Un algorithme matriciel pour le calcul des composantes connectées d'un réseau complexe temporel

Rémi Vaudaine, Pierre. Borgnat, Paulo Gonçalves, Rémi Gribonval, Marton Karsai

▶ **To cite this version:**

Rémi Vaudaine, Pierre. Borgnat, Paulo Gonçalves, Rémi Gribonval, Marton Karsai. Un algorithme matriciel pour le calcul des composantes connectées d'un réseau complexe temporel. 29e Colloque GRETSI sur le traitement du signal et des image, GRETSI, Aug 2023, Grenoble, France. pp.1181-118. hal-04227004

HAL Id: hal-04227004
https://hal.science/hal-04227004

Submitted on 3 Oct 2023

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.

Un algorithme matriciel pour le calcul des composantes connectées d'un réseau complexe temporel

Rémi VAUDAINE[1] Pierre BORGNAT[2] Paulo GONÇALVES[1] Rémi GRIBONVAL[1] Márton KARSAI[3]

[1]Laboratoire de l'Informatique du Parallélisme, ENS de Lyon, CNRS, Inria, Univ Lyon 1, Lyon, France

[2]Univ Lyon, ENS de Lyon, CNRS, Laboratoire de Physique, F-69342 Lyon, France

[3]Department of Network and Data Science, Central European University, Quellenstraße 51, 1100 Wien, Austria
Rényi Institute of Mathematics, Reáltanoda u. 13-15, 1053 Budapest, Hungary

Résumé – Les interactions temporelles par paires d'entités peuvent être représentées par des réseaux temporels. Les processus dynamiques, tels que la propagation d'épidémies ou les cascades d'informations, qui évoluent sur ces réseaux, sont contraints à respecter sa structure. En effet, un nœud du réseau ne peut pas affecter plus de nœuds qu'il ne peut atteindre selon l'ordre des interactions. Cet ensemble de nœuds atteignables est appelé la composante sortante et son calcul est coûteux. Nous proposons un algorithme matriciel efficace pour résoudre ce problème et nous montrons qu'il est plus performant que l'état de l'art.

Abstract – Temporal pairwise interactions of entities are depicted by temporal networks. Dynamical processes, such as epidemic spreading or information cascades, evolve on top of these networks. The largest outcome of these processes is directly linked the structure of the underlying network. Indeed, a node of the network cannot affect more nodes than it can reach. This set of nodes that are reachable from a source is called the out-component and its computation for any source is costly. In this paper, we propose an efficient matrix algorithm to tackle this issue and show that it outperforms the state-of-the-art method.

1 Introduction

Les réseaux temporels représentent la séquence des interactions entre des entités qui évoluent dans le temps [4]. Plus riches que les réseaux complexes [1], ils servent de structures sous-jacentes à de nombreux processus dynamiques : propagation d'une épidémie, cascades d'informations, adoption collective de normes comportementales ou de produits. Dans ces réseaux temporels, les informations d'une entité ne peuvent être transférées qu'au moment des interactionse, respectant l'ordre temporel qui peuvent induire des chemins temporels longs et, à leur tour, des structures connectées grandes. Ces dernières, appelées composantes sortantes, codifient l'ensemble des entités, ou nœuds, qui peuvent être atteintes par l'information d'origine. La taille et la structure de ces composantes déterminent le résultat de tout processus dynamique sur un réseau temporel. Aucun processus ne peut atteindre plus que la plus grande composante sortante connectée.

La caractérisation de ces composantes sortantes est une tâche difficile, car l'ordre des interactions introduit une certaine complexité pour détecter les chemins temporels. Il est nécessaire de développer des algorithmes rapides pour caractériser ses composantes, par exemple calculer leurs tailles. L'état de l'art actuel est d'utiliser les graphes d'évènements [7, 6, 2] : c'est une représentation statique et sans pertes d'un réseau temporel sous la forme d'un graphe pondéré, acyclique et dirigé. Pour un réseau temporel de n nœuds et m interactions (ou évènements), cela demande un temps $O(m \log m)$ pour calculer l'ensemble des tailles des composantes sortantes.

