birth in recent years to the so-called Big Data science. Indeed, almost all kinds of human-generated content can be used to model graphs: online social networks, mobility traces, information regarding collaborations or trade exchanges, and Web traffic are all examples of contexts in which it is possible to define relations among objects and design networks to analyze them. Unfortunately, so far, only a few easily accessible online resources have been made available to researchers to test dynamic network analytical tools. This situation is mostly due to the novelty of this research field: whereas static network datasets of any size (often enriched by valuable semantic annotations) have always been objects of analysis—and thus collected and catalogued—dynamic ones are often small, not well organized, and rare. Nevertheless, DCD approaches are often tested on these type of datasets to evaluate their performance and compare runtimes on real-world scenarios. In the following, we discuss the main online and offline sources used to extract dynamic networks and provide a catalog of currently available datasets.

Collaboration networks. An extensively studied type of dynamic networks is the one that models work-related collaborations, such as coauthoring of scientific publications or movie creation. Collaboration networks are extensively used by methods that work on snapshot graphs due to the natural time scale they provide (each scientific paper has publication year, each movie a released date). One drawback of networks built upon this type of data is their lack of adherence to classical canons of real social networks: they are composed of interactions between groups more than activities between pairs of entities. Such a characteristic reflects into networks structures composed of clique chains whose density is usually higher than what is expected in traditional social networks. Table 1 reports the most used online resources.

Table 1. Dynamic Collaboration Networks

Dataset	URL	Works
DBLP	https://kdl.cs.umass.edu/display/public/DBLP	Tang et al. (2008), Wang et al. (2008), Miller and Eliassi-Rad (2009), Takaffoli et al. (2011), Goldberg et al. (2011), and Ma and Huang (2013)
cond-mat	http://arxiv.org	Wang et al. (2008)
cit-HepTh/Ph	http://arxiv.org	Shang et al. (2014), Görke et al. (2013), and İlhan and Öğüdücü (2015)
IMDB	http://imdb.com	Goldberg et al. (2011)

Table 2. Online/Offline Social Networks

Dataset	URL	Works
Twitter	https://twitter.com	Lee et al. (2014)
Delicious.com	https://del.icio.us	Sun et al. (2010)

Online/offline social networks. The second class of sources often used to generate dynamic graphs is provided by online/offline social contexts. This kind of data is characterized by a clear semantic link: self-declared social relationships. These relationships can be considered as mutual (undirected graph representation) or not necessarily so (directed graph representation). Social structures are often analyzed in different snapshots, whose durations are defined by the analyst since there is no natural temporal resolution to apply (as discussed in Sections 2 and 2.2). They can also be studied as temporal networks of relations. Table 2 lists some online data sources used to test the algorithms discussed in Section 4.

Communication networks. Data regarding communications (phone call logs, chats/emails, face-to-face interactions, etc.) are often released with very precise timestamp annotations. Such fine-grain temporal information makes it possible to build upon them not only snapshot graphs but also temporal networks of interactions, or of relations using a chosen TTL. Those datasets, listed in Table 3, are primarily used by partition update by a set of rules approaches (see Section 4.2).

Technological networks. The last class of networks widely adopted to test dynamic algorithms is the technological one. Peer-to-peer networks, as well as graphs built on top of autonomous systems routing information, are classical datasets, often used to validate even static CD approaches. Table 4 shows some examples of technological networks used in revised algorithms.

6.2 Visualization

Dynamic network visualization is a challenging task: so far, to the best of our knowledge, few solutions have been proposed to visualize the evolution of communities. Effectively representing community dynamics is difficult, as it needs to combine the challenges presented by both evolving networks and communities visualization. For a survey on the visualization of dynamic networks, please refer to Beck et al. (2017). For a survey on the visualization of community structures, please refer to Vehlow et al. (2015b).

We can classify visualization methods in two categories: (i) dynamic visualizations, such as videos, in which the state of communities at different timesteps is represented by independent visualizations, and (ii) static drawings that summarize the whole network/communities evolution.

Table 3. Examples of Real World Dynamic Graphs: Communication Networks

Dataset	URL	Works
ENRON	https://goo.gl/gQqo8l	Falkowski et al. (2008), Tang et al. (2008), Wang et al. (2008), Folino and Pizzuti (2010, 2014), Li et al. (2011), Shang et al. (2014), and Ferry and Bumgarner (2012)
KIT	https://goo.gl/oVrdb7	Görke et al. (2013)
Mobile phone	—	Folino and Pizzuti (2010, 2014), Gong et al. (2012), and Guo et al. (2014)
Wiki-vote	https://goo.gl/pzhst2	Shang et al. (2014)
Senate/court voting	—	Crane and Dempsey (2015)
Thiers	http://goo.gl/jhAcY8	Viard et al. (2016)
CAIDA	http://goo.gl/72y9iT	Miller and Eliassi-Rad (2009)
Facebook	http://goo.gl/8J18cw	Rossetti et al. (2017)
Weibo	http://goo.gl/bcoQiy	Rossetti et al. (2017)
Digg	http://goo.gl/ybKQS2	Zakrzewska and Bader (2015)
Slashdot	http://goo.gl/FGgEiM	Zakrzewska and Bader (2015)

Table 4. Technological Networks

Dataset	URL	Works
AS routers	https://goo.gl/tGk1lJ	Görke et al. (2013) and Alvari et al. (2014)
p2p-Gnutella	https://goo.gl/Tgl5NC	Ma and Huang (2013)

6.2.1 Dynamic Visualization. Methods falling in this category have been designed to represent communities in static networks and have been extended to handle dynamic networks by the means of animations. This adaptation usually consists of using a layout of nodes adapted to favor the stability of their position despite the changes in their connectivity.

Frishman and Tal (2004) focus on minimizing the change of positions of each cluster as a whole, and of the nodes within them. Invisible nodes representing communities are used to ensure the stability of community centroids. Communities are represented by node coloring and/or by drawing a square around nodes belonging to the same group.

In Beiró et al. (2010), the whole community is represented as a single node, whose size corresponds to its number of components. Edges between communities represent all of the links that connect their respective nodes. In Boyack et al. (2008), communities are shown as pie charts—a visualization used to summarize some property of the nodes of which they are composed.

