TEMPO

Management and Processing of Temporal Networks

H.F.R.I. Project No. 03480

D4.1: State of the Art in Temporal Community Detection

Computer Engineering & Informatics Department
University of Patras
Greece
01/12/2022

State of the Art in Temporal Community Detection

Kostas Christopoulos and Kostas Tsichlas

December 01, 2022

Abstract

Community detection is a prominent process on networks and has been extensively studied on static networks the last 25 years. This problem concerns the structural partitioning of networks into classes of nodes that are more densely connected when compared to the rest of the network. However, a plethora of real-world networks are highly dynamic, in the sense that entities (nodes) as well as relations between them (edges) constantly change. As a result, many solutions have also been applied in dynamic/temporal networks under various assumptions concerning the modeling of time as well as the emerging communities. The problem becomes quite harder when the notion of time is introduced, since various unseen problems in the static case arise, like the identity problem. In the last few years, a few surveys have been conducted regarding community detection in time-evolving networks. In this survey, our objective is to give a rather condensed but up-to-date overview, when compared to previous surveys, of the current state-of-the-art regarding community detection in temporal networks. We also extend the previous classification of the algorithmic approaches for the problem by discerning between global and local dynamic community detection. The former aims at identifying the evolution of all communities and the latter aims at identifying the evolution of a partition around a set of seed nodes. Moreover, we also discuss community detection methods tailored to distributed systems that can handle massive graphs.

Keywords: Temporal Graphs, Community Detection

1 Introduction

Networks are widely used as a method for analyzing data in many scientific fields, such as social sciences, transportation and biology. The process of community detection (henceforth also referred as CD) that has its origins in graph partitioning, is concerned with node intra-connectivity and its goal is to identify highly linked groups (communities) of nodes. For example, finding clusters of users in social networks and functional protein complexes in bioinformatics networks are two widely used applications of this problem.

In general, a static network is represented as $G = (V, E)$, where V is the set of vertices (entities) and E is the set of edges (interactions/relations between entities). An edge can be directed, such as the connection between two people where one sends an email to another or undirected, such as the connection between two collaborating peers. Lastly, edges among nodes can be associated with weights (e.g., frequency of interactions) or nodes can be associated to weights (e.g., specific properties of nodes). In many cases, real-world networks are dynamic, in the sense that new edges or nodes appear and existing edges or nodes disappear. As a result, the communities themselves change because of the evolution of the network. The appearance of a new community, the disappearance or the split of an existing community, are examples of changes in the community structure.

Generally, the representation of a temporal network can either be based on a sequence of static graphs (snapshots) possibly with logs between them or as a network with time annotations on its nodes/edges that represent its time evolution. The former approach requires, among others, the specification of the time window that defines the time instances of snapshot construction. The latter, is related to events, like edge/node insertion or deletion or their existence interval. The notion of a time annotation may have different aspects/interpretations depending on the application. The following three aspects have been used in the literature [60]:

1. *Point Networks:* every link among two vertices x and y, which has been created at certain time t, can be represented as a triplet $e = (x, y, t)$.

2. *Time Interval Networks:* the time-interval connection of two nodes is represented as a quadruplet $e = (x, y, t, \Delta t)$. Δt is the duration of the link between the vertices x and y.

3. *Incremental Networks:* edges/nodes can only be added and deletions are forbidden.

It is worth mentioning that multilayer networks [47] can be used for dynamic community detection as well. In particular, a multilayer network is a network made of multiple networks, called layers, where each layer has the same number of nodes, but different edge connections. The multilayer network model is commonly employed in the study of temporal networks, in which each snapshot is represented as a layer and all layers are interconnected based on their time relationship.

Working with big data in a centralized system e.g., personal computer, is very difficult. Indeed, dealing with networks that consist of billions/trillions entities and edges is rather impossible in a single system. For instance, the number of Facebook active users worldwide is more than 2.6 billions. Consequently, as the data are becoming bigger, the need for computing power is also increasing. One potential issue is the scaling-up process's limitation. At some point, by providing more resources to the server will not improve analogously its performance. Another disadvantage is that the server acts as a single point of failure. As a result, if the server fails, the entire service goes down.

To deal with these issues, centralized systems are shifted to distributed, decentralized systems e.g., a peer to peer network. An apparent advantage of adopting a distributed system is eradicating the single point of failure compared to centralized computing. For instance, a peer can fail while the service is still available. This means that other peers can inform the requesting entities that a particular peer failed and/or take over the task of the failed peer. One more advantage of utilizing distributed systems is that we can add extra computational resources to the entities in the system in order to scale-up and/or new entities in order to provide new services in the system.

In the last few years, many surveys in the community detection field in temporal networks have been published. These are discussed briefly in Section 2. The basic notions on temporal networks are discussed in Section 3. The main contribution of this report is in Section 4, which can be summarized as follows: 1) we provide an updated overview of the current state-of-the-art methods for CD in temporal networks, since the last years there are quite a few new related results, and 2) we further classify the approaches in global and local, distributed and non-distributed community detection methods. The local CD concerns the discussion on new methods related as to how a community around a given set of nodes evolves in time. This approach is appropriate in cases where one is not interested in discovering all communities, leading to large efficiency and effectiveness gains. Furthermore, commonly used datasets in the related literature are presented in Section 5. Finally, we conclude in Section 6.

2 Related Work

In this section we discuss existing surveys on CD in temporal networks. In [12], a general classification of methods is proposed into two classes: 1) *Online* (real time, incremental detection) and 2) *Offline* (prior knowledge of network changes). Similarly, in [42], authors identify the same two classes but they focus on online approaches dividing them into two sub-classes: 1) *Temporal Smoothness*, where at each snapshot a static CD algorithm is run from scratch and 2) *Dynamic Update*, where the communities are updated based on the differences of two consecutive snapshots.

Two very interesting surveys are [79, 22], where the dynamic CD algorithms have been classified based on the strategy they use for detecting meaningfull evolving communities. They propose three classes of approaches: 1) *Instant - optimal*, where the algorithms detect communities from scratch at each snapshot and then match them between consecutive snapshots, 2) *Temporal Trade-off*, where the algorithms detect communities comparing the topology of two consecutive snapshots and 3) *Cross-Time*, where the algorithms discover communities using the information of all snapshots. In the same manner, in [6], the authors identify three similar categories as well: 1) *Two-Stage methods* that detect the communities from scratch at each snapshot and then match them across different snapshots, 2) *Evolutionary Clustering* that detect communities based on the changes in the topology between two consecutive snapshots and 3) *Coupling Graph* that creates an aggregate network containing all snapshots and then uses a static CD algorithm.

In [26] the authors classified the dynamic CD algorithms into four classes: 1) *Independent Detection*, where the communities are detected from scratch at each snapshot and then they are matched

among consecutive snapshots, 2) *Dependent Detection*, where communities are identified based on the changes of the topology between two consecutive snapshots, 3) *Simultaneous Detection*, where communities are detected by using the information from all snapshots and 4) *Dynamic Detection*, where the communities are updated based on the network updates. Additionally, in [36] four classes of evolving clustering methods are provided that are similar to the preceding classification: 1) *Sequential mapping-driven*, 2) *Temporal smoothing-driven*, 3) *Milestone detection-driven* and 4) *Incremental adaptation-driven*.

In [16], a survey is conducted exclusively for incremental (online) CD methods in temporal networks. The proposed classification contains two subcategories of incremental methods: 1) *Community Detection in Fully Temporal Networks*, where insertions and deletions of nodes and edges are permitted and 2) *Community Detection in Growing Temporal Networks*, where only insertions of nodes and edges are permitted. Authors in [9], in order to deal with the problem of local community detection, implement and employ a number of existing local community detection algorithms in dynamic networks. The results from the experiments, which are conducted using the LFR synthetic dynamic network generator, help to analyze the weaknesses and strengths of the existing algorithms and subsequently to develop an efficient algorithm. The most recent survey in local community detection in both static and dynamic networks is described in [10]. Among others, authors propose useful tools (GUIs or scripting/programming languages) that are used in the detection of local communities.

Finally, regarding the distributed systems, we mention a survey based on vertex-centric frameworks [65], in large scale networks, that uses the Think Like A Vertex paradigm (TLAV). It is a type of framework that implements user defined programs from the perspective of a vertex rather than a graph. This method enhances locality, exhibits linear scalability, and provides a simple way to define and calculate various iterative graph algorithms, community detection among others. In [7], researchers analyze a number of distributed algorithms that are applied to detect communities in both static and dynamic networks. Authors discuss extensively the definition of the concept of local communities and a variety of algorithms in order to detect them. They segregate them based on the structural and semantic properties of the graph, stating methods such as: LP algorithm, LabelRank, LabelRankT, LPA on graphs, Dynamic Planted Bisection Model based algorithms, Dynamic Bayesian Nonnegative Matrix Factorization (DBNMF) and Huang and Yang's approach to semantic community clustering.

3 Basic Notions on Temporal Graphs

In this section we discuss preliminary notions related to temporal graphs. We discern mainly the following issues: a) A formalism for temporal graphs, b) The notion of time and c) The model of computation.

3.1 Unified Frameworks

There is a plethora of results on temporal graphs related to concepts, formalisms as well as algorithms. Research has been done on how to unify all these under a common framework.

Time-Varying Graphs (TVGs) [20] are one such notable effort to accomplish this unification. Temporal systems can be described by a TVG $\mathcal{G} = (V, E, \mathcal{T}, \rho, \zeta)$ as follows:

V : A set of entities (nodes)

E : A set of relations between entities as well as an alphabet L accounting for a set of properties that this relation could have; that is $E \subseteq V \times V \times L$.