Bien que cette méthode soit efficace, le temps de calcul devient élevé si le nombre d'interactions est grand. Comment faire pour diminuer ces coûts de calcul ? Dans ce travail, nous définissons une représentation à l'aide d'une matrice que nous appelons matrice des composantes, qui décrit les plus grandes composantes sortantes et entrantes pour chacune des entités d'un réseau temporel. Ensuite, nous proposons un algorithme en ligne et efficace pour calculer cette matrice.

2 Problème et état de l'art

2.1 Définition du problème

L'objectif est le calcul de la distribution de la taille des composantes sortantes pour les nœuds d'un réseau temporel.

Nous définissons un réseau temporel $\mathcal{G} := (\mathcal{V}, \mathcal{E}, \mathcal{T})$ comme une série d'événements temporels $e = (u, v, t) \in \mathcal{E}$ qui enregistrent les interactions entre les nœuds $u, v \in \mathcal{V}$ aux temps t (sur durée d'observation $\mathcal{T}$). On note $n = |\mathcal{V}|$ le nombre de nœuds et $m = |\mathcal{E}|$ le nombre d'événements.

Dans $\mathcal{G}$, deux événements $e_i \in \mathcal{E}$, $e_j \in \mathcal{E}$ sont adjacents s'ils partagent au moins un nœud ($\{u_i, v_i\} \cap \{u_j, v_j\} \neq \emptyset$) et leur temps inter-événement est $\Delta t = t_j - t_i > 0$, c'est-à-dire que les deux événements ne sont pas simultanés. Une séquence d'événements adjacents définit un chemin temporel entre u et v à partir du temps t, si le premier évènement du chemin part de u à t, le dernier arrive à v et chaque événement consécutif de la séquence est adjacent par paire [4]. L'ensemble des nœuds qui peuvent être atteints par n'importe quel chemin partant du nœud u à l'instant t définit sa composante sortante.

La taille de la composante externe d'un nœud u à un moment donné t est mesurée comme le nombre de nœuds uniques qui peuvent être atteints par des chemins temporels. En fait, elle détermine le plus grand phénomène possible (par exemple, la plus grande épidémie ou cascade d'informations) qui a été initié à partir d'un nœud source et qui a évolué dans le futur. Le calcul des composantes externes d'un réseau temporel est

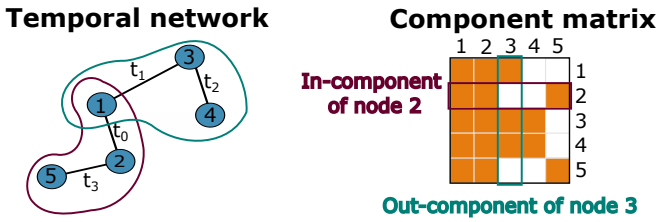

FIGURE 1 : Réseau temporel avec ses événements ordonnés dans le temps ($t_i < t_j$ pour $i < j$). La composante sortante du nœud 3 est entourée en bleu. La composante entrante de 2 l'est en violet. Dans la matrice des composante, une ligne représente la composante entrante d'un nœud, une colonne la composante sortante. Si l'élément de la colonne u, ligne v est non nul, alors v appartient à la composante externe de u.

un défi informatique car il nécessite le suivi de chaque chemin temporel à partir de chaque nœud à chaque instant. seule une solution approximative a été proposée dans [2] pour un défi partiel, à savoir estimer uniquement la taille des composantes sortantes sans suivre les nœuds impliqués.

2.2 Graphes d'évènements et HyperLogLog

Une solution s'appuie sur la représentation des graphes d'événements (GE) [6, 7] des réseaux temporels. Un graphe d'événements $G := (V, E, \Delta t)$ est une représentation statique, pondérée, grâce à un graphe acyclique dirigé (DAG) d'un réseau temporel $\mathcal{G}$. Les événements temporels sont associés à des nœuds dans G (c'est-à-dire $V = \mathcal{E}$), les arêtes dirigées dans G correspondent à des paires d'événements adjacents dans le réseau temporel d'origine, avec une direction indiquant leur ordre temporel. De cette manière, un graphe d'événements a $m = |\mathcal{E}|$ sommets et η arêtes dirigées. Cette représentation fournit une description sans perte d'un réseau temporel et peut être exploitée pour déduire des propriétés de $\mathcal{G}$ sans calculs sur la structure temporelle [6]. La distribution des tailles des composantes externes de $\mathcal{G}$ peut alors être calculée suivant [2], mais avec des coûts de calcul et de mémoire élevés.

