Monitoring User-System Interactions through Graph-Based Intrinsic Dynamics Analysis

Sébastien Heymann, Bénédicte Le Grand

▶ **To cite this version:**

Sébastien Heymann, Bénédicte Le Grand. Monitoring User-System Interactions through Graph-Based Intrinsic Dynamics Analysis. 7th IEEE International Conference on Research Challenges in Information Science, May 2013, Paris, France. pp.1-10, 10.1109/RCIS.2013.6577695 . hal-00828778

HAL Id: hal-00828778
https://hal.science/hal-00828778v1

Submitted on 6 Sep 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Monitoring User-System Interactions through Graph-Based Intrinsic Dynamics Analysis

Sébastien Heymann
LIP6 - CNRS - Université Pierre et Marie Curie
4 place Jussieu, 75252 Paris, France
Email: sebastien.heymann@lip6.fr

Bénédicte Le Grand
CRI - Université Paris 1 Panthéon – Sorbonne
90 rue de Tolbiac, 75013 Paris, France
Email: Benedicte.Le-Grand@univ-paris1.fr

***Abstract*—Monitoring the evolution of user-system interactions is of high importance for complex systems and for information systems in particular, especially to raise alerts automatically when abnormal behaviors occur. However current methods fail at capturing the intrinsic dynamics of the system, and focus on evolution due to exogenous factors like day-night patterns. In order to capture the intrinsic dynamics of user-system interactions, we propose an innovative graph-based approach relying on a novel concept of time. We apply our method on two large real-world systems (the Github.com social network and the eDonkey peer-to-peer system) to automatically detect statistically significant events in a real-time fashion. We finally validate our results with the successful interpretation of the detected events.**

I. Introduction

A. Motivation

An information system is, like any complex system, made of interrelated elements with emergent features, i.e. which result from the interactions of the system's constituents and cannot be directly inferred from these individual constituents. In other words, a complex system cannot be reduced to the sum of its constituents; this is precisely what makes it "complex". Such systems therefore raise difficult challenges. For instance, how to design a communication network that is robust against the failure of some elements? How to guarantee that its growth will respect the initial design? One of the famous examples in the history of technology is the growth of the Internet, in which computers forward messages to one another through physical or wireless connections. Internet was initially designed to enable the communication between any connected computer inside the United States, even in the case of jamming, interference and the destruction of a large portion of computers [1]. However a fifteen-year-old boy paralyzed many of the Internet's major sites for one week in February 2000 [2]. How did such a weakness appear? The observation of the evolution of the Internet over time may have helped in preventing such issue. It would be the first step to avoid that such event happens again.

Until recently, real-world complex systems have mostly been studied as static objects, however most of these networks are not static but dynamic, as elements and connections appear and disappear over time. The main body of research is therefore of little help to monitor, track significant changes, and predict the evolution of complex networks. Revealing the underlying phenomena which lead to their evolution, and answer **how and why these networks change over time**, is of high importance to system administrators and designers, and for users of such systems in general.

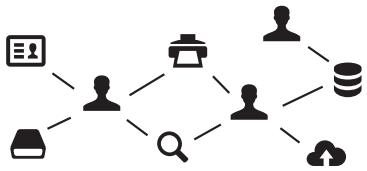

Fig. 1. Example of user-services interactions.

In the context of information system engineering, one objective is to design systems which meet the objectives of a given organization by allowing its users to perform expected tasks efficiently. However, the specification of these requirements is difficult as users goals may vary according to their contexts (e.g. the devices they use, their environment, or their location) and may evolve over time if the objectives of the organization change.

The goal of this paper is to monitor the interactions which take place in a complex system in order to have a better understanding of its operation and to detect -and ultimately anticipate- its evolution over time. We propose a generic approach to address this strategic issue, based on the analysis of the underlying graph's dynamics. **Our contribution therefore consists in proposing an innovative graph analysis methodology for the analysis of complex systems dynamics.** This methodology may be applied to any interaction system to model its behavior, understand its "normal" evolution and detect potential anomalies, which we call "events".

Complex systems may indeed be modeled as graphs where nodes represent the elements of the system and edges represent interactions between these elements. More specifically, many interaction systems may be represented as bipartite graphs when interactions occur between two types of nodes. Also called *two-mode networks*, bipartite graphs are made of nodes (i.e. elements) which belong to two sets usually called *top* and *bottom*, and in which links exist only between nodes of different sets. Natural applications in the context of information systems include client-server architectures where machines connected as clients make use of resources provided by machined connected as servers, or processes-messages graphs where each process is connected to the messages it exchanges, or file-provider graphs where each file is connected to the individuals providing it, e.g. in peer-to-peer architectures. The

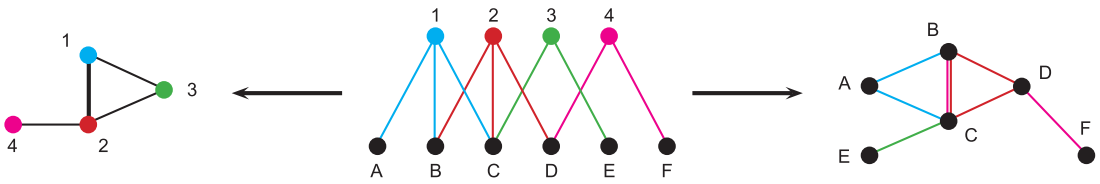

Fig. 2. Example of bipartite graph (center), together with its $\top$-projection (left) and its $\bot$-projection (right).

invocations of various services of an information system by a set of users typically correspond to such a bipartite graph, as illustrated in Figure 1 The classical approach for studying such graphs is to turn them into unipartite graphs using a projection, despite several drawbacks [3]. For instance, one can build the graphs of clients where two clients are linked together if they make use of the same server; one can build the graphs of processes where two processes are linked together if they have exchanged a message.

B. Contribution and Organization of the Paper

Our methodology consists in studying the evolution of specific (internal) features of the graph by considering a sliding time window, which may be defined with two different time units: traditional -extrinsic- time, and what we call *intrinsic time*. We show that intrinsic time provides more reliable results in terms of event detection than extrinsic time. Moreover, the width of the time window has an impact on the observed dynamics and we make interesting observations on the "optimal" value of the window's width when using intrinsic time. Finally, in addition to this contribution related to temporal aspects, we propose to focus on the most stable interactions in the system (i.e. internal links of the bipartite graph) to capture the essential system's dynamics (and neglect marginal -noisy- variations).

We apply this methodology to event detection in two large interaction networks involving human users and computer systems: the Github social network, and the eDonkey file exchange peer-to-peer (P2P) network. The Github dataset consists in the capture of the stream of public activity on Github during 18 weeks, involving 336 000 nodes and 2.2 million links. The P2P dataset contains the query logs of an eDonkey server during 10 weeks, involving 9 billion links and 365 millions nodes recorded every second.

