On the Use of Intrinsic Time Scale for Dynamic Community Detection and Visualization in Social Networks

Alice Albano*, Jean-Loup Guillaume*
*Sorbonne Universités, UPMC Univ Paris 06,
UMR 7606, LIP6, F-75005, Paris, France
*CNRS, UMR 7606, LIP6, F-75005, Paris, France
Email: alice.albano@lip6.fr
Email: jean-loup.guillaume@lip6.fr

Bénédicte Le Grand‡
‡Université Paris 1 Panthéon-Sorbonne, CRI
Email: benedicte.le-grand@univ-paris1.fr

***Abstract*—The analysis of social networks is a challenging research area, in particular because of their dynamic features. In this paper, we study such evolving graphs through the evolution of their community structure. More specifically, we build on existing approaches for the identification of stable communities over time. This paper presents two contributions. We first propose a new way to compute such stable communities, using a different time scale, called intrinsic time. This intrinsic time is related to the dynamics of the graph (e.g., in terms of link appearance or disappearance) and independent from traditional (extrinsic) time units, like the second. We then show how visualization both at intrinsic and extrinsic time scales can help validating and interpreting the obtained communities. Our results are illustrated on a social network made of contacts among the participants of the 2006 edition of the Infocom conference.**

Keywords*—*intrinsic time, temporal networks, stable communitites, visualization

I. Introduction

The growth of social networks has raised new challenges for the complex network research community. The interactions of their members (virtual in online networks or physical in contact networks) are indeed extremely dynamic and the structure of the corresponding graphs keeps evolving quickly over time. Detecting communities in complex networks is a usual approach to gain insight about their structure, however in the case of dynamic networks, community detection remains a widely open research topic despite promising approaches. Good techniques for computing evolving communities in dynamic networks would give a set of (overlapping) communities at each time step but also a mapping between the similar communities computed at different time slots. Both problems are difficult, especially for large or highly dynamic networks. Figure 1 illustrates the decomposition into communities of a small evolving network and their evolution (mapping) over time.

Rather than proposing another technique to compute communities on evolving networks or to map communities between consecutive time steps, we propose to study the evolving community structure using a rescaling of time. Instead of using only the natural notion of time (the second and its derivatives) in a classic way, we use a different concept to measure time in order to gain a different and complementary insight on the community structure and its evolution.

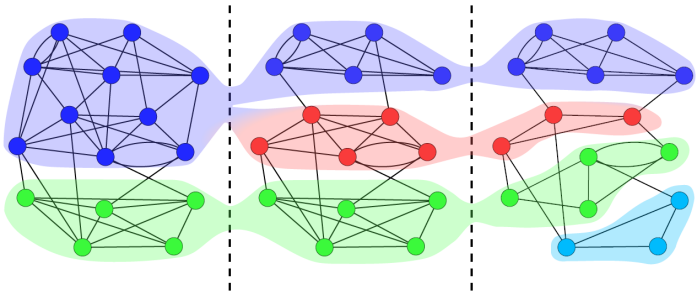

Fig. 1. Communities computed at three consecutive time steps with a possible mapping between the found communities.

Our proposal consists in making the dynamics of the network more regular by considering an alternative time unit, called intrinsic time. This new time is obtained through a non linear rescaling of usual (extrinsic) time unit such as the second. Using this new notion of time, we then search for periods of intrinsic time over which the community structure of the network remains stable enough. Our second contribution is the use of visualization technique to interpret and validate the resulting time periods and corresponding stable communities.

This paper is organized as follows. After a presentation of the state of the art on dynamic community detection in complex networks, we detail the rescaling of time that forms the base of our methodology in Section II. Section III is then dedicated to the evaluation of our proposal through a global experimentation conducted on the social network made of contacts among participants of the 2006 Infocom conference. Then, we deepen the evaluation on two specific time windows in section IV to enlighten the difference between the different notions of time. We also use some classical graph visualization techniques, and a more complex version of the algorithm used to detect stable community structure over time. We finally conclude and present the perspectives of our work on this topic.

II. Background and methodology

In this section we first present classical community detection algorithms for evolving networks. Then, we detail the algorithm we use to compute communities and a hierarchical clustering of time. Finally, we present the notion of *intrinsic time* and the way classical *extrinsic time* can be rescaled in intrinsic time.

A. State of the art on evolving communities detection

Most approaches for community detection in evolving networks can be classified in two distinct classes. First, some techniques compute communities at every time step of the evolution before mapping communities found at consecutive time steps. Since many algorithms have been defined to extract communities in static networks, the main problem of these approaches resides in mapping communities found at consecutive time steps. The second class consists in computing communities directly on the evolving network. In the following we only present some of the existing techniques, and for a more extensive presentation of the field, we refer to [1] which surveys most approaches for evolving communities computation.

Concerning the mapping problem, the first issue to be resolved concerns the instability of algorithms. Indeed, most algorithms defined to compute communities on static graphs are either non deterministic or very sensitive to modifications of the topology: very small modifications (or even no modification) can greatly modify the community structure. As a consequence, trying to map communities makes no sense if the stability problem has not been considered. Indeed most changes will not be related to actual modifications of the network, but rather to artefact of the algorithm.

In [2], the authors propose to follow communities by associating each community at time t to its most likely successor at time $t+1$, i.e., the community with which it shares the highest number of nodes. A restriction is made to compensate the instability issue. This approach has been generalized in [3] to account for more complex evolution rules such as splitting, merge, etc. Other very similar techniques have been introduced in [4]–[8] using various ways to compute the similarity between communities. Another technique, developed in parallel in many papers [9]–[12] consists in identifying core nodes and following them as the network evolves. A related, yet different technique, consists in creating a network between the communities found at different time steps which is, in turn, decomposed using classical community detection algorithms [13]–[15]. These meta communities give a natural mapping between communities.

The second research direction consists in computing communities directly on evolving networks. A solution consists in modifying the quality function to ensure that a partition computed at time $t+1$ is both good in terms of quality of the partition and similar to the partition found at time t (measured by any distance between partitions) [16]–[18]. Another direction consists in adding temporal links between every node at time t and its counterpart at time $t+1$ [19], [20]. This leads to a larger graph where the evolution is somehow encoded in classical links and classical community detection algorithms can be used. Finally, some techniques are trying to reflect each modification of the network on the community structure [21]–[24].

B. Detection of stable communities

A very promising method has been proposed by [25]. The basic principle is that if a partition in communities has been found at time t and that the network barely evolves between time t and time $t+1$, then the partition found at time t is certainly still valid at time $t+1$ (without any modification). The authors then generalize this idea and attempt to find time ranges of two or more consecutive time steps so that a single community partition can adequately represent the communities over the full time range.

To achieve this goal, the authors define the average modularity of a partition π of an evolving graph G over a set of time steps T of size n as

$$Q_{avg}\left(G,\pi,T\right)=\frac{1}{n}\sum_{t\in T}Q\left(G_t,\pi\right)$$

and then try to find partitions and sets of time steps for which the average modularity is good.

