L'exploration des réseaux de transport [1]

La puissance de l'attente aux stations

Ahmed Mouhamadou Wade — David Ilcinkas

LaBRI
CNRS & Université de Bordeaux
France
{ilcinkas, wade}@labri.fr

RÉSUMÉ. Nous étudions le problème d'exploration, par une entité mobile (agent), d'une classe de graphes dynamiques, appelés graphes périodiquement variables (PV-graphes). Ils sont définis par un ensemble de transporteurs suivants infiniment leur route le long des stations du réseau. Flocchini, Mans et Santoro [6] ont étudié ce problème dans le cas où l'agent doit toujours rester sur les transporteurs. Dans ce papier, nous étudions l'impact de la capacité d'attendre sur les stations. Nous prouvons que l'attente sur les stations permet à l'agent, au pire cas, de réduire le nombre de mouvements d'un facteur multiplicatif d'au moins $\Theta(p)$, et la complexité en temps de $\Theta(kp^2)$ à $\Theta(np)$, où n est le nombre de stations, k le nombre de transporteurs, et p la période maximale.

ABSTRACT. We study the problem of exploration by a mobile entity (agent) of a class of dynamic networks, namely the periodically-varying graphs (PV-graphs). These are defined by a set of carriers following infinitely their prescribed route along the stations of the network. Flocchini, Mans, and Santoro [6] studied this problem in the case when the agent must always travel on the carriers and thus cannot wait on a station. In this paper, we study the impact of the ability to wait at the stations. We prove that waiting at the stations allows the agent to reduce the worst-case optimal number of moves by a multiplicative factor of at least $\Theta(p)$, while the time complexity is reduced from $\Theta(kp^2)$ to $\Theta(np)$, where n is the number of stations, k is the number of carriers, and p is the maximum period.

MOTS-CLÉS : Exploration, Graphes dynamiques, Agent mobile, PV-graphes

KEYWORDS : Exploration, Dynamic graphs, Mobile agent, PV-graphs

1. Ce travail a été réalisé avec le soutien du projet ANR ALADDIN, du projet INRIA CEPAGE, et du projet européen EULER. Une version étendue de ces travaux a été publiée dans les actes de la 15th International Conference on Principles of Distributed Systems (OPODIS 2011).

1. Introduction

L'exploration de graphes est un problème classique très étudié depuis sa formulation initiale en 1951 par Shannon [10]. Explorer un graphe consiste, pour une entité mobile (agent), à visiter tous les sommets au moins une fois puis à quitter le système en un temps fini. Ce problème étant l'un des plus classiques dans le cadre du calcul par agent mobile, il a reçu beaucoup d'intérêt jusqu'ici. La complexité en temps, en espace ou l'impact d'une connaissance à priori ont été largement étudiés au cours des 40 dernières années (voir, par exemple, [4, 8, 9]). Cependant, la plus grande partie de ces travaux concernent les graphes statiques. Les nouvelles générations d'environnements interconnectés sont extrêmement dynamiques et évoluent dans le temps. Pour cela, des chercheurs ont commencé à étudier ces réseaux dynamiques. Dans ce papier, nous étudions le problème d'exploration de graphes dans un modèle de réseaux dynamiques, à savoir les graphes périodiquement variables (PV-graphes).

Grossièrement, un PV-graphe consiste en un ensemble de transporteurs suivant périodiquement leur route respective parmi les sites (sommets) du système. Les PV-graphes modélisent en particulier les systèmes de transport publics, par exemple les systèmes de bus ou de métros. Ils modélisent également les systèmes de satellites en orbite autour de la Terre, ou les systèmes de sécurité composés d'agents de sécurité faisant leur ronde. Explorer ces environnements permet par exemple d'effectuer une opération de maintenance etc. En effet, un agent peut vérifier si tout est en ordre au cours de l'exploration. Cet agent peut être un logiciel, un robot ou un être humain.