Hu et al. (2012) proposes to position nodes in the network based on a classic node positioning layout and then overlay a shape on top of the nodes belonging to the same community, in a manner appropriate for an easy understanding of the community structure, inspired by geographical maps. In Lin et al. (2010), a similar approach is used, but a fuzzy contour map is used to represent communities (Figure 9).

6.2.2 Static Visualization. In Mucha et al. (2010), a vertical position is attributed to each node, according to its first apparition and its community affiliation, to put similar nodes close to each other. The horizontal axis corresponds to time. A color is attributed to each community, and for each timestamp, the node, if present, is represented by a dot colored according to the community to which it belongs (Figure 10).

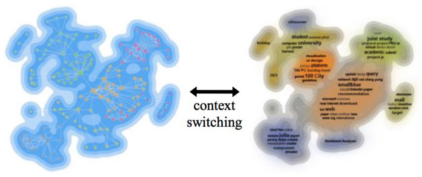

Fig. 9. Visualization using contour map on top of nodes positioned with an appropriate layout (Lin et al. 2010). (Copyright 2010 Society for Industrial and Applied Mathematics. Reprinted with permission. All rights reserved.)

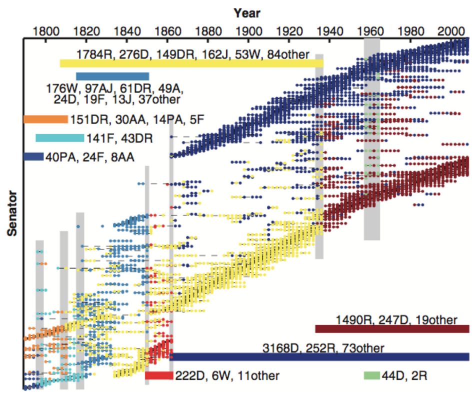

Fig. 10. Visualization used by Mucha et al. (2010).

In Reda et al. (2011), each community is represented as a fixed-width horizontal space and nodes as colored lines that switch between them when community affiliation changes occur.

A similar idea is proposed in Rosvall and Bergstrom (2010). There, alluvial diagrams are used to represent community evolution: each node is represented as a continuous line, whose vertical position is determined by its community affiliation. Nodes belonging to the same community at a given timestamp always appear next to each other and share the same color. Therefore, communities appear as horizontal strips whose width corresponds to the number of nodes that they contain.

The authors of Vehlow et al. (2015a) propose to combine views of the structure of the communities with the alluvial representation. They employ a custom sorting of the communities and vertices per timestep to minimize crossing and automatic color assignment methods to improve readability (Figure 11).

The authors of Morini et al. (2015) use a variation of the alluvial diagram in which the vertical position of communities is less constrained, facilitating the tracking of community life cycles (Figure 12).

In Vehlow et al. (2016), a method is proposed to visualize both a hierarchical organization of communities and their evolution through time. The method use matrices to represent each step, and the relations between communities in different timesteps or different hierarchical levels are represented by edges between the communities.

6.3 DCD as a Tool

Several articles (Agarwal et al. 2012; Lee et al. 2014; Cazabet et al. 2012) proposed using DCD algorithms to identify events in online social networks. The general approach that they follow consists of creating a dynamic network of keywords or n-grams, in which edges correspond to

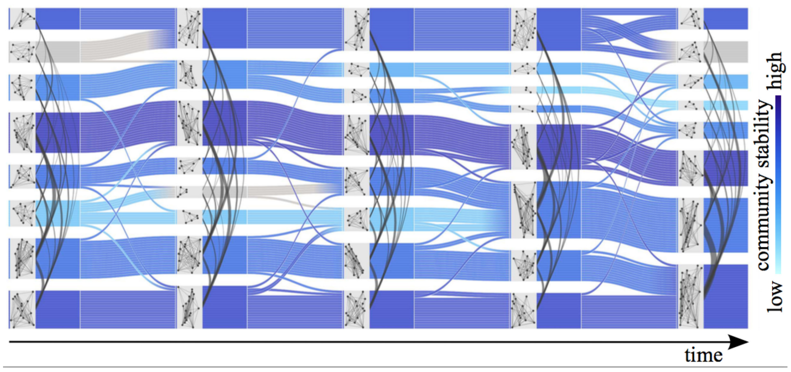

Fig. 11. Visualization by alluvial diagrams (Vehlow et al. 2015a).

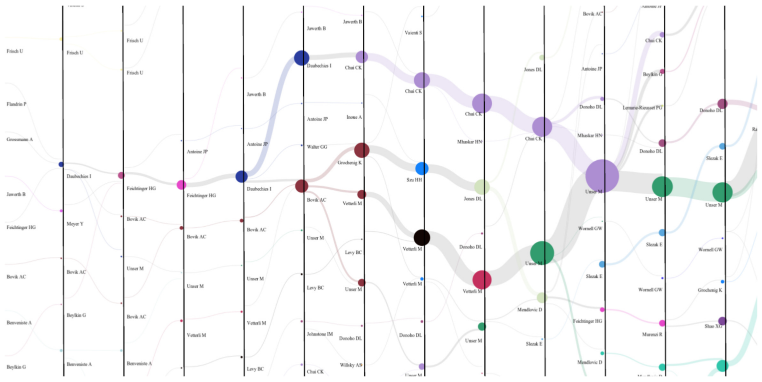

Fig. 12. Variation of the alluvial diagram to improve the readability of the life cycle of communities (Morini et al. 2015).

a semantic relationship between terms observed in the studied OSN. The dynamic communities found on such networks correspond to "popular events" aggregating similar bursting terms. The evolution of groups of people in social media has also been addressed in Gliwa et al. (2012) and Saganowski et al. (2015) on a Polish blogosphere service, Facebook, and DBLP.

DCD approaches could also be used as a tool for telecommunication networks. As such methods allow keeping up-to-date information on the community structure in real time, they enable for the adoption of algorithms able to extract knowledge from highly dynamical network substructures. In Nguyen et al. (2011b), the authors propose using DCD for improving routing in mobile ad hoc networks (MANETs), whereas in Nguyen et al. (2011a), the authors propose using them to contain worms, notably by concentrating on the immunization of nodes at the interface between the infected community and the rest of the network. In Schlitter and Falkowski (2009), the authors study music listened to by a set of 1,800 users on the platform Last.fm over a period of 3 years. They detect groups of users capturing specific music genres and study their evolution. For instance, they observe the fusion between a group of users listening to progressive metal and another group listening to death metal.