$\mathcal{T}$: All these relations are assumed to hold over a time lifespan $\mathcal{T} \subseteq \mathbb{T}$, where the temporal domain $\mathbb{T}$ is assumed to be $\mathbb{N}$ for discrete systems or $\mathbb{R}$ for continuous systems.

ρ : is the presence function $\rho : E \times \mathcal{T} \to \{0, 1\}$, which indicates whether a given edge is valid at some specific time instance (a similar function can be defined for nodes).

ζ : is the latency function $\zeta : E \times \mathcal{T} \to \mathbb{T}$, which indicates the time it takes to cross an edge given the starting time of the traversal (a similar function can be defined for nodes).

In [20] it is shown how TVGs can cope with the notion of time in various network settings. For example, the latency functions can be used to define journeys on such a temporal network. In a journey we require a path from a node to another node such that all time constraints are satisfied.

Stream Graphs and Stream Links [55] is another more recent unified framework for reasoning on temporal graphs. Note that streaming is not related to the notion of streaming as a model of computation.

A simple undirected **stream graph** $S = (T, V, W, E)$ is defined as follows:

V : set of nodes.

T : set of time instants (it can be discrete or continuous)

W : a set of temporal nodes $W \subseteq T \times V$

E : a set of links $E \subseteq T \times V \otimes V$ [1]

In case where there are no temporal nodes we get a stream link. We define by T_v and T_e the set of instances where the node v and the edge e are present in the graph. In [55] there is an extensive discussion about how stream graphs can be used as a formalism to transfer various notions of static graphs to temporal graphs.

3.2 The Notion of Time

The addition of the time dimension in graphs gives rise to temporal graphs. The notion of time may have different aspects/interpretations depending on the application. The following aspects in order of generality can be used:

1. **Transaction Time:** represents the time that an event takes place (i.e., the moment that an edge is inserted or deleted from the graph). In the transaction time setting, updates can only occur in an append-like manner (i.e. an update changes the most recent version).

2. **Valid Time:** it signifies the time period in which an object is valid [17, 18] (i.e. the time interval that an object existed in the graph). Transaction time can be emulated in the valid time setting by restricting updates to intervals that begin on the time moment of the update. Valid time is the time period during which a fact is true in the real world.

3. **Multiple Universe Valid Time:** assumes that the history has a tree-like shape, instead of a linear structure. This is reminiscent of the notion of full persistence in data structures [28], in the sense that the history of the data structure is fully characterized by a tree structure. Similarly, since history is not linear, we require its explicit representation by a history tree. Instead of talking about time that implies a linear evolution we now talk about versions of graphs. New version instances are created by making updates to existing versions of the history tree. For example, let a node v of the history tree correspond to the graph of version v. An update at version v gives a new instance that is represented by node u (with version u) that is a child of v in the version tree. We refer the interested reader for a more detailed analysis to [28]. The crucial point is that navigation in history requires the efficient support of nearest common ancestor queries on the history tree. In this way, searching for a version v in a node is equivalent to finding the version u that exists in this node and is the nearest ancestor among all versions in the node of v. By employing an auxiliary structure for such a tree-like history one can answer efficiently such nearest ancestor queries without further changes to the rest of the structure. Apparently, a special case of this notion of time is the previously mentioned valid time where the history tree degenerates to a single path. We could further generalize this notion, e.g., by allowing merges of graphs of different versions.

[1] $\otimes$ corresponds to an unordered pair of elements.

4 Detecting Communities in Temporal Networks: Classification

In this section, a classification of dynamic CD methods is provided. Firstly, the different versions of the dynamic CD problem are discerned into two categories: *Global* and *Local*. The former, concerns the identification of all communities and their evolution in the temporal network, while the latter concerns the identification of a community around a given set of seed nodes and its evolution in the temporal network. This corresponds to the division between global and local CD in static graphs. The main body of the literature concerns the global version of the problem; however, there is recently an admittedly small number of publications on the local version. Although the local version uses techniques especially from the global online dynamic CD category we believe that it constitutes a class by itself since there are many differences in terms of efficiency as well as effectiveness of the methods. We further divide this category into two: distributed and non-distributed methods. Thus we identify the following 6 classes of methods:

1. Global:
   (a) Community Detection from Scratch and Match
   (b) Dependent or Temporal Trade - off Community Detection
   (c) Simultaneous or Offline Community Detection
   (d) Online Community Detection in fully Temporal Networks and in Growing Temporal Networks
2. Local:
   (a) Non-distributed Community Detection in Temporal Networks using Seed Nodes
   (b) Distributed Community Detection in Temporal Networks in a vertex-centric way

In the following, we present in detail these classes by discussing recent representative methods.

4.1 Global Techniques in Community Detection in Temporal Networks

4.1.1 Community Detection from Scratch and Match

In this class, a static CD algorithm is applied on each snapshot from scratch and then the communities that have been found at snapshot $t+1$ are matched (by using a similarity metric like Jaccard similarity) with the communities found at snapshot t. The advantage is that communities can be detected in parallel and existing methods for static CD can be used. On the other hand, instability (the communities may have a lot of changes between consecutive snapshots) and inefficiency (in each snapshot a static community detection algorithm is invoked) are its main two drawbacks. In the following, we discuss some representative methods of this category (more such methods can be found in [89]).

In [71], an algorithm for community detection is provided that uses an iterative similarity-based approach. It is applied on general weighted networks and a streaming scenario is assumed with respect to snapshots. In each snapshot, CPM [72] is used to find the community partition. The user needs to determine the size of the clique (k-clique) as well as a weight threshold W (links weaker than W are ignored) in order to define the cliques. Then, the identification of evolving communities is realized, for different time steps $t, t+1$. Firstly, for these consecutive time-steps a joined graph is constructed which is composed of the links of both graphs. The communities in the united graph may grow, merge or remain unchanged. In general, if a community in the constructed graph contains a single community from time steps t or $t+1$, then they are matched. Otherwise, when the joined graph contains more than one community, then the matching is performed in descending order (death or creation of a community) of their relative node overlap. There is one significant observation regarding the lifetime of the communities. The authors state that the survival key for large communities is by taking part in merging processes. On the contrary, for small communities the stability is guaranteed.

In [24], authors track the evolution of community and graph representatives. This is a method that is based on independently discovering communities in each snapshot and then matching them.

Table 1: Overview of the proposed temporal community detection classification.

Global Temporal Community Detection		
Class	Description	References
From Scratch and Match	Static algorithm at each snapshot and matching	[89, 71, 24, 66, 29, 52]
Dependent or Temporal Trade-off	Based on the topology of two adjacent snapshots	[84, 81, 83, 39, 108, 102, 58, 100, 94, 37, 61]
Simultaneous or Offline	Creation of single graph-run static algorithm on it	[88, 63, 34, 50, 64, 35, 44]
Online Community Detection	Update in proportion to network modifications	[80, 57, 105, 106, 21, 70, 69, 74, 93, 97, 56, 109, 25, 13, 41, 91, 90, 86, 87, 15, 5, 110, 2, 107]
Local Temporal Community Detection		
Class	Description	References
Using Seed Nodes	Update only the area around the seed node	[68, 27, 49, 40, 23, 96, 76, 32, 62, 11, 33, 46, 101, 59, 43, 104, 103]
Distributed Community Detection	Vertex-centric approach to graph processing	[82, 48, 73, 8, 3, 1, 67, 51, 30, 31]

Given as input a sequence of different time steps of the graph, initially the graph representatives are found, based on the common nodes between two different time steps of the graph, and then the communities are listed. Then, the community representatives are identified (the minimum number of presences of a node in different communities of the same graph). Consequently, the relation of communities between different time steps (G_t, G_{t+1}, G_{t+2}) is established by finding the predecessors and successors for each community (utilizing the community representatives). Finally, the dynamic community detection is performed and six events (Grown, Merged, Split, Shrunken, Born and Vanished) may happen. For all the events, we need to track forward the graph sequence, except for the cases of shrunken and split communities, where a backtracking process is applied. For evaluation purposes, this algorithm was compared to another version of it without graph or community representatives. The experiments shown that the proposed version was 11 to 46 times slower because each community of the current graph should be compared with all the detected communities from the next time step.

In [66] the authors use sliding windows to track the dynamics. They compute partitions for each time slice and they modify the community description at time t using the structures found at times $t-1$ and $t+1$. More specifically, the data set is divided into time windows and for each one a known static algorithm is used. Then, all similarities between communities at times $t-2, t-1, t, t+1, t+2$ are computed. Thus, for each community at time t, the ancestor, predecessor, successor and grandchild are defined. The above information makes it easy to distinguish noise from real evolution. Consequently, this information is used to smooth out the community evolution. Then, communities that appear to be unduly split by the independent community detection are merged, and communities that appear to be artificial merges are separated. At the end of the procedure, a description of the network evolution is obtained.

An interesting work based on an independent method is proposed in [29]. In each network snapshot, the Ensemble Louvain algorithm is utilized, which is an extension of Louvain. In order to measure the community similarity between to consecutive snapshots, the Bcubed measure is used. Lastly, by giving the notion of internal and external influence, this method can also identify super communities.

A recent paper in dynamic community detection is presented in [52]. The authors divide the problem of tracking the community evolution into three steps. First, they define the time window (time step); second, in each time window the communities are detected. Finally, they use for each time window a hash-map ((key,value) pair) technique in order to store the entity affiliations and

similarity lists in order to track the community evolution between different time windows.