Pour réduire ces coûts au détriment du calcul exact, Badie-Modiri et al. [2] ont proposé une approximation pour estimer la distribution de la taille des composantes externes d'un réseau temporel en employant l'algorithme HyperLogLog (HLL) [3]. Cet algorithme estime le cardinal d'un ensemble à partir d'un flux d'éléments avec une complexité de temps et d'espace constante, sans stocker tous les éléments pendant le calcul. Pour estimer la taille d'un ensemble, HLL construit une structure de s registres qui contiennent un nombre. Ensuite, chaque élément de l'ensemble est haché en un vecteur binaire qui est ensuite coupé en deux parties. La première partie indique l'identifiant du registre qui sera utilisé et la position du 1 le plus à gauche dans la deuxième partie est stockée dans ce registre si elle est plus grande que la valeur actuelle. Enfin, la taille de l'ensemble est estimée à l'aide d'une fonction indicatrice d'ensemble basée sur les registres. Le principal avantage de HLL est que l'ensemble n'est pas stocké pour estimer sa taille et que l'estimation peut être effectuée avec un espace constant et une complexité de temps constante $O(s)$. L'erreur d'estimation est de $O(1/\sqrt{s})$. La taille de l'union de deux ensembles peut également être estimée en temps et en espace constants en fusionnant deux ensembles de registres. HLL

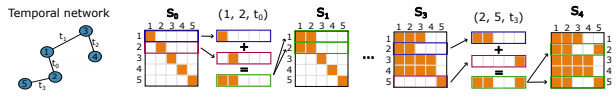

FIGURE 2 : À partir d'un réseau temporel, nous calculons la matrice des composantes en parcourant la série d'événements. Pour chaque événement, le calcul de l'opération OU entre les lignes de la matrice correspondant aux nœuds en interaction donne la nouvelle valeur.

donne une approximation de la taille d'un ensemble, mais avec une précision d'estimation qui peut être arbitrairement ajustée par s, au détriment du temps.

L'algorithme HLL peut être alors utilisé pour estimer la taille des composants externes d'un graphe d'évènements sans suivre l'ensemble exact des nœuds concernés [2]. Cela réduit la complexité temporelle du calcul de la distribution des composantes sortantes à $O(m \log m + \eta)$.

3 Algorithme matriciel pour le calcul de la taille des composantes sortantes

Nous développons ici un algorithme matriciel qui permet une solution exacte d'un défi plus simple : celui de calculer la plus grande composante de sortie de chaque nœud dans un réseau temporel. Notre solution peut traiter en temps réel les nœuds et les événements d'un réseau temporel en flux chronologique avec une complexité spatiale qui ne dépend pas du nombre m d'événements. Ceci est réalisé grâce à la matrice des composantes et un exemple est donnée en Fig. 1. Dans cette matrice, une ligne décrit l'ensemble des nœuds qui peuvent envoyer une information au nœud de la ligne. Une colonne donne l'ensemble des nœuds qui peuvent recevoir une information depuis le nœud de la colonne.

3.1 Description de l'algorithme

A t_0, la matrice des composantes S_0 est définie par :

$$S_0 = I_n \tag{1}$$

où I_n est la matrice booléenne Identité de taille $n \times n$ comme indiqué sur la Fig. 2. Pour les étapes temporelles suivantes, la règle de mise à jour de la matrice des composantes est décrite dans l'Algorithme 1, qui peut être résumée comme suit. Pour chaque événement, dans l'ordre chronologique, (u_i, v_i, t_i), nous calculons l'opération binaire OU entre les lignes des nœuds u et v au moment t_{i-1}.

$$r = S_{i-1}(u_i, .) \text{ OR } S_{i-1}(v_i, .) \tag{2}$$

On obtient une nouvelle ligne r décrivant l'*union* des composantes entrantes de u_i et v_i. Pour obtenir la matrice S_i au pas de temps actuel, nous mettons à jour la matrice S_{i-1} du pas de temps précédent aux lignes u_i et v_i en les remplaçant par r.

