Détection de composantes connexes persistantes non-dominées dans un graphe dynamique

Mathilde Vernet, Yoann Pigné, Eric Sanlaville

Détection de composantes connexes persistantes non-dominées dans un graphe dynamique

Mathilde Vernet et Yoann Pigné et Eric Sanlaville

Normandie Univ, UNIHAVRE, UNIROUEN, INSA Rouen, LITIS, 76600 Le Havre, France

Ce travail s'intéresse au problème de connexité dans un graphe dynamique. Si cette question est assez triviale dans un graphe statique, elle ne l'est pas autant lorsque le graphe varie au cours du temps. Nous étudions, sur un intervalle de temps défini, un graphe non-orienté dont les arêtes peuvent apparaitre et disparaitre. Nous définissons une composante connexe persistante comme un ensemble de noeuds qui restent connectés pendant un certain temps. Nous introduisons une notion de dominance définie selon deux critères : le nombre de noeuds d'une composante et le nombre de pas de temps successifs pendant lesquels elle est présente. Nous proposons un algorithme incrémental de complexité polynomiale permettant d'extraire, pour un graphe dynamique donné, toutes les composantes connexes persistantes non-dominées.

Mots-clefs : graphe dynamique, connexité, algorithme polynomial, composante connexe persistante

1 Introduction

Le problème de connexité dans un graphe, plutôt simple dans le contexte statique, est une question importante dans de nombreux cas. Cela peut être une condition initiale nécessaire pour la résolution de certains problèmes, comme les flots par exemple pour lesquels sources et puits doivent être connectés. La connexité peut aussi être utilisé pour décomposer des problèmes afin de les résoudre indépendamment sur les différentes composantes connexes du graphe, comme les problèmes de coloration de graphes par exemple. La connexité présente également un intérêt en soi dans de nombreux domaines d'applications, comme les réseaux de communication ou les réseaux logistiques par exemple.

Un graphe statique est limité en terme de modélisation car il ne peut rendre compte de l'évolution d'un système au cours du temps. C'est la raison pour laquelle on étudie les graphes dynamiques qui eux, prennent en compte la variable temporelle en permettant aux noeuds et aux arêtes ainsi qu'à toutes les informations portées par ceux-ci de varier au cours du temps.

Mais alors, que devient la notion de connexité dans un graphe dynamique ? C'est la question que nous abordons dans ce travail. Nous proposons une définition qui permet de rendre compte de la persistance de la connexité des noeuds du graphe au cours du temps. Nous présentons ces éléments en section 3 après avoir présenté brièvement quelques travaux traitant de problèmes similaires en section suivante. Nous proposons ensuite un algorithme capable de calculer les composantes non-dominées. Cet algorithme est incrémental, il n'est donc pas nécessaire de connaitre à l'avance l'évolution du graphe pour l'exécuter.

Dans le cas des réseaux de communication, comme les réseaux de capteurs mobiles ou actifs par intermittence, les composantes connexes persistantes caractérisent un ensemble de noeuds restant connectés pendant une période de durée maximum.

2 État de l'art

La question de la connexité dans un graphe dynamique s'est déjà posée dans la littérature. On trouve des travaux dans lesquels la définition de connexité dans un graphe dynamique est basée sur des chemins

dynamiques, c'est-à-dire une suite d'arêtes dont les pas de temps de présence sont croissants. C'est le cas pour [BF03], où deux noeuds u et v sont dans une même composante connexe, au sens dynamique, si au cours de l'évolution de graphe, il existe un chemin dynamique permettant d'aller de u à v et un de v à u. Une définition assez similaire est proposée dans [GCCLL15] où le nombre de pas de temps associé au chemin dynamique est borné par une valeur Δ. Toujours utilisant les chemins dynamiques, dans [HDBXM16], l'accessibilité de u à v et de v à u doit se faire par un chemin dynamique, non pas seulement au cours de l'évolution du graphe mais dans chaque fenêtre de temps de taille fixée pendant l'évolution du graphe.