Scientific collaboration network analysis is a topic on which DCD algorithms have been applied several times, notably in Cazabet et al. (2010), Lin et al. (2009), Van Nguyen et al. (2012), and Rosvall and Bergstrom (2010). One of the most in-depth analyses can be found in Rosvall and Bergstrom

(2010), where a merge between the fields of neurology, psychology, and some areas of molecular and cell biology is identified, forming a new field of neuroscience between 2003 and 2005.

In Braun et al. (2015), DCD is used to study the reconfiguration of the brain network during executive cognition in humans.

Finally, in Mucha et al. (2010), political affiliation among senators is studied based on vote similarities. Known historical changes in U.S. politics are compared to communities of senators found by a DCD algorithm. The same method has been applied for the United Nations General Assembly (Macon et al. 2012), exploring different processes to construct the network from data.

7 OUTLOOK

Despite the significant amount of literature already published on the topic of DCD, numerous lines of research remain open. The problem per se is not trivial: indeed, we have seen that it cannot be reduced to simply running a static algorithm at several points in time.

As the scientific community is still tackling the problem of what is a good community, researchers on DCD are confronted with the problem of defining what it means for a community to persist and evolve. Moreover, they need to solve the apparent paradox that "optimal" partitions found during different network evolution steps do not always form a coherent dynamic structure.

An underlying problem is due to the heterogeneous nature of the dynamic networks used by different methods. Although snapshots and temporal networks have the same expressivity (i.e., both of them can be used to model the same network topology dynamics), they often have to be considered differently in practice. Snapshots are often used when the network varies greatly between subsequent observations, whereas temporal networks represent a fine-grain evolution. As a consequence, a method designed for one of these approaches often solves a different problem than one designed for the other.

The classification that we have adopted highlights the amazing variety of approaches adopted by different authors. Such DCD methodologies not only propose different ways of solving the same problem but also explore different definitions of what a dynamic community is, and how to find network partitions that are both coherent in the long run and meaningful at given points in time.

Despite this variety of definitions and approaches, common grounds have begun to emerge, particularly regarding the definition of *events* or *operations*.

What we believe the field lacks the most is a common ground to evaluate and compare the different methods among themselves. Indeed, the earliest methods for community detection have compared themselves to the Zachary Karate Club, although there is no such widespread toy model for dynamic networks.

More importantly, the field of CD has been structured by the introduction of advanced benchmark and systematic, quantitative comparison of algorithms. Although a few steps have been taken in this direction, by introducing simplistic synthetic benchmarks, there is still no universally recognized benchmark equivalent to the LFR for the dynamic case, and no systematic comparison of methods has been conducted yet. The problem is indeed arduous: to compare methods and their results, it is mandatory to provide to each method an appropriate input and convert outputs to comparable pieces of information. Indeed, the problem is not only a technical one: it involves the lack of standard formal definitions. How to compare methods working on a few snapshots and methods on temporal networks? How to compare results that do not share the same possible operations of communities? And how to design a benchmark that generates realistic evolution of communities when such evolution is not well known yet?

A question many will ask is this: what method should I use for my data? Unfortunately, there is no final answer to this question, particularly due to the lack of systematic evaluation. Nevertheless, the classification of methods according to their characteristics already gives us some hints. A first aspect to consider is the number of steps of evolution present in the original data. If there are few steps (i.e., less than 10), typically corresponding to aggregated observations over months or years, the snapshot representation is usually the most suitable. If the number of steps is high (in the thousands or more), methods using temporal networks are more appropriate. Another aspect is the priority given to the quality of communities found at a particular instant rather than to the coherence of the continuity of communities. For instance, a tool using DCD to improve the routing in a point-to-point network in real time might give higher importance to the current state of the network. Conversely, a retrospective analysis of past events aimed at interpreting the emergence of scientific fields by clustering journals or other works should favor the long-term coherence of communities.

Finally, we observed that despite a large number of methods proposed, few applications of DCD to concrete problems were proposed—except the ones by the authors themselves. As of today, it seems that the field lacks a visibility and that authors who need a DCD method tend to develop their own, ad hoc algorithm, often without considering existing ones. We believe that the future of the field will rely on its greater cohesion and ease of access for newcomers to reach an audience as large as conventional CD.

REFERENCES

Manoj K. Agarwal, Krithi Ramamritham, and Manish Bhide. 2012. Real time discovery of dense clusters in highly dynamic graphs: Identifying real world events in highly dynamic environments. *Proceedings of the VLDB Endowment* 5, 10, 980–991.

Uri Alon. 2007. Network motifs: Theory and experimental approaches. *Nature Reviews Genetics* 8, 6, 450–461.

Hamidreza Alvari, Alireza Hajibagheri, and Gita Sukthankar. 2014. Community detection in dynamic social networks: A game-theoretic approach. In *Proceedings of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM'14)*. IEEE, Los Alamitos, CA, 101–107.

Sitaram Asur, Srinivasan Parthasarathy, and Duygu Ucar. 2009. An event-based framework for characterizing the evolutionary behavior of interaction graphs. *Transactions on Knowledge Discovery From Data* 3, 4, 16.

Thomas Aynaud, Eric Fleury, Jean-Loup Guillaume, and Qinna Wang. 2013. Communities in evolving networks: Definitions, detection, and analysis techniques. In *Dynamics on and of Complex Networks, Volume 2*. Springer, 159–200.

Thomas Aynaud and Jean-Loup Guillaume. 2010. 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'10)*. IEEE, Los Alamitos, CA, 513–519.

Thomas Aynaud and Jean-Loup Guillaume. 2011. Multi-step community detection and hierarchical time segmentation in evolving networks. In *Proceedings of the 5th Social Network Mining and Analysis Workshop, (SNA-KDD Workshop'11)*.

Shweta Bansal, Sanjukta Bhowmick, and Prashant Paymal. 2011. Fast community detection for dynamic complex networks. In *Complex Networks*. Springer, 196–207.

Albert-László Barabási and Réka Albert. 1999. Emergence of scaling in random networks. *Science* 286, 5439, 509–512.