Le problème d'exploration de PV-graphes était déjà étudié par Flocchini, Mans et Santoro [6]. Ils ont considéré que l'agent ne peut pas quitter le transporteur pour rester sur un site. Ne pas être capable de rester sur un site est particulièrement légitime dans les systèmes de satellites en orbite autour de la Terre par exemple, ou les sites ne correspondent pas à une quelconque station physique. Cependant, dans la plupart des systèmes de transport public, il est possible pour l'agent (humain ou non) de rester sur un site afin d'attendre un transporteur (éventuellement différent). Dans ce papier, nous considérons le même problème mais dans le cas où l'agent peut quitter les transporteurs pour attendre sur un site. Nous étudions l'impact de cette nouvelle capacité sur la complexité (temps et nombre de mouvements) du problème d'exploration de PV-graphes.

Dans cet article, nous étendons le travail sur les PV-graphes de Flocchini, Mans et Santoro [6] dans le cas où l'agent peut descendre d'un transporteur et rester sur un site. Cette possibilité permet à l'agent de pouvoir explorer même les PV-graphes qui ne sont pas hautement connexes. Nous prouvons que dans le cas général la complexité en mouvements est réduite de $\Theta(kp^2)$ à $\Theta(min\{kp, np, n^2\})$, et la complexité en temps de $\Theta(kp^2)$ à $\Theta(np)$. (Notons que dans les PV-graphes connexes, nous avons $n \leq kp$). Si le PV-graphe est homogène et hautement connexe, Flocchini, Mans et Santoro prouvent que la complexité en temps est $O(kp)$. Dans cet article nous allons prouver que si le PV-graphe satisfait uniquement une des deux restrictions, la complexité en temps est $\Theta(np)$. Par ailleurs, il se trouve que notre algorithme effectue non seulement l'exploration, mais réalise également la cartographie, c'est à dire qu'il peut produire une copie isomorphe du PV-graphe. Enfin, notons que notre algorithme n'utilise pas forcement les identifiants des sites.

2. Etat de l'art

Ces dernières années, un travail de recherche important a été fait dans les graphes dynamiques, notamment dans l'exploration et la cartographie de ces environnements très dynamiques par des agents mobiles. Motivés par l'exploration robotisée du Web, Cooper et Frieze [3] ont étudié en 2003 la question du temps de couverture minimal d'un graphe qui évolue au cours du temps. Ils ont considéré un modèle particulier de graphes du web et montrent que si après chaque nombre constant de pas de la marche un nouveau sommet apparaît et est raccordé au graphe, une marche aléatoire probabiliste sur le graphe manque de visiter une fraction constante de sommets. C. Avin et al. [1] montrent en utilisant une marche aléatoire que le temps de couverture minimale de certains graphes dynamiques est exponentiel. Dans le même papier, les auteurs montrent que la couverture minimale d'un graphe dynamique connexe est polynomiale en temps, en utilisant la marche aléatoire paresseuse "lazy random walk". Flocchini et al. [5] étudient la cartographie d'un PV-graphe contenant des trous noirs. Pour cela ils considèrent que le système est composé d'un nombre k d'agents et leur but est de construire la carte sans perdre un grand nombre d'agents. Casteigts et al. [2] intègrent dans un cadre unifié une vaste collection de concepts, de formalismes et de résultats obtenus dans la littérature des graphes variables dans le temps.

En 2009, Flocchini et al. [6] considèrent une famille de graphes variables, les graphes périodiquement variables, et montrent que si les sites du graphe sont étiquetés, la connaissance d'une borne supérieure sur la plus grande période ou du nombre n de sites est nécessaire pour qu'un agent puisse explorer le graphe. Si les sites du graphe sont anonymes, la connaissance d'une borne supérieure sur la plus grande période est nécessaire. Dans les deux cas, ils prouvent que la complexité en temps et en mouvements de l'agent est, dans le cas général, $\Theta(kp^2)$, où k est le nombre de transporteurs et p la période maximale des transporteurs. Dans le cas particulier des PV-graphes homogènes (PV-graphes pour lesquels tous les transporteurs ont la même période), ils prouvent que la complexité en temps et en mouvements passe à $\Theta(kp)$.

