Detecting Stable Communities in Link Streams at Multiple Temporal Scales

Souâad Boudebza[1(✉)], Rémy Cazabet[2], Omar Nouali[3], and Faiçal Azouaou[1]

[1] Ecole nationale Supérieure d'Informatique,
BP 68M, 16309 Oued-Smar, Alger, Algeria
s_boudebza@esi.dz
[2] Univ Lyon, Universite Lyon 1, CNRS, LIRIS UMR5205, 69622 Lyon, France
[3] Division de Recherche en Théorie et Ingénierie des Systèmes Informatiques,
CERIST, Rue des Frères Aissiou, Ben Aknoun, Alger, Algeria

Abstract. Link streams model interactions over time in a wide range of fields. Under this model, the challenge is to mine efficiently both temporal and topological structures. Community detection and change point detection are one of the most powerful tools to analyze such evolving interactions. In this paper, we build on both to detect stable community structures by identifying change points within meaningful communities. Unlike existing dynamic community detection algorithms, the proposed method is able to discover stable communities efficiently at multiple temporal scales. We test the effectiveness of our method on synthetic networks, and on high-resolution time-varying networks of contacts drawn from real social networks.

1 Introduction

In recent years, studying interactions over time has witnessed a growing interest in a wide range of fields, such as sociology, biology, physics, etc. Such dynamic interactions are often represented using the snapshot model: the network is divided into a sequence of static networks, i.e., snapshots, aggregating all contacts occurring in a given time window. The main drawback of this model is that it often requires to choose arbitrarily a temporal scale of analysis. The link stream model [9] is a more effective way for representing interactions over time, that can fully capture the underling temporal information.

Real world networks evolve frequently at many different time scales. Fluctuations in such networks can be observed at yearly, monthly, daily, hourly, or even smaller scales. For instance, if one were to look at interactions among workers in a company or laboratory, one could expect to discover clusters of people corresponding to *meetings* and/or *coffee breaks*, interacting at **high frequency** (e.g., every few seconds) for **short periods** (e.g., few minutes), *project members*

This work was supported by the ACADEMICS grant of the IDEXLYON, project of the Universite de Lyon, PIA operated by **ANR-16-IDEX-0005**, and of the project **ANR-18-CE23-0004** of the French National Research Agency (ANR).

P. Cellier and K. Driessens (Eds.): ECML PKDD 2019 Workshops, CCIS 1167, pp. 353–367, 2020.
https://doi.org/10.1007/978-3-030-43823-4_30

interacting at medium frequency (e.g., once a day) for medium periods (e.g., a few months), *coordination groups* interacting at low frequency (e.g., once a month) for longer periods (e.g., a few years), etc.

An analysis of communities found at an arbitrary chosen scale would necessarily miss some of these communities: low latency ones are invisible using short aggregation windows, while high frequency ones are lost in the noise for long aggregation windows. A multiple temporal scale analysis of communities seems therefore the right solution to study networks of interactions represented as link streams.

To the best of our knowledge, no such method exists in the literature. In this article, we propose a method having roots both in the literature on change point detection and in dynamic community detection. It detects what we call **stable communities**, i.e., *groups of nodes* forming a *coherent community* throughout a *period of time*, at a given *temporal scale*.

The remainder of this paper is organized as follows. In Sect. 2, we present a brief review of related works. Then, we describe the proposed framework in detail in Sect. 3. We experimentally evaluate the proposed method on both synthetic and real-world networks in Sect. 4.

2 Related Work

Our contribution relates to two active body of research: (i) dynamic community detection and (ii) change point detection. The aim of the former is to discover groups of nodes that persist over time, while the objective of the latter is to detect changes in the overall structure of a dynamic network. In this section, we present existing work in both categories, and how our proposed method relates to them.

2.1 Dynamic Community Detection

The problem of detecting communities in dynamic networks has attracted a lot of attention in recent years, with various approaches tackling different aspects of the problem, see [16] for a recent survey. Most of these methods consider that the studied dynamic networks are represented as sequences of snapshots, with each snapshot being a well formed graph with meaningful community structure, see for instance [5,12]. Some other methods work with interval graphs, and update the community structure at each network change, e.g., [3,17]. However, all those methods are not adapted to deal with link streams, for which the network is usually not well formed at any given time. Using them on such a network would require to first aggregate the links of the stream by choosing an arbitrarily temporal scale (aggregation window).

