Investigation visuelle d'événements dans un grand flot de liens

Sébastien Heymann*, Bénédicte Le Grand**

*Université Pierre et Marie Curie, LIP6-CNRS
sebastien.heymann@lip6.fr
**Université Paris 1 Panthéon-Sorbonne, CRI
benedicte.le-grand@univ-paris1.fr

Résumé. Nous présentons une nouvelle méthode d'analyse exploratoire de grands flots de liens que nous appliquons à la détection d'événements significatifs dans plus de 2 millions d'interactions (pendant 4 mois) entre utilisateurs du réseau social en ligne Github. Nous combinons une méthode statistique de détection automatique d'événements dans une série temporelle, *Outskewer*, avec un système de visualisation de graphes. *Outskewer* identifie des instants de l'évolution du graphe d'interactions méritant d'être étudiés, et un analyste peut valider et interpréter ces événements par la visualisation de motifs anormaux dans les sous-graphes correspondants. Nous montrons par de multiples exemples que cette approche 1) permet de détecter des événements pertinents et de rejeter ceux qui ne le sont pas, 2) est adaptée à une démarche exploratoire car elle ne nécessite pas de connaissance a priori sur les données.

1 Introduction

L'activité d'un réseau social en ligne, d'un réseau de télécommunication, de requêtes sur un moteur de recherche, ou encore d'un système de paiement par carte bancaire, peut être modélisé par un grand réseau d'interactions ; chaque interaction peut en effet être vue comme un lien entre deux entités, apparaissant au déclenchement de l'interaction. Le réseau est alors représenté par un **flot de liens** ordonnés chronologiquement.

Malgré la diversité des systèmes modélisables par des flots de liens, la dynamique de ces flots de liens reste peu explorée. La dynamique des graphes étant elle-même un sujet d'étude scientifique récent, il a fallu attendre ces dernières années et la publication de grands jeux de données à ce niveau de précision pour que l'étude des flots de liens puisse émerger. La plupart des graphes dynamiques étudiés jusqu'ici sont en effet constitués d'instantanés capturés à une certaine fréquence (ex. un graphe par jour). L'enjeu principal est de caractériser l'évolution de ces systèmes pour mieux les comprendre et en particulier différencier une dynamique normale de comportements anormaux. Pour un analyste surveillant un système et levant des alertes en cas d'anomalies, comme une fraude à la carte bancaire ou une intrusion dans un intranet, il est crucial de pouvoir identifier et valider de tels événements rapidement. En d'autres termes, il s'agit de déterminer *où* et *quand* la structure du réseau d'interactions est anormalement altérée.

Dans cet article nous proposons une méthode d'investigation à la fois statistique et visuelle. Celle-ci repose sur une méthode de fouille de données statistique pour détecter des événements significatifs, suivie de visualisations interactives pour les valider et les interpréter.

1.1 État de l'art

La plupart des méthodes actuelles de fouille de graphes dynamiques (i.e. qui évoluent dans le temps) reposent sur des séries de graphes statiques représentant soit l'état du graphe à l'instant de chaque capture, soit l'agrégation des nœuds et liens apparus entre cet instant et l'instant de la précédente capture (par exemple le graphe des interactions du mois d'avril dans un réseau social). Ces graphes sont communément représentés par des diagrammes nœuds-liens, que l'on trouve dans la majorité des systèmes de visualisation scientifique tels que Visone (Brandes et Wagner (2004)), SoNIA (Moody et al. (2005)), Gephi (Bastian et al. (2009)), ViENA (Windhager et al. (2011)), et GraphDiaries (Bach et al. (2012)).

Quatre stratégies émergent pour la détection visuelle d'événements dans l'évolution d'un graphe : 1) le *morphing* est une correspondance de entre deux instants montrant l'animation du diagramme nœuds-liens (Ghani et al. (2012)) comme une vidéo (dont la position des nœuds peut varier ou non) via une chronologie ou un *slider* (Moody et al. (2005)), voir Gephi, NodeXL, SoNIA, TempoVis. Cette approche est intuitive mais l'analyse est difficile si la densité de liens est élevée et le nombre de changements important (Brandes et al. (2012); Archambault et al. (2011)). 2) la *comparaison de couches* montre l'évolution du graphe sur une vue unique via une série de *small multiples* représentant l'état du graphe à différents instants (Archambault et al. (2011)). Ces *small multiples* sont intégrables à une chronologie (Bach et al. (2012)), voir GraphDiaries. Cette approche permet de comparer rapidement les différences, mais se limite à de petits graphes. 3) la *fusion de couches* intègre deux graphes statiques en un seul. Cette représentation "deux-en-un" distingue les deux instants du graphe par des couleurs et effets de style. Elle amplifie les différences structurelles, mais se limite à des graphes de faible densité (Krempel (2005)), voir ViENA. 4) la *vue en 2.5D* utilise la troisième dimension pour afficher les changements du graphe, mais son efficacité n'est pas démontrée.