3. Définitions et modèles

Nous considérons un système $S = \{s_1, \ldots, s_n\}$ de n sommets (sites) parcouru par k transporteurs. Chaque transporteur c a un identifiant Id(c) et parcourt une séquence ordonnée $R(c) = (s_{i_1}, \cdots, s_{i_{p(c)}})$ de sites, appelée route, de manière périodique. L'entier positif $p(c)$ est appelé période du transporteur c. Plus précisément, le transporteur c commence à un site s_{i_1} au temps 0 et à chaque unité de temps, il passe d'un site à son suivant de manière cyclique (quand c est au site s_{i_p}, il revient au site s_{i_1}).

Un PV-graphe (graphe périodiquement variable) est l'ensemble (S, C), où S est l'ensemble des sites, et C l'ensemble des transporteurs qui circulent sur les sites. Nous notons par n, k et p, respectivement, le nombre de sites, le nombre de transporteurs et le maximum sur les périodes des transporteurs. Un PV-graphe est dit homogène si et seulement si tous les transporteurs ont la même période.

Pour tout PV-graphe G, nous allons définir deux graphes (classiques) $H_1(G)$ et $H_2(G)$ comme suit. Les deux graphes ont l'ensemble des transporteurs comme ensemble de sommets. Il existe une arête entre deux transporteurs c et c' dans $H_1(G)$ si et seulement si il

existe un site qui apparaît dans les routes de c et de c'. Il existe une arête entre deux transporteurs c et c' dans $H_2(G)$ si et seulement si il existe un site s et un temps $t \geq 0$ tel que c et c' se rencontrent sur s au temps t. Un PV-graphe G est dit *connexe* si et seulement si $H_1(G)$ est connexe. Un PV-graphe est dit *hautement connexe* si et seulement si $H_2(G)$ est connexe. Dans cet article, nous considérons les PV-graphes qui sont au moins connexes. (Les PV-graphes non connexes ne peuvent pas être explorés).

Une entité externe, appelée *agent*, opère sur ces PV-graphes. Cette entité peut voir un transporteur et connaître son identifiant. Elle peut monter avec un transporteur ou changer de transporteur. Contrairement au modèle utilisé dans [6], l'agent a la possibilité de quitter un transporteur et de rester sur le site courant. Il pourra plus tard remonter avec le même transporteur ou un autre. Nous ne faisons aucune restriction sur la taille de la mémoire de l'agent ou sur ses capacités de calcul. Nous considérons deux modèles concernant les identités des sites. Dans le cas des PV-graphes *anonymes*, les sites n'ont pas d'identifiant, ou l'agent ne peut pas les voir. Dans le cas des PV-graphes *étiquetés*, les sites ont des identifiants distincts et l'agent peut les voir et les mémoriser.

Nous disons qu'un agent explore un PV-graphe si et seulement si, en commençant au temps 0 sur le site de départ du premier transporteur (sans perte de généralité), l'agent peut visiter l'ensemble des sites du PV-graphe en un temps fini et se mettre dans un état terminal. Cet état terminal exprime le fait que l'agent détecte que l'exploration est terminée.

De même que dans le cas où l'agent ne peut pas attendre sur un site, un agent sans information sur les PV-graphes à explorer ne peut pas explorer tous les PV-graphes (même si on se limite aux PV-graphes étiquetés homogènes et hautement connexes). Nous supposerons donc par la suite que l'agent dispose d'une borne inférieure B sur la période maximale p.

4. Bornes inférieures

Dans cette section, nous ne faisons aucune hypothèse sur les PV-graphes (sauf l'hypothèse de connexité). Nous allons montrer que le fait de descendre permet à l'agent d'explorer l'ensemble des PV-graphes (et pas seulement les PV-graphes hautement connexes). La possibilité de descendre permet aussi une diminution de la borne inférieure sur le nombre de mouvements d'au moins un facteur $\Theta(p)$. La complexité en mouvement, dans le cas général, passe de $\Theta(kp^2)$ à $\Theta(\min\{kp, np, n^2\})$. En outre, la complexité en temps passe de $\Theta(kp^2)$ à $\Theta(kn)$.