4.1.2 Dependent or Temporal Trade-off Community Detection

Methods in this class process repeatedly network changes. First, by using a static CD algorithm they find partitions for the initial state (first snapshot) of the network, and then they identify communities at snapshot t by using information from both the current snapshot (t) and previous snapshots ($< t$). These methods don't suffer from the instability problem and are faster than those from the previous category. On the other hand, its two main drawbacks are the avalanche effect [2] as well as the fact that these methods are generally not parallelizable. Global and multi-objective optimization methods are the most common subcategories in this class.

In [84] an efficient real-time, incremental modularity-based algorithm is proposed for tracking community structures of dynamic networks. A two-step approach is adopted. First, the BGL static algorithm [14] is applied in order to obtain an initial community structure and then incremental updating strategies are applied in order to track the dynamic communities. Initially, each node is placed in a separate community and the first phase of the BGL algorithm starts. BGL checks each node as to whether its transfer to one of its neighbor communities could incur a modularity increase. If this is the case, then the node is moved to the community that gives the highest modularity gain. When this phase is completed for all nodes, then each community is converted to a single node and the algorithm applies iteratively the same procedure and terminates when the modularity is lower than a given threshold. At the next step, the proposed incremental algorithm takes action. When an edge is updated, four different cases are discerned based on the type of the respective nodes.

1. *Inner community edge:* the two nodes incident to the edge already exist and belong to the same community.
2. *Cross community edge:* the two nodes incident to the edge already exist and belong to different communities.
3. *Half-new edge:* one of the nodes incident to the edge is new.
4. *New edge:* both of the nodes incident to the edge are new.

Based on these four types, the communities may be updated as follows: 1) The community remain unchanged; 2) Merge two communities in one; 3) Assign the nodes to an existing community and 4) Assign the new nodes to a new community. The operation must be chosen so that the modularity has the largest gain or the less loss in case no increase in the modularity is possible. The greatest strength of the algorithm is its low computing complexity. The algorithm is compared with both CNM and BGL and the experimental results show that the proposed incremental algorithm has reasonably good performance on both modularity and computing time.

Similarly, in [81], four possible edge transitions take place and the dynamic algorithm CRep aims to maximize the time-dependent log-Likelihood. This is achieved by utilizing an iterative process. In this process, interchangeably either the parameters are kept fixed or the probability distribution of edge transitions is kept fixed. The process iterates until the log-Likelihood converges.

A modified Louvain (C-Blondel) method is proposed in [83]. The historical information from the previews snapshot $(t-1)$ and the current one (t) communities are used in order to construct the compressed graph G^H, which contains more information compared to the graph of the current time step while having smaller size at the same time, reducing the computational complexity of the algorithm. In this method the notion of the destructive node is introduced. During the construction of the compressed graph, when a destructive node disappears a community collapse happens.

The method introduced in [39] detects the evolutionary community structure in weighted dynamic networks. Weighted networks are considered, whose number of nodes, edges and communities are different from time-step t to time-step $t+1$. The main steps of this method are the following. First, for each time-step they detect the initial community (using the input matrix that considers the previews community structure) and then they expand the community. This is iterated until all nodes have been assigned to a community. Finally, the communities are merged. To trace the initial community the input matrix $U_{ij} = (u_{ij}^t)_{N^t \times N^t}$ should be calculated:

[2] The avalanche effect describes the phenomenon when communities can experience substantial drifts compared to what a static algorithm would find on the static network at a particular time instance.

$$u_{ij} = \begin{cases} w_{ij}^t \text{ if } t = 1 \\ (1-\alpha)w_{ij}^t + \alpha u_{ij}^{t-1}\delta_{ij}^{t-1} \text{ if } t \geq 2 \end{cases}$$

where w_{ij} is the adjacency matrix, threshold $\alpha \in [0, 1]$, δ_{ij}^{t-1} is equal to 1 when nodes i and j belong to the same community and 0 otherwise. As it can be seen from this piece-wise function, the community structure remains faithful to the current data and it is expected that it will not shift a lot from one time step to the next. Then, by using the input matrix, the belonging degree $BD^t(u_i^i, C_p^t)$ is calculated and the initial community is found. The greater the belonging degree, the more the node belongs to the community. At the next step, the nodes (one by one) are attached to the community C_p^t (community C at time t) that provides the highest modularity gain. In the last step, When all the communities have been identified, they are merged given that the modularity gain of the new community is positive than when the two communities are separate. Regarding the experimental evaluation, the algorithm is tested on a synthetic dataset and two real network datasets. Three quality functions have been used (snapshot, history and total quality function). According to the experimental results, the algorithm works effectively in dynamic weighted networks, identifying successfully the evolutionary community structure. It is mentioned that its main drawback is the difficulty to handle networks that change extremely fast (churn).

The CCPSO (Consensus Community Particle Swarm Optimization) approach is proposed in [108]. In this paper, the optimal population from the previous time step (intrapopulation) is compared to the population of the current time step, using the high support rate. The subset of the same communities between these two consecutive time steps are recognized as consensus communities (interpopulation). In addition, two objective functions, KKM (kernel k-means) and RC (Ratio Cut), have to be optimized. A similar approach is given in [102], where they utilize different objective functions and an improved version of the PSO method.

Another multi-objective optimization method is described in [58]. In its ititialization step, this approach uses the probability fusion method in order to produce high quality partitions. The crowd and diversity neighbor fusion strategies are used in order to obtain a better partition and to avoid local optima. Finally, modularity and DCEC metrics were used for the purpose of evaluating the objective function.

Another representative algorithm of this subclass is proposed in [100]. The DSBM (Dynamic Stochastic Block Model) algorithm, which is the extended version of SBM [45], detects both the structure of communities and their evolution in a consolidated probabilistic model. The model has online (the model is updated repetitively over time) and offline (the model is trained with data from all time steps) learning versions. The model is based on the Beyesian treatment. Many approaches, in order to estimate the unknown parameters, compute the most credible values. Instead, the DSBM algorithm calculates the prior distribution of the parameters. Assuming that nodes remain unchanged, then in the first snapshot ($t = 1$), the network is generated by applying the SBM. Each node is assigned to a community k with probability π_k and the link between two nodes is generated by a Bernoulli distribution with parameter $P_{k\ell}$ (link between one node from the k community to one node of the ℓ community). For the following time steps, one more parameter $A_{k\ell}$ (transition matrix) is introduced. This parameter estimates the probability that a node will be assigned to the same community, as in the previous time step, or to a new one. The conjugate priors for the three parameters π , P and A are introduced and then by using the Gibbs sampling method (inference algorithm) the posterior probabilities are optimized in order to decide the community assignments of all the nodes (simultaneously) in offline learning and each time (incrementally) in online learning. Furthermore, some extensions are proposed in order to handle different type of links and insertion/deletion of nodes. The experimental evaluation on synthetic and real datasets verified that the DSBM when compared to baseline algorithms (SSBM, SGFC and SSpect), converges very quickly, with many meaningful community changes. The effectiveness of the algorithm, when ground truth is available, is measured by using the Normalized Mutual Information (NMI - cost associated to the current snapshot), or by using modularity (the smoothness with respect to the past history).

[94] proposes the DYN-OPGPBA algorithm. This method works in two stages. First, in order to obtain the optimal snapshot quality, they optimize the modularity metric and for the second stage, in order to obtain both the optimal snapshot quality and temporal cost, they optimize the medularity and NMI. Finally, two recent methods are discussed in [37, 61], both uses deep learning

methods. The former is based on generative adversarial networks while the latter is based on RNN networks.

4.1.3 Simultaneous or Offline Community Detection

Methods in this class discover partitions by considering all states of the temporal network at the same time. A single multilayer network is created from all snapshots using edges based on the relationship between nodes at the same snapshot and at adjacent (preceding and succeeding) snapshots. Then, the communities are detected by using an appropriately modified static algorithm on the multilayer network. Methods in this category don't suffer from instability and the avalanche effect. On the other hand, they have certain limitations: they usually require the assumption of a fixed number of communities and they lack a mechanism to determine operations between temporally successive partitions (like merge of communities). In the following, we discuss some representative methods of this category.

GraphScope [88] is a parameter-free algorithm, and was proposed to address the problems of community detection in a bipartite graph (source and destination partition associated to each other) and change detection (a change of community structure) simultaneously. This method is based on MDL [77] (Minimum Description Length) principle, where the objective function is used to calculate the number of bits we need to encode the full graph stream. First, they divide the problem into two subproblems: 1) Assuming that a graph segment $G^{(s)}$ (set of similar graphs between timestamps) is given, they discover the partitions and 2) they detect changes in the graph stream. Given the graph segment, by using the modified entropy (the lower the encoding cost the better), they determine the source and destination partition by merging or splitting the communities. Then, by finding the encoding cost of assigning a node to a particular partition, the algorithm considers assigning each source/destination node to the source/destination partition that yields the smallest encoding cost. On the other hand, when the graph segments are not known, the GraphScope algorithm constructs them incrementally, by comparing the union of the segment and the newly arrived snapshot with this new snapshot, and finding the encoding cost for both of them. Then it decides if the new snapshot stays with the current segment or a new segment starts with this snapshot. Several real and large graph datasets are used for the its experimental evaluation. The experimental results show that the source and destination groups forms meaningful clusters, and the groups evolve over time and time segments indicate interesting change-points. Regarding the quality and scalability of two initialization steps, Fresh-Start and Resume, are compared. The results show that Resume is much faster than fresh-start, especially for large graphs.

