Suivi de la dynamique intrinsèque des interactions entre utilisateur et SI

Sébastien Heymann* — **Bénédicte Le Grand****

* *Université Pierre et Marie Curie* – CNRS – *Laboratoire d'Informatique de Paris 6 – 4 place Jussieu, 75252 Paris, France*
sebastien.heymann@lip6.fr
** *Université Paris 1 Panthéon Sorbonne – Centre de Recherche en Informatique – 90 rue de Tolbiac, 75013 Paris, France*
Benedicte.Le-Grand@univ-paris1.fr

RÉSUMÉ. *Le suivi de l'évolution des interactions utilisateur-système est de première importance pour les systèmes complexes et les systèmes d'information en particulier, notamment pour déclencher des alertes automatiquement quand survient un comportement anormal. Cependant, les méthodes actuelles ne parviennent pas à capturer la dynamique intrinsèque du système et font apparaître des évolutions dûes à des facteurs externes comme les cycles jour-nuit. Dans le but de capturer la dynamique intrinsèque des interactions utilisateur-système, nous proposons une approche innovante à base de graphes reposant sur une nouvelle conception du temps. Nous appliquons notre méthode à un grand système réel (le réseau social Github.com) pour détecter automatiquement des événements statistiquement significatifs en temps réel. Nous validons enfin nos résultats par l'interprétation réussie des événements détectés.*

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 a large real-world system (the Github.com social network) to automatically detect statistically significant events in a real-time fashion. We finally validate our results with the successful interpretation of the detected events.*

MOTS-CLÉS : *surveillance, système d'information, détection d'événement, système complexe, graphe biparti, dynamique, temps-réel, métrologie, réseaux*

KEYWORDS: *monitoring, information system, event detection, complex system, bipartite graph, dynamics, real-time, metrology, networks*

1. Introduction

1.1. *Motivation*

Un système d'information (SI) est, comme tout système complexe, composé d'éléments inter-connectés ayant des propriétés émergentes, c'est-à-dire qui résultent des interactions entre les composants du système et ne peuvent être inférées des seules propriétés des composants. En d'autres termes, un système complexe n'est pas réductible à la somme de ses parties ; c'est précisément ce qui le rend « complexe ». De tels systèmes soulèvent dès lors des défis ambitieux. Par exemple, comment concevoir un réseau de communication capable de résister à la défaillance d'une partie de ses éléments ? Comment garantir que sa croissance respectera les règles de conception initiales ?

Jusqu'à récemment, les systèmes complexes réels ont principalement été étudiés comme des objets statiques, alors que la plupart d'entre eux sont dynamiques : des éléments ou connexions apparaissent ou disparaissent au cours du temps. La plupart des recherches sont donc peu utiles pour analyser un système complexe, détecter ses variations significatives, et en prédire l'évolution. Révéler les phénomènes sous-jacents menant à leur évolution, et déterminer **comment et pourquoi les systèmes changent au cours du temps**, est d'une importance capitale pour les administrateurs et les concepteurs de systèmes, et pour les utilisateurs en général.

Dans le contexte de l'ingénierie des SI, un objectif est de concevoir des systèmes répondant aux buts d'une organisation donnée, en permettant aux utilisateurs d'effectuer les tâches attendues de manière efficace. Cependant, la spécification de ces besoins est difficile car les buts des utilisateurs peuvent varier selon le contexte (par exemple le terminal utilisé, l'environnement, ou la localisation) et peuvent évoluer au cours du temps si les objectifs de l'organisation changent.

L'objectif de ce papier est d'analyser au fil du temps les interactions se produisant dans un système complexe pour offrir une meilleure compréhension de son fonctionnement et détecter son évolution temporelle. Nous proposons une approche générique pour aborder ce problème stratégique, basée sur l'analyse de la dynamique du graphe d'interactions entre les utilisateurs et les SI. **Notre contribution consiste donc en la proposition d'une méthodologie innovante d'analyse de dynamique des systèmes complexes par les graphes.** Cette méthodologie peut être appliquée à tout système d'interaction pour aider à modéliser son comportement, comprendre son évolution « normale » et détecter des anomalies potentielles, que nous appelons « événements » dans ce papier.

Les systèmes complexes peuvent être modélisés par des graphes où les nœuds représentent les éléments du système et les liens représentent les interactions entre ces éléments. Plus spécifiquement, beaucoup de systèmes d'interactions

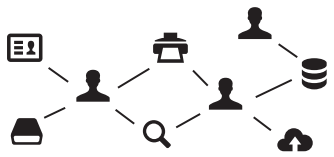

Figure 1 – Exemple d'interactions utilisateur-services.

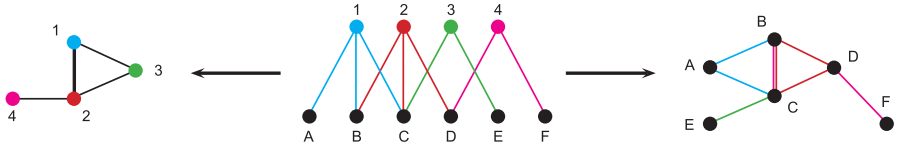

Figure 2 – Exemple de graphe biparti (au centre), avec sa $\top$-projection (à gauche) et sa $\bot$-projection (droite).

peuvent être représentés comme des graphes bipartis ; les graphes bipartis sont constitués de nœuds (éléments) où un nœud appartient à l'un des deux ensembles habituellement appelés *haut* and *bas*, et où les liens n'existent qu'entre nœuds d'ensembles différents. Des applications naturelles dans le contexte des SI incluent les architectures client-serveur où des machines connectées comme clients utilisent des ressources fournies par des machines appelées serveurs, ou encore les graphes processus-messages où chaque processus est relié aux messages échangés. L'invocation de services variés d'un SI par un ensemble d'utilisateurs correspond typiquement à un graphe biparti, tel qu'illustré dans la Figure 1. L'approche classique pour l'étude de tels graphes est de les convertir en graphes unipartis par une projection des relations sur l'un des ensembles (*haut* ou *bas*), en dépit de sévères pertes d'informations (2008, a). Par exemple, on peut construire le graphe des clients où deux clients sont liés s'ils sont connectés au même serveur ; on peut construire le graphe des processus où deux processus sont liés s'ils ont échangé un message, voir Figure 2.