The search for such time windows and partition is done by the definition of a similarity between time windows which rely on the associated partitions. The idea is that if the graph is similar over two time steps or two time windows, then the average community decomposition of one will also be a good decomposition for the other and conversely. The authors evaluate this, for two time windows T_i and T_j whose associated partitions are π_i and π_j, by the similarity

$$Sim(T_i,T_j)=Q_{avg}(G,\pi_i,T_j)+Q_{avg}(G,\pi_j,T_i).$$

Finally, the proposed algorithm defined by the authors of [25] that we will use in the following is described in Algorithm 1. If the evolving graph is made of n time steps, the algorithm requires that n community detections are performed, one for each time step. Then, the two most similar time steps (given the similarity function Sim) are searched for, aggregated (if their modularity is higher than a given threshold) and a new average partition associated is computed (using Q_{avg}). This procedure is repeated as long as the modularity remains higher than the threshold and provides a hierarchical clustering of time steps.

Algorithm 1 Hierarchical time window merge algorithm.

```
G the initial network
L the list of potential snapshots, initially empty
for all snapshots t of G do
    Compute the communities π_{t} on the snapshot t
    Insert {t} in L
end for
while L not empty do
    Find t_i and t_j in L which maximizes Sim(T_i, T_j)
    Remove t_i and t_j of L
    Add t = t_i ∪ t_j in L
    Output that t_i and t_j have been merged
    Compute π_t the communities of G on the window t
end while
```

The algorithm used here does not require to make assumption on the graph. Nodes and links can appear and disappear during graph evolution. This algorithm can merge any pair of time windows and not only consecutive ones. The result of this algorithm is a tree of time periods. At the top of the tree, there are the longest time windows. According to the window size needed, it possible to consider different heights (or levels) in the tree. A more simple version of the algorithm only allows merges if time steps or time windows are consecutive. This version is more simple and faster to compute since the maximization of the similarity has to be computed only for pairs of consecutive time windows and not for all pairs. However, if communities can disappear and reappear later then a non adjacent version is necessary to identify such resurgence of communities. In the following we will mainly use the adjacent version for efficiency reasons but we will also present some results using the non-adjacent version.

This algorithm, in both the simple and non adjacent versions, suffers from several limitations. The first one is the choice of the appropriate level in the resulting hierarchical clustering of time periods. At the lower levels, only few merges have been performed and time windows are therefore small. On the contrary at the higher levels many merges have been made which results in fewer periods but the corresponding community structure may not be stable enough, and therefore not meaningful enough. The choice of the threshold for the modularity value below which we consider communities are not stable enough is therefore a problem. This value has been set to 0 in the original version of the algorithm, i.e. we stop merging time periods when the obtained average modularity becomes negative. This value is not always restrictive enough; in particular, when the dynamics of the network is very high, the algorithm sometimes keeps merging time periods when it should not. We decided not to focus on this problem and to keep the original value of 0 for the experiments in this paper. The direct consequence is that the algorithm does not produce a hierarchical tree of time windows but a forest. We will mainly focus on the highest levels of the trees of this forest.

Another problem comes from the fact that the algorithm for detecting stable communities over time is not deterministic, since it is based on the Louvain method described in [26]. Two computations on the same dataset can therefore provide different results. The use of non deterministic algorithms for community detection is by itself a research domain and we have no objective reason to prefer a given execution rather than another one, we therefore chose to take a random output of the algorithm. This means that we could have obtained better (or maybe worse) results using another output. Note that the work of [27] extended to the case of networks in [28] or others gives hints to deal with this non-determinism by combining multiple executions.

To conclude, the results presented in this paper correspond to one execution of Algorithm 1. We performed more than one execution and we validated that the results do not qualitatively vary much. Indeed, out of ten executions of the algorithm, we observed almost the same time windows (with modifications of a few minutes for some windows).

C. Different notions of time

The study of dynamic phenomena requires the use of a time scale. In most cases, time is an abstraction and can be considered as absolute, i.e., changes happen independently from the flow of time [29], [30]. However, if we consider time as a relative concept, time then depends on space and many techniques exist to measure it. The unit adopted by the International System of Units is the second, which is defined as the transition between two states of the caesium-133 atom [31]. This unit is therefore related to movements measured in the physical space.

In this paper we use a concept of relative time based on a network perspective, called *intrinsic time* of the network, as opposed to *extrinsic time*, measured with the second (or its derivative units like days or years). We call it *extrinsic* because its flow is independent from the changes that occur in the network. On the contrary, let the *intrinsic time* of the network be the time measured by the transition between two states of the network. The unit is thus the (spatial) change of the network, i.e., the addition or removal of one node or one link for instance. We call it *intrinsic* time because it is strongly related to network dynamics. Since the notion of change in the network is quite general, it allows for many different definitions of intrinsic time.

This notion of *intrinsic time* is not a new concept: discrete time Markov chains can be seen in the same manner. When something happens (a random event), the chain changes its state. In our graph, the event is not random, but when it occurs, we increment the intrinsic time. Asynchronous clocks use exactly the same concept as the notion of intrinsic time: when an event occurs, the time is incremented. It does not depend on the classical flow of time.

Whereas *extrinsic time* is broadly used without notice, previous works have shown that the choice between extrinsic and intrinsic time has a very significant impact both on the measurement of statistical properties of temporal networks and on observation of diffusion processes [32], [33]. Previous results on social networks suggest that intrinsic time is better at characterizing the endogenous dynamics of the network, because extrinsic time is more likely to capture exogenous patterns like day-night activity of users. In the case of diffusion processes, the use of intrinsic time allows to isolate in some way the network dynamics from the diffusion dynamics. In this paper, we use the concept of intrinsic time to study dynamic communities over time and in particular to compute stable communities more efficiently than with extrinsic time.

D. Rescaling extrinsic time into intrinsic time

Most networks are naturally measured or presented using extrinsic time in order to synchronize different measurements. In general we know, for each link, the times (in second) of its appearance and disappearance. In order to convert this extrinsic time into intrinsic time, we need first to define which evolution of the network topology will be representative of the network dynamics. In our case, we decided to use the simplest notions which are the appearance and disappearance of links. This is indeed a simple, yet efficient, measure for capturing topology evolution. Other simple notions could be used, for instance considering nodes instead of links. However in most datasets

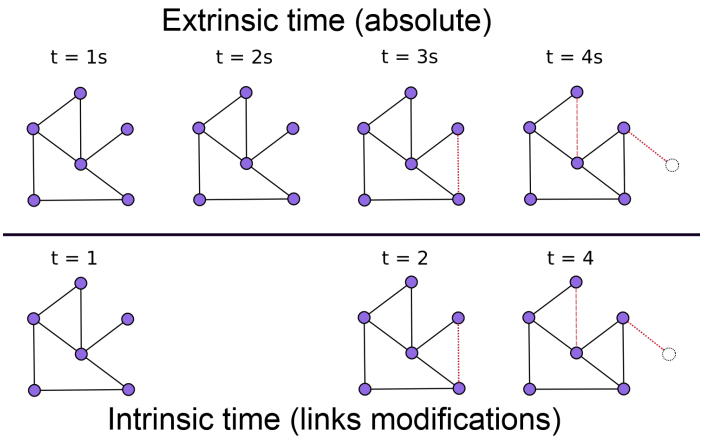

Fig. 2. Comparison between intrinsic and extrinsic times. Appearing links and nodes are dotted, disappearing links are dashed.