2.2 Change Point Detection

Our work is also related to research conducted on change point detection considering community structures. In these approaches, given a sequence of snapshots, one wants to detect the periods during which the network organization

and/or the community structure remains stable. In [15], the authors proposed the first change-point detection method for evolving networks that uses generative network models and statistical hypothesis testing. Wang et al. [19] proposed a hierarchical change point detection method to detect both inter-community (local change) and intra-community (global change) evolution. A recent work by Masuda et al. [11] used graph distance measures and hierarchical clustering to identify sequences of system state dynamics.

From those methods, our proposal keeps the principle of stable periods delimited by change points, and the idea of detecting changes at local and global scales. But our method differs in two directions: (*i*) we are searching for stable individual communities instead of stable graph periods, and (*ii*) we search for stable structures at multiple levels of temporal granularity.

3 Method

The goal of our proposed method is (*i*) to detect stable communities (*ii*) at multiple scales without redundancy and (*iii*) to do so efficiently. We adopt an iterative approach, searching communities from the coarser to the more detailed temporal scales. At each temporal scale, we use a three step process:

1. **Seed Discovery**, to find relevant community seeds at this temporal scale.
2. **Seed Pruning**, to remove seeds which are redundant with communities found at higher scales.
3. **Seed Expansion**, expanding seeds in time to discover stable communities.

We start by presenting each of these three steps, and then we describe the method used to iterate through the different scales in Sect. 3.4.

Our work aims to provide a general framework that could serve as baseline for further work in this field. We define three generic functions that can be set according to the user needs:

- **CD**(g), a static community detection algorithm on a graph g.
- **QC**(N, g), a function to assess the quality of a community defined by the set of nodes N on a graph g.
- **CSS**(N_1,N_2), a function to assess the similarity of two sets of nodes N_1 and N_2.

See Sect. 3.5 on how to choose proper functions for those tasks.

We define a stable dynamic community c as a triplet $c = (N, p, \gamma)$, with $c.N$ the list of nodes in the community, $c.p$ its period of existence defined as an interval, e.g., $c.p = [t_1, t_2]$[1] means that the community c exists from t_1 to t_2, and $c.\gamma$ the temporal granularity at which c has been discovered.

We denote the set of all stable dynamic communities $\mathcal{D}$.

[1] We use right open intervals such as a community starting at t_x and another one ending at the same t_x have an empty intersection, which is necessary to have coherent results when handling discrete time steps.

3.1 Seed Discovery

For each temporal scale, we first search for interesting seeds. A temporal scale is defined by a granularity γ, expressed as a period of time (e.g.; 20 min, 1 h, 2 weeks, etc.). We use this granularity as a window size, and, starting from a time t_0 – by default, the date of the first observed interaction – we create a cumulative graph (snapshot) for every period $[t_0, t_0+\gamma[, [t_0+\gamma, t_0+2\gamma[, [t_0+2\gamma, t_0+3\gamma[, etc.$, until all interactions belong to a cumulative graph. This process yields a sequence of static graphs, such as $G_{t_0,\gamma}$ is a cumulated snapshot of link stream G for the period starting at t_0 and of duration γ. G_γ is the list of all such graphs.

Given a static community detection algorithm CD yielding a set of communities, and a function to assess the quality of communities QC, we apply CD on each snapshot and filter promising seeds, i.e., high quality communities, using QC. The set of valid seeds $\mathcal{S}$ is therefore defined as:

$$\mathcal{S} = \{\forall g \in G_\gamma, \forall s \in CD(g), QC(s, g) > \theta_q\} \tag{1}$$

With θ_q a threshold of community quality.

Since community detection at each step is independent, we can run it in parallel on all steps, this is an important aspect for scalability.

3.2 Seed Pruning

The seed pruning step has a twofold objective: (i) reducing redundancy and (ii) speed up the multi-scale community detection process. Given a measure of structural similarity CSS, we prune the less interesting seeds, such as the set of filtered seeds $\mathcal{FS}$ is defined as:

$$\mathcal{FS} = \{\forall s \in \mathcal{S}, \forall c \in \mathcal{D}, (CSS(s.N, c.N) > \theta_s) \vee (s.p \cap c.p = \{\emptyset\}) \tag{2}$$

Where $\mathcal{D}$ is the set of stable communities discovered at coarser (or similar, see next section) scales, $s.p$ is the interval corresponding to the snapshot at which this seed has been discovered, and θ_s is a threshold of similarity.

Said otherwise, we keep as interesting seeds those that are not redundant topologically (in term of nodes/edges), OR not redundant temporally. A seed is kept if it corresponds to a situation never seen before.