1.2. *Contribution et organisation du papier*

Notre méthodologie consiste en l'étude de l'évolution de propriétés spécifiques (internes) au graphe en considérant une fenêtre temporelle glissante définie avec une unité de *temps intrinsèque*, jugée plus fiable que le temps traditionnel -*extrinsèque*- en termes de détection d'événements. Nous nous concentrons sur les interactions les plus stables dans le système (i.e. les liens dits *internes* du graphe biparti) pour capturer la dynamique essentielle du système (par opposition aux variations marginales -bruit-).

Nous appliquons cette méthodologie à la détection d'événements dans un grand réseau d'interactions impliquant des utilisateurs humains et des systèmes informatiques : le réseau social Github comprenant 2,2 millions de liens.

Les méthodes actuelles échouent à révéler la dynamique intrinsèque des systèmes. Nous utilisons un nouveau concept de temps qui permet de voir une dynamique totalement différente de la dynamique *extrinsèque*. En outre, notre approche est utilisée pour détecter les comportements anormaux du système, i.e. les événements statistiquement marginaux. L'observation des liens internes nous permet d'identifier de nouveaux événements que l'on ne peut détecter autrement, tandis qu'elle confirme les événements détectés par des métriques classiques. Nous étudions l'impact d'échelles de temps variées sur la détection d'événements. Nous montrons que ces événements peuvent être détectés automatiquement avec *Outskewer*, une méthode statistique que nous avons introduite dans un papier précédent (Heymann *et al.*, 2012). Nous validons finalement la pertinence des événements détectés. Notre approche peut donc être utilisée pour surveiller les SI vus comme des graphes bipartis en temps réel, et de déclencher des alertes automatiquement quand un comportement anormal survient. La connaissance obtenue à travers ces expériences peut être utilisée pour la conception de nouvelles méthodes d'analyse des systèmes complexes.

Ce document est organisé comme suit. Dans la Section 2, nous introduisons la modélisation de systèmes complexes par les graphes bipartis et nous présentons notre jeu de données. Dans la Section 3, nous présentons notre approche pour le suivi de l'évolution d'interactions utilisateur-système avec une fenêtre temporelle glissante. Dans la Section 4, nous montrons l'impact des deux concepts de temps sur la caractérisation de l'évolution du système. Dans la Section 5, nous étudions la dynamique des interactions utilisateur-système à l'aide d'une propriété spécifique aux graphes bipartis, appelée *liens internes*, et nous détectons des événements statistiquement significatifs. Nous montrons dans la Section 6 que ces événements peuvent être détectés automatiquement en temps réel. Nous les interprétons et validons leur pertinence sur un système réel. Enfin nous concluons et présentons nos perspectives dans la Section 7.

2. Modèle du système & jeu de données

2.1. *Modèle de graphe biparti*

Un graphe biparti est un triplet $G = (\top, \bot, E)$ où $\top$ est l'ensemble des nœuds du *haut*, $\bot$ est l'ensemble des nœuds du *bas*, et $E \subseteq \top \times \bot$ est l'ensemble des liens. Les graphes bipartis ne constituent pas un type particulier de réseaux comme l'on pourrait le croire. Au contraire, tous les graphes réels ont une structure sous-jacente bipartie (Guillaume et Latapy, 2004). Les graphes qui ne présentent pas cette structure sont en effet des projections de graphes bipartis. Tel que posé dans (2008, a), le $\bot$-projeté de G est le graphe $G_\bot = (\bot, E_\bot)$

dans lequel deux nœuds de $\perp$ sont liés s'ils ont au moins un voisin en commun dans G : $E_\perp = \{(u, v), \exists x \in \top : (u, x) \in E \text{ et } (v, x) \in E\}$. Le $\top$-projeté $G_\top$ est définie de manière duale, comme illustré sur la Figure 2.

2.2. *Jeu de données Github*

Github.com est une plateforme en ligne créée en 2008 pour aider les développeurs à partager du code *open source* et à collaborer. Construite autour du système de versionnement décentralisé Git, elle facilite le partage et les discussions à travers une interface Web. Le 16 janvier 2013, Github a atteint 3 millions d'utilisateurs qui collaborent sur 5 millions de projets de développement de logiciels (dits « *repositories* »)[1].

Le jeu de données Github décrit toute l'activité visible publiquement entre utilisateurs et projets sur la plateforme, du 11 mars 2012 au 18 juillet 2012. Nous avons extrait les données depuis le site Github Archive[2], qui enregistre toute l'activité entre utilisateurs et projets publics. Nous avons ensuite construit le graphe biparti de « qui contribue à quel dépôt de logiciel », où les nœuds représentent les utilisateurs et les projets, et où les liens représentent tout type d'interaction entre utilisateurs et projets.Le jeu de données contient un peu plus de 336 000 nœuds et 2,2 millions de liens.