4.1. Borne inférieure sur le nombre de mouvements

Flocchini, Mans et Santoro [6] ont prouvé une borne inférieure sur le nombre de mouvements de $\Omega(kp)$ pour explorer les PV-graphes avec k transporteurs et une période maximum p (même si le PV-graphe est homogène, étiqueté et hautement connexe). Cette borne inférieure ne s'applique pas directement dans notre cadre, car l'agent, ayant la possibilité d'attendre, pourrait être en mesure d'explorer en faisant moins de déplacements. Nous démontrons que c'est effectivement le cas : la complexité en mouvements de notre algorithme est bornée par $O(\min\{kp, np, n^2\})$. Nous allons prouver ici que cette complexité est optimale.

Théorème 1 *Pour tout n, k et p, il existe un PV-graphe $G_{n,k,p}$ étiqueté, homogène et hautement connexe de n sites, k transporteurs et de période p tel que tout agent (i.e. tout algorithme) a besoin d'au moins $\Omega(min\{kp, np, n^2\})$ mouvements pour l'explorer.*

Éléments de preuve. Le principe général de la preuve est de construire une famille de PV-graphes dans lesquels la construction des routes obligent l'agent à visiter de nombreuses fois certains sites pour explorer les autres sites. Trois constructions différentes sont utilisées (suivant les valeurs relatives des différents paramètres) qui correspondent aux trois arguments du minimum. □

4.2. Bornes inférieures sur le temps

Dans le cas où l'agent ne pouvait pas descendre, le temps (en unités de temps) qu'il mettait pour explorer le PV-graphe était égal au nombre de mouvements. Le fait de donner à l'agent la possibilité de descendre sur un site et d'attendre un transporteur permet maintenant de faire la différence entre le nombre de mouvements et le temps mis par l'agent pour explorer un PV-graphe. Nous montrons ici une borne inférieure sur le temps plus élevée que celle sur le nombre de mouvements.

Théorème 2 *Pour tout n, k, p, il existe une famille $\mathcal{G}_{n,p,k}$ de PV-graphes hautement connexes et une famille $\mathcal{G}'_{n,p,k}$ de PV-graphes homogènes telles que chaque PV-graphe G d'une des deux familles est fait avec n sites, k transporteurs et de période maximale p. Pour tout agent (i.e. tout algorithme), il existe un PV-graphe dans chacune des deux familles où il utilisera au moins $\Omega(np)$ unités de temps pour l'explorer.*

Éléments de preuve. Nous ne donnons l'intuition de la preuve que pour la famille de PV-graphes hautement connexes. La preuve pour la famille de PV-graphes homogènes est similaire.

Les PV-graphes utilisés pour prouver ce théorème sont construits de la manière suivante. Les transporteurs sont numérotés de 1 à k. Les transporteurs de numéro impair, respectivement pair, sont de période p, respectivement $p-1$. Un transporteur i n'a de sites en commun qu'avec les transporteurs $i-1$ et $i+1$. (L'alternance des périodes assurent donc la haute connexité.) Plus précisément, le transporteur i passe une et une seule fois par un unique site visité par le transporteur $i-1$.

La preuve est ensuite basée sur le fait que l'agent ne sait pas précisément dans quel PV-graphe il se trouve. En particulier, l'agent ne sait pas sur quel site du transporteur $i-1$ ni à quels temps passera le transporteur suivant (numéro i). Il est possible de prouver, grossièrement, que l'agent doit attendre au moins $p-1$ unités de temps sur chaque site pour être sûr de trouver le transporteur suivant, d'où la borne obtenue. □

5. Notre algorithme

Dans la première partie du papier, nous avons fourni des bornes inférieures sur la complexité en mouvements et en temps. Nous allons maintenant prouver que tous ces résultats sont optimaux en décrivant et en prouvant un algorithme d'exploration de PV-graphes dont les performances correspondent aux bornes inférieures sur la complexité en mouvements et en temps, à condition que l'agent connaisse une borne supérieure, notée

B, sur la période maximale p du PV-graphe. De ce fait, nous montrons que la possibilité d'attendre permet de diminuer à la fois la complexité en mouvements et en temps, la première par un facteur multiplicatif d'au moins $\Theta(p)$.