The evolution of bipartite graphs of such sizes, as far as we know, has never been studied before. Current methods indeed fail at revealing the intrinsic dynamics of the system. We introduce a novel concept of time which allows us to see totally different dynamics. Besides, our approach is used to detect abnormal behaviors of the system, i.e. events which are statistically different from most others. The observation of internal links enables us to find novel events, which one could not see otherwise, while it confirms events detected using basic metrics. We study the impact of various time scales for the detection of events. We show that such events can be detected automatically using *Outskewer*, a statistical method that we introduced in a previous paper [27]. We finally validate the relevance of the detected events. Our approach can thus be used to monitor bipartite networks in a real-time fashion, and to raise alerts automatically when abnormal behaviors occur. The knowledge gained through these experiments can help in the design of novel monitoring methods of complex networks.

This document is organized as follows. In Section II we introduce the modeling of complex systems as bipartite graphs and we present our datasets. In Section III we introduce our approach for the monitoring of evolving user-interaction systems using a sliding window. In Section IV we describe two concepts of time and study their impacts on the characterization of the evolution of the system. In Section V we experiment various time scales and study their impacts. In Section VI we study the dynamics of user-system interactions using a specific property of graphs, called *internal links*, and we detect statistically significant events. In Section VII we show that these events can be automatically detected in a real-time fashion. We interpret them and validate their relevance on a real-world system. We finally conclude and present our perspectives in Section VIII.

II. Model of the System & Datasets

A. Bipartite Graph Model

A bipartite graph is a triplet $G = (\top, \bot, E)$ where $\top$ is the set of *top* nodes, $\bot$ is the set of *bottom* nodes, and $E \subseteq \top \times \bot$ is the set of links. Bipartite graphs do not form a particular type of networks as one could think. Quite the opposite, all complex networks have an underlying bipartite structure [4]. Graphs which do not display a bipartite structure are indeed projections of bipartite graphs. As defined in [3], the $\bot$-projection of G is the graph $G_\bot = (\bot, E_\bot)$ in which two nodes of $\bot$ are linked together if they have at least one neighbor in common in G: $E_\bot = \{(u, v), \exists x \in \top : (u, x) \in E \text{ and } (v, x) \in E\}$. The $\top$-projection $G_\top$ is defined dually, as shown in Figure 2.

B. Datasets

Github.com is an online platform created in 2008 to help developers share open source code and collaborate. Built on the Git decentralized versioning system, it facilitates the contributions and discussions by providing a Web interface. Github reached 3 million users on January 16, 2013, who collaborate on 5 million source code repositories [5].

1) Github: The Github dataset describes the complete activity between users and repositories on the platform from March 11, 2012 to July 18, 2012. We extracted the data from the Github Archive [6], which is a record of every public activity on Github. Then we built the bipartite graph of "who contributes to which repository", where nodes represent users and repositories, and where links represent any kind of activity the users have on repositories: commit and push source code, open and close issues for bug reports, comment on issues, commits or pull request (i.e. asking for a patch to be merged), create or delete branches and tags, and edit the repository wiki. We ignore the other activities: fork (i.e. repository duplication), mark repositories as favorite, and follow of the timeline of

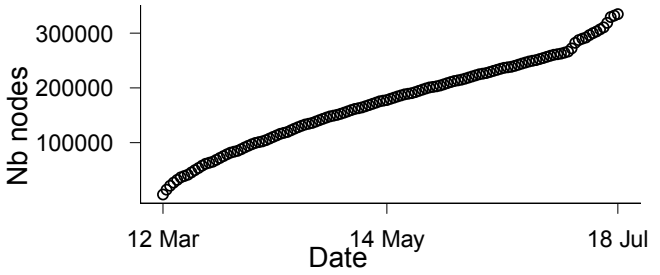

Fig. 3. Github: number of distinct nodes as a function of the total number of links observed.

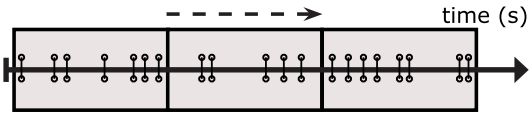

Fig. 4. Stream of appearing links split in contiguous time windows.

another user or repository. There are a bit more than 336 000 nodes and 2.2 million links.

In the Github bipartite graph, *top* nodes represent users and *bottom* nodes represent repositories. A link represents an activity between a user and a repository.

2) Peer-to-peer: The data comes from a peer-to-peer network in which the activity of an eDonkey server was collected during almost ten weeks, leading to the observation of 9 billion messages involving almost 90 million users and more than 275 million distinct files [7]. The eDonkey protocol is a half-centralized peer-to-peer protocol: client peers make requests for some files, and the eDonkey server answers the client peer with potential source peers. These server responses have been collected, thus containing a set of sources for each given file, intended for a specific client. This client then retrieves the file from sources, which is not visible in the traces because the server is no longer involved at this stage.

In the P2P bipartite graph, *top* nodes represent users and *bottom* nodes represent files. A link represents a user query for a file.

III. Our Approach: Use of a Sliding Window

We collected all data necessary to monitor the evolution of the graph, as we stored all nodes and links over time. Each link

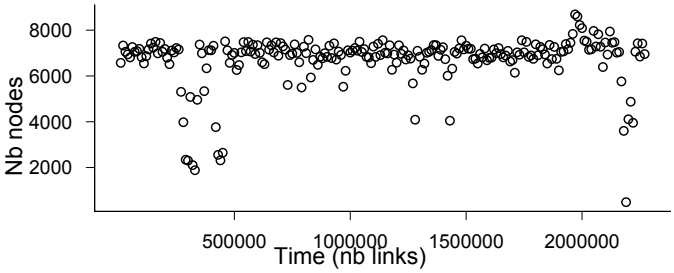

Fig. 5. Github: number of distinct nodes in the union of 10,000 consecutive links, computed every 10,000 links, as a function of the number of observed links.

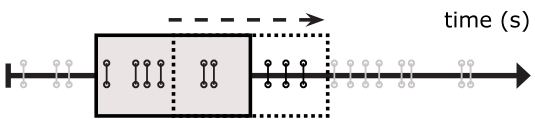

Fig. 6. Sliding time window over a stream of appearing links.

is associated with a timestamp indicating the moment when it has been observed. The data is thus a stream of observed links, ordered by their timestamp. A node is considered to appear in the graph when it is attached to an observed link for the first time. However there is no information in data about the duration of nodes and links existence. A node may indeed be observed only once even if it exists during a long period. It means that we do not observe the nodes which appeared before the beginning of the measurement and for which no link is observed during the measurement, i.e. who do not contribute or for which there is no activity during the studied period. We thus miss the registered users who are not active in the social network during the measurement, and we also miss the existing repositories on which there is no activity.