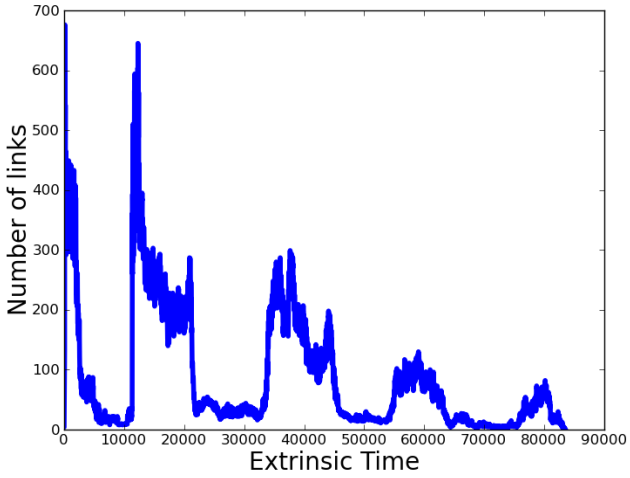

Fig. 3. Number of links as a function of (aggregated) extrinsic time in the Infocom 2006 network.

nodes are quite stable and this choice would not capture all dynamics of the network. More complex notions can also be relevant, for instance the creation or destruction of triangles which are very commonly studied in social sciences ("any friend of yours is a friend of mine"). However, as we want in this case to understand the relations between notions of time and the evolution of the community structure, using a simple definition of intrinsic time is a good starting point.

Figure 2 illustrate the relations between extrinsic and intrinsic time. In this case the time unit in intrinsic time is the appearance or disappearance of a link. In extrinsic time, there is no modification between time $t = 1s$ and $t = 2s$, therefore in intrinsic time, there is no new time step. Then, there is a link creation at time $t = 3s$ in extrinsic time which in return creates a new time step $t = 2$ in intrinsic time. Finally, at extrinsic time $t = 4s$, there is a node and link creation, and also a link disappearance. This corresponds to two time steps in intrinsic time, therefore intrinsic time $t = 3$ does not exist, unless we can order the appearance and disappearance of links.

Note that the rescaling made to convert extrinsic time to intrinsic time is not linear. Indeed, when there are many link appearances and disappearances during a short extrinsic time range, there will be many corresponding intrinsic time steps. Therefore, a burst of activity in a short extrinsic time will correspond to a long intrinsic time period. On the contrary, a long extrinsic time range with very little activity will be very short in intrinsic time.

III. Experiments

A. Dataset

We experiment our methodology on a social network dataset. This dataset is a contact network measured during the 2006 Infocom conference[1]. Participants volunteered to wear an RFID equipment. When two people wearing this equipment have been close enough, the RFID device has recorded this contact as a link between the two persons. In the dataset, nodes represent the participants of the experiment, and links therefore represent a contact (physical proximity) between participants.

[1]http://www.ieee-infocom.org/2006/

We know the instants (in seconds) when each link appears and disappears. The experiment lasted for nearly 4 days (334 015 seconds) with a total of 78 participants and 2953 different links on the aggregated graph, i.e., 2953 recordings of proximity between distinct participants. If two persons have been close during one hour then far away and close again later on, this counts for only one link. This means that nearly all of potential interactions have been observed at least one.

As the algorithm we use here is costly in terms of computation, we have reduced the number of extrinsic time steps by aggregating 4 consecutive time steps: seconds 1, 2, 3 and 4 are aggregated into instant 1, seconds 5, 6, 7 and 8 are aggregated into instant 2, etc.

Figure 3 shows the number of links as a function of (aggregated) extrinsic time in the Infocom 2006 network. We see that this network is very dynamic: there are many variations in the number of links. The measurement started at the end of the first day of the conference, and is followed by the first night, which is clearly visible (between approximately time 5000 and time 10 000) since it contains very few links which certainly corresponds to roommates. After the first night, we observe a burst of activity, corresponding to the second day (between time 10 000 and 20 000): during the second day, the number of links varies a lot, but is always quite high, with more than 3 contacts per individual on average. The burst at the beginning of the second day corresponds to the keynote session. At the end of the same day, the burst is due to the social event. The sudden increases during the days correspond to lunch break and coffee breaks. This day/night pattern is repeated on the whole dataset: night phases are very stable, with only a few links, and day phases are much more active, with many variations. Moreover, the number of links progressively decreases during the experiment: this decrease is induced by some people leaving before the end of the conference. The last day of the conference was dedicated to the workshops.

Overall, this network is very dynamic, with different phases. On this type of network, the detection of a single

community structure without taking the evolution into account makes no sense because the aggregated graph lacks a lot of information. For instance, links present during a given day are rarely present the other days. Therefore, the graph structure is completely different from one day to another, since they have very few links in common.

On the opposite, detecting stable communities on time windows can be very relevant here, since there are clearly different phases in the dynamics. Furthermore, the number of links as a function of extrinsic time gives us a good understanding of the dynamics of the network. Therefore, using the appearance and disappearance of links for the definition of intrinsic time should capture well enough the evolutions of the topology.

B. Stable communities computed and observed in extrinsic time

We use the algorithm detailed in section II to find extrinsic time windows over which communities are stable. This means that we can compute a single partition of the graph which makes sense for the whole time window. In this paper, we want to validate and interpret these windows using different (extrinsic and intrinsic) time scales.

To achieve this goal, we compute evolving communities on the original graph as well as on the graph where time has been converted from extrinsic to intrinsic time. On this converted graph, we use the same community detection algorithm to detect meaningful time windows. We do not expect to obtain the same number of time windows in extrinsic and in intrinsic time. Indeed, as the intrinsic time slows down bursty events, and aggregates stable periods, the natural intuition is to think there will be a higher number of time windows in intrinsic time during phases with many bursts.

A simple computation of stable communities in extrinsic time returns 9 large time windows at the top level of the tree. We removed all time windows whose duration is smaller than 500 time steps, i.e., 33 minutes.

Figure 4 represents these nine time windows, together with the number of links as a function of extrinsic time. The first and the fourth windows correspond to the first two nights. We expected the algorithm to detect these windows, as they correspond to long periods with very little activity and, furthermore, these periods are very different from the days before and after, which are very active. The third and fourth nights are not clearly detected which can be due to the fact that the day in between is less active and the algorithms fails to detect the difference, at least in the upper level of the tree.

During phases which correspond to days, there is much more activity, and since the graph structure evolves more quickly, we expect the time windows to be smaller. The second, third, fifth and sixth time windows are indeed much shorter, which confirms this intuition. On the opposite, the seventh window is very large, and a night phase (third night) is grouped with the following day. This result is quite surprising, because the number of links seems to vary enough to change the graph structure. The absence of separation is due to a problem of modularity threshold in the algorithm. If we take a low threshold, we may group windows during which the graph structure evolves a lot. On the contrary, using a high threshold,

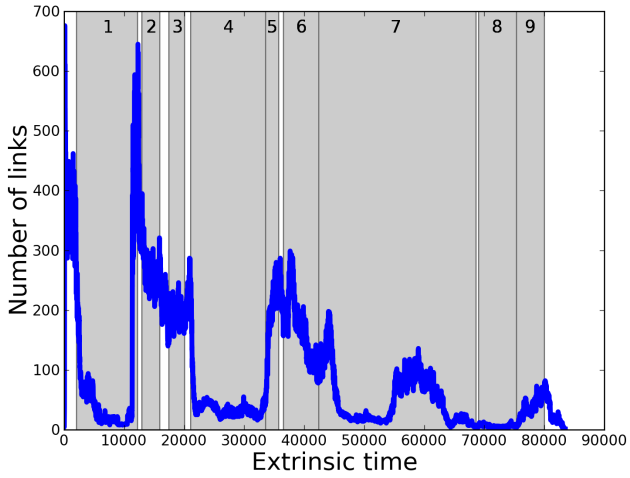

Fig. 4. Number of links as a function of (aggregated) extrinsic time in the Infocom 2006 network. Top level stable communities are represented with grey stripes.

we may miss some relevant grouping of time windows which would be relevant. We can also observe that bursts at the end of the first day, the beginning and the end of the second day (corresponding to keynote session and social event) are not included in the time windows. During these events, the graph structure changes a lot during very short periods of time, and the algorithm cannot detect a stable community structure over time.