Dans le graphe biparti correspondant, les nœuds du *haut* représentent les utilisateurs et les nœuds du *bas* les projets. Un lien représente une interaction entre l'utilisateur et le dépôt de logiciels.

3. Notre approche : utilisation d'une fenêtre glissante

Nous avons collecté toutes les données nécessaires à l'analyse de l'évolution du graphe, puisque nous conservons tous les nœuds et liens au cours du temps. Chaque lien est associé à un *timestamp* indiquant le moment où il a été observé. Les données forment donc un flux de liens observés, ordonnés par leur *timestamp*. On considère qu'un nœud apparaît dans le graphe quand il est attaché à un lien observé pour la première fois. Cependant, les données ne contiennent aucune information sur la durée de vie des nœuds et liens. Un nœud peut en effet être observé une seule fois même s'il existe durant une longue période. Cela signifie que nous n'observons pas les nœuds apparus avant le début de la mesure et pour lesquels aucun lien n'est observé au cours de la mesure, i.e. les utilisateurs inactifs durant la période étudiée. Nous ne voyons donc pas non plus les projets qui ne présentent aucune activité sur la période observée.

---

1. `https://github.com/about/press`
2. `http://www.githubarchive.org`

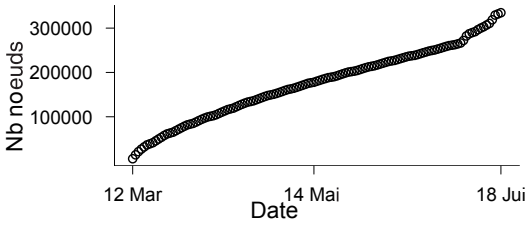

Figure 3 – Github : nombre de nœuds distincts en fonction du nombre total de liens observés.

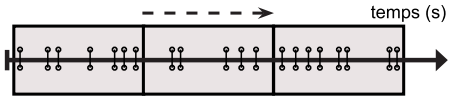

Figure 4 – Flux de liens divisé en fenêtres temporelles contiguës.

Trois approches classiques peuvent être distinguées pour l'étude des réseaux dynamiques. La première consiste à étudier la croissance du graphe au cours du temps, représentée de façon cumulative. Par exemple, la Figure 3 représente le nombre de nœuds comulé en fonction du nombre total de liens observés depuis le début de la mesure. Cette courbe affiche une croissance régulière avec un changement de régime à la fin, mais nous n'obtenons que peu d'informations sur la dynamique sous-jacente. La seconde approche consiste à diviser le flux de liens en fenêtes temporelles contiguës pour construire une série de sous-graphes, comme illustré en Figure 4. Nous calculons alors la propriété statistique choisie sur chaque sous-graphe. Par exemple, le nombre de nœuds de chaque sous-graphe capturé au cours du temps est représenté sur la Figure 5. On observe sur cette courbe une tendance régulière et quelques pics, cependant nous manquons peut-être des événements plus subtils et le moment précis de leur apparition. Nous employons donc une troisième approche, qui généralise

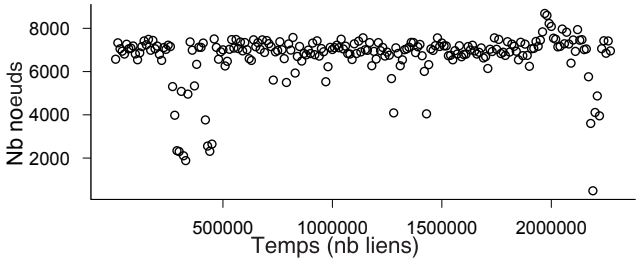

Figure 5 – Github : nombre de nœuds distincts dans l'union de 10 000 liens consécutifs, tous les 10 000 liens, en fonction du nombre de liens observés.

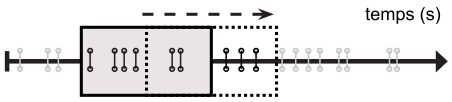

Figure 6 – Fenêtre temporelle glissante sur un flux de liens.

la précédente. Elle consiste en l'extraction de sous-graphes consécutifs depuis une fenêtre temporelle *glissante* (Figure 6).

Notre approche se déroule comme suit : depuis un flux de liens, nous mesurons une propriété statistique donnée[3] du graphe observé dans la fenêtre temporelle glissante. Soit $\{e_0, e_1, ..., e_n\}$ un flux de liens. Soit une fenêtre glissante de taille w. Si w est fonction du temps, par exemple si sa valeur s'exprime en secondes, alors la fenêtre glissante est l'ensemble de tous les liens observés durant w secondes.

A notre connaissance, toutes les études sur les réseaux dynamiques utilisant une fenêtre glissante définissent la taille de cette dernière en secondes. L'apparente trivialité de cette approche attire peu l'attention car elle est facile à mettre en oeuvre et implique une unité de temps commune. Cependant elle soulève des questions non triviales (détaillées en section IV) qui ne sont pas abordées dans la plupart des éudes. Par ailleurs, la taille w de la fenêtre glissante peut correspondre au nombre de liens (indépendamment de l'intervalle temporel entre ces liens) ; dans ce cas, la fenêtre glissante est définie ainsi :

Soit $\{e_0, e_1, ..., e_n\}$ un flux de liens. Soit une fenêtre glissante de taille w liens : $E_i = \{e_{i-w+1}, ..., e_i\}$. Tout lien e_i de la série appartient à $E_i, E_{i+1}, ..., E_{i+w-1}$. Nous calculons la valeur d'une propriété sur la série de graphes, où chaque graphe est composé de l'ensemble des liens de la fenêtre temporelle : soit le graphe $G_i = (V_i, E_i)$ où V_i est l'ensemble des nœuds attachés aux liens de E_i. Soit la série de graphes $G_w, G_{w+1}, ..., G_{|E|}$ où $|E|$ est le nombre total de liens.