Three classical approaches exist for the study of network dynamics. The first one consists in studying the growth of the graph over time, displayed in a cumulative way. For instance the cumulative number of nodes is shown in Figure 3, where the number of nodes is plotted as a function of the total number of links observed since the beginning of the capture. This plot displays a regular growth with a regime change at the end, but we obtain little information on the underlying dynamics. The second approach consists in splitting the stream using contiguous time windows to build a series of sub-graphs, as illustrated in Figure 4. We then compute the selected statistical property on each sub-graph. For instance, the number of nodes of each sub-graph captured over time is shown in Figure 5. This plot displays a regular trend and a few spikes, however we may miss more subtle events and the precise moment of their appearance. So we use a third approach, which is the generalized version of this approach. It consists in extracting consecutive sub-graphs from a sliding time window, as illustrated in Figure 6.

Our approach is as follows: given a stream of links, we measure a given statistical property[1] of the graph observed inside a sliding time window. Let $\{e_0, e_1, ..., e_n\}$ be a stream of links. Let a sliding window of width w. If w is a function of time, e.g. a value in seconds, the sliding window is the multiset which contain all links observed during w seconds.

As far as we know, all studies on evolving networks which make use of a sliding window define its width in seconds. The apparent simplicity of this approach brings little attention because it is easy to set up and involves a common time unit. However it raises non-trivial questions (detailed in sections IV and V) which are not addressed in most studies. On the other hand, the width w of the sliding window may correspond to a number of links (independently of the time intervals between those links); in this case, the sliding window is defined as follows:

[1] A relevant property for bipartite graphs is studied in Section VI

Let $\{e_0, e_1, ..., e_n\}$ be a stream of links. Let a sliding window of width w links: $E_i = \{e_{i-w+1}, ..., e_i\}$. Any link e_i of the series belongs to $E_i, E_{i+1}, ..., E_{i+w-1}$. We compute the value of the studied property over the series of graphs, where each graph is made from the set of links in the time window: Let the graph $G_i = (V_i, E_i)$ where V_i is the set of nodes attached to the links in E_i. Let a series of graphs $G_w, G_{w+1}, ..., G_{|E|}$ where $|E|$ is the total number of links.

This sliding window, whatever the width unit used, allows us to build a time series corresponding to the evolution of the studied graph property over time.

The use of a sliding window for the analysis of graph dynamics raises several questions: which of these time units (traditional time-based or link-based) should we use to detect events? Moreover, what is the impact of the window length on the evolution of the property? In the following sections we empirically study the impact of these different concepts of time as well as various time scales on a trivial statistical property: the number of nodes observed in the network over time. We aim at determining the consequences of such choices on our ability to characterize dynamics, and to detect statistically significant events. We have found that these parameters have an important impact on the observed results.

IV. Which Time Unit?

A. Concept

Time is a controversial concept that one can see as a dimension in which changes occur in sequence. In this perspective, time is considered as absolute, i.e. changes happen independently from the flow of time [8], [9]. But if we consider time as a relative concept, time then depends on space. This debate remains open, however in practice time is experienced as relative because we can only measure it through the relative movements of bodies (in space). 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 [10]. This unit is therefore related to movements measured in the physical space.

However networks make the physical space transparent by connecting elements whatever their geographical distances. In graph theory, the distance between two nodes (also called *geodesic distance*) is indeed defined as the number of links in a shortest path connecting them. Under the hypothesis that network distances are independent from geographical distances, we consider the physical space as absolute in a network point of view. Conversely if we reject this hypothesis and correlate network distances to physical distances, the observation of such effects may hide the effects which are not related to the physical space. In the first case, measuring the distances with physical units is not relevant. In the latter case, it brings little information on the network itself. This question is difficult because effects have been found even for social networks and the Web, which are designed to abolish the physical distances between people. For instance, there is a higher probability on Facebook to be friend with someone from the same country [11]. On Github, open source developers based in North America receive a disproportionate amount of attention [12]. These studies shed light on the way the geographical location of users influences the network, but they do not address the reciprocal question of how the network allows users to be connected to one another despite geographic boundaries. Therefore existing works do not study the endogenous effects at stake in the network (i.e. which come from inside).

Notwithstanding the high potential impact of a time unit derived from the physical space, most studies use the absolute time in evolving networks: statistical properties are measured as a function of the second and its derivative units (e.g. days and years). As a consequence, they detect exogenous activities on these networks (i.e. which come from outside) [13]–[16]. For instance, click-stream data of Web traffic naturally reveal a day-night pattern in the network because of usual human activity [17]. While this finding may be of interest, it provides more information on the users activity than on the network itself. Such trends may hide the patterns which are only related to the network, preventing us to characterize the endogenous dynamics of the network.

We thus introduce a concept of relative time in a network point of view, called *intrinsic time* of the network, as opposed to the *extrinsic time*, which is a concept of absolute time. Let the *extrinsic time* of the network be the time measured using the second. We call it *extrinsic* because its flow is independent from the changes that occur in the network. 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. This unit is minimal because nothing can happen in the network between two consecutive changes. We call it *intrinsic* because time depends on the changes that occur in the network, and changes depend on such time to happen. The relation between time and space in networks is however out of scope of this paper.

Whereas the *extrinsic time* is broadly used without notice, we find out in the following section that using it has a high impact on the measurement of statistical properties of evolving networks, and on our ability to detect statistically significant events. We will see that using the *intrinsic time* avoids biases and allows us to reveal network dynamics. In the remainder of the paper, the unit of *intrinsic time* is the appearance of a link, because our datasets consists in streams of observed links.

B. Empirical Impact

We conducted our experiment on the two datasets described in Section II for various network metrics. We report the results related to the evolution of the number of nodes, because they are representative of the impact of both time concepts. We indeed obtain similar results for the following metrics, which are classical properties of networks: the evolution of the number of distinct links[2], the number of connected components[3], the average degree[4] and the maximum degree[5].

[2]The number of links is the window width, thus it is constant.

[3]Let $\mathcal{C}(G)$ be a connected component of $G(V, E)$ (where V is the set of nodes and E is the set of links): it is a connected sub-graph of G, i.e. for each pair of nodes $(u, v) \in \mathcal{C}(G)$, a path exists between u and v. The number of connected components is therefore $|\{C \in \mathcal{C}(G)\}|$.

[4]Let $d(u)$ be the degree of the node u, i.e. its number of neighbors. The average degree of the graph $G(V, E)$ is $2 \times |E|/|V|$.

[5]Let $d(u)$ be the degree of the node u. The maximum degree of the graph $G(V, E)$ is the maximum number of neighbors of nodes in the graph, i.e. $max(D), D = \{d(u), \forall u \in V\}$.

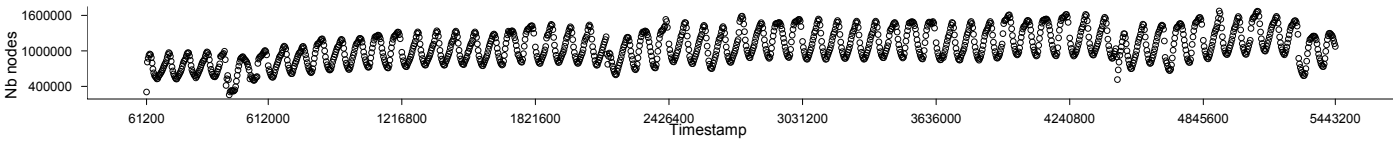

Fig. 7. P2P: number of nodes per hour.