The hierarchical structure of time windows given by the algorithm is shown in Figure 5 (the four highest levels) for the windows 2 and 7. For the window 2 (on the left of the Figure), the upper three levels contain windows of very short durations: on the first level, there is only one time step separated from the initial window (instant 15931) and, on the second level, the new window (between instants 15871 and 15930) has a size of less than a hundred time steps. The only meaningful division is on the fourth level, with the windows from 12917 to 14700 and from 14701 to 15870 (relevant windows are colored in green). If we compare these results with the evolution of the number of links shown in Figure 4, we see that there are many bursts, which are not found in the tree divisions. If we consider the bursty period covered by the extrinsic window 2, the fact that we do not find more significant divisions means that the algorithm has difficulty to compute stable community structure over this kind of network dynamics.

For the window 7 (on the right of the Figure), we observe similar results: the left branch of the tree consists only in small divisions. On the right branch, there is a significant division at time step 64374 (colored in green). When we look at this instant on the Figure with the number of links (Figure 4), it corresponds to the beginning of the night phase. The others divisions are too small to be relevant.

The algorithm using extrinsic time returns interesting results, but the tree division given by the algorithm shows that the windows we expect to find cannot be seen even in lower levels. But even if we consider all windows on a given level of the tree, there are not all relevant, and it is therefore complex

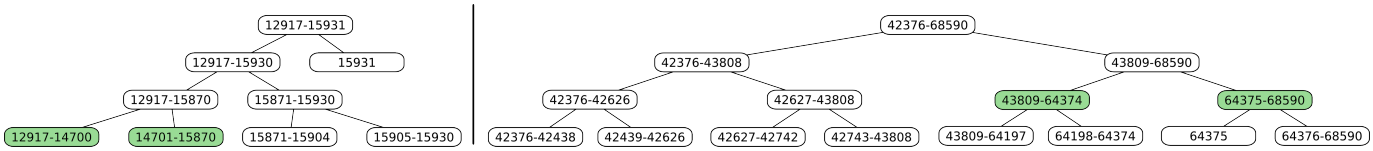

Fig. 5. Hierarchical structure for extrinsic windows 2 (on the left) and 7 (on the right).

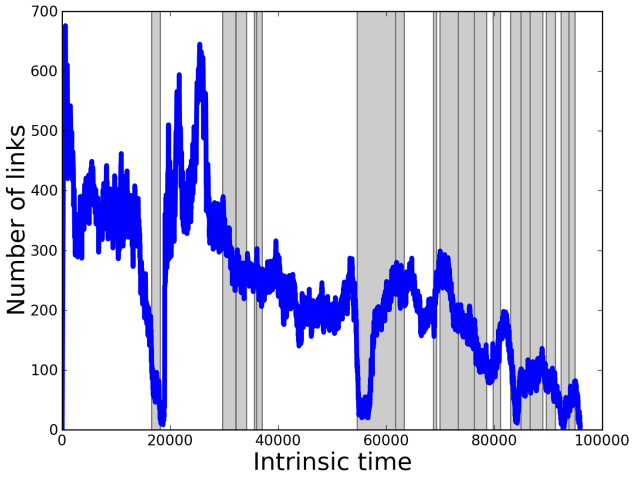

Fig. 6. Number of links as a function of (aggregated) intrinsic time in the Infocom 2006 network. Top level stable communities are represented with grey stripes.

to select the right windows. We expect that using intrinsic time will allow us to obtain directly relevant windows at the upper level of the tree, given that the intrinsic time makes the network dynamics more regular. We thus expect to get more detailed divisions for the windows identified above.

C. Stable communities computed and observed in intrinsic time

We now convert the evolution of the graph into intrinsic time, and then compute the stable communities and the associated time windows in intrinsic time. This conversion should give us more information on phases with a lot of activity, since it stretches them. We obtain 19 large time windows (of more than 500 intrinsic time steps) at the top level of the tree.

Figure 6 represents these time windows, superposed with the number of links as a function of intrinsic time. First, we can see that the number of links represented in intrinsic time has a very different behavior from its representation in extrinsic time. Night phases are naturally shortened, while day phases with a lot of activity occupy most of the intrinsic time span.

Furthermore, the meaningful windows in intrinsic time cover fewer time steps than in extrinsic time: more time windows are extracted but they are shorter. The first window covers the first night of the conference. After that, most windows correspond to day phases, with a lot of variation in the activity level. The end of the measure is covered by a lot of little time windows, while there was only one window in extrinsic time. Generally, the intrinsic time windows are more relevant than extrinsic ones, since they do not group night and day phases. Moreover, they divide the day periods more precisely: there are more intrinsic windows than extrinsic windows on each day.

The computation of stable communities over intrinsic time gives us very different results from the computation over extrinsic time. In the following section, we compare the computations over extrinsic and intrinsic times more easily on a single figure.

D. Comparison of time notions

We have seen in the previous sections that intrinsic time slows down bursty events, and aggregates stable periods, and therefore, there are more different time windows in intrinsic time. To validate this intuition, we compare the results obtained with both methods: on a curve, we represent intrinsic time as a function of extrinsic time. For each time scale, we plot the times windows. Therefore, we can compare the windows obtained in both scales. Figure 7 contains a cross-representation of curves presented in Figures 4 and 6, i.e., a representation of intrinsic time time windows as a function of extrinsic time.

First, we can see on this figure that the intrinsic rescaling of time is not homogeneous at all. Some phases are really shortened, while others become very long. In particular, the four plateaux in the curve correspond to the nights: in extrinsic time, the night lasts a long time (around 28 800 seconds which corresponds to 7200 time steps), so it spreads on many time steps but, in intrinsic time, the night is much shorter as there is very little activity (around 500 time steps). On the opposite, day phases are longer in intrinsic time, because of the important activity during conference days. This corresponds to the steepest portions of the curve

This figure also give us the possibility to compare the time windows computed in extrinsic and intrinsic times. For instance, we see that the first extrinsic window and the first intrinsic window fit quite well as both span the entire first plateau, which corresponds to the first night of the conference.

With this method, we can analyze each time window for both time scales, as we can distinguish four types of crossing. First, some windows cover the same period, as it is the case for the first and eighth extrinsic time windows and the eighth intrinsic one. In this situation, both notions of time give the same information which confirms the relevance of these windows and of the associated partitions of the network. In these cases, the threshold used by the algorithm to stop merging time windows is therefore well adapted.