Lorsque tous les événements ont été traités, nous prenons la matrice $S_\mathcal{T}$ obtenue à l'étape finale et comptons le nombre de nœuds uniques dans chaque composante externe, afin d'obtenir la distribution maximale de la taille de la composante externe sur chaque nœud du réseau. Pour ce faire, il suffit de compter le nombre de 1 dans chaque colonne de $S_\mathcal{T}$. Il convient

de noter que les tailles maximales des composantes externes peuvent être calculées à tout moment pendant le traitement du réseau temporel, ce qui rend cette approche très efficace en tant qu'algorithme de flux.

Algorithme 1 : L'algorithme matriciel

```
Data : Liste d'interaction ordonnée E
Result : Matrice des composantes S
S ← I_n ;                    // Initialisation
for e ∈ ℰ do
    u ← e[0]
    v ← e[1]
    r ← S[u] OR S[v] ;       // Calcule le OU
     binaire
    S[u] ← r ;  // Remplace ligne u par r
    S[v] ← r ;  // Remplace ligne v par r
end
```

Comme nous utilisons une matrice de taille $n \times n$ pour stocker les résultats, la complexité spatiale est $O(n^2)$, ce qui peut être réduit en utilisant des matrices creuses pour le stockage. Pour la complexité temporelle, nous pouvons diviser l'algorithme en plusieurs étapes. La création de la matrice identité peut être réalisée en $O(n)$ en fixant simplement les n éléments diagonaux à la valeur "True" au départ. Pour mettre à jour la matrice, nous effectuons l'opération OU entre deux vecteurs de taille n une fois pour chaque événement m. La complexité de chaque mise à jour est en $O(n)$, limitée par la taille maximale des composantes extérieures n, mais elle pourrait être encore réduite avec un format de matrice creuse. En conséquence, la complexité totale des mises à jour est de $O(nm)$. Enfin, le comptage du nombre d'éléments non nuls (ou d'éléments non "False") peut être effectué en même temps que la mise à jour sans aucune complexité supplémentaire. Ainsi, la complexité temporelle globale de l'algorithme de la matrice des composants est de $O(n) + O(nm) = O(nm)$.

3.2 Distribution des tailles des composantes sortantes grâce au renversement du temps

Il nous manque la distribution de la taille maximale des composants externes dans $\mathcal{G}$. En considérant un parcours de matrices en inversant le temps, nous pouvons en fait facilement obtenir une solution qui permet de calculer l'ensemble de cette distribution à moindre coût. Cependant, nous perdons la propriété de flux de notre algorithme, car cette solution prend en entrée l'ensemble de la séquence d'interaction dans un ordre inversé, pour la traiter de la fin vers le début.

L'inversion de la séquence des événements nous permet d'utiliser la méthode de comptage HyperLogLog pour l'estimation de la distribution des tailles maximales des composantes sortantes. Plus précisément, pour chaque nœud, nous initialisons une structure HyperLogLog qui ne contient que le nœud lui-même. Ensuite, pour chaque événement (u_i, v_i, t_i), *en les prenant dans l'ordre chronologique inverse*, nous fusionnons les structures u_i et v_i (correspondant aux colonnes, c'est-à-dire aux estimations actuelles des composantes externes) en un temps $O(s)$. Enfin, nous estimons la taille de la composante externe maximale de chaque nœud à l'aide des estimations HLL. Cela permet d'obtenir une approximation de l'ensemble de la distribution des tailles des composantes externes en $O(n)$ de complexité spatiale, $O(m)$ de complexité temporelle et un seul parcours de la séquence d'événements.

4 Validation expérimentale

Nous réalisons des expériences pour tester l'efficacité de l'algorithme matriciel et comparer ses performances à la solution correspondante basée sur les graphes d'évènements.