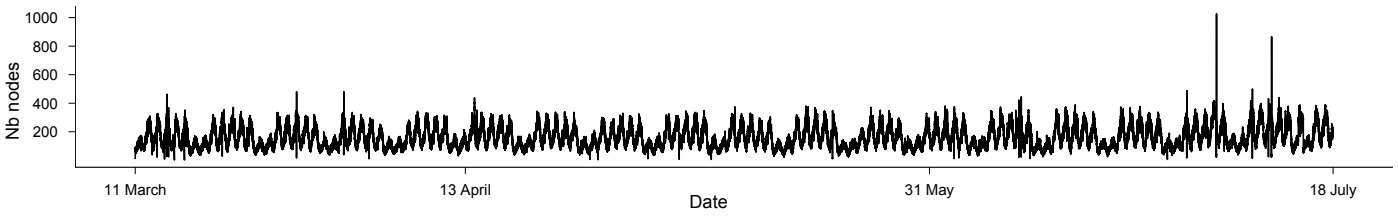

Fig. 8. Github: number of nodes in a sliding window of 10 minutes, for each minute.

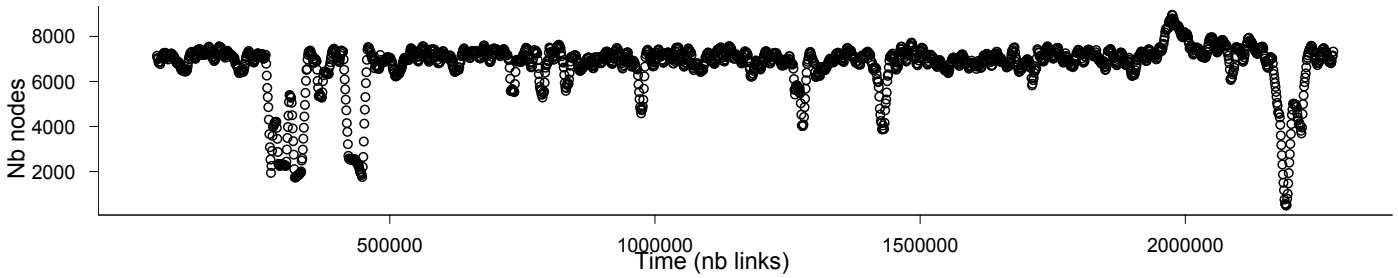

Fig. 9. Github: number of nodes in a sliding window of 10,000 links, for each set of 1000 links.

The observed number of nodes corresponds to the sum of both *top* and *bottom* nodes in our bipartite graphs. In the Github dataset, the total number of nodes therefore reflects both the number of users and the number of projects, whereas in the P2P dataset it represents the number of users and the number of requested files. The temporal evolution of this statistical property when considering *extrinsic time* reveals a daily and weekly pattern. On the contrary, the overall number of nodes is very stable when *intrinsic time* is used, which confirms that different types of dynamics are observed according to the time unit. The events which correspond to peaks may clearly be extracted from the overall trend: this shows that the graph has normal dynamics in the statistical sense (i.e. the mean value is a relevant indicator for the description of the distribution of values) and that statistical anomalies (i.e. values which deviate significantly from the mean) may be identified. Although some events also seem to appear in the curve obtained with *extrinsic time*, their characterizations are in practice much more difficult[6]. Intrinsic time therefore seems to be more relevant to perform dynamic measures.

Figure 7 and Figure 8 represent the evolution of the number of nodes over time, where the size of the sliding window is one hour and ten minutes, on the P2P and Github dataset respectively. Both plots display a daily fluctuation of the number of nodes. We thus observe more nodes during the day than during the night. The Github plot also displays a weekly fluctuation. We thus observe a greater number of nodes during the week than during the weekend.

[6]Section VII is dedicated to the interpretation of detected events.

Figure 9 represents the same property, but the size of the sliding window is 10,000 appearing links, on the Github dataset. This plot does not display such fluctuations. On the contrary, we observe that the number of nodes is globally stable with a few variations and spikes.

While we study the same property, the choice of time unit has a high impact on the resulting curves. We thus show that **so-called *observed* results are bound to an underlying concept of time**. Using the *intrinsic time* of the network instead of the traditional *extrinsic time*, we reveal totally different dynamics for the total number of nodes, which is a trivial property. We also observed different dynamics for the other properties that we have studied further in this paper. This study is hence of primary importance in metrology. Our results support the hypothesis that the intrinsic dynamics of the network is not captured by measures bound to an *extrinsic time* unit. An *extrinsic time* unit seems indeed more likely to capture the dynamics of exogenous activities on the network (i.e. which come from the outside), like the day-night and weekly patterns.

C. Discussion

The number of nodes in the Github and P2P network is very stable in the case of *intrinsic time*, which is also the case for other properties like the number of distinct links, the number of connected components, the average degree and the maximum degree[7].

[7]We have conducted experiments but do not include the corresponding figures in the paper as the results are similar.

The day-night and weekly patterns which occur in the case of *extrinsic time* while using a sliding window reveal the dynamics of users activities on the network. Hence one can see the network as an artifact which is able to capture and reveal phenomena that happen outside of it.

Based on these observations, our intuition is as follows: one should use an *intrinsic time* unit to capture an endogenous phenomenon of the network (i.e. which come from inside); one should use an *extrinsic time* unit to capture an exogenous phenomenon of the network (i.e. which come from outside). Further studies with other datasets are however necessary to draw a firm conclusion (see perspectives in Section VIII).

V. Which Time Scale?

A. Concept

In the case of a stream of links, the evolution is usually captured by the measure of statistical properties of the network over a sliding time window, as explained in Section III. Many studies assume a specific time scale for the measurement of statistical properties [18]–[21]. Some other studies address the issue of time scale. For instance, Benamara *et al.* propose a methodology to estimate the size of the observable window for a rigorous characterization of any network property [22]. Papadimitriou *et al.* compute an optimal scale for pattern detection in time series [23]. Reeves *et al.* raise the issue of downsampling time series for storage, while preserving the capacity to detect anomalies [24]. Assuming specific properties, Zhang *et al.* explore a multi-resolution approach to anomaly detection for the internet [25]. But all of these studies assume an *extrinsic time* unit.

We report our preliminary observations on the detection of events using a wide range of time scales on real-world networks, for both *intrinsic* and *extrinsic time*.

B. Empirical Impact

We have studied the variation of different metrics at various time scales for both extrinsic and intrinsic times, i.e. for different sizes w of the sliding window. We report the results for the evolution of the number of nodes, because this property is representative of the time scale's impact.

1) Extrinsic Time: We computed the number of nodes as a function of time for a sliding window of size $w = 10$ minutes (Figure 8), 1 hour (Figure 12), 12 hours (Figure 11 and 24 hours (Figure 10) on the Github dataset. Each plot for w from 10 minutes to 12 hours clearly exhibits a daily trend. A weekly trend is also observable for all studied w. These patterns are exogenous activities as explained in Section IV. Spikes appear clearly for w equal to 10 minutes and 1 hour; they are less extreme for $w = 12$ hours, and most of them have disappeared for $w = 24$ hours. Surprisingly, the two remaining spikes for $w = 24$ hours are more extreme than for smaller w.