Second, one extrinsic time window can be divided into several intrinsic time windows. It is the case for extrinsic windows 2 (divided in 4 intrinsic windows), 5 (divided in 2), 6 (divided in 4), 7 (divided in 5) and 9 (divided in 2). Extrinsic windows 2, 5 and 6, are all during day phases, while window

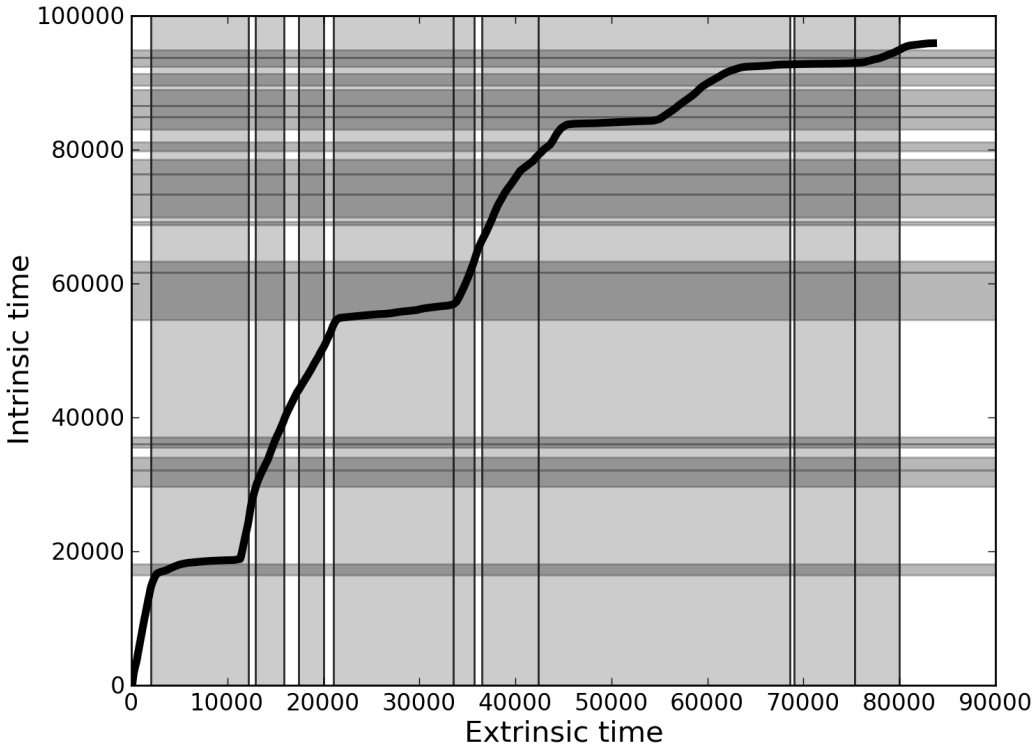

Fig. 7. Intrinsic time represented as a function of extrinsic time. Light grey vertical stripes represent extrinsic time windows, while dark grey horizontal ones represent intrinsic time windows.

7 regroups a night phase and a day phase. This division during day phases is very interesting. Indeed during a day, there is a lot of activity, and the community structure is likely to evolve quickly. The division into several intrinsic time windows can give us a more detailed information on community evolution. For the extrinsic time window 7, this division is also potentially relevant since the extrinsic window covers a night phase and a day phase. In section IV, we study the period covered by this window in more details. We also study further the period covered by the window 2, because it is quite short in extrinsic time, but still divided in four intrinsic windows.

For windows 2 and 7, the intrinsic time divides our extrinsic periods in many intrinsic windows. We have explained before that the algorithm used to detect stable communities returns a hierarchy of time windows: at the top of the tree, there are the biggest windows. It is therefore natural to think that the extrinsic windows we have found may be divided in the tree in the same manner as the intrinsic windows. If we compare our results obtained with the intrinsic time to the tree structure (Figure 5), we see that in the tree, the sub-windows do not appear clearly. For the extrinsic window 2, in the tree, the only division corresponding to intrinsic time is on the fourth level (around time 14 700). The other windows in the tree only split in very short periods. The 3 other intrinsic windows cannot be found in the tree. The case of extrinsic window 7 is very similar: we cannot find in the tree a division that corresponds to an intrinsic time window. All windows in the tree are very short. We see in these two cases that considering a lower level in the tree gives us less relevant results than directly computing stable communities over intrinsic time. Indeed, by stretching the events, the rescaling in intrinsic time allows the algorithm to compute better windows than in extrinsic time, when the dynamics of the network is too irregular.

The third possibility is to have an extrinsic time window larger than the crossing intrinsic time window. This is the case for the extrinsic window 4. In this case, the algorithm gives us a better result in extrinsic time than in intrinsic time: indeed, in intrinsic time, the night is grouped with the beginning of the following day, in spite of important topology changes.

The last possibility observed on the figure is a window with no corresponding window in the other time scale. The extrinsic window 3 has no corresponding meaningful intrinsic window. It corresponds to a period with a lot activity (as shown on Figure 4). As it is not covered by any large intrinsic time window, the quality of the stable community structure detected by the algorithm would need to be studied further.

In conclusion of this analysis, we see that in many cases, the computation in intrinsic time windows is very relevant, especially when extrinsic windows are divided into many intrinsic windows. Moreover, there are a few windows for which the crossing invalidates the computation: when a window does not cross another for instance (like extrinsic time window 3). And finally, for some windows, intrinsic time gives identical results as extrinsic time, and shows that the community structure on this period is relevant. The computation in intrinsic

time can therefore be used as a first step for validating the results found by the algorithm.

In the following, we will study in more depth the behavior of the algorithm on the extrinsic time windows 2 and 7.

IV. Focus on specific windows

A. Interpretation and validation using visualization

After computing time windows in both time scales and comparing them, we want to interpret them further. For instance, why can we observe five intrinsic time windows during the extrinsic window 7? Does this correspond to a major topology change? To answer these questions, we use visualization techniques on our graph.

We consider here the case of an extrinsic time window corresponding to several intrinsic ones (but our visualization method also works the other way). We first visualize the graph aggregated on the extrinsic time window. Then we plot the graph aggregated on each intrinsic time window (which are all smaller than the global one). In this way, we can see if there are major changes between two windows. We can also compare each little intrinsic window to the extrinsic window, and see if the division in many smaller windows is meaningful.

The validation of stable communities by visualization is only efficient when the graph is not too big: otherwise, the visualization would be cluttered. For larger graphs, it would certainly more efficient to use statistical values, like the distance between two aggregated graphs.

We have seen in the previous section that time windows computed in extrinsic or intrinsic times are often very different. In this section, we study these differences further for two extrinsic windows, the second and the seventh.

1) Extrinsic window 2: In section III, we have shown that the extrinsic window 2 was divided in four intrinsic windows. Moreover, we have seen that during the time period covered by this window (which corresponds to a day phase), there are many bursts in the number of links. We want to see if the intrinsic time windows found previously are relevant. Figure 8 represents the graph aggregated on the whole extrinsic window (top), then aggregated only on each intrinsic window (windows 3, 4, 5 and 6).