[34] proposes a community detection model for temporal networks, where nodes can interact (inter/intra-connectivity) during a time interval. However, it is assumed that there is a fixed number of communities and that they don't change with time. In this method, a streaming scenario is assumed with respect to snapshots and the data is represented by a rank-3 tensor. The Non-negative Tensor Factorization Approach (NTFA) is utilized in order to identify the structures that correspond to communities. The non-negativity makes the resulting factorization (decomposition) more interpretative. This is an optimization problem and we approximate it by minimizing the distance (Frobenius norm) of the tensor and the product of the three factors (PARAFAC Decomposition) as follows:

$$min_{A,B,C}\|T - A, B, C\|_F^2 \tag{1}$$

where A and B represent the structure of the community and C the temporal activity. However, in order to reduce the dimension from three to two, they convert the tensor to three matrices:

$$X_{(1)}, X_{(2)}, X_{(3)}$$

. Applying the PARAFAC decomposition in each of these three matrices, three minimization problems are defined like the one defined in Equation 1. These are solved by utilizing the non negative Alternate Least Squares method. The method is evaluated by using the Core Consistency metric. Comparing the NTFA to four other static algorithms, they observed that NTFA has the same performance as INFOMAP and Community WALKTRAP. On the other hand, the OSLOM and Louvain algorithms merge small communities into larger and the number of detected communities is not the same like NTFA.

Another recent study is described in [50]. Given a multi-layer network, authors apply a random walk-based approach in order to calculate the visiting probabilities and to construct the matrix W (by merging the matrices of different layers). At the end, the Leiden algorithm [92] is applied on matrix W in order to obtain the partition of the network.

In [64] an interesting method is proposed. The PPSBM (Poisson Process Stochastic Block Model) is utilized in order to detect communities in dynamic networks. The connection characteristics of partitions is defined by a time function. The EM (Expectation Maximization) is introduced to detect the nodes affiliation and the time function that is suitable for the studied network. Lastly, it is assumed that the node affiliations are fixed.

A significant study of the fundamental limits of detecting community structure in dynamic networks can be found in [35]. In particular, the authors are trying to analyze the boundaries of detectability for a Dynamic Stochastic Block Model (DSBM) where nodes affiliations can change over time (from one community to other), and edges are created separately at each time step. The method exploits the powerful tools of probabilistic generative models and Bayesian inference, and by utilizing the cavity method, they obtain a clearly defined detectability threshold as a function of the rate of change and the community strength. Below this threshold, they claim that no efficient algorithm can identify the communities. Then, they propose two algorithms that are optimal in the sense that they succeed up to this threshold. The first algorithm utilizes Belief Propagation (BP), which provides asymptotically optimal accuracy, and the second is a quick spectral algorithm, founded on linearizing the BP equations.

An algorithm based on clique enumeration is proposed in [44]. In this case, a well-known recursive static backtracking algorithm, Bron-Kerbosch [19], is adapted to temporal graphs. Finally, another method in this category can be found in [63].

4.1.4 Online Community Detection in Fully Temporal Networks and in Growing Temporal Networks

In these methods, the temporal network is not considered as a sequence of snapshots, but instead as a succession of network transformations. The methods are initialized by discovering partitions at time 0 and then the community structure is updated in each update of nodes/edges. This class of methods is further divided into two main subcategories of incremental methods: 1) *Community Detection in Fully Temporal Networks*, where insertions and deletions of nodes and edges are allowed and 2) *Community Detection in Growing Temporal Networks*, where only insertions of nodes and edges are allowed. In addition, the methods of this class can either handle network updates in batches of arbitrary size. One extreme is to consider batches of size 1, that is after each update the community structure is updated. One advantage of this method is that algorithms for static CD can be used with minor modifications. Moreover, this approach does not suffer from instability, and it is quite efficient since updates for the community structure are usually applied locally. In the following, we discuss some representative methods of this category.

In [80], Tiles is proposed as a method that follows the evolution of communities. The method works in a streaming scenario, treating each topological perturbation as a domino tile fall: whenever a new interaction emerges in the network, Tiles first updates the communities locally, then propagates the changes to the node's neighborhood and finally it modifies the neighbors' community memberships. The online nature of Tiles provides several benefits. First, the network sub-structure computation is local and involves a small number of nodes and communities; thus speeding up the updating process. Second, this method allow us to study two forms of evolutionary behaviors: (1) the stability of individuals' attachments to communities, and (2) the emergence of interaction-based communities through time.

In [57] a method for dynamic community detection is presented that is based on incremental analysis (CDBIA). The main assumption behind this method is that the communities evolve as time goes by and will not abruptly appear or disappear. According to this method, five main operations are applied during the incremental computation: add or delete a node outlier, add or delete a link between two nodes and finally change the community affiliation of a node. These operations are involved in an iterative process until all vertices do not change their community affiliation. More specifically, a vertex belongs to the community that shares the highest number of links. Based on their experimental evaluation, they argued that CDBIA is better than existing algorithms in many cases, with a running time $O(dn)$, where d is the number of operations.

Another dynamic algorithm that adopts an incremental approach, is described in both [105] and [106]. They introduce a filtering technique, which is called "Δ-Screening". During two consecutive time steps, the set of changes are applied and potentially affect the structure of the graph. The technique of Δ-Screening is proposed to capture, in each time step, new inserted/deleted nodes, where their changes (e.g., appearance of an edge), affect the structure of the network, in the sense that nodes change community affiliation. At the beginning ($t = 0$), a static algorithm detects the communities of the initial network. Next, for each time step t, all the added nodes are assigned a new community label. Then, Δ-Screening captures a subset R_t of these nodes. Finally, the static algorithm is evoked to detect the evolution of the communities, visiting only the subset of nodes R_t.

In [21], the algorithm iLCD (intrinsic Longitudinal Community Detection) is presented. This method is an alternative to the Clique Percolation Method (CPM) [72] and it is suitable for addressing the problem of overlapping communities. Receiving as input a sorted set of inserted edges, the algorithm is composed of three steps. First, the existing communities are updated. Then, the algorithm looks as to whether it is possible to create a new community based on the incoming update. Finally, the algorithm looks as to whether it is possible to merge communities. This method supports only addition of new nodes and edges and does not allow deletions.

In [70], the Quick Community Adaptation (QCA) method is proposed, which is based on modularity optimization. It is used for identifying and tracing community structure of dynamic online social networks. The current method has the advantage of rapidly and effectively updating the graph structure, through a series of changes: newVertex, removeVertex, newEdge, and removeEdge, by utilizing the structure related to previous snapshots. In addition, it provides the capability to trace the evolution of its communities as time goes by. In [69], a modified version of [70] is discussed that deals with overlapping communities.

A non-parametric (without the need to assume prior knowledge) probabilistic model that utilizes Bayesian treatment and detect communities in an online incremental manner is proposed in [74]. The IC-DRF model forms dynamic random fields as the graph evolves. These fields define and modify the community structure based. Furthermore, the number of communities is determined dynamically and automatically at any given time, so that the model is able to successfully adjust to global changes in the level of detail of the natural community structure. The Gibbs sampling algorithm is utilized with the purpose of accelerating this approach. Its experimental evaluation show that the proposed method compared to other state-of-the-art algorithms is more effective.

In [93], they use the Louvain algorithm in conjunction with the the concept of the core nodes [95]. Their method detects communities utilizing core nodes (strong clusters or stable memberships). They construct the membership matrix P, applying repeatedly the Louvain [14] algorithm until it converges. Choosing a stable (large) cluster from each community, which is based on the overlapping size with its community, the expansion of the core cluster is performed by using a fitness score function. For the next time steps, the discovered cores are used as an initialization step and the Louvain method is used for the purpose of calculating the new communities. Then, the tracking of community evolution is evoked, based on [38], and the new community structure is compared with the previous one. The experimental results show that there is significant influence on the effectiveness of the method from the temporal variation of the community structure.

An incremental community detection algorithm, LabelRankT, which is based on a label-propagation method called LabelRank [98], is discussed in [97]. In both methods four main operations take place: 1) propagation, 2) inflation, 3) cutoff, and finally 4) conditional update. The difference between these two methods lies on the last operation. In LabelRankT, there is an extra conditional update rule for the involved vertices. Vertices that are transformed among two successive snapshots need to be updated, including instances like add or delete edges from a vertex and deletion or insertion of an existing vertex.

In [56], the objective of the incremental clustering algorithm is to monitor the progress of event patterns. For this purpose, a skeletal graph is utilized to represent the whole network and by using sliding windows all the updates can be tracked. Consequently, a sequence of operations is devised in order to help with the expression of the cluster evolution patterns. Differently from other methods and approaches, this algorithm, called eTrack, handles really well real time incremental bulk updates. For experimental purposes, the method is applied on two real data sets from Twitter with the purpose of tracking the event evolution in social streams. Another similar in spirit method is proposed in [109].

An incremental, modified Louvain algorithm [14] is proposed in [25]. In this method, nodes and

edges are inserted or deleted in the network as time evolves, and the Louvain method is applied only on those communities that are affected. Stability and computational cost are the two main advantages of the method, since the local modularity optimization metric is applied only in a part of network, where the changes are taking place.