Cette fenêtre glissante, quelle que soit l'unité de taille choisie, nous permet de construire une série temporelle correspondant à l'évolution de la propriété du graphe étudiée au cours du temps.

L'utilisation d'une fenêtre glissante pour l'analyse des graphes dyanmiques soulève de nombreuses questions : quelle unité de temps (traditionnellement basée sur le temps en secondes ou basée sur les liens) devrait-on utiliser pour détecter des événements ? Quel est l'impact de la taille de la fenêtre sur l'évolution de la propriété ? Dans les sections suivantes, nous étudions empiriquement l'impact de ces différents concepts de temps ainsi que des échelles de temps variées sur une propriété statistique triviale : le nombre de nœuds observés dans

3. Une propriété pertinente pour les graphes bipartis est proposée et étudiée en Section 6

le graphe au cours du temps. Nous voulons déterminer leurs conséquences sur la caractérisation de la dynamique, et sur la détection d'événements statistiquement significatifs.

4. Unité et Échelle de Temps

4.1. *Unité de temps*

Le temps est un concept controversé que l'on peut voir comme une dimension dans laquelle les changements surviennent séquentiellement. Selon cette perspective, le temps est considéré comme absolu, c'est-à-dire que les changements se produisent indépendamment de l'écoulement du temps (Kant, 1781; Newton, 1687). Mais si nous considérons le temps comme un concept relatif, alors le temps dépend de l'espace. Le temps est vécu en pratique comme relatif parce que nous pouvons seulement le mesurer à travers le mouvement relatif (dans l'espace) des corps par rapport aux autres. Plusieurs techniques existent pour le mesurer. L'unité adoptée par le Système International d'Unités est la seconde, définie par la transition entre deux états de l'atome de césium 133 (de la Convention du Mètre, 2006). Cette unité est donc liée aux mouvements mesurés dans l'espace physique.

Nonobstant le fort impact potentiel d'une unité de temps dérivée de l'espace physique, la plupart des études sur les graphes dynamiques utilisent le temps absolu : les propriétés statistiques sont mesurées en secondes ou avec l'une de ses unités dérivées (ex. jours et années). En conséquence, ces études détectent des activités exogènes (i.e. qui viennent de l'extérieur) à ces réseaux (Aynaud et Guillaume, 2011; Panisson *et al.*, 2011; Takeuchi et Yamanishi, 2006; Whitbeck *et al.*, 2012). Par exemple, les données de flux de clics du trafic Web révèlent naturellement un cycle jour-nuit dans le réseau à cause de l'activité humaine habituelle (Meiss *et al.*, 2008). Bien que cette découverte puisse être intéressante, elle fournit plus d'information sur l'activité des utilisateurs que sur le réseau lui-même. De telles tendances peuvent cacher des tendances liées uniquement au réseau, nous empêchant de caractériser la dynamique endogène du réseau.

Nous avons alors introduit dans (Heymann et Le Grand, 2013) le concept de temps relatif du point de vue du réseau, appelé *temps intrinsèque* au réseau, par opposition au *temps extrinsèque*, qui est un concept de temps absolu du point de vue du réseau. Soit le *temps extrinsèque* du réseau le temps mesuré en secondes. Nous l'appelons *extrinsèque* car son écoulement est indépendant des changements survenant dans le réseau. Soit le *temps intrinsèque* du réseau le temps mesuré par la transition entre deux états du réseau. L'unité de temps est donc le changement (spatial) du réseau : l'ajout ou la suppression d'un nœud ou lien. Nous qualifions ce temps d'*intrinsèque* car son écoulement dépend des

changements sur le réseau, et les changements dépendent de l'écoulement d'un tel temps pour se produire.

Nous avons aussi montré que l'unité de temps a un fort impact sur la mesure de propriétés statistiques, et sur notre capacité à détecter des événements statistiquement significatifs. Dans la suite du papier, nous utilisons l'observation d'un lien comme unité de *temps intrinsèque*, et étudions la dynamique endogène du réseau.

4.2. *Échelle de temps*

Dans le cas d'un flux de liens, l'évolution est habituellement capturée par la mesure de propriétés statistiques du graphe sur une fenêtre temporelle glissante, comme expliqué dans la Section 3. De nombreuses études présupposent une échelle de temps spécifique pour la mesure de ces propriétés (Aggarwal *et al.*, 2011; Akoglu *et al.*, 2010; 2008, b). D'autres études abordent le problème de l'échelle temporelle. Par exemple, (Benamara et Magnien, 2010) propose une méthodologie pour estimer la taille de la fenêtre observable pour établir une caractérisation rigoureuse de n'importe quelle propriété du graphe. (Papadimitriou, 2006) propose une échelle optimale pour la détection de formes dans les séries temporelles. (Reeves *et al.*, 2009) soulève le problème du sous-échantillonnage de séries temporelles pour le stockage, tout en préservant la capacité à détecter des anomalies.