4.1 Cadre expérimental

Pour tester les performances de ces algorithmes, nous utilisons un réseau temporel issu d'un modèle aléatoire. En pratique, nous générons d'abord un graphe aléatoire statique non dirigé en utilisant la solution $G(n, p)$ du modèle d'Erdos-Rényi avec n nœuds et une probabilité de liens $p = 2/n$. De cette manière, le graphe statique construit contiendra probablement une grande composante connectée de taille $O(n^{2/3})$. Pour générer un réseau temporel, nous définissons un processus de Poisson indépendant sur chaque lien de ce graphe afin de déterminer les moments où les liens sont présents et peuvent transmettre des informations entre les nœuds connectés [2]. De cette manière, la structure sous-jacente et la dynamique des liens sont générées par des processus aléatoires, qui induisent des hétérogénéités de degré limitées (avec une distribution de degré de Poisson émergente) [1] et pas d'éclatement (avec une distribution exponentielle du temps inter-événements) [5]. Dans nos réseaux simulés, le nombre de nœuds n varie entre $\{100, 200, 500, 1000, 2000, 5000, 10000\}$ et le nombre d'événements m entre 100 et 10^8 en puissances de 10. Bien que les réseaux temporels réels puissent présenter plusieurs types d'hétérogénéités structurelles et temporelles, nous supposons qu'elles ne modifieraient pas considérablement la conclusion de notre évaluation des performances à venir.

4.2 Mesures de performance

Naturellement, nous choisissons le temps et la mémoire nécessaire au calcul de la distribution de la tailles des plus grandes composantes sortantes pour comparer les différentes méthodes.

4.3 Résultats

Pour comparer les différentes méthodes, nous indiquons dans la Fig. 3 le ratio de temps de calcul et le ratio d'utilisation de la mémoire entre les algorithmes comparés. Concentrons-nous d'abord sur les performances relatives de la méthode de la matrice de composants dans les Fig. 3a et c. Il est intéressant de noter que les résultats décrits dans le panneau (a) suggèrent que, bien que cette méthode fournisse une solution exacte pour la tâche, elle est toujours plus performante que l'algorithme GE+HLL en termes de temps de calcul. Il en va de même pour la complexité de la mémoire (panneau (c)), bien que le coût quadratique élevé de la matrice des composants rende cette méthode moins performante que la référence pour un petit nombre d'événements ou un grand nombre de nœuds. Néanmoins, nous pouvons conclure que la méthode de la matrice des composants est largement plus performante que la méthode du graphe des événements en termes de temps de calcul et de mémoire pour un grand nombre d'événements,

Méthode	Complex. temps	Complex. espace	Exacte ?	Stream ?	OC ?
GE + HLL	$O(m\log(m)+\eta)$	$O(m+\eta)$	No	No	No
Matrice	$O(mn)$	$O(n^2)$	Yes	Yes	Yes
Matrix + reverse t + HLL	$O(m)$	$O(n)$	No	No	No

TABLE 1 : Résumé des caractéristiques des méthodes de calcul de la distribution maximale de la taille des composantes sortantes. La complexité en termes de temps et d'espace dépend du nombre de nœuds n et d'événements m, ainsi que du nombre d'arêtes η dans le graphe d'événements (GE). La colonne intitulée "Exact" indique si la méthode fournit une solution exacte (Oui) ou approximative (Non). La colonne "Stream" indique si la méthode permet de traiter les événements dans l'ordre chronologique. La colonne "OC" indique si la méthode peut calculer non seulement les tailles des composantes externes, mais aussi les nœuds concernés.