3.3 Seed Expansion

The aim of this step is to assess whether a seed corresponds to a *stable* dynamic community. The *instability* problem has been identified since the early stages of the dynamic community detection field [1]. It means that the same algorithm ran twice on the same network after introducing minor random modifications might yield very different results. As a consequence, one cannot know if the differences observed between the community structure found at t and at $t+1$ are due to structural changes or to the instability of the algorithm. This problem

is usually solved by introducing smoothing techniques [16]. Our method use a similar approach, but instead of comparing communities found at step t and $t-1$, we check whether a community found at t is still relevant in previous and following steps, recursively.

More formally, for each seed $s \in FS$ found on the graph $G_{t,\gamma}$, we iteratively expand the duration of the seed $s.d = [t, t+\gamma[$ (where t is the time start of this duration) at each step t_i in both temporal directions ($t_i \in (...[t-2\gamma, t-\gamma[, [t-\gamma, t]; [t+\gamma, t+2\gamma[, [t+2\gamma, t+3\gamma]...)$) as long as the quality $QC(s.N, G_{t_i,\gamma})$ of the community defined by the nodes $s.N$ on the graph at $G_{t_i,\gamma}$ is good enough. Here, we use the same similarity threshold θ_s as in the seed pruning step. If the final duration period $|s.p|$ of the expanded seed is higher than a duration $\theta_p\gamma$, with θ_p a threshold of stability, the expanded seed is added to the list of stable communities, otherwise, it is discarded. This step is formalized in Algorithm 1.

Algorithm 1: Forward seed expansion. Forward temporal expansion of a seed s found at time t of granularity γ. The reciprocal algorithm is used for *backward* expansion: $t+1$ becomes $t-1$.

Input: $s, \gamma, \theta_p, \theta_s$
$t \leftarrow t^{start} | s.p = [t^{start}, t^{end}[$;
$g \leftarrow G_{t,\gamma}$;
$p \leftarrow [t, t+\gamma[$;
while $QC(s.N, g) > \theta_s$ **do**
  $s.p \leftarrow s.p \cup p$;
  $t \leftarrow t+\gamma$;
  $p \leftarrow [t, t+\gamma[$;
  $g \leftarrow G_{t,\gamma}$;
end
if $|s.p| \geq \theta_p\gamma$ **then**
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{s\}$;
end

In order to select the most relevant stable communities, we consider seeds in descending order of their QC score, i.e., the seeds of higher quality scores are considered first. Due to the pruning strategy, a community of lowest quality might be pruned by a community of highest quality at the same granularity γ.

3.4 Multi-scale Iterative Process

Until then, we have seen how communities are found for a particular time scale. In order to detect communities at multiple scales, we first define the ordered list of studied scales Γ. The largest scale is defined as $\gamma^{max} = |G.d|/\theta_p$, with $|G.d|$ the total duration of the dynamic graph. Since we need to observe at least θ_p successive steps to consider the community stable, γ^{max} is the largest scale at which communities can be found.

We then define Γ as the ordered list:

$$\Gamma = [\gamma^{max}, \gamma^{max}/2^1, \gamma^{max}/2^2, \gamma^{max}/2^3, ..., \gamma^{max}/2^k] \quad (3)$$

With k such as $\gamma^{max}/2^k > \theta_\gamma >= \gamma^{max}/2^{k+1}$, θ_γ being a parameter corresponding to the finest temporal granularity to evaluate, which is necessarily data-dependant (if time is represented as a continuous property, this value can be fixed at least at the sampling rate of data collection).

This exponential reduction in the studied scale guarantees a limited number of scales to study.

The process to find seeds and extend them into communities is then summarized in Algorithm 2.

Algorithm 2: Multi-temporal-scale stable communities finding. Summary of the proposed method. See corresponding sections for the details of each step. G is the link streams to analyze, $\theta_q, \theta_s, \theta_p, \theta_\gamma$ are threshold parameters.

Input: $G, \theta_q, \theta_s, \theta_p, \theta_\gamma$
$\mathcal{D} \leftarrow \{\emptyset\}$;
$\Gamma \leftarrow$studied_scales(G, θ_γ) ;
for $\gamma \in \Gamma$ **do**
  $\mathcal{S} \leftarrow$ Seed_Discovery$(\gamma, CD, QC, \theta_q)$;
  $\mathcal{FS} \leftarrow$Seed_Pruning$(\mathcal{S}, CSS, \theta_s)$;
  for $s \in \mathcal{FS}$ **do**
    Seed_Expansion$(s, \gamma, \theta_p, \theta_s)$;
  end
end

3.5 Choosing Functions and Parameters

The proposed method is a general framework that can be implemented using different functions for CD, QC and CSS. This section provides explicit guidance for selecting each function, and introduces the choices we make for the experimental section.