Visualiser l'ensemble du graphe dynamique entraîne de nombreux problèmes, notamment l'occlusion, le croisement des liens dans des graphes de forte densité, et la difficulté à préserver la carte mentale (Archambault et al. (2011)). Des métaphores et approches orientées pixels (matrices, codes barre (Albano et al. (2011))) contournent ces problèmes mais restent marginales. Pourtant, voir l'ensemble du graphe n'est pas toujours nécessaire ni souhaitable, en particulier dans un flot de plusieurs millions de liens. La visualisation d'un sous-graphe bien choisi (grâce à une méthode statistique par exemple) permet de détecter des événements pertinents (Sun et al. (2007); Chau (2012)). Les approches existantes sont cependant basées sur des séries de graphes statiques, et ne sont pas adaptées aux flots de liens à cause des biais induits par les changements d'échelle (Clauset et Eagle (2007)).

1.2 Contribution et organisation du papier

Nous proposons une méthode pour valider, par la visualisation, les événements statistiquement significatifs détectés par une méthode de fouille de données. On identifie ainsi où et quand un événement statistiquement significatif apparait dans le réseau.

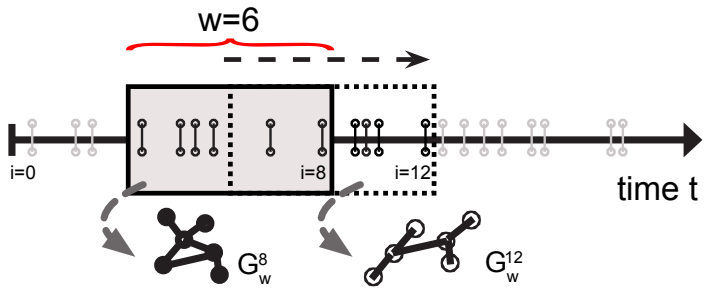

FIG. 1 – *Illustration d'une fenêtre temporelle glissante (de taille w) sur le flot de liens. A chaque position i de la fenêtre correspond un sous-graphe G^i_w.*

Nous décrivons notre méthode en Section 2, l'appliquons sur l'étude d'un grand réseau social en ligne en Section 3, et concluons en Section 4.

2 Méthode

Notre méthode se compose des deux étapes suivantes.

1. Identifier automatiquement des changements statistiquement significatifs dans la structure du graphe sur une fenêtre de temps glissante, en fonction d'une métrique de graphe. Cette étape permet de traiter l'ensemble du graphe rapidement.
2. Valider la pertinence de chaque événement détecté en identifiant visuellement un motif anormal dans le sous-graphe correspondant à la fenêtre temporelle anormale, puis en vérifiant que ce motif est unique au cours du temps.

Nous posons les définitions nécessaires dans la section suivante.

2.1 Définitions

Aucune définition concernant les flots de liens n'étant à notre connaissance disponible dans la littérature, nous posons les définitions suivantes.

Flot de liens : soit $F = \{f_0, f_1, ..., f_m\}$ le multi-ensemble ordonné de triplets $f_i =< n_x, n_y, t_i >$, avec la relation d'ordre $f_i \leq f_{i+1}$ si et seulement si $t_i \leq t_{t+1}$, où $i \in \mathbb{N}, t_i \in \mathbb{R}$. F est appelé *flot de liens*. Chaque triplet représente un lien entre deux nœuds n_x, n_y observé à une date t_i à la position i dans F.

Graphe de flot de liens : soit $G = (V, F)$ le graphe correspondant à ce flot de liens, avec $V = \{n_0, ..n_n\}$. Notons que G possède n nœuds et m liens. C'est un graphe dynamique car chaque lien f_i a une date d'apparition t_i.

Sous-graphe de flot de liens : on considère le multi-ensemble de w triplets $F^i_w = \{f_{i-w+1}, ..., f_i\}$. $G^i_w = (V^i_w, F^i_w)$ est le sous-graphe correspondant. Notons qu'il possède w liens (voir FIG. 1).

Statistique de sous-graphe de flot de liens : soit S^i_w une statistique calculée sur $G^i, \forall i \in [w-1, m]$. La série temporelle $S_w = \{S^{w-1}, ..., S^m\}$ représente l'évolution de cette statistique au cours du temps, calculée sur les sous-graphes de F avec une fenêtre glissante de taille w.