On trouve dans la littérature des travaux qui se sont intéressés à la question de la connexité dans les graphes dynamiques sans utiliser la notion de chemins dynamiques. C'est le cas de [CKNP15], qui étudie la connexité de graphes statiques correspondant à l'intersection du graphe dynamique sur un nombre donné de pas de temps successifs. Dans [AS19], les auteurs s'intéressent à l'intervalle de temps pendant lequel le graphe tout entier, ou un sous-ensemble de noeuds du graphe, reste connexe pour une date de départ donnée.

Nous proposons dans la suite une définition qui permet de rendre compte de la connexité d'un graphe au cours du temps. Nous souhaitons pouvoir identifier les ensembles de noeuds qui restent connexes au cours de l'évolution du graphe, même si cette connexité se fait via des chemins différents. Cette notion semble particulièrement appropriée pour les graphes dynamiques.

3 Modèle de graphe et composantes connexes persistantes

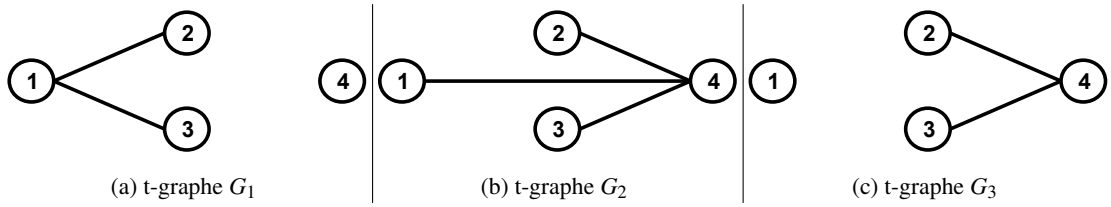
Graphe dynamique Nous considérons un graphe G non orienté de n noeuds et son évolution sur l'intervalle d'étude $\mathcal{T} = \{1, \dots, T\}$ au cours duquel les arêtes du graphe peuvent apparaitre et disparaitre. On appelle *horizon de temps* la fin de l'intervalle d'étude, soit le pas de temps T. On appelle *t-graphe* le graphe statique qui représente l'état de G à un pas de temps i donné de l'intervalle d'étude. Le graphe dynamique $G = (G_i)_{i \in \mathcal{T}}$ est une succession de t-graphes $G_i = (V, E_i)$ définis sur le même ensemble de noeuds. Ce modèle permet de prendre en compte l'apparition ou la disparition de noeuds via l'existence d'arêtes incidentes : tout ce passe comme si un noeud isolé au pas de temps i disparaissait du graphe à cette date.

Composante connexe persistante Une composante connexe persistante (PCC) p est un ensemble K de k noeuds connectés directement ou indirectement pendant l pas de temps successifs de la date $f - l + 1$ à la date f incluses. On la note $p = (K, k, l, f)$ et on appelle k la taille de p, l sa durée, et f sa date de fin.

Dominance Une PCC $p = (K, k, l, f)$ domine une PCC $p' = (K', k', l', f')$ lorsqu'elle possède au moins autant de noeuds et est présente plus longtemps ($k \geq k'$ et $l > l'$) ou bien lorsqu'elle possède au moins un noeud supplémentaire et qu'elle est présente au moins aussi longtemps ($l \geq l'$ et $k > k'$).

Afin de ne conserver qu'une seule PCC pour une taille et une durée données, on pose aussi que p domine p' si elles ont la même taille et la même durée mais que p s'achève avant p' ($k = k'$, $l = l'$ et $f < f'$). Dans le cas où p et p' auraient aussi la même date de fin, un ordre lexicographique sur K et K' est pris en compte.

Exemple Un exemple de graphe dynamique est proposé dans la Figure 1. Ce graphe, sur 3 pas de temps, possède 4 noeuds. Il n'est pas connexe sur tout l'intervalle d'étude. Ce graphe contient plusieurs composantes connexes persistantes maximales en terme de taille et de durée : $p_1 = (\{1,2,3\}, 3, 2, 2)$, $p_2 = (\{1,2,3,4\}, 4, 1, 2)$, $p_3 = (\{2,3,4\}, 3, 2, 3)$ et $p_4 = (\{2,3\}, 2, 3, 3)$. Les PCC p_1 et p_3 ont la même taille (3 noeuds) et la même durée (2 pas de temps) mais p_1 se termine avant p_3 donc p_1 domine p_3.