Danielle S. Bassett, Mason A. Porter, Nicholas F. Wymbs, Scott T. Grafton, Jean M. Carlson, and Peter J. Mucha. 2013. Robust detection of dynamic community structure in networks. *Chaos* 23, 1, 013142. DOI: http://dx.doi.org/10.1063/1.4790830

Marya Bazzi, Lucas G. S. Jeub, Alex Arenas, Sam D. Howison, and Mason A. Porter. 2016. Generative benchmark models for mesoscale structure in multilayer networks. arXiv:1608.06196.

Fabian Beck, Michael Burch, Stephan Diehl, and Daniel Weiskopf. 2017. A taxonomy and survey of dynamic graph visualization. *Computer Graphics Forum* 36, 1, 133–159. DOI: http://dx.doi.org/10.1111/cgf.12791

Mariano G. Beiró, Jorge Rodolfo Busch, and José Ignacio Alvarez-Hamelin. 2010. Visualizing communities in dynamic networks. In *Proceedings of the Latin-American Workshop on Dynamic Networks (LAWDN'10)*. 4.

Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment* 2008, 10, P10008.

András Bóta, László Csizmadia, and András Pluhár. 2010. Community Detection and Its Use in Real Graphs. Available at http://www.academia.edu/17955192/Community_detection_and_its_use_in_Real_Graphs.

András Bóta, Miklós Krész, and András Pluhár. 2011. Dynamic communities and their detection. *Acta Cybernetica* 20, 1, 35–52.

Romain Bourqui, Frédéric Gilbert, Paolo Simonetto, Faraz Zaidi, Umang Sharan, and Fabien Jourdan. 2009. Detecting structural changes and command hierarchies in dynamic social networks. In *Proceedings of the International Conference on Advances in Social Network Analysis and Mining (ASONAM'08)*. IEEE, Los Alamitos, CA, 83–88.

Kevin Boyack, Katy Börner, and Richard Klavans. 2008. Mapping the structure and evolution of chemistry research. *Scientometrics* 79, 1, 45–60.

Ulrik Brandes, Marco Gaertler, and Dorothea Wagner. 2003. Experiments on graph clustering algorithms. In *Algorithms—ESA 2003.* Lecture Notes in Computer Science, Vol. 2832. Springer, 568–579.

Urs Braun, Axel Schäfer, Henrik Walter, Susanne Erk, Nina Romanczuk-Seiferth, Leila Haddad, Janina I. Schweiger, et al. 2015. Dynamic reconfiguration of frontal brain networks during executive cognition in humans. *Proceedings of the National Academy of Sciences* 112, 37, 11678–11683. DOI: http://dx.doi.org/10.1073/pnas.1422487112

Piotr Bródka, Stanisław Saganowski, and Przemysław Kazienko. 2013. GED: The method for group evolution discovery in social networks. *Social Network Analysis and Mining* 3, 1, 1–14.

Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. 2012. Time-varying graphs and dynamic networks. *International Journal of Parallel, Emergent and Distributed Systems* 27, 5, 387–408.

Remy Cazabet and Frederic Amblard. 2011. Simulate to detect: A multi-agent system for community detection. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT'11)*, Vol. 2. IEEE, Los Alamitos, CA, 402–408.

Rémy Cazabet and Frédéric Amblard. 2014. Dynamic community detection. In *Encyclopedia of Social Network Analysis and Mining*. Springer, 404–414.

Remy Cazabet, Frederic Amblard, and Chihab Hanachi. 2010. Detection of overlapping communities in dynamical social networks. In *Proceedings of the 2nd International Conference on Social Computing (SocialCom'10).* IEEE, Los Alamitos, CA, 309–314.

Remy Cazabet, Rathachai Chawuthai, and Hideaki Takeda. 2015. Using multiple-criteria methods to evaluate community partitions. arXiv:1502.05149.

Rémy Cazabet, Hideaki Takeda, Masahiro Hamasaki, and Frédéric Amblard. 2012. Using dynamic community detection to identify trends in user-generated content. *Social Network Analysis and Mining* 2, 4, 361–371.

Zhengzhang Chen, Kevin A. Wilson, Ye Jin, William Hendrix, and Nagiza F. Samatova. 2010. Detecting and tracking community dynamics in evolutionary networks. In *Proceedings of the International Conference on Data Mining Workshops (ICDMW'10).* IEEE, Los Alamitos, CA, 318–327.

Aaron Clauset, Mark E. J. Newman, and Cristopher Moore. 2004. Finding community structure in very large networks. *Physical Review E* 70, 6, 066111.

Anne Condon and Richard M. Karp. 2001. Algorithms for graph partitioning on the planted partition model. *Random Structures and Algorithms* 18, 2, 116–140.

Michele Coscia, Fosca Giannotti, and Dino Pedreschi. 2011. A classification for community discovery methods in complex networks. *Statistical Analysis and Data Mining: The ASA Data Science Journal* 4, 5, 512–546.

Harry Crane. 2014. The cut-and-paste process. *Annals of Probability* 42, 5, 1952–1979.

Harry Crane and Walter Dempsey. 2015. Community detection for interaction networks. arXiv:1509.09254.

Jean Creusefond, Thomas Largillier, and Sylvain Peyronnet. 2016. On the evaluation potential of quality functions in community detection for different contexts. In *Proceedings of the International Conference and School on Network Science.* 111–125.

Leon Danon, Albert Díaz-Guilera, and Alex Arenas. 2006. The effect of size heterogeneity on community identification in complex networks. *Journal of Statistical Mechanics: Theory and Experiment* 2006, 11, P11010.

Zeineb Dhouioui and Jalel Akaichi. 2013. Overlapping community detection in social networks. In *Proceedings of the International Conference on Bioinformatics and Biomedicine (BIBM'13).* IEEE, Los Alamitos, CA, 17–23.

Zeineb Dhouioui and Jalel Akaichi. 2014. Tracking dynamic community evolution in social networks. In *Proceedings of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM'14).* IEEE, Los Alamitos, CA, 764–770.

Dongsheng Duan, Yuhua Li, Yanan Jin, and Zhengding Lu. 2009. Community mining on dynamic weighted directed graphs. In *Proceedings of the 1st International Wworkshop on Complex Networks Meet Information & Knowledge Management.* ACM, New York, NY, 11–18.