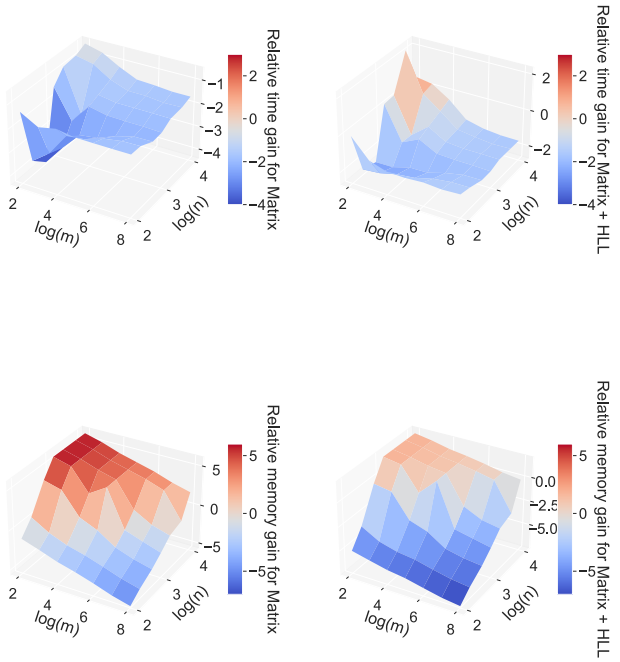

FIGURE 3 : Ratio du temps de calcul (haut) et de l'utilisation de la mémoire (bas) de la méthode proposée par rapport à la méthode [2]. (a) et (c) sont les résultats de la méthode exacte de la matrice des composantes, tandis que les (b) et (d) sont pour la solution approximative à l'aide de HLL. Les échelles sont logarithmiques, la couleur bleue indiquant que la méthode de la matrice des composantes est plus performante (la couleur est rouge dans le cas contraire).

en particulier dans les réseaux de petite taille où le gain peut atteindre plusieurs ordres de grandeur.

Nous avons constaté des performances plus variables en comparant l'algorithme de la matrice des composants combiné à HLL avec l'approche GE+HLL de référence [2]. En termes de temps de calcul (voir le panneau (b) de la Fig. 3), il est un peu moins performant que la référence pour un petit nombre d'événements et de grandes tailles de réseau, mais sinon il peut atteindre des gains similaires pour le reste de l'espace de paramètres que la méthode exacte. Cependant, il est nettement plus performant en termes de mémoire (voir panneau (d)) car il n'utilise pas de matrice quadratique en taille pour représenter la matrice des composants. Pour les réseaux de grande taille, le modèle nécessite approximativement la même taille de mémoire que le modèle de référence, alors qu'il est nettement plus performant pour le reste de l'espace des paramètres.

5 Conclusion

La structure des réseaux temporels détermine largement le résultat des processus dynamiques qui évoluent sur ces premiers. La taille de la composante sortante d'un nœud est la taille maximale d'un processus dynamique commençant à ce nœud.

Ainsi, nous avons proposé une nouvelle méthode matricielle de calcul des composantes sortantes. Cette approche est plus rapide que l'état de l'art et nécessite, dans certains cas, moins de mémoire. De plus, cette méthode requiert moins de mémoire en utilisant HyperLogLog au prix d'un peu de temps.

Notre méthode apporte une réponse efficace au problème du calcul de la taille des composantes sortantes. Cependant, le coût mémoire reste élevé dans certains cas. Il serait sans doute intéressant de considérer des techniques de réduction de dimension pour diminuer la taille des matrices considérées.

Références

[1] Réka ALBERT et Albert-László BARABÁSI : Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002.

[2] Arash BADIE-MODIRI, Márton KARSAI et Mikko KIVELÄ : Efficient limited-time reachability estimation in temporal networks. *Phys. Rev. E*, 101:052303, May 2020.

[3] Philippe FLAJOLET, Éric FUSY, Olivier GANDOUET et Frédéric MEUNIER : Hyperloglog : The analysis of a near-optimal cardinality estimation algorithm. *In AOFA'07 (2007 Int. Conf. on Analysis OF Algorithms*, 2007.

[4] Petter HOLME et Jari SARAMÄKI : Temporal networks. *Physics Reports*, 519(3):97–125, oct 2012.

[5] Márton KARSAI, Hang-Hyun JO, Kimmo KASKI *et al.* : *Bursty human dynamics*. Springer, 2018.

[6] Mikko KIVELÄ, Jordan CAMBE, Jari SARAMÄKI et Márton KARSAI : Mapping temporal-network percolation to weighted, static event graphs. *Scientific reports*, 8(1):12357, 2018.

[7] Andrew MELLOR : The temporal event graph. *Journal of Complex Networks*, 6(4):639–659, oct 2017.

Ce travail a été financé par l'ANR, dans le cadre du projet DATAREDUX ANR19-CE46-0008.