2.2 Détection d'événements statistiquement significatifs

Nous appliquons la méthode *Outskewer* (Heymann et al. (2012)), une méthode de fouille de données non supervisée applicable à un échantillon de valeurs univarié. Cette méthode considère comme anormale une valeur extrême qui rend le coefficient d'asymétrie [1] de la distribution de valeurs éloigné de zéro (i.e. symétrie parfaite de la distribution de valeurs). *Outskewer* peut détecter de multiples valeurs anormales dans un échantillon, et peut être étendu à l'analyse d'une série temporelle (potentiellement en temps réel). Soit $X = \{x_0, x_1, ..., x_n\}$ une série temporelle. On considère le multi-ensemble de w_o valeurs $X^i = \{x_{i-w_o+1}, ..., x_i\}$, où w_o est appelée la taille de la fenêtre temporelle de détection d'événements. Chaque valeur x_i de X appartient à $X^i, X^{i+1}, ..., X^{i+w_o-1}$. Nous utilisons *Outskewer* sur tous ces w_o multi-ensembles, et calculons le nombre de fois où x_i est classée comme *normale*, *probablement anormale*, ou *anormale*. La classe finale est celle ayant obtenu le score le plus grand. Nous utilisons $X = S$ pour appliquer *Outskewer* à une statistique de graphe. Nous posons la définition suivante.

Événement : ensemble consécutif de valeurs $\{x_i, x_{i+1}, ..., x_j\}, i \leq j$ classées comme anormales dans X. Par commodité, nous nous référons aux événements par la valeur i par la suite.

2.3 Analyse visuelle des événements

Cette étape a pour objectif de valider les événements détectés par *Outskewer*, et de les interpréter en les corrélant à des motifs anormaux dans la structure des sous-graphes G_w^i correspondants. Par exemple, la soudaine et fréquente répétition de l'apparition d'un lien contribue à diminuer brusquement le nombre de nœuds uniques observés via une fenêtre temporelle à cet instant. Cet événement peut être détecté automatiquement mais peut avoir plusieurs origines comme nous le verrons.

Nous proposons une visualisation interactive permettant d'investiguer ces événements. Nous faisons l'hypothèse que l'utilisateur sait lire et interpréter un diagramme nœuds-liens. Notre prototype doit respecter les contraintes suivantes : a) représenter la structure du graphe par un diagramme nœuds-liens ; b) ne pas afficher tout le graphe, qui peut compter des millions de nœuds et liens, ce qui rendrait la visualisation illisible ; c) tenir compte de la possibilité que les événements détectés par *Outskewer* soient très espacés des uns des autres dans le temps.

Pour investiguer un événement (cf. section 2.5), l'utilisateur extrait le sous-graphe G_w^i et y identifier un ou plusieurs motifs suspects selon ses critères. Il désambiguïse enfin l'interprétation de l'événement : il détermine si le motif anormal apparaît seulement lors de l'événement. Si c'est le cas, alors il le considère comme corrélé à l'événement.

2.4 Description du prototype

Notre prototype étend les fonctionnalités du logiciel commercial Linkurious [2], réalisé en HTML5 et Javascript pour s'exécuter dans un navigateur Web. Il se connecte à une base de données de graphe Neo4j [3]. L'interface se compose des quatre espaces suivants :

1. $\frac{n}{(n-1)(n-2)} \sum_{x \in X} \left(\frac{x-\bar{x}}{\sigma}\right)^3$, avec la moyenne empirique $\bar{x} = \sum_{x \in X}(x/n)$ et l'écart type $\sigma = \sqrt{1/(n-1) \cdot \sum_{x \in X}(x-\bar{x})^2}$ de l'échantillon de valeurs X.
2. `http://linkurio.us`
3. `http://www.neo4j.org`

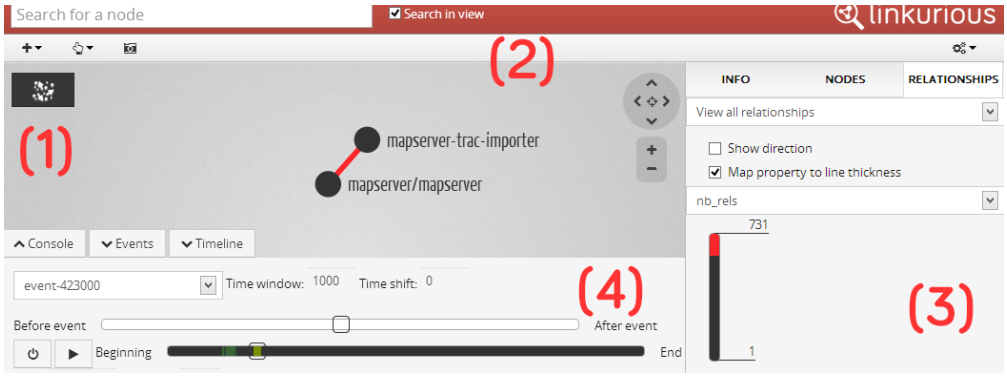

FIG. 2 – *Interface graphique du prototype, dérivée du logiciel commercial Linkurious.*

1) Le **diagramme nœuds-liens** positionne les nœuds dans l'espace par un algorithme à base de force, *ForceAtlas 2* (Jacomy et al. (2011)). Les liens sont pondérés par leur nombre d'apparitions dans le graphe visible. Ainsi, le graphe visible n'est pas un multi-graphe (graphe ayant plusieurs liens entre deux nœuds), qui aurait été difficile à représenter et à lire. Pour détecter les liens ayant un poids de valeur élevé, l'utilisateur utiliser les variables d'épaisseur et de couleur du trait des liens : l'épaisseur est proportionnelle au poids, la couleur varie du gris (petite valeur) au rouge (grande valeur) selon un dégradé linéaire. La taille des disques représente les nœuds et est proportionnelle au degré pondéré. L'utilisateur peut aussi interagir avec la représentation en zoomant et en déplaçant la caméra.