Dongsheng Duan, Yuhua Li, Ruixuan Li, and Zhengding Lu. 2012. Incremental K-clique clustering in dynamic social networks. *Artificial Intelligence Review* 38, 2, 129–147.

N. Duhan, A. K. Sharma, and K. K. Bhatia. 2009. Page ranking algorithms: A survey. In *Proceedings of the International Advance Computing Conference (IACC'09).* IEEE, Los Alamitos, CA, 1530–1537.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD'96)*, Vol. 96. 226–231.

Tanja Falkowski, Jorg Bartelheimer, and Myra Spiliopoulou. 2006. Mining and visualizing the evolution of subgroups in social networks. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence (WI'06).* IEEE, Los Alamitos, CA, 52–58.

Tanja Falkowski, Anja Barth, and Myra Spiliopoulou. 2008. Studying community dynamics with an incremental graph mining algorithm. In *Proceedings of the Americas Conference on Information Systems (AMCIS'08)*. 1–11.

Tanja Falkowski and Myra Spiliopoulou. 2007. Data mining for community dynamics. *Kunstliche Intelligenz* 21, 3 (2007), 23–29.

Ying Fan, Menghui Li, Peng Zhang, Jinshan Wu, and Zengru Di. 2007. Accuracy and precision of methods for community identification in weighted networks. *Physica A: Statistical Mechanics and Its Applications* 377, 1, 363–372.

James P. Ferry and J. Oren Bumgarner. 2012. Community detection and tracking on networks from a data fusion perspective. arXiv:1201.1512.

Gary William Flake, Steve Lawrence, and C. Lee Giles. 2000. Efficient identification of Web communities. In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, 150–160.

Francesco Folino and Clara Pizzuti. 2010. Multiobjective evolutionary community detection for dynamic networks. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*. ACM, New York, NY, 535–536.

Francesco Folino and Clara Pizzuti. 2014. An evolutionary multiobjective approach for community discovery in dynamic networks. *Transactions on Knowledge and Data Engineering* 26, 8, 1838–1852.

Santo Fortunato. 2010. Community detection in graphs. *Physics Reports* 486, 3, 75–174.

Santo Fortunato and Marc Barthélemy. 2007. Resolution limit in community detection. *Proceedings of the National Academy of Sciences* 104, 1, 36–41.

Santo Fortunato and Darko Hric. 2016. Community detection in networks: A user guide. *Physics Reports* 659, 1–44.

Yaniv Frishman and Ayellet Tal. 2004. Dynamic drawing of clustered graphs. In *Proceedings of the Symposium on Information Visualization (INFOVIS'04)*. IEEE, Los Alamitos, CA, 191–198.

Marco Gaertler, Robert Görke, and Dorothea Wagner. 2007. Significance-driven graph clustering. *Algorithmic Aspects in Information and Management.* Lecture Notes in Computer Science, Vol. 4508. Springer, 11–26.

Laetitia Gauvin, André Panisson, and Ciro Cattuto. 2014. Detecting the community structure and activity patterns of temporal networks: A non-negative tensor factorization approach. *PloS One* 9, 1, e86028.

Lise Getoor and Christopher P. Diehl. 2005. Link mining: A survey. *ACM SIGKDD Explorations Newsletter* 7, 2, 3–12.

Amir Ghasemian, Pan Zhang, Aaron Clauset, Cristopher Moore, and Leto Peel. 2016. Detectability thresholds and optimal algorithms for community structure in dynamic networks. *Physical Review X* 6, 3, 031005.

Michelle Girvan and Mark E. J. Newman. 2002. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences* 99, 12, 7821–7826.

Bogdan Gliwa, Stanislaw Saganowski, Anna Zygmunt, Piotr Bródka, Przemyslaw Kazienko, and Jaroslaw Kozak. 2012. Identification of group changes in blogosphere. In *Proceedings of the International Conference on Advances in Social Networks Analysis and Mining (ASONAM'12)*. IEEE, Los Alamitos, CA, 1201–1206.

Mark Goldberg, Malik Magdon-Ismail, Srinivas Nambirajan, and James Thompson. 2011. Tracking and predicting evolution of social communities. In *Proceedings of the 3rd International Conference on Privacy, Security, Risk, and Trust (PASSAT'11) and the 3rd International Conference on Social Computing (SocialCom'11)*. IEEE, Los Alamitos, CA, 780–783.

Mao-Guo Gong, Ling-Jun Zhang, Jing-Jing Ma, and Li-Cheng Jiao. 2012. Community detection in dynamic social networks based on multiobjective immune algorithm. *Journal of Computer Science and Technology* 27, 3, 455–467.

Robert Görke, Tanja Hartmann, and Dorothea Wagner. 2012. Dynamic graph clustering using minimum-cut trees. *Journal of Graph Algorithms and Applications* 16, 2, 411–446.

Robert Görke, Pascal Maillard, Andrea Schumm, Christian Staudt, and Dorothea Wagner. 2013. Dynamic graph clustering combining modularity and smoothness. *Journal of Experimental Algorithmics* 18, 1–5.

Robert Görke, Pascal Maillard, Christian Staudt, and Dorothea Wagner. 2010. Modularity-driven clustering of dynamic graphs. In *Proceedings of the 9th International Conference on Experimental Algorithms (SEA'10)*. 436–448.

Clara Granell, Richard K. Darst, Alex Arenas, Santo Fortunato, and Sergio Gómez. 2015. Benchmark model to assess community structure in evolving networks. *Physical Review E* 92, 1, 012805.

Derek Greene, Donal Doyle, and Padraig Cunningham. 2010. Tracking the evolution of communities in dynamic social networks. In *Proceedings of the International Conference on Advances in Social Networks Analysis and Mining (ASONAM'10)*. IEEE, Los Alamitos, CA, 176–183.

Roger Guimerà, Marta Sales-Pardo, and Luís A. Nunes Amaral. 2007. Module identification in bipartite and directed networks. *Physical Review E* 76, 3, 036102.

Chonghui Guo, Jiajia Wang, and Zhen Zhang. 2014. Evolutionary community structure discovery in dynamic weighted networks. *Physica A: Statistical Mechanics and Its Applications* 413, 565–576.

Manish Gupta, Jing Gao, Yizhou Sun, and Jiawei Han. 2012. Integrating community matching and outlier detection for mining evolutionary community outliers. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, 859–867.