A density-based framework, called HOCTracker, is proposed in [13]. For the purpose of monitoring the evolution of overlapping and hierarchical communities in temporal social networks, differently from other methods, this method adjusts a preparatory community structure to the transformations that are happening in a temporal network, and handle systematically only "active" nodes. Furthermore, HOCTracker recognizes all the events (birth, growth, contraction, merge, split, and death) that can occur during the evolution of the community.

In [41], a dynamic community detection algorithm based on distance dynamics is proposed, which is an extended version of the static algorithm in [85]. By utilizing a local interaction model, based on Jaccard distance, and a disturbance factor,the method in [41] overcomes the disadvantages of modularity-based algorithms by detecting small communities or outliers. In addition, regardless of the processing order of the increment set, it can achieve the same community partition results in near-linear time.

An effective, parametric-free and incremental method, based on Matthew effect, is proposed in [91], aiming at detecting dynamic communities. Unlike other incremental approaches, a batch of changes is processed. Between two consecutive snapshots, where a batch of deletions and insertions of nodes and edges takes place, the changed subgraph is extracted by using the notion of node and group attractiveness. Then the changed and unchanged communities are calculated between each two consecutive snapshots. The same dynamic community detection framework, based on information dynamics, is used in [90]. a similar approach is also used in [86].

An incremental density-based approach is proposed in [87]. This method consists of two basic parts: an online and an offline part. In the online part, the algorithm produces the core-connected chain (using the idea of similarity based on core-connectivity), in which the internal partition knowledge of the network can be preserved during the evolution of the network. In the offline part, the extraction of the partitions from the core-connected chain is performed. This part helps the algorithm to react to user queries in an efficient way.

An online version of Dynamic CPM [71] (clique percolation method), using at the same time the LPA [75] (Label Propagation Algorithm), is presented in [15]. The proposed OLCPM (Online Label Propagation Clique Percolation Method) is a two-step framework. First, it uses the Dynamic CPM [71] that updates the communities locally by utilizing a stream model, aiming at improving the time complexity, Then, by using the LPA, the algorithm solves the problem of node affiliations, by allocating each node to one or more partitions. In [5], an incremental method based on the SLPA [99] algorithm is proposed. The ISLPA (Incremental Speaker-Listener Label Propagation Algorithm) can detect overlapping and non-overlapping communities by considering the network modifications between two consecutive time steps. The learning process can be done in an unsupervised or semi-supervised manner.

Another method that uses the static Louvain algorithm for initialization is discussed in [110]. The algorithm, called DynaMo, utilizes six different strategies in order to process a set of network changes (edge/node insertion/deletion). The aim of this method is to maximize the community structure modularity while efficiently handling all incoming updates. In [2], the method CSIMCD is proposed. This method is initialized by extracting the topics of interest, and in each cluster of a specific topic it applies link analysis in order to discover communities. Then, in the incremental part of the method, for each node modification is uses the ACSIMCD (Adaptive CSIMCD) algorithm in order to find the topical cluster of the modified node and then apply Louvain to update the community structure. Finally, an incremental approach based on message distribution and structural and attributes similarities is proposed in [107]. This method applies the Louvain algorithm at the starting graph to obtain the initial communities. Then, considering the modifications in each time step (edge/node insertion or deletion and new messages among nodes that belong to two different partitions), it applies the Louvain algorithm for the affected subgraph or merges the communities if the number of overlapping nodes exceeds a predefined threshold.

4.2 Local Techniques in Community Detection in Temporal Networks

4.2.1 Community Detection in Temporal Networks using Seed Nodes

Given a set of seed nodes Z, our goal is to detect the community that includes Z. The main assumption in this case is that Z is of high importance (e.g., high degree centrality) and act as the community reference point. This problem differs from general temporal CD approaches since our objective is to discover the community defined around the set of seed nodes. This community may not coincide with a community when a global partition is considered. Notice that the online methods described in 4.1.4 are global in the sense that they maintain a partition of the network in communities. The algorithms in this class are very efficient, since by definition it is required from them to maintain a single community by looking only at its neighbor, thus there is no need to access the whole graph.

A dynamic algorithm for local community detection in graphs that solve the problem of seed set expansion is proposed in [103]. This algorithm detects the local community that consists of a set of nodes with high interest. First, a greedy static algorithm is used in order to discover the local community in the starting graph. Initially, the community contains only the seed node and in each iteration one neighbor node is added maximizing the chosen fitness score. At the end of this step, there is a collection of sequences (vertex,interior/border edge sum and fitness score) for each iteration, in an increasing order with respect to the fitness score. Thus, in each iteration the fitness score of the local community always increases. When the fitness score cannot increase any further, the next phase starts. For each stream update of the graph, the algorithm modifies the collection of sequences. If after the modification, the fitness score of a position is higher than the fitness score of the next position, then the node is removed and interior/border edges are modified as well. When this step is finished, the collection of modified sequences is scanned, and if in any position the fitness score is not in an increasing order then the set of collection from this position until the end is removed. Finally, the static algorithm is used one more time in order to add new nodes (community neighbors) to the local community. A full streaming version of seed set expansion method is described in [104]. The seed set expansion dynamic algorithm is tested in three social network graphs and is compared with the static local community detection algorithm. Three metrics (average ratio, precision and recall) were used in order to measure the performance of both algorithms. The results shows that the dynamic method is more efficient as well as more effective in identifying the correct local community of the seeds.

In [43], a hierarchical algorithm is presented. This method discovers communities in temporal networks based on hubs (nodes with large connections) by grouping nodes around them. Each node carries hub information (e.g., distance between nodes, hub and parent nodes, threshold level) and the idea is based on passing this information through the network. Then, the intra-node hubs transfer the information (message) to the outer nodes and in this way, all non-hub elements are assigned to the closest hub. Then, the membership (fuzzy membership) of each node can be calculated. The major advantages of the method is that only a small amount of processing steps have to be done to update the partitions and no predefined parameters are required.

A graph streaming process is described in [59], which is called CoEuS. The authors propose an algorithm in order to detect local dynamic communities by expanding sets of seed nodes. The memory that is used is limited and moreover there is no restriction regarding the order of the edges that are coming in the stream. Another method, called SCDAC, which is similar to CoEuS is proposed in [101]. This method is based on a single pass streaming community detection procedure. In order to find the optimal community, the "approximate conductance" metric is used. The experimental comparison in real networks has shown that SCDAC is more effective and efficient than CoEuS. A dynamic community detection algorithm, based on static local seeding, is proposed in [46]. The authors use hybrid centrality measures to evaluate the best seed set of nodes. This set in each time step can be updated only by nodes that are affected by network changes.

Another approach is Evoleaders [33], which employs leader nodes with followers in order to identify the evolution of the communities. The "Top leaders" algorithm [53] initialize D leader nodes, one for each community, and associate the nodes of the network to an appropriate leader. In this way, the communities are constructed and by utilizing the highest centrality node, a new leader is picked and the old one is replaced. Then, in each time step, for the initialization of the leader nodes, their common neighbors from the previous and the current time step are taken into consideration and the Top leaders algorithm is used iteratively. Then, the process of community

splitting starts and all small communities can be merged, in an appropriate manner, so that the quality (in terms of modularity) of the community structure is improved. Finally, [11] proposed a framework that strengthens the vicinity of the seed set (called anchors) exploiting the fact that the seed set is of central importance for the evolving community. As a matter of fact, it is assumed that the importance of seeds for the community is established by external knowledge that is not related to topological information of the network.

To address the problem of multiple local community detection in static and dynamic networks, authors in [62] proposed the HqsMLCD and HqsDMLCD methods respectively. The notion of high quality seeds is introduced, which are obtained by the (dynamic for HqsDMLCD) embedding candidate subgraph, and then are (dynamicly for HqsDMLCD) expanded to the detected communities. In addition, in [32], by proposing three main techniques (edge filtering, motif push operation and incremental sweep cut), they can achieve high effectiveness and efficiency in local community detection.

A different method that utilizes differential geometry properties is described in [76]. In this method, the affiliation of a (boundary) node into the local community is determined by the velocity function that uses the derivatives, curvature and gradient, and then tracks the temporal smoothness transformations of the network.

In [96], authors present a multi-step local community detection method. First, by measuring the influence of a node in the community, the seed nodes are selected. Then, communities are expanded by measuring the closeness degree between seed community and non-seed nodes. Finally, the communities are optimized (using conductance) and free nodes are assigned to the communities when high degree similarity between the community and the free node is reached. Another recent research is presented in [23]. In this case, authors generate a k-NN graph based on the high and low order structure and attributed similarity. By processing both categorical and numerical attributes, they can find and expand the seed nodes to the communities. Finally, in [40], in order to identify the best seed nodes, they introduce the node fitness score and then they propose the node contribution measure to detect clusters incrementally. More methods in this category can be found in [68, 27, 49].

4.2.2 Vertex-Centric Distributed Community Detection in Temporal Networks

In this subsection we summarize recent research in the field of community detection utilizing a distributed system. A distributed system is a collection of autonomous computer systems (clusters) that are connected by a centralized computer network equipped with distributed system software. The autonomous computers communicate among each other by sharing resources and files and performing the tasks assigned to them. The most recent papers on community detection related to the vertex-centric approach, are discussed below. This is a subcategory of local community detection since distributed community detection methods tend to allow nodes to access only their immediate neighbor. This tendency has been enforced on distributed algorithms by the vertex-centric approach for computation on networks. Thus, the methods described can in principle be used for local community detection but it would not fit well the distributed computation unless we considered a concurrent distributed system (queries and updates executed in parallel).