2) La **barre d'outils** est proposée pour trouver un nœud selon la valeur d'une propriété qui lui est associée. La vue est centrée sur le nœud correspondant au résultat sélectionné. La barre d'outil permet de sélectionner et de cacher des nœuds et liens.

3) Les **panneaux latéraux** permettent d'afficher les propriétés des nœuds et des liens sélectionnés, de colorer les nœuds selon une propriété, et d'activer les variations d'épaisseur et de couleur des liens. Un diagramme indique la distribution du poids des liens par couleur, ce qui permet d'associer la couleur d'un lien à sa position dans la distribution.

4) Les **panneaux du bas** sont la nouveauté apportée à Linkurious. Ils permettent de sélectionner un événement i dans une liste déroulante, de préciser une taille de fenêtre temporelle w' (qui peut être différente de w), et d'afficher le sous-graphe correspondant $G^i_{w'}$. Un curseur permet de décaler la position de la fenêtre un peu avant ou un peu après la date de l'événement (jusqu'à 10 000 liens en amont ou en aval) pour observer l'évolution du graphe autour de l'événement. Un second diagramme représente la totalité du flot de liens F permettant à l'utilisateur de déplacer la fenêtre temporelle à n'importe quelle date via un curseur. Un bouton permet de colorer le diagramme en fonction du nombre de liens apparaissant entre les nœuds visibles à l'écran à chaque pourcentage de la largeur du diagramme : de gris (pas d'apparition) à vert clair (le maximum d'apparitions). Si un seul nœud est visible, alors la couleur correspond au nombre de liens ayant ce nœud pour extrémité. Ce diagramme fournit des indications sur la fréquence des apparitions des liens entre nœuds d'un motif durant toute la durée de l'évolution du graphe. Ce diagramme est compacte (10 pixels par 500 pixels) pour répondre à une contrainte de hauteur, mais la précision d'affichage et de positionnement du curseur dépend de

sa largeur l et du nombre de liens m. Une largeur de 1 pixel représente en effet m/l liens. Une fonction de zoom serait une amélioration possible pour gagner en précision.

2.5 Validation expérimentale des événements

L'étape de validation et d'interprétation des événements est effectuée par l'analyste. Elle repose sur la visualisation de chaque sous-graphe G_w^i dont la statistique S_w^i a une valeur classée comme *anormale* par *Outskewer* (i.e. l'événement). Bien que des motifs anormaux puissent apparaître dans G_w^i, cette vue n'est pas suffisante pour corréler un motif à un événement. Nous considérons que celui-ci doit apparaître exceptionnellement à cet instant de F.

1) Sélection des événements à investiguer : nous rappelons qu'un événement est une valeur anormale de S_w^i ou un ensemble consécutif de valeurs anormales $\{S_w^i, S_w^{i+1}, ..., S_w^j\}, i \leq j$. Nous étudions le sous-graphe G_w^i correspondant à l'événement ou à la première valeur pour observer le début de l'événement. D'autres choix sont possibles, comme étudier le sous-graphe ayant le score d'anomalie le plus élevé dans $[i, j]$.

2) Identification d'un motif anormal : L'analyste sélectionne un événement i dans la liste déroulante des événements. Le sous-graphe G_w^i s'affiche ; il peut diminuer la taille de la fenêtre temporelle pour l'affichage (ainsi réduire la taille du sous-graphe) ou conserver la taille utilisée lors de la détection d'événements (w). Il observe le diagramme nœuds-liens à la recherche de motifs suspects (au regard de la statistique utilisée par *Outskewer*), comme une étoile, une composante connexe, ou un lien très redondant. S'il trouve un motif suspect, il explore l'évolution du graphe autour de la date de l'événement. Si ce motif reste visible avant ou après cette date, alors il est évident qu'il n'est pas corrélé à l'événement. L'analyste doit alors chercher un autre motif, ou considérer que l'événement détecté par *Outskewer* n'est pas valide.