We expect that a large window width smoothes the resulting curve. The plot corresponding to a 24 hours window (Figure 10) confirms this intuition: a regular variation can be observed, which looks more disrupted with the 12 hours window (Figure 11) and 1 hour window (Figure 12). This "noise" is due to users daily activities on the system; 5 oscillations may be identified during the week and 2 oscillations during the week-end, see Figure 13. The plot using 24 hours window therefore only reflects the Monday to Friday period, displaying lower user activity during the week-end. However we may observe the appearance of events or the growth of their amplitude (positive peaks) when the size of the window increases (circled spikes in these figures). The consequences of these results are the following:

- Selecting a large window size in order to smooth the curve is not a good option, as it modifies the shape of the plot and changes the perception of events.
- Selecting a very small window size in order to detect all events is not a good option either, as events taking place at higher time scales are missed. Moreover, this approach is extremely costly in terms of computing time.

As explained below, the same statements hold for curves based on intrinsic time: the curve is smoother when the size of the window increases. However an event (increasing peak) appears also.

2) Intrinsic Time: We computed the number of nodes as a function of time for a sliding window of size $w = 50{,}000$ links (Figure 14) and 1000 links (Figure 15). We observe in these figures that the global trend and the regime changes (i.e. sudden changes of the mean of the time series) are similar for all studied w. Spikes observed on small w values disappear progressively when w increases.

Our intuition led us to set a large w for removing non-significant events in order to smooth the global trend while keeping significant events. But we discovered that this strategy alters the trend because spikes not present for smaller w can appear. One should thus consider the duration of expected events to set the size of the sliding window accordingly, and consider the results to be valid for the specific time scale only. In our data, $w = 10,000$ is a good tradeoff to observe all events, see Figure 9.

In this section we have discussed which concept of time is relevant to characterize the intrinsic dynamics of the system. We have also seen that there is no optimal time scale for the characterization of events. In the following section we propose a specific property for studying the stability of bipartite graphs and therefore monitor user-system interactions.

VI. Monitoring Dynamics of User-System Interactions

The property we suggest to consider for the monitoring of the dynamics of user-system interactions is the number of *internal links*. The intrinsic bipartite notion of *internal links* has been introduced recently and studied for static networks [26] to bring novel insights on the characterization of these networks. An internal link is such that its removal does not change the projection of the graph for a given set of nodes, either *top* or *bottom*. In the example of an information system of users (i.e. *top* nodes) interacting with services (i.e. *bottom* nodes), an internal link from the *top* (resp. *bottom*) point of view is a link which is not mandatory to connect two users (resp. services) in the corresponding projection. One can interpret internal links as a measure of links redundancy.

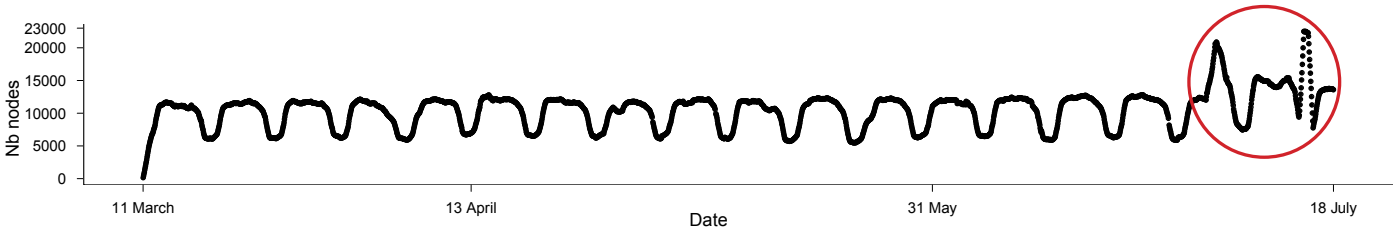

Fig. 10. Github: number of nodes in a sliding window of 24 hours, for each hour. Significant spikes are circled.

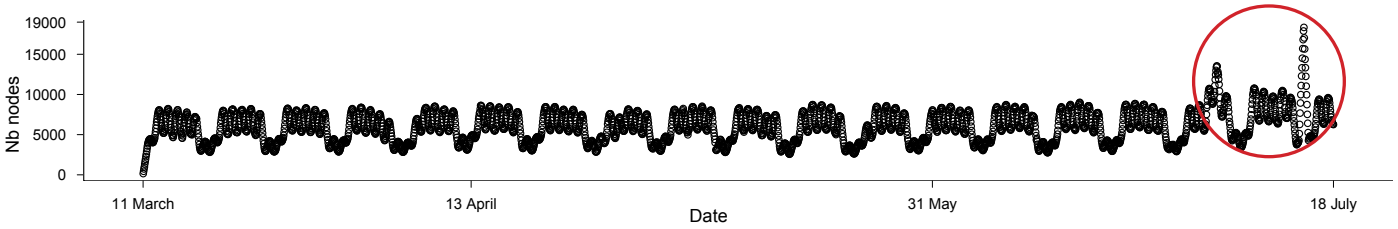

Fig. 11. Github: number of nodes in a sliding window of 12 hours, for each hour. Significant spikes are circled.

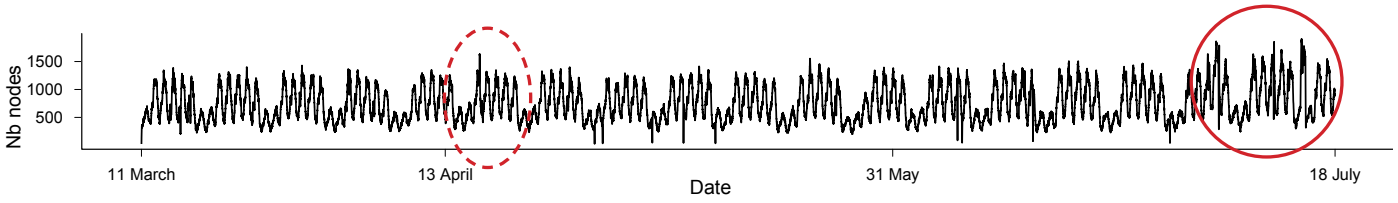

Fig. 12. Github: number of nodes in a sliding window of 1 hour, for each 5 minutes. Spikes which are significant at larger scales are circled. We zoom on the dotted area in Figure 13

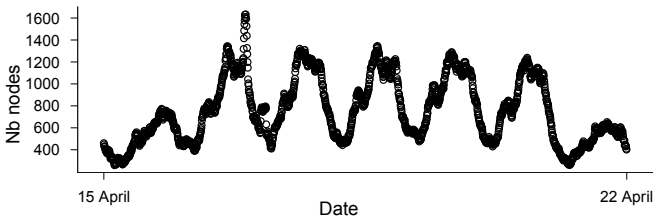

Fig. 13. Github: number of nodes in a sliding window of 1 hour, for each 5 minutes. Zoom during the 6th week.