Tanja Hartmann, Andrea Kappes, and Dorothea Wagner. 2016. Clustering evolving networks. In *Algorithm Engineering*. Springer, 280–329.

Keith Henderson and Tina Eliassi-Rad. 2009. Applying latent Dirichlet allocation to group discovery in large graphs. In *Proceedings of the ACM Symposium on Applied Computing*. ACM, New York, NY, 1456–1461.

Tue Herlau, Morten Mørup, and Mikkel N. Schmidt. 2013. Modeling temporal evolution and multiscale structure in networks. In *Proceedings of the 30th International Conference on Machine Learning*. 960–968.

Anne-Sophie Himmel, Hendrik Molter, Rolf Niedermeier, and Manuel Sorge. 2016. Enumerating maximal cliques in temporal graphs. In *Proceedings of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM'16)*. IEEE, Los Alamitos, CA, 337–344.

Petter Holme and Jari Saramäki. 2012. Temporal networks. *Physics Reports* 519, 3, 97–125.

John Hopcroft, Omar Khan, Brian Kulis, and Bart Selman. 2004. Tracking evolving communities in large linked networks. *Proceedings of the National Academy of Sciences* 101, 1, 5249–5253.

Yifan Hu, Stephen G. Kobourov, and Sankar Veeramoni. 2012. Embedding, clustering and coloring for dynamic maps. In *Proceedings of the Pacific Visualization Symposium (PacificVis'12)*. IEEE, Los Alamitos, CA, 33–40.

Nagehan İlhan and Şule Gündüz Öğüdücü. 2015. Predicting community evolution based on time series modeling. In *Proceedings of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM'15)*. ACM, New York, NY, 1509–1516.

Katsuhiko Ishiguro, Tomoharu Iwata, Naonori Ueda, and Joshua B. Tenenbaum. 2010. Dynamic infinite relational model for time-varying relational data analysis. In *Advances in Neural Information Processing Systems 23 (NIPS'10)*. 919–927.

Manel Ben Jdidia, Céline Robardet, and Eric Fleury. 2007. Communities detection and analysis of their dynamics in collaborative networks. In *Proceedings of the 2nd International Conference on Digital Information Management (ICDIM'07)*, Vol. 2. IEEE, Los Alamitos, CA, 744–749.

Vikas Kawadia and Sameet Sreenivasan. 2012. Online detection of temporal communities in evolving networks by estrangement confinement. arXiv:1203.5126.

Min-Soo Kim and Jiawei Han. 2009. A particle-and-density based evolutionary clustering method for dynamic networks. *Proceedings of the VLDB Endowment* 2, 1, 622–633.

Mikko Kivel, Alex Arenas, Marc Barthelemy, James P. Gleeson, Yamir Moreno, and Mason A. Porter. 2014. Multilayer networks. *Journal of Complex Networks* 2, 3, 203–271. DOI: http://dx.doi.org/10.1093/comnet/cnu016arXiv:/oup/backfile/content_public/journal/comnet/2/3/10.1093_comnet_cnu 016/2/cnu016.pdf

Lauri Kovanen, Márton Karsai, Kimmo Kaski, János Kertész, and Jari Saramäki. 2011. Temporal motifs in time-dependent networks. *Journal of Statistical Mechanics: Theory and Experiment* 2011, 11, P11005.

Andrea Lancichinetti and Santo Fortunato. 2009. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Physical Review E* 80, 1, 016118.

Andrea Lancichinetti, Santo Fortunato, and János Kertész. 2009. Detecting the overlapping and hierarchical community structure in complex networks. *New Journal of Physics* 11, 3, 033015.

Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. 2008. Benchmark graphs for testing community detection algorithms. *Physical Review E* 78, 4, 046110.

Matthieu Latapy, Tiphaine Viard, and Clémence Magnien. 2017. Stream graphs and link streams for the modeling of interactions over time. arXiv:1710.04073.

Pei Lee, Laks V. S. Lakshmanan, and Evangelos E. Milios. 2014. Incremental cluster evolution tracking from highly dynamic network data. In *Proceedings of the 30th International Conference on Data Engineering (ICDE'14)*. IEEE, Los Alamitos, CA, 3–14.

Sungmin Lee, Luis E. C. Rocha, Fredrik Liljeros, and Petter Holme. 2012. Exploiting temporal network structures of human interaction to effectively immunize populations. *PloS One* 7, 5, e36439.

Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2005. Graphs over time: Densification laws, shrinking diameters and possible explanations. In *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*. ACM, New York, NY, 177–187.

Xiao-Li Li, Aloysius Tan, S. Yu Philip, and See-Kiong Ng. 2011. ECODE: Event-based community detection from social networks. In *Proceedings of the International Conference on Database Systems for Advanced Applications (DASFAA'11)*. 22–37.

Yu-Ru Lin, Yun Chi, Shenghuo Zhu, Hari Sundaram, and Belle L. Tseng. 2008. FacetNet: A framework for analyzing communities and their evolutions in dynamic networks. In *Proceedings of the 17th International Conference on World Wide Web (WWW'08)*. ACM, New York, NY, 685–694.

Yu-Ru Lin, Yun Chi, Shenghuo Zhu, Hari Sundaram, and Belle L. Tseng. 2009. Analyzing communities and their evolutions in dynamic social networks. *Transactions on Knowledge Discovery From Data* 3, 2, 8.

Yu-Ru Lin, Jimeng Sun, Nan Cao, and Shixia Liu. 2010. Contextour: Contextual contour visual analysis on dynamic multi-relational clustering. In *Proceedings of the International Conference on Data Mining*. 418–429.

Hao-Shang Ma and Jen-Wei Huang. 2013. Cut: Community update and tracking in dynamic social networks. In *Proceedings of the 7th Workshop on Social Network Mining and Analysis*. ACM, New York, NY, 6.

Kevin T. Macon, Peter J. Mucha, and Mason A. Porter. 2012. Community structure in the United Nations general assembly. *Physica A: Statistical Mechanics and Its Applications* 391, 1, 343–361.

Naoki Masuda and Renaud Lambiotte. 2016. *A Guide to Temporal Networks*. Vol. 4. World Scientific.