3) Interprétation de l'événement : Une fois un motif suspect identifié, l'analyste cherche à déterminer s'il est exceptionnel ou s'il est récurrent. Cette étape permet de tester la corrélation de motifs pour un événement. Comme illustré dans la section suivante, l'analyste sélectionne les nœuds du motif et cache les autres. Il applique alors la coloration de la barre du flot de liens, et voit en un coup d'œil si les liens entre ces nœuds apparaissent à d'autres moments (périodes en vert). Si ce n'est pas le cas, alors le motif entre ces nœuds est unique et corrélé à l'événement. Sinon il déplace le curseur sur chaque période d'apparition pour voir la forme des relations entre ces nœuds. Lors de cette opération la position des nœuds est stable, ce qui permet de comparer plusieurs versions du motif au cours du temps en réalisant un export de l'image du graphe (via la barre d'outils). Si le motif est redondant, alors l'analyste ne peut pas affirmer qu'il soit corrélé à l'événement, bien qu'il puisse être anormal si nous ne considérons que le graphe visible. Dans ce cas l'événement est ambigu, et une analyse approfondie est nécessaire. Au contraire, si le motif apparaît uniquement lors de l'événement, alors il est corrélé à cet événement ; l'analyste peut valider l'événement en interprétant le contenu de ce motif.

3 Application

3.1 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

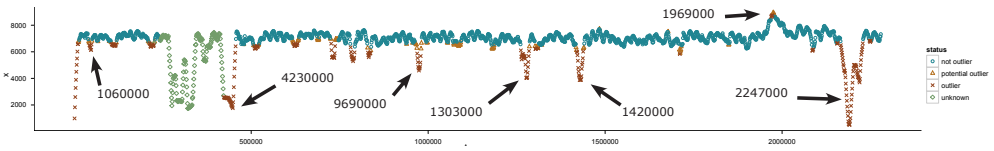

FIG. 3 – *Série temporelle du nombre de nœuds uniques sur le flot de liens de Github, avec* $w = 10000$ *et* $w_o = 200$ *liens. Les valeurs anormales sont en rouge, les événements analysés dans le papier sont pointés par une flèche.*

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* ») [4].

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 [5], 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 : *commit* et envoi de code source, ouverture et fermeture de rapports de bogue, commentaire sur ces rapports, demandes d'intégration de patch, création et suppression de branches et de *tags*, et édition du wiki. Nous ne prenons pas en compte les autres interactions : *fork* (duplication du dépôt de logiciels), mise en favori du dépôt de logiciels, et abonnement au flux d'activités des autres utilisateurs ou d'un dépôt de logiciels. Le jeu de données est ainsi un flot de liens contenant un peu plus de 336 000 nœuds et 2,2 millions de liens ordonnés par date d'apparition.

3.2 Détection automatique d'événements

Nous illustrons l'intérêt de notre approche sur une statistique basique : le nombre de nœuds uniques $S_w^i = card(V^i)$. Suite aux travaux de Heymann et Le Grand (2013) sur l'impact des tailles de fenêtres temporelles sur la détection d'événements dans ce jeu de données, nous calculons S_w^i sur F avec $w = 10000$ liens et appliquons *Outskewer* sur S avec $w_o = 200$ liens. Ces valeurs offrent en effet un bon compromis pour détecter beaucoup d'événements.

Nous observons sur la FIG. 3 que les valeurs sont généralement stables au cours du temps (en bleu), mais des pics apparaissent à certains moments. Les valeurs de ces pics sont classées comme anormales (en rouge), et certaines valeurs sont limites (en orange). Il arrive qu'une période subisse trop de fluctuations pour que *Outskewer* détermine un comportement normal, et classe ces valeurs comme *inconnues* (en vert). Nous en extrayons une liste de 7 événements.

3.3 Validation expérimentale des événements

Événement 106000 : $G_{w=10000}^{i=106000}$ possède 6509 nœuds uniques, 4313 liens uniques, et 2233 composantes connexes. Sa visualisation met en évidence la relation entre l'utilisateur "goneri"

4. `https://github.com/about/press`
5. `http://www.githubarchive.org`

FIG. 4 – *Diagramme des interactions entre l'utilisateur "goneri" et le projet "fusinv/glpi". Nous voyons clairement que l'utilisateur arrête de travailler sur le projet à une certaine date.*

FIG. 5 – *Diagramme des interactions du projet "mxcl/homebrew". Ce projet reçoit des contributions tout au long de la période de capture des données.*