Community Detection - CD. Any algorithm for community detection could be used, including overlapping methods, since each community is considered as an independant seed. Following literature consensus, we use the Louvain method [2], which yields non-overlapping communities using a greedy modularity-maximization method. The louvain method performs well on static networks, it is in particular among the fastest and most efficient methods. Note that it would be meaningful to adopt an algorithm yielding communities of good quality according to the chosen QC, which is not the case in our experiments, as we wanted to use the most standard algorithms and quality functions in order to show the genericity of our approach.

Quality of Communities - QC. The QC quality function must express the quality of a set of nodes w.r.t a given network, unlike functions such as the modularity, which express the quality of a whole partition w.r.t a given network.

Many such functions exist, like *Link Density* or *Scaled Density* [7], but the most studied one is probably the *Conductance* [10]. Conductance is defined as the ratio of (i) the number of edges between nodes inside the community and nodes outside the community, and (ii) the sum of degrees of nodes inside the community (or outside, if this value is larger). More formally, the conductance ϕ of a community C is:

$$\phi(C) = \frac{\sum_{i \in C, j \notin C} A_{i,j}}{Min(A(C), A(\bar{C}))}$$

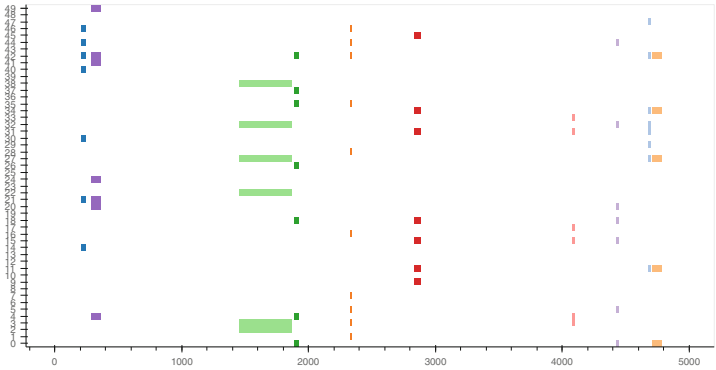
Where A is the adjacency matrix of the network, $A(C) = \sum_{i \in C} \sum_{j \in V} A_{i,j}$ and $\bar{C}$ is the complement of C. Its value ranges from 0 (Best, all edges starting from nodes of the community are internal) to 1 (Worst, no edges between this community and the rest of the network). Since our generic framework expects good communities to have QC scores higher than the threshold θ_q, we adopt the definition $QC = 1\text{-conductance}$.

Community Seed Similarity - CSS. This function takes as input two sets of nodes, and returns their similarity. Such a function is often used in dynamic community detection to assess the similarity between communities found in different time steps. Following [5], we choose as a reference function the Jaccard Index. Given two sets A and B, it is defined as: $J(A, B) = \frac{|A \cap B|}{|A \cup B|}$

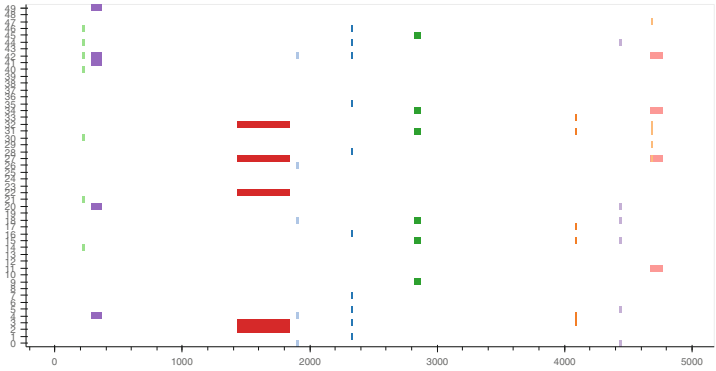
3.6 Parameters

The algorithm has four parameters, $\theta_\gamma, \theta_q, \theta_s, \theta_p$, defining different thresholds. We explicit them and provide the values used in the experiments.

1. θ_γ is data-dependant. It corresponds to the smallest temporal scale that will be studied, and should be set at least at the collection rate. For synthetic networks, it is set at 1 (the smallest temporal unit needed to generate a new stream), while, for SocioPatterns dataset, it is set to 20 s (the minimum length of time required to capture a contact).
2. θ_q determines the minimal quality a seed must have to be preserved and expanded. The higher this value, the more strict we are on the quality of communities. We set $\theta_q = 0.7$ in all experiments. It is dependent on the choice of the QC function.
3. θ_s determines the threshold above which two communities are considered redundant. The higher this value, the more communities will be obtained. We set $\theta_s = 0.3$ in all experiments. It is dependent on the choice of the CSS function.
4. θ_p is the minimum number of consecutive periods a seed must be expanded in order to be considered as stable community. We set $\theta_s = 3$ in all experiments. The value should not be lower in order to avoid spurious detections due to pure chance. Higher values could be used to limit the number of results.