(a) t-graphe G_1 (b) t-graphe G_2 (c) t-graphe G_3

FIGURE 1: Évolution d'un graphe dynamique sur 3 pas de temps. Chaque t-graphe est représenté.

4 Algorithme PICCNIC

Nous proposons l'algorithme PICCNIC (Persistent Connected CompoNent Incremental Algorithm) dont l'objectif est de trouver les composantes connexes persistantes non dominées dans un graphe dynamique donné. Cet algorithme fonctionne de manière incrémentale sur l'intervalle d'étude du graphe. À la fin de l'itération t, ont été calculées toutes les PCC non-dominées qui existent dans le graphe jusqu'au temps t. Chaque itération de PICCNIC s'intéresse à un pas de temps du graphe et est composée de deux étapes.

La première étape a pour but de calculer les PCC présentes au pas de temps courant, c'est-à-dire celles qui apparaissent au pas de temps courant ainsi que celles qui étaient déjà présentes au pas de temps précédent et qui existent encore. Pour cela, une intersection est réalisée entre toutes les composantes connexes (au sens statique) du t-graphe courant et toutes les composantes connexes persistantes présentes au pas de temps précédent.

La seconde étape de PICCNIC permet d'identifier les composantes connexes persistantes qui viennent de se terminer et de ne conserver en mémoire que les dominantes. Les PCC qui viennent de se terminer sont celles qui étaient présentes au pas de temps précédent et ne le sont plus au pas de temps courant. Une fois identifiées, elles sont comparées, d'après le critère de dominance, entre elles mais aussi aux PCC qui étaient dominantes jusqu'alors, et seules les dominantes sont conservées.

D'une itération t de PICCNIC à la suivante $t+1$, deux ensembles de PCC sont alors conservés. Un premier ensemble, PCC_c, contient les PCC en cours, c'est-à-dire présentes à t. Un second ensemble, PCC_f, contient les PCC terminées et dominantes.

Pour l'initialisation de l'algorithme au premier pas de temps, les PCC en cours sont les composantes connexes dans G_1, et aucune n'est encore terminée. Enfin, une dernière itération de l'algorithme est nécessaire, à la fin, pour traiter les PCC encore présentes au pas de temps T et donc qui se terminent à T.

On notera que les composantes connexes persistantes composées d'un seul noeud sont ignorées. Il est inutile de les calculer puisqu'un noeud n'étant jamais déconnecté de lui-même, tout graphe dynamique possède n composantes connexes persistantes singleton qui durent T pas de temps et se terminent à la date T.

Correction Nous pouvons prouver qu'à chaque pas de temps t toute PCC dominante terminée au plus tard au temps $t-1$ a été identifiée et est conservée dans l'ensemble PCC_f et que toute PCC dominante terminée exactement au temps t a été identifiée et est conservée dans l'ensemble PCC_c. En prouvant ensuite que l'ensemble PCC_f ne contient que des PCC dominantes, cela montre que PICCNIC est correct.

Complexité Soient deux PCC p et p' contenues dans PCC_c au même pas de temps. Alors leurs ensembles de sommets respectifs K et K' sont soit strictement disjoints, soit l'un est strictement inclus dans l'autre. En effet, si K et K' s'intersectent à une date t donnée, cela signifie que leur union forme une composante connexe sur G_t. Puisque l'on s'intéresse aux composantes maximales en terme de nombre de noeuds, c'est donc cette union qui sera ajoutée à l'ensemble PCC_c à la date t.

On peut en déduire que la taille de l'ensemble PCC_c est bornée par le nombre de noeuds du graphe. On peut aussi prouver que la taille de l'ensemble PCC_f est bornée par le nombre de noeuds du graphe. Sachant cela, on montre que la complexité d'une itération de PICCNIC est $O(n^2)$. La complexité totale de l'algorithme est donc $O(n^2 \cdot T)$.