et le projet "fusinv/glpi", avec 207 apparitions du lien au cours de cette période, ainsi que les relations en étoile du projet "mxcl/homebrew". Ces motifs suspects ne sont cependant pas corrélés à l'événement. En effet les diagrammes d'interactions des FIG. 4 et 5 montrent que ces interactions sont fréquentes durant toute la durée du flot pour "mxcl/homebrew" et jusqu'à une certaine date entre "goneri" et "fusinv/glpi". Nous ne validons donc pas l'événement.
Événement 423000 : $G_{w=10000}^{i=423000}$ possède 2531 nœuds uniques, 1525 liens uniques, et 1007 composantes connexes. Sa visualisation met en évidence la relation entre l'utilisateur "mapserver-trac-importer" et le projet "mapserver/mapserver" [6], avec 6993 apparitions du lien au cours de cette période. La FIG. 2 montre que les interactions sont très intenses, avec 731 apparitions de liens sur une fenêtre de taille $w' = 1000$. Le diagramme des interactions FIG. 6 montre 2 courtes périodes d'interactions dont la plus grande en intensité se passe lors de l'événement. Nous pouvons donc valider l'événement et l'interpréter. Le nom du nœud "mapserver-trac-importer" indique que le compte utilisateur qui y est associé sur Github n'est pas actionné par un humain. Nous le vérifions sur sa page d'activité sur Github.com [7], et nous remarquons que le compte a par la suite été renommé "mapserver-bot". Le but de ce robot a été de migrer le code source et la liste de bogues du projet "mapserver" de Trac vers Github. Nous trouvons des traces de la discussion lancée le 19 mars 2012 sur la mailing-list publique [8]. L'événement correspond donc à la date de migration du code source ou de la liste de bogues sur Github.com : le bug n°1 correspond en effet au début de l'événement le 3 avril 2012 à 17 :37 :55 ($t_{i=413000}$).

6. `https://github.com/mapserver/mapserver`
7. `https://github.com/mapserver-bot?tab=activity`
8. `https://lists.osgeo.org/pipermail/mapserver-dev/2012-March/012100.html`

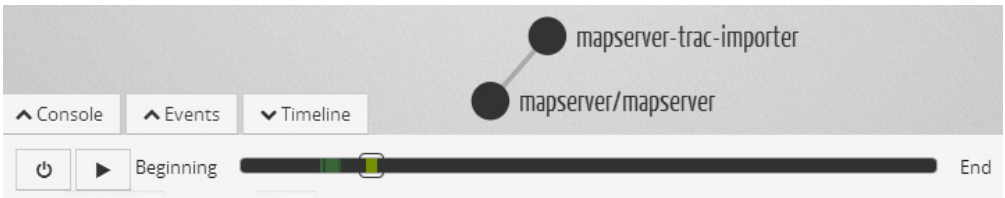

FIG. 6 – *Interactions très redondantes entre cette paire de nœuds lors de l'événement 423000, confirmée par la période vert clair du diagramme des interactions. Nous remarquons une autre période d'interactions plus tôt en vert foncé.*

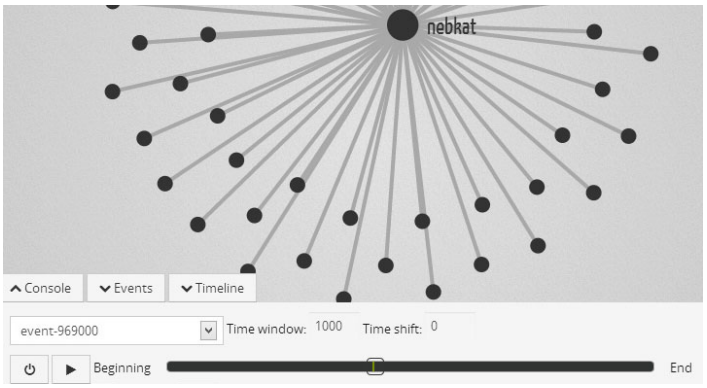

FIG. 7 – *Interactions de l'utilisateur "nebkat" avec un grand nombre de projets lors de l'événement 969000. Le diagramme d'interactions n'est vert qu'à la date de l'événement.*

Événement 969000 : $G_{w=10000}^{i=969000}$ possède 5681 nœuds uniques, 3752 liens uniques, et 1958 composantes connexes. Sa visualisation met en évidence une étoile centrée sur l'utilisateur "nebkat" qui a contribué subitement à plus de 100 projets de l'utilisateur "android-mirror", avec 2138 apparitions de liens. Nous observons aussi une grande composante connexe de projets codés en Rails et en Javascript ayant plus de 240 nœuds, mais qui disparaît à une échelle plus petite (à $w' = 1000$). Nous choisissons donc d'étudier l'étoile de "nebkat". Le diagramme d'interactions de la FIG. 7 montre que "nebkat" n'interagit qu'au moment de l'événement. La quantité d'apparitions de liens à ce moment nous incite à valider l'événement et à le corréler à ce motif. Cependant le compte "android-mirror" a été supprimé de Github.com lors de notre étude, nous ne pouvons donc interpréter cet événement.
Événement 1303000 : $G_{w=10000}^{i=1303000}$ possède 6642 nœuds uniques, 4395 liens uniques, et 2318 composantes connexes. Aucun motif ne se démarque clairement. 17 nœuds ont entre 50 et 110 interactions chacun, et une composante connexe de plus de 380 nœuds apparaît. Nous pouvons donc raisonnablement supposer que la valeur anormale de la statistique est due à une combinaison de facteurs. Nous ne validons pas cet événement.
Événement 1420000 : $G_{w=10000}^{i=1420000}$ possède 4049 nœuds uniques, 2606 liens uniques, et 1458 composantes connexes. Sa visualisation met en évidence une étoile centrée sur l'utilisateur "aruiz" qui a contribué aux projets du groupe "GNOME-Project", avec 5049 apparitions de liens (voir la FIG. 8). Nous observons aussi une étoile centrée sur le projet "twitter/bootstrap", avec 208 apparitions de liens. Nous choisissons d'étudier le motif de l'étoile d'aruiz puisqu'il contient vingt fois plus d'apparitions de liens. Son diagramme d'interactions FIG. 9 montre qu'il n'interagit qu'au moment de l'événement. Nous validons donc l'événement, cependant les informations disponibles sur Github.com ne nous permettent pas de l'interpréter.
Événement 2247000 : $G_{w=10000}^{i=2247000}$ possède 7315 nœuds uniques, 4967 liens uniques, et 2451 composantes connexes. Sa visualisation met en évidence une étoile centrée sur l'utilisateur "Try-Git" avec 300 apparitions de liens. Nous observons aussi deux composantes connexes de resp. 270 et 300 nœuds. L'événement est ambigu. Le diagramme des interactions de "Try-Git" en FIG. 10 montre que la plupart de ses interactions se passent lors de l'événement antérieur 1969000. Nous ne validons donc pas l'événement, et analysons ensuite l'événement 1969000.