(a) Stable communities produced by the generator.



(b) Stable communities discovered by the proposed method.

Fig. 1. Visual comparison between planted and discovered communities. Time steps on the horizontal axis, nodes on the vertical axis. Colors correspond to communities and are randomly assigned. We can observe that most communities are correctly discovered, both in terms of nodes and of duration. (Color figure online)

4 Experiments and Results

The validation of our method encompasses three main aspects: (*i*) the validity of communities found, and (*ii*) the multi-scale aspect of our method, (*iii*) its scalability. We conduct two kinds of experiments: on synthetic data, on which we use planted *ground-truth* to quantitatively compare our results, and on real networks, on which we use both qualitative and quantitative evaluation to validate our method.

4.1 Validation on Synthetic Data

To the best of our knowledge, no existing network generator allows to generate dynamic communities at multiple temporal scale. We therefore introduce a simple solution to do so. Let us consider a dynamic network composed of T steps

and N different nodes. We start by adding some random noise: at each step, an Erdos-Renyi random graph [4] is generated, with a probability of edge presence equal to p. We then add a number SC of random stable communities. For each community, we attribute randomly a set of $n \in [4, N/4]$ nodes, a duration $d \in [10, T/4]$ and a starting date $s \in [0, T - d]$. n and d are chosen using a logarithmic probability, in order to increase variability. The temporal scale of the community is determined by the probability of observing an edge between any two of its nodes during the period of its existence, set as $10/d$. As a consequence, a community of duration 10 will have edges between all of its nodes at every step of its existence, while a community of length 100 will have an edge between any two of its nodes only every 10 steps in average.

Since no algorithm exists to detect communities at multiple temporal scales, we compare our solution to a baseline: communities found by a static algorithm on each window, for different window sizes. It corresponds to *detect & match* methods for dynamic community detection such as [5]. We then compare the results by computing the overlapping NMI as defined in [8], at each step. For those experiments, we set $T = 5000, N = 100, p = 10/N$. We vary the number of communities SC.

Table 1. Comparison of the average NMI scores (over 10 runs) obtained for the proposed method (*Proposed*) and for each of the temporal scales ($\gamma \in \Gamma$) used by the proposed method, taken independently.

t_scale (γ)	5	10	20	30	40	50
Proposed	**0.91**	**0.78**	**0.69**	**0.69**	**0.62**	**0.54**
1666	0.41	0.32	0.24	0.23	0.15	0.19
833	0.36	0.30	0.29	0.27	0.23	0.25
416	0.39	0.40	0.36	0.34	0.32	0.33
208	0.46	0.45	0.40	0.42	0.41	0.37
104	0.47	0.48	0.44	0.46	0.45	0.42
52	0.45	0.47	0.45	0.47	0.47	0.45
26	0.35	0.35	0.38	0.42	0.42	0.41
13	0.28	0.26	0.30	0.31	0.32	0.31
6	0.17	0.16	0.19	0.19	0.20	0.19
3	0.12	0.09	0.11	0.10	0.12	0.11
1	0.05	0.03	0.04	0.03	0.05	0.04

Figure 1 represents the synthetic communities to find for $SC = 10$, and the communities discovered by the proposed method. We can observe a good match, with communities discovered throughout multiple scales (short-lasting and long-lasting ones). We report the results of the comparison with baselines in Table 1. We can observe that the proposed method outperforms the baseline at every scale in all cases in term of average NMI.

The important implication is that the problem of dynamic community detection is not only a question of choosing the right scale through a window size, but that if the network contains communities at multiple temporal scale, one needs to use an adapted method to discover them.

4.2 Validation on Real Datasets

We validate our approach by applying it to two real datasets. Because no ground truth data exists to compare our results with, we validate our method by using both quantitative and qualitative evaluation. We use the quantitative approach to analyze the scalability of the method and the characteristics of communities discovered compared with other existing algorithms. We use the qualitative approach to show that the communities found are meaningful and could allow an analyst to uncover interesting patterns in a dynamic datasets.