The graph aggregated on the extrinsic window is very dense, which is natural since it covers a day phase. We see that each graph for the intrinsic windows is very different from the extrinsic graph. Between intrinsic windows 3 and 4, we can observe a lot of differences in the graph structure: on the window 3, the graph is more dense. It corresponds to the window which contains the highest amount of links (Figure 6). We can also see that a part of the graph which is very dense on the first intrinsic visualization (at the top left) becomes very sparse during the second window. On the other hand, the part on the bottom right becomes a little less dense, but still quite connected.

Graphs of windows 4 and 5 are slightly more similar, but they still exhibit important differences. The part on the bottom right becomes more dense again, but the part at the top left does not evolve a lot. Finally, between graphs of windows 5 and 6, the bottom right part does not change, but in the rest of the graph, there are many link creations, which can explain the change of the community structure between these two windows.

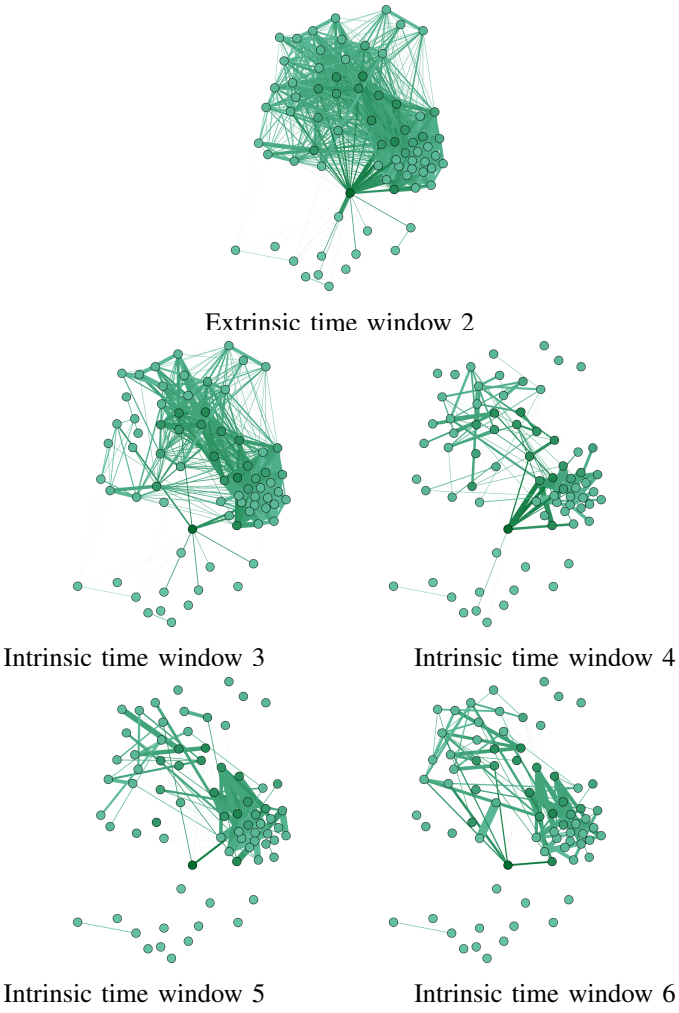

Fig. 8. Graph aggregated on extrinsic time window 2, and graph aggregated on each crossing intrinsic time window (from 3 to 6).

This visualization shows us that all the intrinsic windows found during extrinsic window 2 are relevant. Between each of them, we observe a lot of differences in the graph structure, and we see that the intrinsic division allows us to make a more detailed analysis of the evolution of community structure than in extrinsic time. On a short time period (corresponding to extrinsic window 2), the intrinsic time can still allow a meaningful time division in the community structure.

2) Extrinsic window 7: As we have seen in section III, the meaning of extrinsic window 7 was not clear in our previous results, because it groups a night phase with a day phase. Moreover, this window is divided in five intrinsic time windows. We study here if these windows are relevant. We first visualize the graph aggregated on the whole extrinsic time window, then aggregated only on each intrinsic time window. These visualizations are shown in Figure 9.

On this figure, the aggregated graph over extrinsic time is on the top. Then we see aggregated graphs on each intrinsic

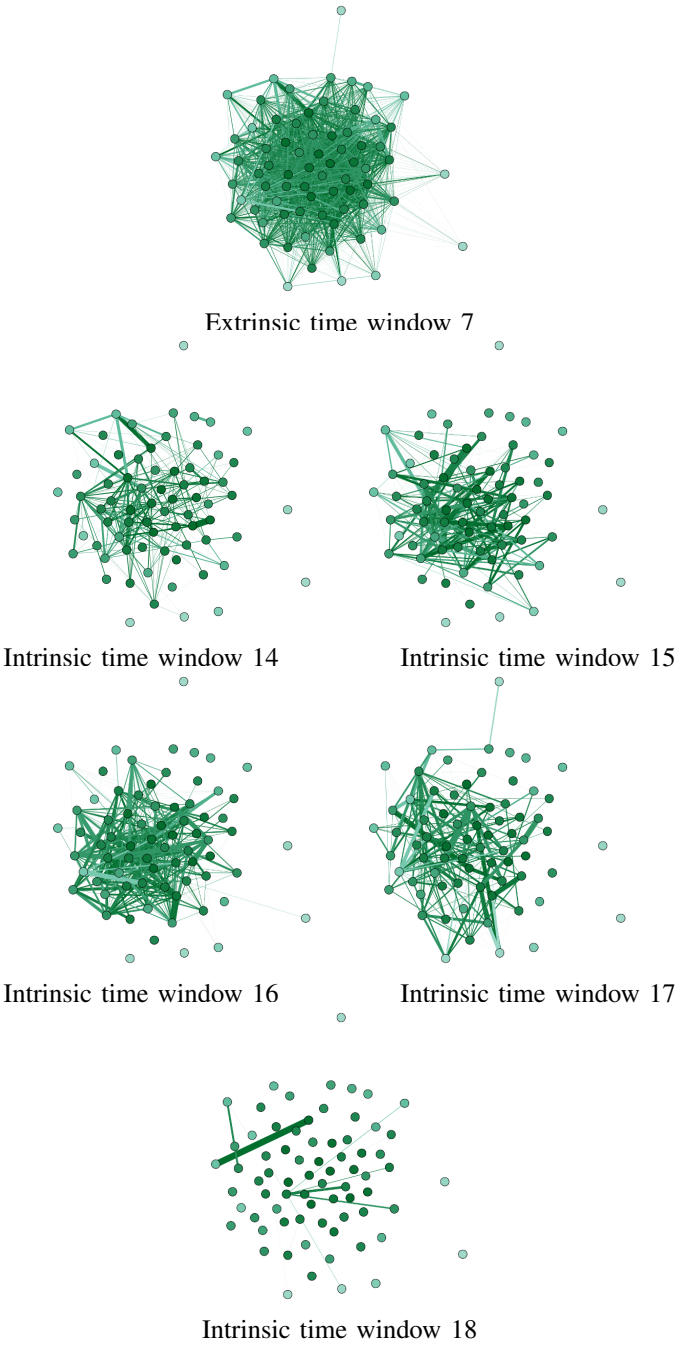

Fig. 9. Graph aggregated on extrinsic time window 7, and graph aggregated on each crossing intrinsic time window (from 14 to 18).

time window. We observe on this figure that the graph aggregated over extrinsic time window is very dense: except for two nodes which have a degree equal to 1, all nodes have many neighbors. In spite of the very small size of the graph, this visualization is very confusing. There are too many links, and we cannot see anything except that the graph is dense.