$\top$-internal (resp. $\bot$-internal) links are links which may be removed from E without altering the $\top$-projection (resp. $\bot$-projection), as shown in Figure 16. Let $(u, v) \in \bot \times \top$ with $(u, v) \in E$ and let $G' = G - (u, v)$, (u, v) is a $\bot$-internal link if and only if $G_\bot = G'_\bot$ where $G'_\bot$ is the $\bot$-projection of G'. $\top$-internal links are defined dually.

Whereas it has been shown that internal links are able to capture interesting statistical properties of bipartite networks, this notion has never been used for the study of evolving networks. We measure the number of $\top$-internal links using a sliding window of $10,000$ links[8]. We observe on Figure 17 that the number of $\top$-internal links is globally stable around 2400 links, thus 24% of links inside the sliding windows are internal links. This result is interesting because it provides us a characterization of a **normal behavior of the system**. We also observe significant events with fast increases and decreases of the values which are statistical anomalies of the system's behavior. These events are however not new to us: we find them also in the evolution of the number of nodes, or in the evolution of the number of distinct links (which plot is not shown because similar to the number of nodes). So the number of internal links is proportional to the number of distinct links.

This led us to study the **normalized number of internal links**, which is the number of internal links divided by the number of distinct links in the sliding window. This property gives the ratio of internal links observed in the sliding window. This ratio is different from the ratio we can compute on the previous property because only distinct links are counted. It is relevant because the measure of internal links is independent from the fact that more than one link exist between two nodes. We annotate the plot on Figure 18 with a rectangle around each set of abnormal values in the plot, or which are abnormal in other plots, and we label the events with a capital letter.

We identify one new event, the small event *A*, while events *E* and *G* have disappeared. The event *I* is interesting: it is the only one with an increasing spike on the plot of the number of internal links, and it is also revealed by the plot of the maximum degree (not shown in this paper). We interpret this event in Section VII. Moreover, we discover that all decreasing spikes in the plot of the number of links are instead increasing in this plot. The ratio of internal links increases when the

[8]This is the size selected in Section V.

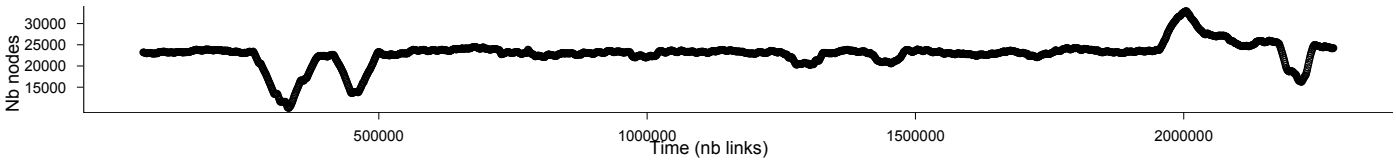

Fig. 14. Github: number of nodes in a sliding window of $w = 50,000$ links, for each 1000 links.

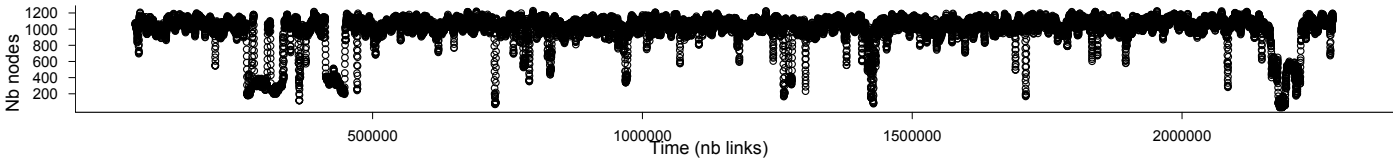

Fig. 15. Github: number of nodes in a sliding window of $w = 1000$ links, for each 100 links.

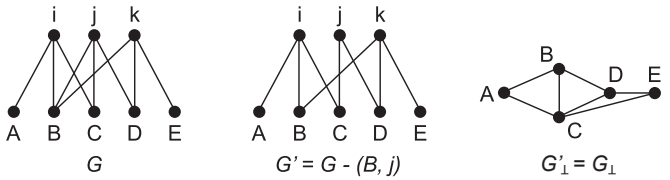

Fig. 16. Example of $\perp$-internal link. Left to right: a bipartite graph G, the bipartite graph G' obtained by removing link (B, j) from G, and the $\perp$-projection of them. As $G'_\perp = G_\perp$, (B, j) is a $\perp$-internal link of G.

number of distinct links decreases. This effect is due to an intrinsic bias of this property, which counts as internal links the links attached to nodes of degree equal to 1[9]. We finally remove this bias by ignoring the nodes of degree 1. We thus obtain the proportion of internal links among the links that connect nodes which have at least two neighbors. We see in Figure 19 that this filter removed most events, thus keeping the most significant connections.

In conclusion, we have seen that a property based on the measures of internal links reveals events of a novel kind, and confirms known events. In the following section we see how to detect such events automatically in a real-time fashion.

VII. Real-Time Event Detection

A. Methodology

How can we automatically and reliably detect events in a real-time fashion? We propose in this section to use the *Outskewer* method which we developed recently [27]. *Outskewer* is indeed able to detect statistically significant outliers (i.e. values which deviate significantly from the remainder of the values) in samples and in time series. It is easy to interpret because values are classified as *outliers*, *potential outliers* or *not outliers*. The class to assigned values is *unknown* when there is no normal behavior in the sample. This method is also easy to use because it requires no prior knowledge on data, and the only parameter is the width of the time window for time series over which the outliers are detected. This width may be different from the one used to measure the property. An implementation of the method can take a stream of values as input for a real-time monitoring.

[9]A node of degree 1 is a node with 1 single neighbor in the graph.

We applied our method on the evolution of the normalized number of internal links. In Figure 18, points of the curve are colored as a function of their outlying class: red for *outliers*, orange for *potential outliers*, blue for *not outliers*, green for *unknown*. We observe that *Outskewer* is able to detect automatically all events we identified manually. They are either identified by a set of *outliers*, or by a set of *unknown* values. The latter case happens when there is no normal behavior in the related sliding windows, but we consider that something unexpected happens at this moment. We can thus use this method to detect events automatically, allowing us to monitor the system in a real-time fashion.

B. Results and Interpretation of Events

The methodology we presented for the automatic detection of events in the graph dynamics allows us to identify some events associated to specific parts of the curve (which correspond to groups of values of the studied property, see Figure 18). Each value of the time series represents the property of the sub-graph which is visible within the corresponding window. Several steps are then needed to interpret the detected event and check the validity of our detection method:

1) We extract the sub-graph associated to the detected event and analyze it by computing the normalized number of $\top$-internal links.
2) We perform a mapping between the event's intrinsic time and the corresponding extrinsic -absolute- time, as this information is available in Github data.
3) We look for abnormal interactions between users and the system at the time of the event.