The datasets used are the following:

- **SocioPatterns** primary school data [18], face-to-face interactions between children in a school (323 nodes, 125 773 interaction).
- **Math overflow** stack exchange interaction dataset [14], a larger network to evaluate scalability (24 818 nodes, 506 550 interactions).

Qualitative Evaluation. For the qualitative evaluation, we used the primary school data [18] collected by the SocioPatterns collaboration[2] using RFID devices. They capture face-to-face proximity of individuals wearing them, at a rate of one capture every 20 s. The dataset contains face-to-face interactions between 323 children and 10 teachers collected over two consecutive days in October 2009. This school has 5 levels, each level is divided into 2 classes (A and B), for a total of 10 classes.

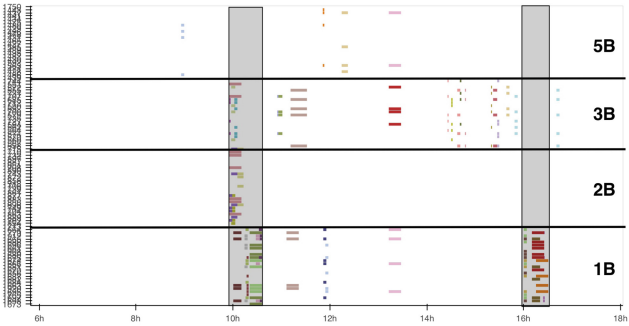
No community ground truth data exists to validate quantitatively our findings. We therefore focus on the descriptive information highlighted on the SocioPatterns study [18], and we show how the results yielded by our method match the course of the day as recorded by the authors in this study.

In order to make an accurate analysis of our results, the visualization have been reduced to one day (the second day), and we limited ourselves to 4 classes (1B, 2B, 3B, 5B)[3]. 120 communities are discovered in total on this dataset. We created three different figures, corresponding to communities of length respectively (i) less than half an hour, (ii) between half an hour and 2 h, (iii) more than 2 h. Figure 2 depicts the results. Nodes affiliations are ordered by class, as marked on the right side of the figure. The following observations can be made:

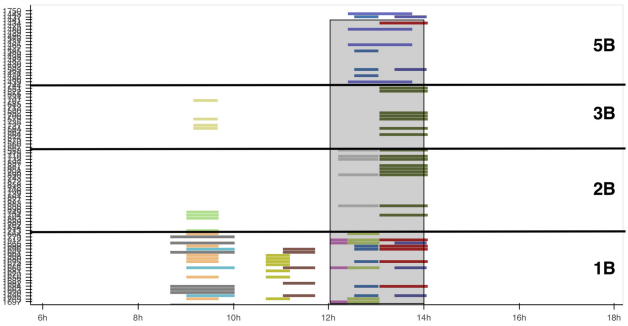
- Communities having the longest period of existence clearly correspond to the class structure. Similar communities had been found by the authors of the original study using aggregated networks per day.

[2] www.sociopatterns.org.

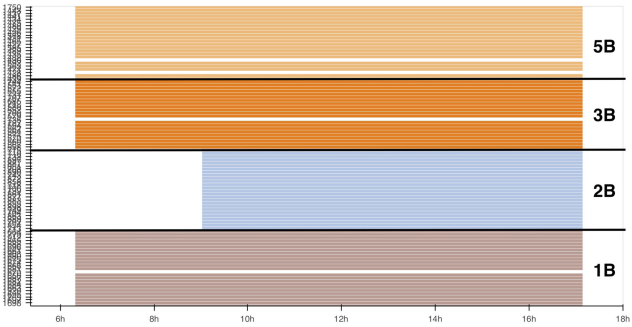
[3] Note that full results can be explored online using the provided notebook (see conclusion section).



(a) Second day, length<30min. Grey vertical areas correspond to most likely break periods.



(b) Second day, 30min<length<2hours. Grey vertical area corresponds to the lunch break



(c) Second day, length>2hours

Fig. 2. Stable communities of different lengths on the SocioPatterns Primary School Dataset. Time on the horizontal axis, children on the vertical axis. Colors are attributed randomly. (Color figure online)

- Most communities of the shorter duration are detected during what are probably breaks between classes. In the original study, it had been noted that break periods are marked by the highest interaction rates. We know from data description that classes have 20/30 min breaks, and that those breaks are not necessarily synchronized between classes. This is compatible with observation, in particular with communities found between 10:00 and 10:30 in the morning, and between 4:00 and 4:30 in the afternoon.
- Most communities of medium duration occur during the lunch break. We can also observe that the most communities are separated into two intervals, 12:00–13:00 and 13:00–14:00. This can be explained by the fact that children have a common canteen, and a shared playground. As the playground and the canteen do not have enough capacity to host all the students at the same time, only two or three classes have breaks at the same time, and lunches are taken in two consecutive turns of one hour. Some children do not belong to any communities during the lunch period, which matches the information that about half of the children come back home for lunch [18].
- During lunch breaks and class breaks, some communities involve children from different classes, see the community with dark-green colour during lunch time (medium duration figure) or the pink community around 10:00 for short communities, when classes 2B and 3B are probably in break at the same time. This confirms that an analysis at coarser scales only can be misleading, as it leads only to the detection of the stronger class structure, ignoring that communities exist between classes too, during shorter periods.