5.1. Principe et performance de notre algorithme

Comme précisé précédemment, notre algorithme utilise une borne supérieure B sur la période maximale p du PV-graphe. L'idée principale de l'algorithme consiste à descendre sur chaque site et, pendant $O(B)$ unités de temps, à noter tous les temps de passages de transporteurs sur ce site. Correctement gérées, ces informations permettent de cartographier le PV-graphe (liste des routes et des horaires de passage de tous les transporteurs).

Plusieurs précautions doivent être prises pour ne manquer aucun site et pour optimiser le nombre de mouvements. Par exemple, après chaque étude d'un site (pendant $O(B)$ unités de temps), l'algorithme calcule la plus petite période possible de chaque transporteur connu étant donné les informations déjà récoltés. Ceci permet de connaître tous les temps de passage futurs sur les sites étudiés. Afin d'éviter des déplacements inutiles, l'algorithme utilise la notion de transporteur courant. L'agent étudie tous les sites du transporteur courant avant de passer au suivant. Les identifiants des transporteurs rencontrés sont stockés sous la forme d'un arbre, un transporteur c' étant un fils d'un transporteur c si c' est découvert lorsque c est le transporteur courant. Les transporteurs sont traités en profondeur d'abord pour des raisons de performance.

Une analyse détaillée de l'algorithme complet permet de prouver le résultat suivant.

Théorème 3 *Un agent exécutant notre algorithme utilise au plus $O(min\{kp, np, n^2\})$ mouvements et $O(nB)$ unités de temps pour explorer et cartographier n'importe quel PV-graphe, où n est le nombre de sites, k le nombre de transporteurs et B une borne supérieure donnée à l'algorithme sur la plus grande période p.*

Notons que les algorithmes proposés dans [6] ne sont optimaux en temps (et donc en mouvements) que si la borne supérieure sur p qu'ils utilisent est linéaire en p. Dans notre cas, d'après les Théorèmes 1 et 2, notre algorithme est de la même façon optimal en temps seulement si la borne supérieure B est linéaire en p, mais il est par contre toujours optimal en nombre de mouvements, aussi mauvaise que soit la borne B.

5.2. Description de notre algorithme EXPLORE-AVEC-ARRET

Outre la borne supérieure B sur la plus grande période p du PV-graphe exploré, notre algorithme utilise les variables décrites ci-après.

L'algorithme utilise une numérotation propre pour identifier les sites, afin de pouvoir fonctionner même si le PV-graphe est anonyme.

– $numeroCourant$: numéro du site actuellement étudié.

Si le PV-graphe est étiqueté, l'agent maintient un tableau de correspondance entre les numéros donnés par l'agent et les identifiants réels des sites.

– $numeroVersID$: $numeroVersID[j]$ est l'identifiant du site numéro j.

L'agent maintient un arbre enraciné dont les différents sommets correspondant aux différents transporteurs rencontrés. Un ordre sur les fils est aussi maintenue.

– $arbre$: l'arbre des transporteurs.

Algorithme EXPLORE-AVEC-ARRET – Notre algorithme d'exploration de PV-graphes

1: $bFini \leftarrow$ **false**
2: $numeroCourant \leftarrow 1$
3: $transporteurCourant \leftarrow 1$
4: $arbre \leftarrow$ arbre réduit à un unique sommet (la racine) correspondant au transporteur 1
5: **while** ($bFini =$ **false**) **do**
6: etudierSiteCourant()
7: miseAuPropre()
8: **if** aucune valeur 0 dans aucun tableau $route[.]$ **then**
9: $bFini \leftarrow$ **true**
10: **else**
11: trouverEtAtteidreProchainSite()
12: **end if**
13: **end while**
14: Terminer en fournissant la carte du PV-graphe (variables $route[.]$, $position[.]$ et éventuellement $numeroVersID$ si le PV-graphe est étiqueté)