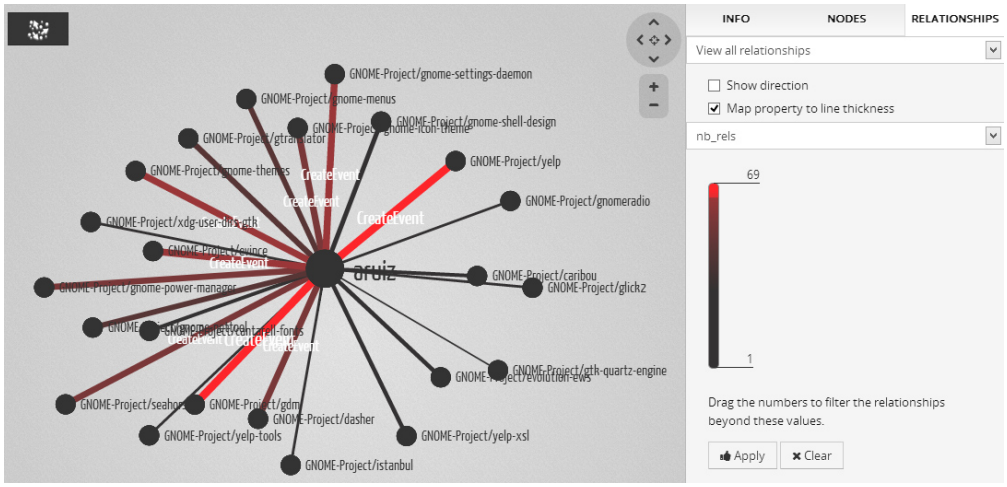

FIG. 8 – *Visualisation des interactions de l'utilisateur "aruiz" lors de l'événement 1420000 avec* $w' = 1000$.

FIG. 9 – *Diagramme d'interactions de l'utilisateur "aruiz" au cours du temps.*

Événement 1969000 : $G^{i=1969000}_{w=10000}$ possède 8651 nœuds uniques, 6817 liens uniques, et 1855 composantes connexes. Sa visualisation met en évidence une étoile centrée sur l'utilisateur "Try-Git" avec 3658 apparitions de liens, ce qui confirme l'événement suggéré par le diagramme d'interactions de "Try-Git" en FIG. 10. Nous apprenons sur la page Web de "Try-Git" 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'événement correspond au moment où Try-Git a été rendu public le 4 juillet 2012 à 17h (cette information est confirmée par un billet sur le blog Github.com [9]). Nous validons donc cet événement.

4 Conclusions et perspectives

Nous avons présenté une nouvelle méthode d'investigation dans un grand flot de liens pour détecter des événements significatifs dans les interactions entre utilisateurs du réseau social en

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

FIG. 10 – *Interactions de "Try-Git" au cours du temps. Les interactions avec ce nœud apparaissent avant l'événement 2247000 et démarrent subitement à l'événement 1969000.*

ligne Github. Des travaux préliminaires ont montré l'intérêt de représenter ces données comme une succession d'apparition de liens dans un graphe dynamique, que nous formalisons par un flot de liens ordonnés chronologiquement. Peu de jeux de données de graphes dynamiques possèdent une telle granularité pour un volume de plusieurs millions de liens, ce qui constitue une opportunité pour développer des méthodes de détection d'événements adaptées.