Quantitative Evaluation. In this section, we compare our proposition with other methods on two aspects: scalability, and aggregated properties of communities found. The methods we compare ourselves to are:

- An Identify and Match framework proposed by Greene et al. [5]. We implement it using the Louvain method for community detection, and the *Jaccard coefficient* to match communities, with a minimal similarity threshold of 0.7. We used a custom implementation, sharing the community detection phase with our method.
- The multislice method introduced by Mucha et al. [12]. We used the authors implementation, with interslice coupling $\omega = 0.5$.
- The dynamic clique percolation method (D-CPM) introduced by Palla et al. [13]. We used a custom implementation, the detection in each snapshot is done using the implementation in the networkx library [6].

For Identify and Match, D-CPM and our approach, the community detection phase is performed in parallel for all snapshots. This is not possible for Mucha et al., since the method is performed on all snapshots simultaneously. On the other hand, D-CPM and Indentify and Match are methods with no dynamic smoothing.

Figure 3 presents the time taken by those methods and our proposition, for each temporal granularity, on the Math Overflow network. The task

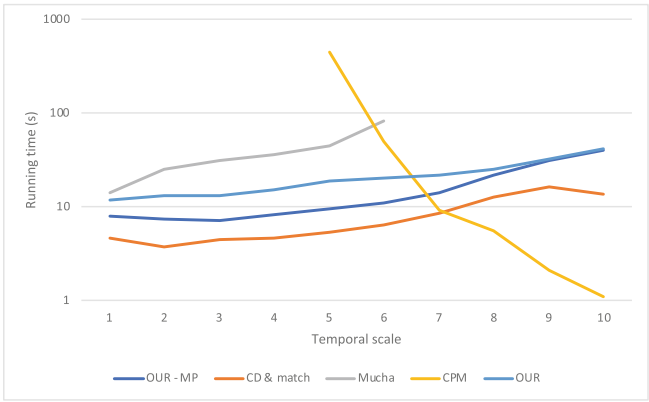


Fig. 3. Speed of several dynamic community detection methods for several temporal granularities, on the Math Overflow dataset. Missing points correspond to computation time above 1000 s. Temporal scales correspond to window sizes and are divided by 2 at every level, from 1 = 67 681 200 s (about 2 years) to 10 = 132 189 s (about 36 h). OUR and OUR-MP corresponds to our method using or not multiprocessing (4 cores)

accomplished by our method is, of course, not comparable, since it must not only discover communities, but also avoid redundancy between communities in different temporal scales, while other methods yield redundant communities in different levels. Nevertheless, we can observe that the method is scalable to networks with tens of thousands of nodes and hundreds of thousands of interactions. It is slower than the Identify and Match (CD&Match) approach, but does not suffer from the scalability problem as for the two other ones (D-CPM and Mucha et al.). In particular, the clique percolation method is not scalable to large and dense networks, a known problem due to the exponential growth in the number of cliques to find. For the method by Mucha et al., the scalability issue is due to the memory representation of a single modularity matrix for all snapshots.

Table 2. Average properties of communities found by each method (independently of their temporal granularity). #Communities: number of communities found. Persistence: number of consecutive snapshots. Size: number of nodes. Stability: average Jaccard coefficient between nodes of the same community in successive snapshots. Density: average degree/size-1. Q: 1-Conductance (higher is better)

Method	#Communities	Persistance	Size	Stability	Density	Q
OUR	179	3.44	10.89	1.00	0.50	0.91
CD& MATCH	29846	1.21	5.50	0.97	0.42	0.96
CPM	3259	1.87	5.37	0.51	0.01	0.53
MUCHA	1097	15.48	9.72	0.62	0.38	0.85

In Table 2, we summarize the number of communities found by each method, their persistence, size, stability, density and conductance. It is not possible to formally rank those methods based on these values only, that correspond to vastly different scenarios. What we can observe is that existing methods yield much more communities than the method we propose, usually at the cost of lower overall quality. When digging into the results, it is clear that other methods yield many noisy communities, either found on a single snapshot for methods without smoothing, unstable for the smoothed Mucha method, and often with low density or Q.