On each intrinsic time window, aggregated graphs are more readable. The first graph corresponds to the night phase, and is therefore quite sparse. On the three following graphs (intrinsic time windows 15, 16 and 17), there are more links. The difference between the two first intrinsic time windows is very clear, and validates the relevance of splitting these two periods.

Windows 15, 16 and 17 are all quite dense, but every one is different from the others. High degree nodes change between each windows, and some connected nodes become almost inactive. It is therefore not surprising that community structure changes between each of these windows. Indeed, the graph topology exhibits important evolutions and differences. Window 18 is a special case: it is very sparse, and different from every other window. This is because this window is between extrinsic windows 7 and 8. On the visualization, it covers only the end on the window 7, and is cut before the end.

We have seen with this visualization that the extrinsic time window can be cut in a more subtle and meaningful manner using intrinsic time. The graph conversion, and the new computation of time windows is therefore very relevant in this case. Visualization for both extrinsic windows confirms the value of the intrinsic time to compute stable communities on time windows.

B. *Non adjacent time windows*

In the previous section, we have detailed the interpretation of the division of an extrinsic time window into several smaller intrinsic windows. We have shown that the algorithm can be easily enhanced by using intrinsic time. In this section, we carry on our analysis of the same two extrinsic windows, but this time, we use the non-adjacent version of the algorithm: we allow the merging of time steps or time windows which are not consecutive. We study if this version of the algorithm can be useful to complete the use of intrinsic time. This non-adjacent version is useful to regroup phases that are similar, but not close in time, like for instance, the night phases. However, the complexity of the non-adjacent version of the algorithm is much greater than the consecutive version, and is too high to compute community structure on the whole graph. Therefore, we apply the algorithm only on the extrinsic windows 2 and 7, to complete the study of these time periods.

The trees obtained in this way are shown on Figure 10: the tree for the window 2 is on the left, and the tree for the window 7 is on the right. For the window 2, on the right branch of the tree, we see that at each level, only one time step is separated from the rest. We want to know if the graph structure at this time step is different from the rest. We compute the percentage of common links between the isolated time step and the surrounding time steps. We find that there are between 99% and 100% of common links between a single time step and those around it. Moreover, it is the case for every isolated time step in the tree (and it is the same for the window 7). These divisions are therefore not relevant for the community structure. On the left branch of the tree, the divisions are a little longer (around 20 time steps), but they are also very similar to the time steps around, if we look at the percentage of common links. In this non-adjacent computation of time windows for stable communities, we do not obtain relevant windows. In particular, we do not find anything close to the intrinsic division, neither do we find meaningful time windows.

For the tree of the time window 7, we have quite the same scenario: each window is separated with only one excluded time step. The similarity between this time step and the rest

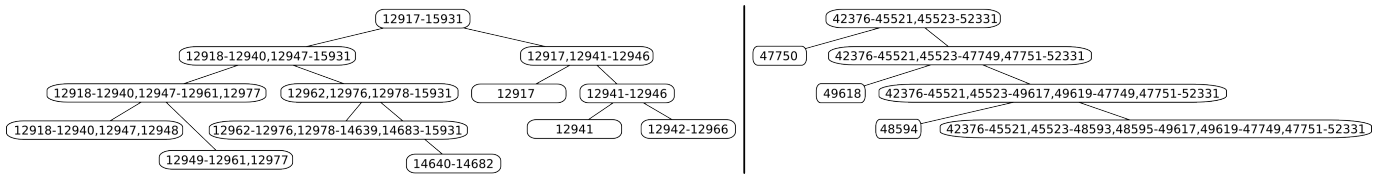

Fig. 10. Hierarchical strucutre for extrinsic windows 2 (on the left) and 7 (on the right)

is very high, and the separation of these windows does not give us information about the community structure. Like the case of the extrinsic window 2, the non-adjacent version of the algorithm does not help us to distinguish meaningful time-windows, whereas during the time period under study, there are clearly several different phases. In intrinsic time, we can finally easily compute relevant and detailed time windows, but the non-adjacent algorithm is very costly, and does not provide a good time division. This confirms the interest of the intrinsic time notion, which is both very light and more relevant.

V. Conclusion

In this article, we have proposed an efficient method for detecting stable communities in dynamic networks, by extending an existing algorithm with the notion of intrinsic time. We have converted the time in our dataset in a non linear way, and we have used this rescaling with the algorithm. This rescaling allows to "smooth" the graph dynamics, by stretching the bursts, and shortening stable phases. With this method, we obtain very promising results: the time windows found are at least as meaningful as in the extrinsic case, and very often with the adequate level of detail compared to extrinsic ones. We have proposed an easy and efficient methodology to compare the results obtained with both time scales. We have described a way to confirm the relevance of the time windows obtained in intrinsic time by using a visualization method. Finally, we have compared our results with a more elaborate version of the community detection algorithm, which allows grouping non-consecutive time windows (this version is very costly in terms of computation).

In our future work, we will study further the definition of intrinsic time. In this article, we have associated the appearance or disappearance of a link to a new time step. For our dataset, this definition is very relevant, since it captures adequately the dynamics of the network. But we can easily imagine other graphs, where it would not be adequate. For instance, in a graph with many triangles, the number of links is not enough to characterize the topology evolutions. The observation of stable community structure over intrinsic time is a good way to validate or invalidate the definition of intrinsic time used.

The study of the intrinsic time definition goes along with the study of stable community structure on other datasets. As the intrinsic time definition is based on a topology change, like the appearance or disappearance of a link for instance, the dataset must have a detailed time step (like the second). Indeed, if we have bigger time steps (like a day), there would be a lot of activity during a single time step and it would imply an artificial scheduling of links. Therefore, to be relevant, the dataset must be a link stream. Moreover, there must be a community structure in it (which is the case in most complex networks). The study of stable community structure on other datasets will show the interest and the limits of this approach according to different dataset types (social contact network, internet graph, etc.)

In this paper, for all our computations of community structure, we have used the Louvain method, which is not deterministic. Therefore, by applying the same method used here on the same dataset, we will observe differences, even if the consistency of the result have been studied. In a future work, we will combine the approach described here through multiple executions with the notion of community cores: the idea is to compute many times community structure with the Louvain method, and to place a node into a community if it was present in the community more than a given threshold.

Generally, the intrinsic time can be used for any dynamical processes: in our previous work, we have shown its interest with diffusion phenomena, and in this article, we have seen that it is also very useful for the computation of stable community structure in a dynamic network. It is therefore legitimate to think that it will give promising results in the study of dynamical processes in general, which is an open and strategic issue.

Acknowledgment

This work is supported in part by the French National Research Agency contract DynGraph ANR-10-JCJC-0202 and by the CODDDE project ANR-13-CORD-0017-01. We would also like to thank Thomas Aynaud for both his ideas, and his previous advices on our work.

References

[1] T. Aynaud, J.-L. Guillaume, E. Fleury, and Q. Wang, *Dynamics On and Of Complex Networks*, 2012, vol. 2, ch. Communities in evolving networks: definitions, detection and analysis techniques, pp. 159–200.