For example, we found out that event I in Figure 18 is correlated with a sudden increase of the maximum node degree in the graph. We also discover in the 1000 nodes sub-graph that the *node-migrator-bot* project[10] interacts with 95 users, which is an unusually high number of neighbors for a node in the graph within a window time. When looking at the Web page of this project on Github, we learnt that this project aims at monitoring the updates of another project, called *NodeJitsu*[11]. We discover that *node-migrator-bot* sent a *PullRequestEvent* message on June 26th, 2012 at midnight to

[10]https://github.com/node-migrator-bot
[11]https://nodejitsu.com/

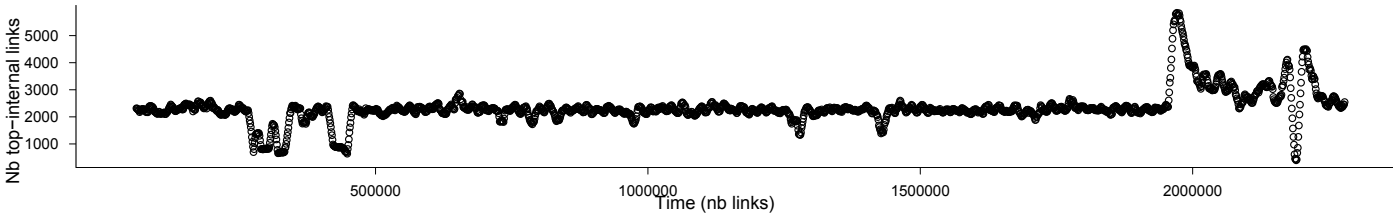

Fig. 17. Github: number of $\top$-internal links in a time window of width $w = 10,000$ links, for each 1000 links. Colors represent the outlying class of values, see Section VII.

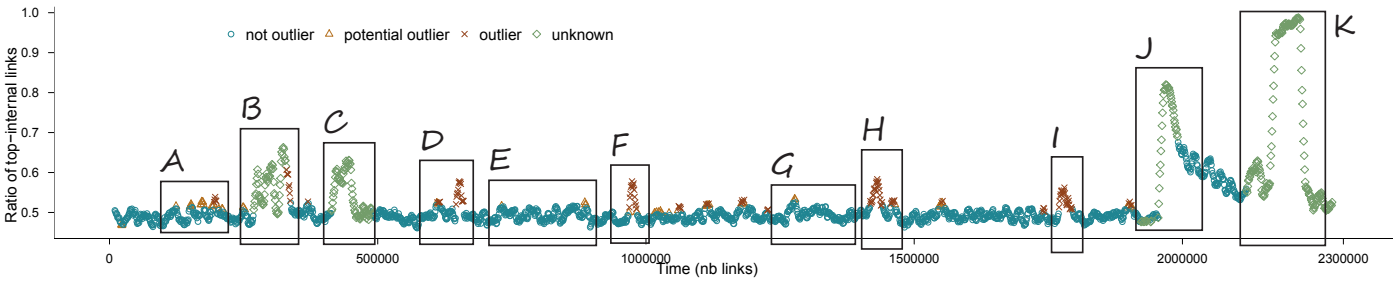

Fig. 18. Github: number of $\top$-internal links divided by the total number of distinct links in the time window of width $w = 10,000$ links, for each 1000 links. Events are circled and labeled by a letter. Colors represent the outlying class of values, see Section VII.

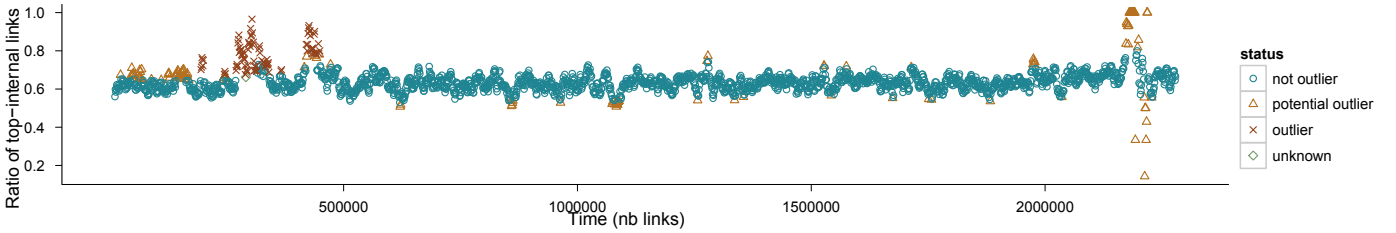

Fig. 19. Github: ratio of $\top$-internal links among the links connecting nodes of degree > 1 in a time window of width $w = 10,000$ links, for each 1000 links. Colors represent the outlying class of values, see Section VII.

the other projects on the Github platform which use an older version of *NodeJitsu*. This bot thus manages the dependencies of other projects with *NodeJitsu*, providing them a patch to apply on their own code. This observation explains the increase in the maximum degree in the graph: we recall that a link in the graph represents an interaction between a project and a user. In this case there are many interactions between the bot and users who own a project which needs an update.

Let us also interpret the event J in Figure 18. This event is also correlated to a sudden increase in the maximum degree of the sub-graph. We discover in the 1000 nodes sub-graph that the *Try-Git* project interacts with 506 users, which explains this high degree. We find out on this project's Web page that it is a tutorial for Git, one of Github's underlying tools; the first action required from the user in this tutorial is to create a clone with a new project (by sending this user a *CreateEvent* message). The instant of the event detected by our tool corresponds to the moment when *Try-Git* was made public, on July 4th, 2012 at 5 pm (this information was confirmed by a post on the Github.com blog[12]).

We showed with the two example interpretations above that the events automatically detected with our method correspond indeed to extraordinary activity on the Github platform and that they actually represent events. All the events detected on the curve could be interpreted this way, demonstrating the validity of our monitoring of the user-system interactions dynamics.

[12]https://github.com/blog/1183-try-git-in-your-browser

VIII. Conclusion and Future Work

A. Conclusion

We proposed in this paper a novel approach for the characterization and monitoring of user-system interactions and especially their evolution over time. We model these interactions as a bipartite graph, where *top* and *bottom* nodes represent users and system elements respectively, and links correspond to the interaction between these two types of nodes. We study the evolution of these graphs to characterize normal behavior and to detect abnormal dynamics, through the measurement of specific statistical properties over a sliding window. This approach is easy to set up but raises difficult questions which are barely addressed in the literature. Which time unit should we use? Using an absolute reference of time such as seconds, the current body of research fails at capturing the intrinsic dynamics of the network and focuses on evolution due to exogenous factors (e.g. day-night patterns). We thus introduce a novel concept of time, called the *intrinsic time* of the network based on link appearance, which reveals

totally different dynamics. Then, which time scale should we set to calibrate the measurement, i.e. which width for the sliding window? We see that there is no optimal time scale because events can appear at various scales, which is counter-intuitive. The smallest resolution is thus not able to capture all events. Using this approach, we successfully capture the normal behavior of the system, and detect abnormal evolution.