5 Conclusion and Future Work

To conclude, this article only scratches the surface of the possibilities of multiple-temporal-scale community detection. We have proposed a first method for the detection of such structures, that we validated on both synthetic and real-world networks, highlighting the interest of such an approach. The method is proposed as a general, extensible framework, and its code is available[4,5] as an easy to use library, for replications, applications and extensions.

As an exploratory work, further investigations and improvements are needed. Heuristics or statistical selection procedures could be implemented to reduce the computational complexity. Hierarchical organization of relations – both temporal and structural–between communities could greatly simplify the interpretation of results.

References

1. Aynaud, T., Guillaume, J.L.: Static community detection algorithms for evolving networks. In: 8th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks, pp. 513–519. IEEE (2010)
2. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. J. Stat. Mech: Theory Exp. **2008**(10), P10008 (2008)
3. Boudebza, S., Cazabet, R., Azouaou, F., Nouali, O.: OLCPM: an online framework for detecting overlapping communities in dynamic social networks. Comput. Commun. **123**, 36–51 (2018)
4. Erdös, P., Rényi, A.: On random graphs i. Publ. Math. Debr. **6**, 290–297 (1959)
5. Greene, D., Doyle, D., Cunningham, P.: Tracking the evolution of communities in dynamic social networks. In: 2010 International Conference on Advances in Social Networks Analysis and Mining, pp. 176–183. IEEE (2010)
6. Hagberg, A., Swart, P., Chult, D.S.: Exploring network structure, dynamics, and function using networkX. Technical report, Los Alamos National Lab. (LANL), Los Alamos, NM, USA (2008)

[4] The full code is available at https://github.com/Yquetzal/ECML_PKDD_2019.
[5] An online notebook to test the method is available at https://colab.research.google.com/github/Yquetzal/ECML_PKDD_2019/blob/master/simple_demo.ipynb.

7. Labatut, V., Orman, G.K.: Community structure characterization. In: Alhajj, R., Rokne, J. (eds.) Encyclopedia of Social Network Analysis and Mining, pp. 1–13. Springer, New York (2017). https://doi.org/10.1007/978-1-4614-7163-9
8. Lancichinetti, A., Fortunato, S., Kertész, J.: Detecting the overlapping and hierarchical community structure in complex networks. New J. Phys. **11**(3), 033015 (2009)
9. Latapy, M., Viard, T., Magnien, C.: Stream graphs and link streams for the modeling of interactions over time. CoRR abs/1710.04073 (2017)
10. Leskovec, J., Lang, K.J., Dasgupta, A., Mahoney, M.W.: Community structure in large networks: natural cluster sizes and the absence of large well-defined clusters. Internet Math. **6**(1), 29–123 (2009)
11. Masuda, N., Holme, P.: Detecting sequences of system states in temporal networks. Sci. Rep. **9**(1) (2019). https://doi.org/10.1038/s41598-018-37534-2
12. Mucha, P.J., Richardson, T., Macon, K., Porter, M.A., Onnela, J.P.: Community structure in time-dependent, multiscale, and multiplex networks. Science **328**(5980), 876–878 (2010)
13. Palla, G., Barabási, A.L., Vicsek, T.: Quantifying social group evolution. Nature **446**(7136), 664 (2007)
14. Paranjape, A., Benson, A.R., Leskovec, J.: Motifs in temporal networks. In: Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, pp. 601–610. ACM (2017)
15. Peel, L., Clauset, A.: Detecting change points in the large-scale structure of evolving networks. CoRR abs/1403.0989 (2014)
16. Rossetti, G., Cazabet, R.: Community discovery in dynamic networks: a survey. ACM Comput. Surv. (CSUR) **51**(2), 35 (2018)
17. Rossetti, G., Pappalardo, L., Pedreschi, D., Giannotti, F.: Tiles: an online algorithm for community discovery in dynamic social networks. Mach. Learn. **106**(8), 1213–1241 (2017)
18. Stehlé, J., et al.: High-resolution measurements of face-to-face contact patterns in a primary school. PLOS ONE **6**(8) (2011). https://doi.org/10.1371/journal.pone.0023176
19. Wang, Y., Chakrabarti, A., Sivakoff, D., Parthasarathy, S.: Fast change point detection on dynamic social networks. In: Proceedings of the 26th International Joint Conference on Artificial Intelligence, IJCAI 2017, pp. 2992–2998. AAAI Press (2017)