Catherine Matias and Vincent Miele. 2016. Statistical clustering of temporal networks through a dynamic stochastic block model. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 79, 4, 1119–1141.

Catherine Matias, Tabea Rebafka, and Fanny Villers. 2015. Estimation and Clustering in a Semiparametric Poisson Process Stochastic Block Model for Longitudinal Networks: Semiparametric Estimation in PPSBM. Available at https://hal.archives-ouvertes.fr/hal-01245867v1.

Aaron F. McDaid, Derek Greene, and Neil Hurley. 2011. Normalized mutual information to evaluate overlapping community finding algorithms. arXiv:1110.2515.

K. Miller and T. Eliassi-Rad. 2009. Continuous time group discovery in dynamic graphs. In *Proceedings of the NIPS 2009 Workshop on Analyzing Networks and Learning With Graphs*.

Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. 2002. Network motifs: Simple building blocks of complex networks. *Science* 298, 5594, 824–827.

Matteo Morini, Patrick Flandrin, Eric Fleury, Tommaso Venturini, and Pablo Jensen. 2017. Revealing evolutions in dynamical networks. arXiv:1707.02114.

Matteo Morini, Pablo Jensen, and Patrick Flandrin. 2015. Temporal evolution of communities based on scientometrics data. In *Proceedings of Sciences Des Données et Humanités Numeriques*.

Peter J. Mucha, Thomas Richardson, Kevin Macon, Mason A. Porter, and Jukka-Pekka Onnela. 2010. Community structure in time-dependent, multiscale, and multiplex networks. *Science* 328, 5980, 876–878.

Paul Newbold. 1983. ARIMA model building and the time series analysis approach to forecasting. *Journal of Forecasting (pre-1986)* 2, 1, 23.

Mark E. J. Newman. 2003. The structure and function of complex networks. *SIAM Review* 45, 2, 167–256.

Mark E. J. Newman. 2004. Fast algorithm for detecting community structure in networks. *Physical Review E* 69, 6, 066133.

Mark E. J. Newman and Michelle Girvan. 2004. Finding and evaluating community structure in networks. *Physical Review E* 69, 2, 026113.

Nam P. Nguyen, Thang N. Dinh, Sindhura Tokala, and My T. Thai. 2011a. Overlapping communities in dynamic networks: Their detection and mobile applications. In *Proceedings of the 17th Annual International Conference on Mobile Computing and Networking*. ACM, New York, NY, 85–96.

Nam P. Nguyen, Thang N. Dinh, Ying Xuan, and My T. Thai. 2011b. Adaptive algorithms for detecting community structure in dynamic social networks. In *Proceedings of the 30th International Conference on Computer Communications (INFOCOM'11)*. IEEE, Los Alamitos, CA, 2282–2290.

Gergely Palla, Albert-László Barabási, and Tamás Vicsek. 2007. Quantifying social group evolution. *Nature* 446, 7136, 664–667.

Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. 2005. Uncovering the overlapping community structure of complex networks in nature and society. *Nature* 435, 7043, 814–818.

Leto Peel and Aaron Clauset. 2015. Detecting change points in the large-scale structure of evolving networks. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI'15)*. 2914–2920.

Leto Peel, Daniel B. Larremore, and Aaron Clauset. 2017. The ground truth about metadata and community detection in networks. *Science Advances* 3, 5, e1602548.

Pascal Pons and Matthieu Latapy. 2006. Computing communities in large networks using random walks. *Journal of Graph Algorithms and Applications* 10, 2, 191–218.

Filippo Radicchi, Claudio Castellano, Federico Cecconi, Vittorio Loreto, and Domenico Parisi. 2004. Defining and identifying communities in networks. *Proceedings of the National Academy of Sciences of the United States of America* 101, 9, 2658–2663.

Khairi Reda, Chayant Tantipathananandh, Andrew Johnson, Jason Leigh, and Tanya Berger-Wolf. 2011. Visualizing the evolution of community structures in dynamic social networks. In *Computer Graphics Forum*, Vol. 30. Wiley Online Library, 1061–1070.

A. Rényi and P. Erdos. 1959. On random graphs. *Publicationes Mathematicae* 6, 290–297.

Giulio Rossetti. 2015. *Social Network Dynamics*. Ph.D. Dissertation. Computer Science Department, University of Pisa, Italy.

Giulio Rossetti. 2017. RDYN: Graph benchmark handling community dynamics. *Journal of Complex Networks* 5, 6, 893–912. DOI: http://dx.doi.org/10.1093/comnet/cnx016

Giulio Rossetti, Riccardo Guidotti, Diego Pennacchioli, Dino Pedreschi, and Fosca Giannotti. 2015. Interaction prediction in dynamic networks exploiting community discovery. In *Proceedings of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM'15)*. ACM, New York, NY, 553–558.

Giulio Rossetti, Luca Pappalardo, Dino Pedreschi, and Fosca Giannotti. 2017. Tiles: An online algorithm for community discovery in dynamic social networks. *Machine Learning* 106, 8, 1213–1241.

Giulio Rossetti, Luca Pappalardo, and Salvatore Rinzivillo. 2016. A novel approach to evaluate community detection algorithms on ground truth. In *Complex Networks VII*. Springer, 133–144.

Martin Rosvall and Carl T. Bergstrom. 2010. Mapping change in large networks. *PloS One* 5, 1, e8694.

Stanisław Saganowski, Bogdan Gliwa, Piotr Bródka, Anna Zygmunt, Przemysław Kazienko, and Jarosław Koźlak. 2015. Predicting community evolution in social networks. *Entropy* 17, 5, 3053–3096.

Erin N. Sawardecker, Marta Sales-Pardo, and Luıs A. Nunes Amaral. 2009. Detection of node group membership in networks with group overlap. *European Physical Journal B: Condensed Matter and Complex Systems* 67, 3, 277–284.

Nico Schlitter and Tanja Falkowski. 2009. Mining the dynamics of music preferences from a social networking site. In *Proceedings of the International Conference on Advances in Social Network Analysis and Mining (ASONAM'09)*. IEEE, Los Alamitos, CA, 243–248.

Jiaxing Shang, Lianchen Liu, Feng Xie, Zhen Chen, Jiajia Miao, Xuelin Fang, and Cheng Wu. 2014. A real-time detecting algorithm for tracking community structure of dynamic networks. arXiv:1407.2683.