Étude expérimentale Nous avons réalisé une étude expérimentale que nous choisissons de ne pas présenter ici par manque d'espace. Nous avons implémenté notre algorithme en utilisant la bibliothèque GraphStream [DGOP07]. L'objectif premier était de déterminer l'impact de la structure du graphe sur la taille et la durée des composantes connexes persistantes. Nous souhaitions également comparer le temps d'exécution de l'algorithme en pratique avec son comportement théorique déterminé par sa complexité. Cette étude est présentée en détails dans [VPS20], de même que les démonstrations détaillées de la correction et de la complexité de l'algorithme.

5 Conclusion

Nous avons proposé, dans ce travail, une définition permettant d'étendre la notion de connexité et de composantes connexes aux graphes dynamiques. Cette définition permet de rendre compte de la manière dont les noeuds restent connectés au cours de l'évolution du graphe dynamique.

Nous avons proposé PICCNIC, un algorithme de complexité polynomiale $O(n^2 \cdot T)$ capable de calculer les composantes connexes persistantes non-dominées dans un graphe dynamique. L'avantage de cet algorithme est qu'il est incrémental. À la fin de chaque itération, nous avons calculé toutes les composantes connexes persistantes dominantes jusqu'au pas de temps correspondant. Il n'est pas nécessaire de savoir à l'avance comment le graphe va évoluer.

Nous nous intéressons ici aux graphes non-orientés, mais nous pouvons tout à fait étendre nos définitions pour prendre en compte des graphes orientés. Pour cela, nous devons considérer les composantes fortement connexes des graphes statiques. L'algorithme PICCNIC peut alors être utilisé dans ce cas.

Nous pouvons aussi envisager des applications pour lesquelles la durée effective entre deux pas de temps n'est pas constante sur toute l'évolution du graphe. L'algorithme PICCNIC peut être utilisé dans ce cas, après des modifications mineures sur le calcul de l'âge d'une composante. En effet, plutôt que le nombre de pas de temps, il faudrait prendre en compte la différence entre la date de début et la date de fin de la composante.

Références

[AS19] Eleni C Akrida and Paul G Spirakis. On verifying and maintaining connectivity of interval temporal networks. *Parallel Processing Letters*, 29(02) :1950009, 2019.

[BF03] Sandeep Bhadra and Afonso Ferreira. Complexity of connected components in evolving graphs and the computation of multicast trees in dynamic networks. In *International Conference on Ad-Hoc Networks and Wireless*, pages 259–270. Springer, 2003.

[CKNP15] Arnaud Casteigts, Ralf Klasing, Yessin M Neggaz, and Joseph G Peters. Efficiently testing t-interval connectivity in dynamic graphs. In *International Conference on Algorithms and Complexity*, pages 89–100. Springer, 2015.

[DGOP07] Antoine Dutot, Frédéric Guinand, Damien Olivier, and Yoann Pigné. Graphstream : A tool for bridging the gap between complex systems and dynamic graphs. In *Emergent Properties in Natural and Artificial Complex Systems. Satellite Conference within the 4th European Conference on Complex Systems (ECCS'2007)*, 2007.

[GCCLL15] Carlos Gómez-Calzado, Arnaud Casteigts, Alberto Lafuente, and Mikel Larrea. A connectivity model for agreement in dynamic systems. In *European Conference on Parallel Processing*, pages 333–345. Springer, 2015.

[HDBXM16] Charles Huyghues-Despointes, Binh-Minh Bui-Xuan, and Clémence Magnien. Forte Δ-connexité dans les flots de liens. In *ALGOTEL 2016-18èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications*, 2016.

[VPS20] Mathilde Vernet, Yoann Pigne, and Eric Sanlaville. A Study of Connectivity on Dynamic Graphs : Computing Persistent Connected Components. working paper, preprint, available at `https://hal.archives-ouvertes.fr/hal-02473325`, February 2020.