Mais toutes ces études utilisent une unité de *temps extrinsèque*. De plus, nous avons constaté dans (Heymann et Le Grand, 2013) qu'il n'existe aucune échelle temporelle optimale pour la caractérisation d'événements, et que l'échelle doit être choisie selon la taille de l'événement considéré. Dans la section suivante, nous utilisons une unité de *temps intrinsèque* et proposons une propriété spécifique pour l'étude de la stabilité des graphes bipartis et ainsi suivre les interactions utilisateur-système.

5. Suivi de la dynamique des interactions entre l'utilisateur et le système

La propriété que nous suggérons de prendre en considération pour le suivi de la dynamique des interactions entre utilisateur et système est le nombre de liens internes. Le concept intrinsèquement biparti de lien interne a été introduit et étudié récemment pour des réseaux statiques (Allali *et al.*, 2012) afin d'apporter des éléments nouveaux pour la caractérisation de ces réseaux. Un lien interne est un lien dont la suppression ne modifie pas la projection du graphe pour l'un des deux ensembles de nœuds, *haut* ou *bas*. Si l'on prend l'exemple d'un système d'information où les utilisateurs (i.e. nœuds *haut*) interagissent avec des services (i.e. nœuds *bas*), un lien interne partant du point de

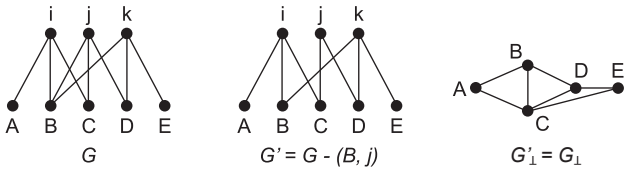

Figure 7 – Exemple de lien $\perp$-interne. De gauche à droite : un graphe biparti G, le graphe biparti G' obtenu en supprimant le lien (B, j) de G, et leur $\perp$-projection. Comme $G'_\perp = G_\perp$, (B, j) est un lien $\perp$-interne de G.

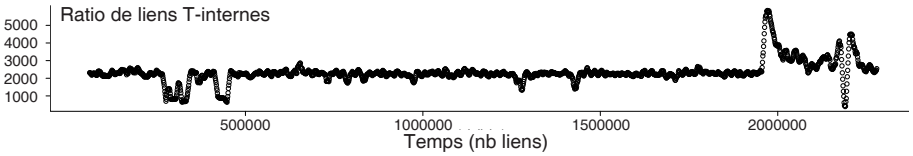

Figure 8 – Github : nombre de $\top$-liens internes dans une fenêtre glissante de $w = 10000$ liens, tous les 1000 liens. Les couleurs représentent les valeurs de classe d'*outlier*, voir Section 7.

vue *haut* (respectivement *bas*) est un lien qui n'est pas indispensable pour que deux utilisateurs (respectivement services) soient connectés dans la projection correspondante. On peut interpréter ces liens internes comme une mesure de redondance de liens.

Les liens $\top$-internes (respectivement $\perp$-internes) sont des liens qui peuvent être supprimés de E sans modifier la $\top$-projection (respectivement la $\perp$-projection), comme le montre la Figure 7. Soit $(u, v) \in \perp \times \top$ avec $(u, v) \in E$ et soit $G' = G - (u, v)$, (u, v) est un lien $\perp$-interne si et seulement si $G_\perp = G'_\perp$ où $G'_\perp$ est la $\perp$-projection de G'. Les liens $\top$-internes sont définis de manière duale.

Alors qu'il a été montré que les liens internes permettent de capturer des propriétés statistiques intéressantes des réseaux bipartis, ce concept n'a jamais été utilisé pour l'étude des réseaux qui évoluent dans le temps. Nous mesurons le nombre de liens $\top$-internes en utilisant une fenêtre glissante de 10000 liens[4]. Nous observons sur la Figure 8 que le nombre de liens $\top$-internes est globalement stable, autour de 2400 liens, ce qui signifie que 24% des liens à l'intérieur de la fenêtre glissante sont des liens internes. Ce résultat est intéressant car il nous fournit une caractérisation du comportement normal du système. Nous observons également des événements significatifs avec des croissances et des décroissances rapides des valeurs, qui constituent des anomalies statistiques du comportement du système. Ces événements ne sont cependant pas nouveaux pour nous : nous les observons aussi dans l'évolution du nombre de nœuds, ou

4. C'est la taille choisie en Section 5

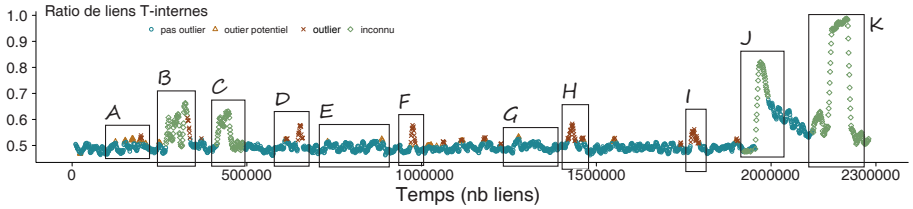

Figure 9 – Github : nombre de $\top$-liens internes dans une fenêtre glissante de $w = 10000$ liens, tous les 1000 liens. Les événements sont entourés et étiquetés par une lettre. Les couleurs représentent les valeurs de classe d'*outlier*, voir Section 7.