An incremental community detection algorithm for distributed dynamic networks using the WCC metric is discussed in [1]. The authors begin by noting that the existence of such metrics is important as they provide information about the quality of a community. They continue with the analysis of the WCC algorithm and mention the three basic conditions that should be satisfied: 1. Each community should contain a central node and a set of nodes that will be connected to the central nodes, 2. The central node should be the one with the largest clustering coefficient, and 3. Given a central node x and a border node y within a community a, the clustering coefficient of x should be greater than the clustering coefficient of any neighbor z of y that is the center of its own community. Then, based on the Distributed Weighted Community Clustering (DWCC) algorithm, they devise the Incremental Distributed Weighted Community Clustering (IDWCC) method. The two algorithms were implemented in the Spark and GraphX environment and the experimental evaluation shows that IDWCC is two to three times faster in detecting local communities than DWCC.

In [67], the authors present an alternative version of the PHASR (Prune-Hash-Refine) method, in a way that it is consistent with known distributed models. For this reason, the algorithm is executed using the big data analysis engine Apache Spark. Although most distributed local community detection algorithms aim to discover a subset of edges within the network using some metric, the

techniques presented in this paper attempt to discover local communities that have the lowest temporal conductivity in a distributed way. The algorithm uses notions like temporal conductivity in order to identify the local communities. Then, they use the Personalized PageRank metric, which will be used in the refinement step of the algorithm and is based on random interactions between the nodes. The analysis of the modified PHASR algorithm is provided. Finally, there are extensive experiments that yield interesting conclusions.

In [51], the authors present a method called randomized Speaker-Listener Label Propagation Algorithm (rLSPA), which is based on LPA and its goal is to detect overlapping communities on distributed dynamic graphs. The authors succinctly present the four steps of their algorithm (Initialization, Label Propagation, Incremental Updating, Post-Processing) and then proceed to its analysis. In a nutshell, the algorithm operates as follows: for the starting graph, the Initialization and Label Propagation stages will be executed and a chart of the graph will be extracted. Then, if the user wants to capture the local communities that are created, the chart is passed to the Post-Processing stage and the result is exported. However, if changes are made to the graph, then it goes to the Incremental Updating stage in order to handle efficiently these updates.

In [30], authors propose a distributed algorithm for detecting local communities in the Stochastic-block model or the Planted Partition Model, which is a model related to random graphs. The proposed algorithm is called CDRW (Community Detection by Random Walks) and is based on random interactions between nodes. A characteristic of the algorithm is that it uses the idea of local mixing sets to prove that specific nodes are part of a specific community. In addition to the Planted Partition Model, they analyze the performance of the CDRW algorithm on CONGEST (widespread model for distributed computing which captures the inherent bandwidth limitations in modern real-world networks) and in the k-machine distributed model through a communication device, as well. These are models for large-scale distributed computing and highlight the significant efficiency of the CDRW algorithm. Finally, they succinctly analyze the complexity of CDRW by analyzing probabilistic techniques and drawing necessary conclusions through graphical representations that highlight the efficiency and accuracy of the method.

An information-theoretic approach for the community detection problem in large networks is described in [31]. The authors begin by providing background into the InfoMap method, which is an information theoretic approach to community detection. This is a sequential algorithm known for providing high quality results. InfoMap uses a standard data compression technique utilizing the PageRank algorithm in order to balance the workload and keep communication, among processes, minimal. Nonetheless, a scalable parallel execution version of InfoMap was needed due to the very high execution time to process large scale networks. Thus, the authors present their solution of a distributed-memory parallel algorithm based on the InfoMap method. The authors conclude that the quality of communities is similar to the sequential InfoMap, providing good scalability and reduced communication cost across processors by transferring a minimal amount of information More papers related to this category are [82, 48, 73, 8, 3].

5 Datasets

In this section we describe various datasets (real or synthetic generators) used in the context of community detection in temporal networks. There are various repositories for temporal networks. We list the most important below accompanied by a small discussion.

5.1 Real Datasets

There is a plethora of real datasets that are used to experiment with proposed approaches. Most of these datasets, followed by a small description, can be found in the following repositories.

- **SNAP** (`https://snap.stanford.edu/snap/`) A large repository of networks of various types among which one can find temporal networks as well. It also provides a Python API.
- **SocioPatterns** (`http://www.sociopatterns.org/datasets/`) It contains social networks (people/animals), some of which have also temporal information.
- **KONECT** (`http://konect.cc/`) Another repository of general purpose networks that also contain temporal information (although no explicit such category).

5.2 Synthetic Generators

A synthetic generator for dynamic networks with communities is given in [78]. They introduce the extensively used dynamic network generator RDyn, which is suitable for generating community dynamics. RDyn is intended to ensure power-law distributions of node degrees and community sizes and can be instantiated in order to crete planted communities that make optimal several different quality measures. The suggested technique first generates the community size and degree distribution and utilize both of them to connect every node to a partition. Then it uses network dynamics by repeatedly adding and removing links. When RDyn recognizes a stable state (a partition of high quality), it provides an output as ground truth and generates community perturbations (split and merge), changing node-community connections of chosen communities.

Another synthetic generator for social graphs is described in [4]. The LDBC Social Network Benchmark (LDBC SNB) has two components: a synthetic generator for social graphs as well as a suite of benchmark queries. LDBC SNB is meant to be a believable look-alike of all the aspects of in operation a social network site. LDBC SNB includes the Interactive workload, that consists of user-centric transactional-like interactive queries, as well as the Business Intelligence Workload, which incorporates analytic queries that corresponds to business essential queries. At first, a graph analytics workload was conjointly enclosed within the roadmap of LDBC SNB, but this was finally delegated to the Graphalytics benchmark project, that was adopted as a formal LDBC graph analytics benchmark. LDBC SNB and Graphalytics joined, target a broad variety of systems with totally different nature and characteristics. LDBC SNB and Graphalytics are intended to capture the essential options of those scenarios, whereas abstracting away details of specific business deployments.

Finally, in [54], the LFR Benchmark network is proposed. They utilize features of real networks, (e.g., heterogeneity), in order to generate graphs. The experimental results show that these features contribute to a harder test of the existing methods.

6 Conclusion

Our aim in this survey is to reexamine all the recent surveys in the field of CD in temporal networks and to propose a new category of methods based on local community detection. Thus, we propose six classes of algorithms and discuss some representative methods. The advantages and drawbacks of each class are also discussed. In future work, it will be beneficial to delve into the local community detection class and to enrich the current survey with more literature.

Acknowledgments

This research was supported by the Hellenic Foundation for Research and Innovation (H.F.R.I.) under the "2nd Call for H.F.R.I. Research Projects to support Faculty Members & Researchers" (Project Number: 3480).

References

[1] Tariq Abughofa, Ahmed A Harby, Haruna Isah, and Farhana Zulkernine. Incremental community detection in distributed dynamic graph. In *2021 IEEE Seventh International Conference on Big Data Computing Service and Applications (BigDataService)*, pages 50–59. IEEE, 2021.

[2] Elyazid Akachar, Brahim Ouhbi, and Bouchra Frikh. Acsimcd: A 2-phase framework for detecting meaningful communities in dynamic social networks. *Future Generation Computer Systems*, 125:399–420, 2021.

[3] Hamidreza Alvari, Alireza Hajibagheri, and Gita Sukthankar. Community detection in dynamic social networks: A game-theoretic approach. In *2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2014)*, pages 101–107. IEEE, 2014.

[4] Renzo Angles, János Benjamin Antal, Alex Averbuch, Peter Boncz, Orri Erling, Andrey Gubichev, Vlad Haprian, Moritz Kaufmann, Josep Lluís Larriba Pey, Norbert Martínez, József Marton, Marcus Paradies, Minh-Duc Pham, Arnau Prat-Pérez, Mirko Spasić, Benjamin A. Steer, Gábor Szárnyas, and Jack Waudby. The ldbc social network benchmark, 2021.

[5] Mohammad Asadi and Foad Ghaderi. Incremental community detection in social networks using label propagation method. In *2018 23rd Conference of Open Innovations Association (FRUCT)*, pages 39–47. IEEE, 2018.

[6] Thomas Aynaud, Eric Fleury, Jean-Loup Guillaume, and Qinna Wang. Communities in evolving networks: definitions, detection, and analysis techniques. In *Dynamics On and Of Complex Networks, Volume 2*, pages 159–200. Springer, 2013.

[7] Mehdi Azaouzi, Delel Rhouma, and Lotfi Ben Romdhane. Community detection in large-scale social networks: state-of-the-art and future directions. *Social Network Analysis and Mining*, 9(1):1–32, 2019.

[8] Seung-Hee Bae and Bill Howe. Gossipmap: A distributed community detection algorithm for billion-edge directed graphs. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–12, 2015.

[9] Sahar Bakhtar and Hovhannes A. Harutyunyan. Dynamic local community detection algorithms. In *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, pages 1–6, 2022.

[10] Georgia Baltsou, Konstantinos Christopoulos, and Konstantinos Tsichlas. Local community detection: A survey. *IEEE Access*, 10:110701–110726, 2022.

[11] Georgia Baltsou and Konstantinos Tsichlas. Dynamic community detection with anchors.

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

[13] Sajid Yousuf Bhat and Muhammad Abulaish. Hoctracker: Tracking the evolution of hierarchical and overlapping communities in dynamic social networks. *IEEE Transactions on Knowledge and Data engineering*, 27(4):1019–1013, 2014.