Nous avons combiné une méthode statistique, *Outskewer*, avec un système de visualisation. *Outskewer* détecte des événements statistiquement significatifs selon une statistique du graphe, et donne des instants de l'évolution du graphe à investiguer. Ces événements n'étant pas tous pertinents, nous avons proposé une étape de validation supplémentaire par la visualisation. Cette dernière montre un sous-graphe correspondant au moment des événements et facilite le suivi longitudinal des motifs anormaux présents dans ce sous-graphe, ce qui permet de localiser des motifs anormaux à ces instants, et de vérifier s'ils semblent corrélés à l'événement. Nous avons illustré notre méthode sur les événements à investiguer, et montrons par plusieurs exemples que nous détectons des événements pertinents, et rejetons des événements proposés par *Outskewer* pour lesquels nous ne trouvons pas d'anomalie dans le graphe. Nous avons ainsi illustré la complémentarité des statistiques et de la visualisation pour détecter des événements dans un grand flot de liens.

Dans des travaux futurs nous proposerons une évaluation du système avec un analyste, expert de son domaine, pour en évaluer l'approche et en critiquer l'IHM. Une comparaison avec des méthodes alternatives est aussi souhaitable. Enfin, nous tenterons de généraliser notre méthode par plusieurs cas réels comme la détection de fraudes ou la détection d'intrusions.

Une telle méthode sans a priori sur les données ni sur les événements est idéale lors d'une démarche exploratoire. La visualisation, en particulier, offre une grande flexibilité quant à la nature des motifs anormaux à observer. Une fois une typologie des événements identifiée en fonction de la tâche d'investigation, il est envisageable d'automatiser entièrement la détection et la validation des événements en replaçant la visualisation par des statistiques sur les motifs. Des méthodes comme *Oddball* (Akoglu et al. (2010)) permettraient de détecter des motifs anormaux dans les sous-graphes correspondant aux fenêtres anormales.

Références

Akoglu, L., M. McGlohon, et C. Faloutsos (2010). Oddball : Spotting anomalies in weighted graphs. In *Advances in Knowledge Discovery and Data Mining*, pp. 410–421. Springer.

Albano, A., B. Le Grand, et M. Latapy (2011). Détection visuelle d'événements dans des grands réseaux d'interaction dynamiques. application à l'internet. In *Atelier Visualisation et Extraction de Connaissances de la conférence EGC 2011*.

Archambault, D., H. Purchase, et B. Pinaud (2011). Animation, small multiples, and the effect of mental map preservation in dynamic graphs. *Visualization and Computer Graphics, IEEE Transactions on 17*(4), 539–552.

Bach, B., E. Pietriga, J.-D. Fekete, et al. (2012). Temporal navigation in dynamic networks. In *IEEE Vis Week*.

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

Brandes, U., N. Indlekofer, et M. Mader (2012). Visualization methods for longitudinal social networks and stochastic actor-oriented modeling. *Social Networks 34*(3), 291–308.

Brandes, U. et D. Wagner (2004). Analysis and visualization of social networks. In *Graph drawing software*, pp. 321–340. Springer.

Chau, D. H. (2012). *Data Mining Meets HCI : Making Sense of Large Graphs*. Ph. D. thesis.

Clauset, A. et N. Eagle (2007). Persistence and periodicity in a dynamic proximity network. In *Proc. DIMACS Workshop on Computational Methods for Dynamic Interaction Networks*.

Ghani, S., N. Elmqvist, et J. S. Yi (2012). Perception of animated node-link diagrams for dynamic graphs. In *Computer Graphics Forum*, Volume 31, pp. 1205–1214. Wiley.

Heymann, S., M. Latapy, et C. Magnien (2012). 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.

Heymann, S. et B. Le Grand (2013). Monitoring user-system interactions through graph-based intrinsic dynamics analysis. *Proc. IEEE International Conference on Research Challenges in Information Science (RCIS 2013), IEEE*.

Jacomy, M., S. Heymann, T. Venturini, et M. Bastian (2011). Forceatlas2, a graph layout algorithm for handy network visualization. *Paris http ://www. medialab. sciences-po. fr/fr/publications-fr*.

Krempel, L. (2005). *Visualisierung komplexer Strukturen*. Campus.

Moody, J., D. McFarland, et S. Bender-deMoll (2005). Dynamic network visualization. *American Journal of Sociology 110*(4), 1206–1241.

Sun, J., C. Faloutsos, S. Papadimitriou, et P. S. Yu (2007). Graphscope : parameter-free mining of large time-evolving graphs. In *Proc. 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 687–696. ACM.

Windhager, F., L. Zenk, et P. Federico (2011). Visual enterprise network analytics-visualizing organizational change. *Procedia-Social and Behavioral Sciences 22*, 59–68.

Summary

We introduce a novel method of exploratory analysis in large link streams, which we apply to significant event detection in more than 2 millions interactions (during 4 months) among users in the Github online social network. We combine a statistical method to automatically detect events in time series, *Outskewer*, with an interactive graph visualization system. *Outskewer* points to interesting moments of the graph evolution, then an analyst validates and interprets the events through visualization of abnormal patterns in the corresponding sub-graphs. We show using several examples that this approach allows to detect relevant events and to reject irrelevant ones, and is suitable for exploratory analysis because no prior knowledge is required.