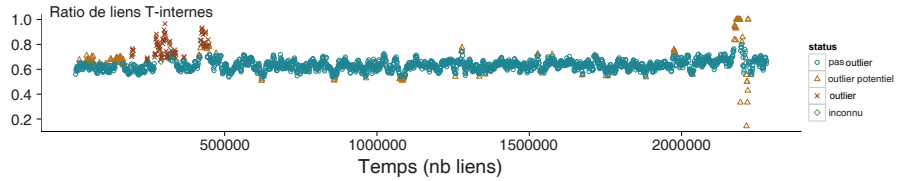

Figure 10 – Github : ratio de $\top$-liens internes parmi les connexions des nœuds de degré > 1 dans une fenêtre glissante de $w = 10000$ liens, tous les 1000 liens. Les couleurs représentent les valeurs de classe d'*outlier*, voir Section 7.

bien dans l'évolution du nombre de liens distincts (dont nous n'affichons pas la courbe puisque son allure est similaire à celle du nombre de nœuds). Par conséquent, le nombre de liens internes est proportionnel au nombre de liens distincts.

Ceci nous a conduits à étudier le nombre normalisé de liens internes, qui correspond au nombre de liens internes divisé par le nombre de liens distincts dans la fenêtre glissante. Cette propriété fournit la proportion de liens internes observés dans la fenêtre glissante. Ce ratio est différent de celui que l'on peut calculer à partir de la propriété précédente puisque seuls les liens distincts sont pris en compte. Ce ratio est pertinent car la mesure de liens internes est indépendante du fait que plus d'un lien existe entre deux nœuds. Nous complétons la courbe de la Figure 9 avec un rectangle autour de chaque ensemble de valeurs anormales de la courbe, ou de points qui sont anormaux dans d'autres courbes, et nous étiquetons ces événements par une lettre.

Nous identifions un nouvel événement, le petit événement A, alors que les événements E et G ont disparu. L'événement I est intéressant : il est le seul qui corresponde à un pic croissant sur la courbe du nombre de liens internes ; il est également observé sur la courbe du degré maximal (non incluse dans ce papier). Nous interprétons cet événement dans la Section 7. De plus, nous découvrons que tous les pics décroissants de la courbe du nombre de liens sont au contraire

croissants dans cette courbe. La proportion de liens internes augmente lorsque le nombre de liens distincts diminue. Cet effet est dû au biais intrinsèque de cette propriété, qui considère comme liens internes les liens attachés à des nœuds ayant un seul voisin. Finalement, nous supprimons ce biais en ignorant ces nœuds. Nous obtenons ainsi la proportion de liens internes parmi les liens qui relient les nœuds ayant au moins deux voisins. Nous constatons dans la Figure 10 que ce filtre a fait disparaître la plupart des événements, conservant ainsi les connexions les plus significatives.

En conclusion, nous avons vu qu'une propriété reposant sur la mesure des liens internes révèle des événements d'un nouveau type, et confirme les événements connus. Dans la section suivante, nous montrons comment détecter ces événements de manière automatique et en temps réel.

6. Détection d'événements en temps réel

6.1. *Méthodologie*

Comment détecter des événements de manière automatique, fiable et en temps réel ? Dans cette section, nous proposons d'utiliser la méthode *Outskewer* que nous avons développée récemment (Heymann *et al.*, 2012). Outskewer permet en effet de détecter les anomalies statistiquement significatives (i.e. les valeurs qui s'éloignent sensiblement des autres) dans des échantillons et dans des séries temporelles. Les résultats sont faciles à interpréter car les valeurs sont classifiées comme *outliers*, *outliers potentiels* ou *non outliers*. La classe est *inconnue* lorsqu'il n'y a pas de comportement « normal » dans l'échantillon étudié. Cette méthode est également facile à utiliser car elle ne requiert aucune connaissance a priori sur les données, et l'unique paramètre est la largeur de la fenêtre temporelle pour les séries temporelles sur lesquelles les *outliers* sont détectées. Cette largeur peut être différente de celle utilisée pour mesurer la propriété. Une implémentation de cette méthode peut prendre en entrée un flux de valeurs pour une analyse en temps réel.

Nous avons appliqué notre méthode pour suivre l'évolution du nombre normalisé de liens internes. Sur la Figure 9, la couleur des points de la courbe correspond à leur classe d'*outlier* : rouge pour les *outliers*, orange pour les *outliers potentiels*, bleu pour les *non outliers* et vert pour les *inconnues*. Nous observons qu'*Ouskewer* est capable de détecter automatiquement tous les événements identifiés à la main. Ils sont identifiés soit par un ensemble d'*outliers*, soit par un ensemble de valeurs inconnues. Ce dernier cas se présente lorsqu'il n'y a pas de comportement normal au cours des fenêtres glissantes correspondantes, mais nous considérons que quelque chose d'imprévu se produit à ce moment-là. Nous pouvons donc utiliser cette méthode pour détecter des événements de manière automatique, et éventuellement de surveiller le système en temps réel.

6.2. *Résultats et interprétation des événements*

La méthodologie que nous avons présentée pour la détection automatique d'événements dans la dynamique du graphe nous permet d'identifier des événements associés à des parties spécifiques de la courbe (qui correspondent aux groupes de valeurs de la propriété étudiée, voir Figure 9). Chaque valeur de la série temporelle représente la propriété du sous-graphe qui est observable pendant la fenêtre correspondante. Plusieurs étapes sont alors nécessaires pour interpréter l'événement détecté et vérifier la validité de notre méthode de détection d'anomalies : 1) Nous extrayons le sous-graphe associé à l'événement détecté et nous l'analysons en calculant le nombre normalisé de liens $\top$-internes ; 2) Nous mettons en correspondance le temps intrinsèque de l'événement et le temps extrinsèque (absolu) correspondant ; cette information est disponible dans les données Github ; 3) Nous recherchons des interactions anormales entre les utilisateurs et le système au moment de l'événement.