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

[15] Souâad Boudebza, Rémy Cazabet, Faiçal Azouaou, and Omar Nouali. Olcpm: An online framework for detecting overlapping communities in dynamic social networks. *Computer Communications*, 123:36–51, 2018.

[16] Fariza Bouhatem, Ali Ait El Hadj, Fatiha Souam, and Abdelhakim Dafeur. Incremental methods for community detection in both fully and growing dynamic networks. *Acta Universitatis Sapientiae, Informatica*, 13(2):220–250, 2021.

[17] Nieves R Brisaboa, Diego Caro, Antonio Farina, and M Andrea Rodriguez. Using compressed suffix-arrays for a compact representation of temporal-graphs. *Information Sciences*, 465:459–483, 2018.

[18] Luiz FA Brito, Bruno AN Travençolo, and Marcelo K Albertini. A review of in-memory space-efficient data structures for temporal graphs. *arXiv preprint arXiv:2204.12468*, 2022.

[19] Coen Bron and Joep Kerbosch. Algorithm 457: finding all cliques of an undirected graph. *Communications of the ACM*, 16(9):575–577, 1973.

[20] Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. Time-varying graphs and dynamic networks. In *Proceedings of the 10th International Conference on Ad-Hoc, Mobile, and Wireless Networks*, ADHOC-NOW'11, page 346–359, Berlin, Heidelberg, 2011. Springer-Verlag.

[21] Remy Cazabet, Frederic Amblard, and Chihab Hanachi. Detection of overlapping communities in dynamical social networks. In *2010 IEEE Second International Conference on Social Computing*, pages 309–314, 2010.

[22] Remy Cazabet and Giulio Rossetti. Challenges in community discovery on temporal networks. In *Temporal Network Theory*, pages 181–197. Springer, 2019.

[23] Wenju Chen, Kun Guo, and Yuzhong Chen. Adaptive seed expansion based on composite similarity for community detection in attributed networks. In *CCF Conference on Computer Supported Cooperative Work and Social Computing*, pages 214–227. Springer, 2022.

[24] Zhengzhang Chen, Kevin A. Wilson, Ye Jin, William Hendrix, and Nagiza F. Samatova. Detecting and tracking community dynamics in evolutionary networks. In *2010 IEEE International Conference on Data Mining Workshops*, pages 318–327, 2010.

[25] Mário Cordeiro, Rui Portocarrero Sarmento, and Joao Gama. Dynamic community detection in evolving networks using locality modularity optimization. *Social Network Analysis and Mining*, 6(1):1–20, 2016.

[26] Narimene Dakiche, Fatima Benbouzid-Si Tayeb, Yahya Slimani, and Karima Benatchba. Tracking community evolution in social networks: A survey. *Information Processing and Management*, 56(3):1084–1102, 2019.

[27] Daniel J DiTursi, Gaurav Ghosh, and Petko Bogdanov. Local community detection in dynamic networks. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 847–852. IEEE, 2017.

[28] J R Driscoll, N Sarnak, D D Sleator, and R E Tarjan. Making data structures persistent. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, STOC '86, page 109–121, New York, NY, USA, 1986. Association for Computing Machinery.

[29] Bojan Evkoski, Igor Mozetič, Nikola Ljubešić, and Petra Kralj Novak. Community evolution in retweet networks. *Plos one*, 16(9):e0256175, 2021.

[30] Reza Fathi, Anisur Rahaman Molla, and Gopal Pandurangan. Efficient distributed community detection in the stochastic block model. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pages 409–419. IEEE, 2019.

[31] Md Abdul Motaleb Faysal and Shaikh Arifuzzaman. Distributed community detection in large networks using an information-theoretic approach. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 4773–4782. IEEE, 2019.

[32] Dongqi Fu, Dawei Zhou, and Jingrui He. Local motif clustering on time-evolving graphs. In *Proceedings of the 26th ACM SIGKDD International conference on knowledge discovery & data mining*, pages 390–400, 2020.

[33] Wenhao Gao, Wenjian Luo, and Chenyang Bu. Evolutionary community discovery in dynamic networks based on leader nodes. In *2016 International Conference on Big Data and Smart Computing (BigComp)*, pages 53–60, 2016.

[34] Laetitia Gauvin, André Panisson, and Ciro Cattuto. Detecting the community structure and activity patterns of temporal networks: A non-negative tensor factorization approach. *PLOS ONE*, 9(1):e86028, 2014.

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

[36] Maria Giatsoglou and Athena Vakali. Capturing social data evolution using graph clustering. *IEEE Internet Computing*, 17(1):74–79, Jan 2013.

[37] Changwei Gong, Changhong Jing, Yanyan Shen, and Shuqiang Wang. Dynamic community detection via adversarial temporal graph representation learning. *arXiv preprint arXiv:2207.03580*, 2022.

[38] Derek Greene, Donal Doyle, and Padraig Cunningham. Tracking the evolution of communities in dynamic social networks. In *2010 international conference on advances in social networks analysis and mining*, pages 176–183. IEEE, 2010.

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

[40] Kun Guo, Ling He, Jiangsheng Huang, Yuzhong Chen, and Bing Lin. A local dynamic community detection algorithm based on node contribution. In *CCF Conference on Computer Supported Cooperative Work and Social Computing*, pages 363–376. Springer, 2019.

[41] Qian Guo, Lei Zhang, Bin Wu, and Xuelin Zeng. Dynamic community detection based on distance dynamics. In *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 329–336, 2016.

[42] Tanja Hartmann, Andrea Kappes, and Dorothea Wagner. Clustering evolving networks. In *Algorithm engineering*, pages 280–329. Springer, 2016.

[43] Pascal Held and Rudolf Kruse. Detecting overlapping community hierarchies in dynamic graphs. In *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 1063–1070. IEEE, 2016.

[44] Anne-Sophie Himmel, Hendrik Molter, Rolf Niedermeier, and Manuel Sorge. Enumerating maximal cliques in temporal graphs. In *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 337–344. IEEE, 2016.

[45] Paul W Holland and Samuel Leinhardt. Local structure in social networks. *Sociological methodology*, 7:1–45, 1976.

[46] Yanmei Hu, Yingxi Zhang, Xiabing Wang, Jing Wu, and Bin Duo. A local seeding algorithm for community detection in dynamic networks. In *International Conference on Advanced Data Mining and Applications*, pages 97–112. Springer, 2022.

[47] Xinyu Huang, Dongming Chen, Tao Ren, and Dongqi Wang. A survey of community detection methods in multilayer networks. *Data Mining and Knowledge Discovery*, 35:1–45, 2021.

[48] San-Chuan Hung, Miguel Araujo, and Christos Faloutsos. Distributed community detection on edge-labeled graphs using spark. In *12th International Workshop on Mining and Learning with Graphs (MLG)*, volume 113, 2016.

[49] Saeed Haji Seyed Javadi, Pedram Gharani, and Shahram Khadivi. Detecting community structure in dynamic social networks using the concept of leadership. In *Sustainable interdependent networks*, pages 97–118. Springer, 2018.

[50] Tao Jia, Chenxi Cai, Xin Li, Xi Luo, Yuanyu Zhang, and Xuesong Yu. Dynamical community detection and spatiotemporal analysis in multilayer spatial interaction networks using trajectory data. *International Journal of Geographical Information Science*, pages 1–22, 2022.

[51] Xun Jian, Xiang Lian, and Lei Chen. On efficiently detecting overlapping communities over distributed dynamic graphs. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 1328–1331. IEEE, 2018.

[52] Kaveh Kadkhoda Mohammadmosaferi and Hassan Naderi. Evolution of communities in dynamic social networks: An efficient map-based approach. *Expert Systems with Applications*, 147:113221, 2020.

[53] Reihaneh Rabbany Khorasgani, Jiyang Chen, and Osmar R Zaiane. Top leaders community detection approach in information networks. In *4th SNA-KDD workshop on social network mining and analysis*. Citeseer, 2010.

[54] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. Benchmark graphs for testing community detection algorithms. *Physical review E*, 78(4):046110, 2008.

[55] Matthieu Latapy, Tiphaine Viard, and Clémence Magnien. Stream graphs and link streams for the modeling of interactions over time. *Social Network Analysis and Mining*, 8:1–29, 2018.

[56] Pei Lee, Laks V.S. Lakshmanan, and Evangelos E. Milios. Incremental cluster evolution tracking from highly dynamic network data. In *2014 IEEE 30th International Conference on Data Engineering*, pages 3–14, 2014.

[57] Jingyong Li, Lan Huang, Tian Bai, Zhe Wang, and Hongsheng Chen. Cdbia: A dynamic community detection method based on incremental analysis. In *2012 International Conference on Systems and Informatics (ICSAI2012)*, pages 2224–2228. IEEE, 2012.

[58] Weimin Li, Xiaokang Zhou, Chao Yang, Yuting Fan, Zhao Wang, and Yanxia Liu. Multi-objective optimization algorithm based on characteristics fusion of dynamic social networks for community discovery. *Information Fusion*, 79:110–123, 2022.

[59] Panagiotis Liakos, Katia Papakonstantinopoulou, Alexandros Ntoulas, and Alex Delis. Rapid detection of local communities in graph streams. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[60] Panagiotis Liakos, Katia Papakonstantinopoulou, Theodoros Stefou, and Alex Delis. On compressing temporal graphs. 05 2022.

[61] Hao Liu, Langzhou He, Fan Zhang, Zhen Wang, and Chao Gao. Dynamic community detection over evolving networks based on the optimized deep graph infomax. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 32(5):053119, 2022.