Junming Shao, Zhichao Han, and Qinli Yang. 2014. Community detection via local dynamic interaction. arXiv:1409.7978.

Jianbo Shi and Jitendra Malik. 2000. Normalized cuts and image segmentation. *Transactions on Pattern Analysis and Machine Intelligence* 22, 8, 888–905.

Jimeng Sun, Christos Faloutsos, Spiros Papadimitriou, and Philip S. Yu. 2007. 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, 687–696.

Yizhou Sun, Jie Tang, Jiawei Han, Cheng Chen, and Manish Gupta. 2014. Co-evolution of multi-typed objects in dynamic star networks. *Transactions on Knowledge and Data Engineering* 26, 12, 2942–2955.

Yizhou Sun, Jie Tang, Jiawei Han, Manish Gupta, and Bo Zhao. 2010. Community evolution detection in dynamic heterogeneous information networks. In *Proceedings of the 8th Workshop on Mining and Learning With Graphs*. ACM, New York, NY, 137–146.

Lionel Tabourier, Anne-Sophie Libert, and Renaud Lambiotte. 2016. Predicting links in ego-networks using temporal information. *EPJ Data Science* 5, 1, 1.

Mansoureh Takaffoli, Farzad Sangi, Justin Fagnan, and Osmar R. Zaïane. 2011. MODEC-modeling and detecting evolutions of communities. In *Proceedings of the 5th International Conference on Weblogs and Social Media (ICWSM'11)*. 30–41.

Biying Tan, Feida Zhu, Qiang Qu, and Siyuan Liu. 2014. Online community transition detection. In *Proceedings of the International Conference on Web-Age Information Management*. 633–644.

Lei Tang, Huan Liu, Jianping Zhang, and Zohreh Nazeri. 2008. Community evolution in dynamic multi-mode networks. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, 677–685.

Chayant Tantipathananandh, Tanya Berger-Wolf, and David Kempe. 2007. A framework for community identification in dynamic social networks. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, 717–726.

Minh Van Nguyen, Michael Kirley, and Rodolfo García-Flores. 2012. Community evolution in a scientific collaboration network. In *Proceedings of the Congress on Evolutionary Computation (CEC'12)*. IEEE, Los Alamitos, CA, 1–8.

Corinna Vehlow, Fabian Beck, Patrick Auwärter, and Daniel Weiskopf. 2015a. Visualizing the evolution of communities in dynamic graphs. In *Computer Graphics Forum*, Vol. 34. Wiley Online Library, 277–288.

Corinna Vehlow, Fabian Beck, and Daniel Weiskopf. 2015b. The state of the art in visualizing group structures in graphs. In *Proceedings of the Eurographics Conference on Visualization (EuroVis'15)-STARs*, Vol. 2.

Corinna Vehlow, Fabian Beck, and Daniel Weiskopf. 2016. Visualizing dynamic hierarchies in graph sequences. *IEEE Transactions on Visualization and Computer Graphics* 22, 10, 2343–2357. DOI: http://dx.doi.org/10.1109/TVCG.2015.2507595

Tiphaine Viard, Matthieu Latapy, and Clémence Magnien. 2016. Computing maximal cliques in link streams. *Theoretical Computer Science* 609, 245–252.

Yi Wang, Bin Wu, and Xin Pei. 2008. CommTracker: A core-based algorithm of tracking community evolution. In *Proceedings of the 4th International Conference on Advanced Data Mining and Applications (ADMA'08)*. 229–240.

Duncan J. Watts and Steven H. Strogatz. 1998. Collective dynamics of small-world networks. *Nature* 393, 6684, 440–442.

Jierui Xie, Mingming Chen, and Boleslaw K. Szymanski. 2013a. LabelRankT: Incremental community detection in dynamic networks via label propagation. In *Proceedings of the Workshop on Dynamic Networks Management and Mining*. ACM, New York, NY, 25–32.

Jierui Xie, Stephen Kelley, and Boleslaw K. Szymanski. 2013b. Overlapping community detection in networks: The state-of-the-art and comparative study. *ACM Computing Surveys* 45, 4, 43.

Jierui Xie and Boleslaw K. Szymanski. 2013. LabelRank: A stabilized label propagation algorithm for community detection in networks. In *Proceedings of the 2nd Network Science Workshop (NSW'13)*. IEEE, Los Alamitos, CA, 138–143.

Hao Xu, Zhenwen Wang, and Weidong Xiao. 2013a. Analyzing community core evolution in mobile social networks. In *Proceedings of the International Conference on Social Computing (SocialCom'13)*. IEEE, Los Alamitos, CA, 154–161.

Hao Xu, Weidong Xiao, Daquan Tang, Jiuyang Tang, and Zhenwen Wang. 2013b. Community core evolution in mobile social networks. *Scientific World Journal* 2013, 781281.

Kevin S. Xu and Alfred O. Hero. 2014. Dynamic stochastic blockmodels for time-evolving social networks. *Journal of Selected Topics in Signal Processing* 8, 4, 552–562.

Jaewon Yang and Jure Leskovec. 2014. Structure and overlaps of ground-truth communities in networks. *Transactions on Intelligent Systems and Technology* 5, 2, 26.

Jaewon Yang and Jure Leskovec. 2015. Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems* 42, 1, 181–213.

Tianbao Yang, Yun Chi, Shenghuo Zhu, Yihong Gong, and Rong Jin. 2009. A Bayesian approach toward finding communities and their evolutions in dynamic social networks. In *Proceedings of the International Conference on Data Mining*. 990–1001.

Tianbao Yang, Yun Chi, Shenghuo Zhu, Yihong Gong, and Rong Jin. 2011. Detecting communities and their evolutions in dynamic social networks—a Bayesian approach. *Machine Learning* 82, 2, 157–189.

Anita Zakrzewska and David A. Bader. 2015. A dynamic algorithm for local community detection in graphs. In *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM'15).* ACM, New York, NY, 559–564.

Ding Zhou, Isaac Councill, Hongyuan Zha, and C. Lee Giles. 2007. Discovering temporal communities from social network documents. In *Proceedings of the 7th International Conference on Data Mining (ICDM'07).* IEEE, Los Alamitos, CA, 745–750.

Received October 2016; revised November 2017; accepted December 2017