Par exemple, nous avons découvert que l'événement I de la Figure 9 est corrélé à une croissance soudaine du degré maximum des nœuds dans le graphe. Nous avons également trouvé dans le sous-graphe comprenant 1000 nœuds que le projet *node-migrator-bot*[5] interagit avec 95 utilisateurs, ce qui est un nombre inhabituellement élevé de voisins pour un nœud du graphe dans une fenêtre temporelle. En regardant la page Web de ce projet sur Github, nous avons appris que ce projet vise à surveiller les mises à jour d'un autre projet, appelé *NodeJitsu*[6]. Nous avons constaté que *node-migrator-bot* a envoyé le 26 juin 2012 à minuit un message *PullRequestEvent* aux autres projets de la plateforme Github qui utilisent une version plus ancienne de *NodeJitsu*. Ce *bot* gère donc les dépendances entre *NodeJitsu* et les autres projets, en leur fournissant un patch à appliquer à leur propre code. Cette observation explique la croissance du degré maximal dans le graphe : rappelons qu'un lien du graphe représente une interaction entre un projet et un utilisateur. Dans ce cas, il y a beaucoup d'interactions entre le *bot* et les utilisateurs propriétaires d'un projet qui a besoin d'être mis à jour.

Interprétons à présent l'événement J de la Figure 9. Cet événement est également corrélé à une augmentation soudaine du degré maximal dans le sous-graphe. Nous découvrons dans ce sous-graphe contenant 1000 nœuds que le projet *Try-Git* interagit avec 506 utilisateurs, ce qui explique ce degré élevé. Nous apprenons sur la page Web du projet qu'il s'agit d'un tutoriel sur Git, l'un des outils sous-jacents à Github. La première action requise de l'utilisateur dans ce tutoriel est de créer un clone avec un nouveau projet (en envoyant à cet utilisateur un message *CreateEvent*). L'instant de détection de l'événement par

5. `https://github.com/node-migrator-bot`
6. `https://nodejitsu.com/`

notre outil correspond au moment où *Try-Git* a été rendu public, le 4 juillet 212 à 17h (cette information a été confirmée par un post sur le blog Github.com[7]).

Nous avons montré avec les deux exemples d'interprétation ci-dessus que les événements détectés de manière automatique avec notre méthode correspondent réellement à une activité extra-ordinaire de la plateforme Github : il s'agit donc bien d'événements. Tous les événements détectés sur la courbe peuvent être interprétés de la même manière, démontrant la validité de notre surveillance de la dynamique des interactions utilisateur-système.

7. Conclusion et perspectives

7.1. *Conclusion*

Nous avons proposé dans cet article une approche originale pour la caractérisation et la surveillance des interactions entre utilisateur et système, et en particulier leur évolution au fil du temps. Nous modélisons ces interactions sous la forme d'un graphe biparti, où les nœuds *haut* et *bas* représentent respectivement les utilisateurs et les éléments du système, et où les liens correspondent aux interactions entre ces deux types de nœuds. Nous étudions l'évolution de ces graphes pour caractériser le comportement *normal* et détecter une dynamique *anormale*, à travers la mesure de propriétés statistiques spécifiques pendant une fenêtre de temps. Cette approche est facile à mettre en œuvre mais elle soulève des questions ardues qui ne sont quasiment pas abordées dans la littérature. Quelle unité de temps devrions-nous utiliser ? En utilisant une référence temporelle absolue comme la seconde, les recherches actuelles ne parviennent pas à capturer la dynamique intrinsèque du réseau et se concentrent sur l'évolution liée à des facteurs exogènes (par exemple les cycles jour/nuit). Nous utilisons ici un concept de temps que nous avons récemment introduit, appelé *temps intrinsèque*, qui repose sur l'apparition des liens, et révèle une dynamique totalement différente. Quelle échelle de temps doit alors être choisie pour calibrer la mesure, i.e. quelle largeur choisir pour la fenêtre glissante ? Comme il n'existe pas d'échelle de temps optimale car les événements se produisent à différentes échelles, nous utilisons une échelle offrant un bon compromis en termes d'événements détectés. En utilisant cette approche, nous capturons avec succès le comportement normal du système et nous détectons les évolutions anormales.

Nous avons appliqué notre approche à un système réel de grande taille : la plateforme sociale Github, où les utilisateurs interagissent avec des projets de développement logiciel. Nous avons suivi l'évolution de ce système en utilisant une propriété appelée *liens internes* qui permet de capturer les interactions essentielles et de révéler les événements significatifs.

---

7. `https://github.com/blog/1183-try-git-in-your-browser`

Nous avons finalement surveillé le système en temps réel en utilisant la méthode *Outskewer*, afin de déclencher des alarmes automatiquement lorsque des comportements anormaux apparaissent.