[62] Jiaxu Liu, Yingxia Shao, and Sen Su. Multiple local community detection via high-quality seed identification over both static and dynamic networks. *Data Science and Engineering*, 6(3):249–264, 2021.

[63] Catherine Matias and Vincent Miele. Statistical clustering of temporal networks through a dynamic stochastic block model series b statistical methodology. 2017.

[64] Catherine Matias, Tabea Rebafka, and Fanny Villers. A semiparametric extension of the stochastic block model for longitudinal networks. *Biometrika*, 105(3):665–680, 2018.

[65] Robert Ryan McCune, Tim Weninger, and Greg Madey. Thinking like a vertex: a survey of vertex-centric frameworks for large-scale distributed graph processing. *ACM Computing Surveys (CSUR)*, 48(2):1–39, 2015.

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

[67] Apostolos N. Papadopoulos and Georgios Tzortzidis. Distributed time-based local community detection. In *24th Pan-Hellenic Conference on Informatics*, pages 390–393, 2020.

[68] Eisha Nathan, Anita Zakrzewska, Jason Riedy, and David A Bader. Local community detection in dynamic graphs using personalized centrality. *Algorithms*, 10(3):102, 2017.

[69] Nam P. Nguyen, Thang N. Dinh, Sindhura Tokala, and My T. Thai. Overlapping communities in dynamic networks: Their detection and mobile applications. In *Proceedings of the 17th Annual International Conference on Mobile Computing and Networking*, MobiCom '11, page 85–96, New York, NY, USA, 2011. Association for Computing Machinery.

[70] Nam P. Nguyen, Thang N. Dinh, Ying Xuan, and My T. Thai. Adaptive algorithms for detecting community structure in dynamic social networks. In *2011 Proceedings IEEE INFOCOM*, pages 2282–2290, 2011.

[71] Gergely Palla, Albert-László Barabási, and Tamás Vicsek. Quantifying social group evolution. *Nature*, 446(7136):664–667, Apr 2007.

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

[73] Harris Papadakis, Costas Panagiotakis, and Paraskevi Fragopoulou. Distributed detection of communities in complex networks using synthetic coordinates. *Journal of Statistical Mechanics: Theory and Experiment*, 2014(3):P03013, 2014.

[74] Guo-Jun Qi, Charu C. Aggarwal, and Thomas S. Huang. Online community detection in social sensing. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, WSDM '13, page 617–626, New York, NY, USA, 2013. Association for Computing Machinery.

[75] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical review E*, 76(3):036106, 2007.

[76] M Amin Rigi, Irene Moser, M Mehdi Farhangi, and Chengfei Lui. Finding and tracking local communities by approximating derivatives in networks. *World Wide Web*, 23(3):1519–1551, 2020.

[77] J. Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978.

[78] Giulio Rossetti. RDYN: graph benchmark handling community dynamics. *Journal of Complex Networks*, 5(6):893–912, 07 2017.

[79] Giulio Rossetti and Rémy Cazabet. Community discovery in dynamic networks: A survey. *ACM Comput. Surv.*, 51(2), feb 2018.

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

[81] Hadiseh Safdari, Martina Contisciani, and Caterina De Bacco. Reciprocity, community detection, and link prediction in dynamic networks. *Journal of Physics: Complexity*, 3(1):015010, 2022.

[82] Matthew Saltz, Arnau Prat-Pérez, and David Dominguez-Sal. Distributed community detection with the wcc metric. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1095–1100, 2015.

[83] Mahsa Seifikar, Saeed Farzi, and Masoud Barati. C-blondel: An efficient louvain-based dynamic community detection algorithm. *IEEE Transactions on Computational Social Systems*, 7(2):308–318, 2020.

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

[85] Junming Shao, Zhichao Han, Qinli Yang, and Tao Zhou. Community detection based on distance dynamics. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1075–1084, 2015.

[86] Xing Su, Jianjun Cheng, Haijuan Yang, Mingwei Leng, Wenbo Zhang, and Xiaoyun Chen. Incnsa: Detecting communities incrementally from time-evolving networks based on node similarity. *International Journal of Modern Physics C*, 31(07):2050094, 2020.

[87] Heli Sun, Jianbin Huang, Xin Zhang, Jiao Liu, Dong Wang, Huailiang Liu, Jianhua Zou, and Qinbao Song. Incorder: Incremental density-based community detection in dynamic networks. *Knowledge-Based Systems*, 72:1–12, 2014.

[88] Jimeng Sun, Christos Faloutsos, Spiros Papadimitriou, and Philip S. Yu. Graphscope: Parameter-free mining of large time-evolving graphs. KDD '07, page 687–696, New York, NY, USA, 2007. Association for Computing Machinery.

[89] Yang Sun, Junhua Tang, Li Pan, and Jianhua Li. Matrix based community evolution events detection in online social networks. In *2015 IEEE international conference on smart city/SocialCom/SustainCom (SmartCity)*, pages 465–470. IEEE, 2015.

[90] Zejun Sun, Jinfang Sheng, Bin Wang, Aman Ullah, and FaizaRiaz Khawaja. Identifying communities in dynamic networks using information dynamics. *Entropy*, 22(4):425, 2020.

[91] ZeJun Sun, YaNan Sun, Xinfeng Chang, Feifei Wang, Zhongqiang Pan, Guan Wang, and Jianfen Liu. Dynamic community detection based on the matthew effect. *Physica A: Statistical Mechanics and its Applications*, page 127315, 2022.

[92] Vincent A Traag, Ludo Waltman, and Nees Jan Van Eck. From louvain to leiden: guaranteeing well-connected communities. *Scientific reports*, 9(1):1–12, 2019.

[93] Qinna Wang and Eric Fleury. Mining time-dependent communities. *LAWDN - Latin-American Workshop on Dynamic Networks*, 11 2010.

[94] Yan-Jiao Wang, Jia-Xu Song, and Peng Sun. Research on dynamic community detection method based on an improved pity beetle algorithm. *IEEE Access*, 10:43914–43933, 2022.

[95] Yi Wang, Bin Wu, and Nan Du. Community evolution of social network: Feature, algorithm and model, 2008.

[96] Simeng Wu, Jun Gong, Fei Liu, and Laizong Huang. Multi-step locally expansion detection method using dispersed seeds for overlapping community. In *ITM Web of Conferences*, volume 47, page 02008. EDP Sciences, 2022.

[97] Jierui Xie, Mingming Chen, and Boleslaw K Szymanski. Labelrankt: Incremental community detection in dynamic networks via label propagation. In *Proceedings of the workshop on dynamic networks management and mining*, pages 25–32, 2013.

[98] Jierui Xie and Boleslaw K Szymanski. Labelrank: A stabilized label propagation algorithm for community detection in networks. In *2013 IEEE 2nd Network Science Workshop (NSW)*, pages 138–143. IEEE, 2013.

[99] Jierui Xie, Boleslaw K Szymanski, and Xiaoming Liu. Slpa: Uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process. In *2011 ieee 11th international conference on data mining workshops*, pages 344–349. IEEE, 2011.

[100] Tianbao Yang, Yun Chi, Shenghuo Zhu, Yihong Gong, and Rong Jin. A bayesian approach toward finding communities and their evolutions in dynamic social networks. pages 990–1001, 04 2009.

[101] Yanhao Yang, Meng Wang, David Bindel, and Kun He. Streaming local community detection through approximate conductance. *arXiv preprint arXiv:2110.14972*, 2021.

[102] Ying Yin, Yuhai Zhao, He Li, and Xiangjun Dong. Multi-objective evolutionary clustering for large-scale dynamic community detection. *Information Sciences*, 549:269–287, 2021.

[103] Anita Zakrzewska and David A. Bader. A dynamic algorithm for local community detection in graphs. ASONAM '15, page 559–564, New York, NY, USA, 2015. Association for Computing Machinery.

[104] Anita Zakrzewska and David A Bader. Tracking local communities in streaming graphs with a dynamic algorithm. *Social Network Analysis and Mining*, 6(1):1–16, 2016.

[105] Neda Zarayeneh and Ananth Kalyanaraman. A fast and efficient incremental approach toward dynamic community detection. In *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, ASONAM '19, page 9–16, New York, NY, USA, 2019. Association for Computing Machinery.

[106] Neda Zarayeneh, Nitesh Kumar, Ananth Kalyanaraman, and Aurora Clark. Dynamic community detection decouples hierarchical timescale behavior of complex chemical systems. 2022.

[107] Hedia Zardi, Bushra Alharbi, Walid Karamti, Hanen Karamti, and Eatedal Alabdulkreem. Detection of community structures in dynamic social networks based on message distribution and structural/attribute similarities. *IEEE Access*, 9:67028–67041, 2021.

[108] Xiangxiang Zeng, Wen Wang, Cong Chen, and Gary G. Yen. A consensus community-based particle swarm optimization for dynamic community detection. *IEEE Transactions on Cybernetics*, 50(6):2502–2513, 2020.

[109] Zhongying Zhao, Chao Li, Xuejian Zhang, Francisco Chiclana, and Enrique Herrera Viedma. An incremental method to detect communities in dynamic evolving social networks. *Knowledge-Based Systems*, 163:404–415, 2019.

[110] Di Zhuang, J Morris Chang, and Mingchen Li. Dynamo: Dynamic community detection by incrementally maximizing modularity. *IEEE Transactions on Knowledge and Data Engineering*, 33(5):1934–1945, 2019.