We applied our approach on two large real-world systems: the Github social platform where users interact with coding projects, and a peer-to-peer system where users search for files to download. We monitored these systems using a property called the *internal links* which is able to capture the core interactions and reveal significant events.

We finally monitored the system in a real-time fashion using the *Outskewer* method, to raise alerts automatically when abnormal behaviors occur.

B. Future Work

In the future, we will consider the combination of this monitoring of user-system interaction dynamics with an innovative approach in method engineering, called *intention mining* [28]. Intention mining aims at discovering users current and future intentions from the analysis of activity traces. The joint use of both methods could be applied to various types of information systems, in particular pervasive information systems [29], [30].

We will also dig further in the use of *intrinsic time* to explore its potential and its implications for the study of networks. In particular, we have seen that the properties computed from the Github and P2P network are very stable. Such phenomenon may be a property of our datasets, however it could also be a side effect of the way we observe the property. The time window is indeed of fixed width w, thus it contains at most w distinct links and $2 \times w$ nodes. Further studies are required to determine if such phenomenon is necessarily caused by the method of observation.

Finally, events are sometimes difficult to interpret. Data visualization is a promising approach to the characterization of events. A few network visualization software like Gephi [31] and VIENA [32] provide a graphical user interface to explore temporal networks, but they are not designed for event analysis. We will consider novel techniques of visual investigation.

REFERENCES

[1] B. M. Leiner, V. G. Cerf, D. D. Clark, R. E. Kahn, L. Kleinrock, D. C. Lynch, J. B. Postel, L. G. Roberts, and S. S. Wolff, "A brief history of the internet," *Computing Research Repository*, vol. cs.NI/9901, 1999.

[2] A.-L. Barabási and M. Aldana-Gonzales, "Linked: The New Science of Networks," *The Journal of Artificial Societies and Social Simulation*, vol. 6, pp. 71–72, 2003.

[3] M. Latapy, C. Magnien, and N. Del Vecchio, "Basic notions for the analysis of large two-mode networks," *Social Networks*, vol. 30, no. 1, 2008.

[4] J.-L. Guillaume and M. Latapy, "Bipartite structure of all complex networks," *Information Processing Letters (IPL)*, vol. 90, no. 5, 2004.

[5] Github press page. [Online]. Available: https://github.com/about/press

[6] Github archive. [Online]. Available: http://www.githubarchive.org

[7] F. Aidouni, M. Latapy, and C. Magnien, "Ten weeks in the life of an edonkey server," in *Proc. IEEE Sixth International Workshop on Hot Topics in Peer-to-Peer Systems (Hot-P2P 2009)*. IEEE, 2009.

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

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

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

[11] J. Ugander, B. Karrer, L. Backstrom, and C. Marlow, "The anatomy of the facebook social graph," Nov. 2011.

[12] Y. Takhteyev and A. Hilts, "Investigating the geography of open source software through github," 2010.

[13] T. Aynaud and J.-L. Guillaume, "Multi-step community detection and hierarchical time segmentation in evolving networks," 2011.

[14] A. Panisson, A. Barrat, C. Cattuto, W. V. den Broeck, G. Ruffo, and R. Schifanella, "On the dynamics of human proximity for data diffusion in ad-hoc networks," *Ad Hoc Networks*, 2011.

[15] J. ichi Takeuchi and K. Yamanishi, "A unifying framework for detecting outliers and change points from time series," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, pp. 482–492, 2006.

[16] J. Whitbeck, M. D. de Amorim, V. Conan, and J.-L. Guillaume, "Temporal reachability graphs," pp. 377–388, 2012.

[17] M. R. Meiss, F. Menczer, S. Fortunato, A. Flammin, and A. Vespignan, "Ranking web sites with real user traffic," in *Proc. ACM International Conference on Advances in Social Networks Analysis and Mining (WSDM'08)*. ACM, 2008.

[18] C. C. Aggarwal, Y. Zhao, and P. S. Yu, "Outlier detection in graph streams," in *International Conference on Data Engineering*, 2011, pp. 399–409.

[19] L. Akoglu, M. McGlohon, and C. Faloutsos, "Event detection in time series of mobile communication graphs." in *Army Science Conference*, 2010.

[20] M. Latapy, C. Magnien, and F. Ouedraogo, "A radar for the internet," in *IEEE International Conference on Data Mining*, vol. abs/0807.1, 2008, pp. 901–908.

[21] Q. Zhao, T. yan Liu, S. S. Bhowmick, and W. ying Ma, "Event detection from evolution of click-through data," in *Knowledge Discovery and Data Mining*, 2006, pp. 484–493.

[22] L. Benamara and C. Magnien, "Estimating properties in dynamic systems: The case of churn in p2p networks," in *IEEE Conference on Computer Communications Workshops, INFOCOM Wksps*, 2010.

[23] S. Papadimitriou, "Optimal multi-scale patterns in time series streams," in *Proc. ACM International Conference on Management of Data (SIGMOD'06)*. ACM, 2006, pp. 647–658.

[24] G. Reeves, J. Liu, S. Nath, and F. Zhao, "Managing massive time series streams with multiscale compressed trickles," *Proceedings of The Vldb Endowment*, vol. 2, pp. 97–108, 2009.

[25] L. Zhang, Z. Zhu, K. Jeffay, J. S. Marron, and F. D. Smith, "Multi-resolution anomaly detection for the internet," 2008.

[26] O. Allali, L. Tabourier, C. Magnien, and M. Latapy, "Internal links and pairs as a new tool for the analysis of bipartite complex networks," *Social Network Analysis and Mining*, 2012.

[27] S. Heymann, M. Latapy, and C. Magnien, "Outskewer: Using skewness to spot outliers in samples and time series," in *Proc. IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012)*. IEEE, 2012.

[28] C. Hug, R. Deneckere, and C. Salinesi, "Map-tbs: Map process enactment traces and analysis," in *Research Challenges in Information Science (RCIS), 2012 Sixth International Conference on*, may 2012, pp. 1 –6.

[29] G. M. G. Panos E. Kourouthanassis, "A design theory for pervasive information systems," in *Proc. 3rd Int. Workshop on Ubiquitous Computing (IWUC'06)*, 2006, pp. 62–70.

[30] S. Najar, M. Pinheiro, C. Souveyet, and L. Steffenel, "Service discovery mechanism for an intentional pervasive information system," in *Web Services (ICWS), 2012 IEEE 19th International Conference on*, june 2012, pp. 520 –527.

[31] M. Bastian, S. Heymann, and M. Jacomy, "Gephi: An open source software for exploring and manipulating networks," in *Proc. AAAI International Conference on Weblogs and Social Media (ICWSM'09)*, 2009.

[32] F. Windhager, L. Zenk, and P. Federico, "Visual enterprise network analytics - visualizing organizational change," *Procedia - Social and Behavioral Sciences*, vol. 22, pp. 59–68, 2011.