7.2. *Perspectives*

Dans l'avenir, nous considérerons la combinaison de ce suivi de la dynamique des interactions utilisateur/système selon une approche innovante en ingénierie des méthodes, appelée *fouille d'intentions* (Hug *et al.*, 2012). La fouille d'intentions vise à découvrir les intentions actuelles et futures des utilisateurs à partir de l'analyse des traces d'activités. L'utilisation conjointe des deux méthodes pourrait être appliquée à différents types de systèmes d'information, en particulier les systèmes d'information pervasifs (Kourouthanassis et Giaglis, 2006; Najar *et al.*, 2012). Nous approfondirons également l'utilisation de la notion de *temps intrinsèque* pour explorer son potentiel et ses implications pour l'analyse des réseaux. En particulier, nous avons vu que les propriétés calculées pour le réseau Github sont très stables. Ce phénomène pourrait être spécifique à nos données, cependant il pourrait aussi s'agir d'un effet de bord lié à la manière dont nous observons cette propriété. La fenêtre glissante a en effet une largeur fixée w ; elle contient donc au maximum w liens distincts et $2w$ nœuds. Des études plus poussées sont nécessaires pour déterminer si ce phénomène est nécessairement lié à la méthode d'observation.

Enfin, les événements sont parfois difficiles à interpréter. La visualisation de données est une approche prometteuse pour la caractérisation des événements. Un petit nombre de logiciels de visualisation tels que Gephi (Bastian *et al.*, 2009) et VIENA (Windhager *et al.*, 2011) fournissent une interface utilisateur graphique pour l'exploration de réseaux temporels ; ils ne sont cependant pas conçus pour la détection d'événements. Nous travaillerons donc à la conception de techniques d'investigation visuelle.

Références

Aggarwal C. C., Zhao Y., Yu P. S., « Outlier detection in graph streams », *International Conference on Data Engineering*, 2011, p. 399–409.

Akoglu L., McGlohon M., Faloutsos C., « Event Detection in Time Series of Mobile Communication Graphs. », *Army Science Conference*, 2010.

Allali O., Tabourier L., Magnien C., Latapy M., « Internal links and pairs as a new tool for the analysis of bipartite complex networks », *Social Network Analysis and Mining*, , 2012, Springer Vienna.

Aynaud T., Guillaume J.-L., « Multi-Step Community Detection and Hierarchical Time Segmentation in Evolving Networks », , 2011.

Bastian M., Heymann S., Jacomy M., « Gephi : An Open Source Software for Exploring and Manipulating Networks », *Proc. AAAI International Conference on Weblogs and Social Media (ICWSM'09)*, 2009.

Benamara L., Magnien C., « Estimating Properties in Dynamic Systems : The Case of Churn in P2P Networks », *IEEE Conference on Computer Communications Workshops, INFOCOM Wksps*, 2010.

de la Convention du Mètre O. I., « The International System of Units (SI) », rapport n° 8, 2006, Bureau International des Poids et Mesures.

Guillaume J.-L., Latapy M., « Bipartite Structure of all Complex Networks », *Information Processing Letters (IPL)*, vol. 90, n° 5, 2004, Elsevier.

Heymann S., Latapy M., Magnien C., « Outskewer : Using Skewness to Spot Outliers in Samples and Time Series », *Proc. IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012)*, IEEE, 2012.

Heymann S., Le Grand B., « Monitoring User-System Interactions through Graph-Based Intrinsic Dynamics Analysis », *to appear in Proc. IEEE International Conference on Research Challenges in Information Science (RCIS 2013)*, IEEE, 2013.

Hug C., Deneckere R., Salinesi C., « Map-TBS : Map process enactment traces and analysis », *Research Challenges in Information Science (RCIS), 2012 Sixth International Conference on*, may 2012, p. 1 -6.

Kant I., *Kritik der reinen Vernunft*, 1781.

Kourouthanassis P. E., Giaglis G. M., « A design theory for pervasive information systems », *Proc. 3rd Int. Workshop on Ubiquitous Computing (IWUC'06)*, 2006, p. 62–70.

Latapy M., Magnien C., Del Vecchio N., « Basic Notions for the Analysis of Large Two-mode Networks », *Social Networks*, vol. 30, n° 1, 2008, Elsevier.

Latapy M., Magnien C., Ouedraogo F., « A Radar for the Internet », *IEEE International Conference on Data Mining*, vol. abs/0807.1, 2008, p. 901–908.

Meiss M. R., Menczer F., Fortunato S., Flammin A., Vespignan A., « Ranking Web Sites with Real User Traffic », *Proc. ACM International Conference on Advances in Social Networks Analysis and Mining (WSDM'08)*, ACM, 2008.

Najar S., Pinheiro M., Souveyet C., Steffenel L., « Service Discovery Mechanism for an Intentional Pervasive Information System », *Web Services (ICWS), 2012 IEEE 19th International Conference on*, june 2012, p. 520 -527.

Newton I., *PhilosophiæNaturalis Principia Mathematica*, 1687.

Panisson A., Barrat A., Cattuto C., Van den Broeck W., Ruffo G., Schifanella R., « On the dynamics of human proximity for data diffusion in ad-hoc networks », *Ad Hoc Networks*, , 2011.

Papadimitriou S., « Optimal multi-scale patterns in time series streams », *Proc. ACM International Conference on Management of Data (SIGMOD'06)*, ACM, 2006, p. 647–658.

Reeves G., Liu J., Nath S., Zhao F., « Managing Massive Time Series Streams with MultiScale Compressed Trickles », *Proceedings of The Vldb Endowment*, vol. 2, 2009, p. 97–108.

Takeuchi J.-i., Yamanishi K., « A unifying framework for detecting outliers and change points from time series », *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, 2006, p. 482–492.

Whitbeck J., Dias de Amorim M., Conan V., Guillaume J.-L., « Temporal reachability graphs », , 2012, p. 377–388.

Windhager F., Zenk L., Federico P., « Visual Enterprise Network Analytics - Visualizing Organizational Change », *Procedia - Social and Behavioral Sciences*, vol. 22, 2011, p. 59–68.