[2] J. Hopcroft, O. Khan, B. Kulis, and B. Selman, "Tracking evolving communities in large linked networks," in *National Academy of Sciences of the United States of America*, vol. 101, no. Suppl 1. National Acad Sciences, 2004, p. 5249. [Online]. Available: http://www.pnas.org/content/101/suppl.1/5249.full

[3] M. Spiliopoulou, I. Ntoutsi, Y. Theodoridis, and R. Schult, "Monic: modeling and monitoring cluster transitions," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM New York, NY, USA, 2006, pp. 706–711. [Online]. Available: http://portal.acm.org/citation.cfm?id=1150402.1150491

[4] S. Asur, S. Parthasarathy, and D. Ucar, "An event-based framework for characterizing the evolutionary behavior of interaction graphs," in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2007, p. 921. [Online]. Available: http://portal.acm.org/citation.cfm?id=1281290

[5] T. Falkowski, M. Spiliopoulou, and J. Bartelheimer, "Community dynamics mining," in *Proceedings of 14th European Conference on Information Systems (ECIS 2006)*. Citeseer, 2006. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.89.5334\&rep=rep1\&type=pdf

[6] D. Greene and D. Doyle, "Tracking the evolution of communities in dynamic social networks," in *Advances in Social Networks Analysis and Mining (ASONAM)*, vol. 2010, no. 2010. IEEE, 2010, pp. 1–13. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5562773

[7] M. Oliveira and J. Gama, "Bipartite graphs for monitoring clusters transitions," *Advances in Intelligent Data Analysis IX*, vol. M, pp. 114–124, 2010. [Online]. Available: http://www.springerlink.com/index/L475802466032221.pdf

[8] ——, "Understanding Clusters Evolution," in *Workshop on Ubiquitous Data Mining*, vol. D, no. 19, 2010, pp. 16 – 20. [Online]. Available: http://www.liaad.up.pt/\~{}udm/procUDMECAI2010.pdf\#page=7

[9] M. Beiró and J. Busch, "Visualizing communities in dynamic networks," in *Latin American Workshop on Dynamic Networks*, vol. 1, no. C, 2010. [Online]. Available: http://cnet.fi.uba.ar/lawdn/submissions/lawdn2010_Beiro_Busch_AlvarezHamelin.pdf

[10] Z. Chen, K. A. Wilson, Y. Jin, W. Hendrix, and N. F. Samatova, "Detecting and Tracking Community Dynamics in Evolutionary Networks," in *IEEE International Conference on Data Mining Workshops*, 2010, pp. 318–327.

[11] Q. Wang and E. Fleury, "Mining time-dependent communities," in *Latin American Workshop on Dynamic Networks*, 2010. [Online]. Available: http://cnet.fi.uba.ar/lawdn/submissions/lawdn2010_Wang_Fleury.pdf

[12] Y. Wang, B. Wu, and N. Du, "Community Evolution of Social Network: Feature, Algorithm and Model," *Science And Technology*, no. 60402011, 2008. [Online]. Available: http://uk.arxiv.org/abs/0804.4356v1

[13] Y. Chi, S. Zhu, X. Song, J. Tatemura, and B. L. Tseng, "Structural and temporal analysis of the blogosphere through community factorization," *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '07*, p. 163, 2007. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1281192.1281213

[14] T. Falkowski and M. Spiliopoulou, "Users in volatile communities: Studying active participation and community evolution," *Lecture Notes in Computer Science*, vol. 4511, p. 47, 2007. [Online]. Available: http://www.springerlink.com/index/324q127x20n4211g.pdf

[15] C. Tantipathananandh, T. Berger-Wolf, and D. Kempe, "A framework for community identification in dynamic social networks," *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '07*, p. 717, 2007. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1281192.1281269

[16] P. H. S.-Y. Chan and K. Xu, "Community Detection of Time-Varying Mobile Social Networks," *Complex Sciences*, pp. 1154–1159, 2009.

[17] A. T. R. Kumar and D. Chakrabarti, "Evolutionary clustering," in *12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006.

[18] X. Song, Y. Chi, B. Tseng, D. Zhou, and K. Hino, "Evolutionary spectral clustering by incorporating temporal smoothness," in *13th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2007, pp. 153–162.

[19] C. R. M. B. Jdidia and E. Fleury, "Communities detection and analysis of their dynamics in collaborative networks," in *IEEE ICDIM*, 2007, pp. 744–749.

[20] P. Mucha, T. Richardson, K. Macon, and M. Porter, "Community structure in time-dependent, multiscale, and multiplex networks," *science*, vol. 876, pp. 10–13, 2010. [Online]. Available: http://www.sciencemag.org/cgi/content/abstract/328/5980/876

[21] T. Dinh, I. Shin, N. Thai, and M. Thai, "A General Approach for Modules Identification in Evolving Networks," *Dynamics of Information*, vol. 40, no. 4, pp. 83–100, 2010. [Online]. Available: http://www.springerlink.com/index/N7N652L3G6181J02.pdf

[22] T. Falkowski, A. Barth, and M. Spiliopoulou, "Studying community dynamics with an incremental graph mining algorithm," in *Proc. of the 14 th Americas Conference on Information Systems (AMCIS 2008)*, 2008, pp. 1–11. [Online]. Available: http://www.tanjafalkowski.de/downloads/FalBarSpi08.pdf

[23] R. Görke, P. Maillard, and C. Staudt, "Modularity-Driven Clustering of Dynamic Graphs," *Experimental Algorithms*, vol. Cl, no. 1, 2010. [Online]. Available: http://www.springerlink.com/index/5GN1W87H48035TU1.pdf

[24] H. Ning, W. Xu, Y. Chi, Y. Gong, and T. Huang, "Incremental spectral clustering with application to monitoring of evolving blog communities," *In SIAM Int. Conf. on Data Mining*, 2007.

[25] T. Aynaud and J.-L. Guillaume, "Multi-step community detection and hierarchical time segmentation in evolving networks," in *Proceedings of the 5th SNA-KDD workshop*, 2011.

[26] V. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, 2008.

[27] E. Diday, "Une nouvelle mthode en classification automatique et reconnaissance des formes la mthode des nues dynamiques," *Revue de Statistique Applique*, vol. 19, no. 2, pp. 19–33, 1971.

[28] M. Seifi, I. Junier, J.-B. Rouquier, S. Iskrov, and J.-L. Guillaume, "Stable community cores in complex networks," in *Complex Networks*, 2013, pp. 87–98.

[29] I. Newton, *PhilosophiæNaturalis Principia Mathematica*, 1687.

[30] I. Kant, *Kritik der reinen Vernunft*, 1781.

[31] O. I. de la Convention du Mètre, "The international system of units (SI)," Bureau International des Poids et Mesures, Tech. Rep. 8, 2006.

[32] S. Heymann and B. L. Grand, "Monitoring user-system interactions through graph-based intrinsic dynamics analysis," in *Proceedings of the 7th IEEE International Conference on Research Challenges in Information Science*, 2013.

[33] A. Albano, J.-L. Guillaume, S. Heymann, and B. Le Grand, "A matter of time - intrinsic or extrinsic - for diffusion in evolving complex networks," in *In Proceedings of the International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, 2013.