Procédure etudierSiteCourant() – collecte toute les informations possibles sur le site courant

1: Rester sur le site courant pendant $2B$ unités de temps
2: **for** chaque unité de temps **do**
3: **for** chaque transporteur i présent sur le site courant à l'instant courant **do**
4: **if** i absent dans $arbre$ **then**
5: Ajouter i comme dernier fils de $transporteurCourant$ dans $arbre$
6: $route[i] \leftarrow$ tableau de longueur $3B$ rempli de 0
7: $position[i] \leftarrow 0$
8: **end if**
9: $route[i][position[i]] \leftarrow numeroCourant$
10: **end for**
11: **end for**

Procédure miseAuPropre() – utilise les connaissances aquises pour mettre à jour les variables

1: **for** chaque transporteur i présent dans $arbre$ **do**
2: $periode[i] \leftarrow$ période minimale de $route[i]$ entre $position[i] - 2B + 1$ et $position[i]$
3: Rendre tout le tableau $route[i]$ périodique de période $periode[i]$ à partir des valeurs de $route[i]$ entre $position[i] - 2B + 1$ et $position[i]$
4: $position[i] \leftarrow position[i] \bmod periode[i]$
5: **end for**

Procédure trouverEtAtteindreProchainSite() – déterminer le prochain site à étudier et s'y rendre

1: $transporteurCourant \leftarrow$ identifiant i du premier transporteur suivant l'ordre DFS dans $arbre$ tel que la valeur 0 apparaît dans $route[i]$
2: Calculer à partir des tables $route[.]$ le trajet minimum, en nombre de mouvements puis en unité de temps, entre le site courant et un site marqué 0 dans $route[transporteurCourant]$
3: Suivre ce trajet
4: $numeroCourant \leftarrow numeroCourant + 1$
5: $numeroVersID[numeroCourant] \leftarrow$ identifiant du site courant

Après chaque mouvement de l'agent le long du trajet :

1: **for** chaque transporteur i présent dans $arbre$ **do**
2: $position[i] \leftarrow position[i] \bmod periode[i]$
3: **end for**

– $transporteurCourant$: identifiant du transporteur actuellement étudié.

Pour chaque transporteur d'identifiant i présent dans $arbre$:

– $route[i]$ est un tableau de longueur $3B$ (indexé de 0 à $3B-1$) ; il sert à mémoriser la suite des sites visités par le transporteur i.

– $position[i]$ est un entier compris entre 0 et $3B-1$ (inclus) ; il indique la position courante du transporteur i vis-à-vis de $route[i]$.

– $periode[i]$ est un entier compris entre 1 et B (inclus) ; il indique la période minimale du transporteur i étant donné les connaissances actuelles.

Notons que les compteurs $position[i]$ sont incrémenté de 1 à chaque unité de temps. Ceci n'est pas indiqué dans le pseudocode afin de ne pas trop l'allourdir.

6. Conclusion

Dans cet article, nous avons étudié le problème d'exploration de PV-graphes en donnant à l'agent la possibilité d'attendre sur un site. Nous avons prouvé que cette possibilité permet à l'agent de réduire la complexité en mouvements et en temps de l'exploration.

Il serait intéressant d'approfondir ces résultats par des expérimentations, mieux à même de capturer les performances en moyenne. Ce travail pourrait aussi être étendu au cas probabiliste, pour les transporteurs (site de départ ou route aléatoire) comme pour les agents (utilisation de marche aléatoire).

7. Bibliographie

[1] C. AVIN, M. KOUCKY, Z. LOTKER, « How to explore a fast-changing world (cover time of a simple random walk on evolving graphs », *35th International Colloquium on Automata, Languages and Programming (ICALP)*, vol. LNCS 5125, p. 121–132, 2008.

[2] A. CASTEIGTS, P. FLOCCHINI, W. QUATTROCIOCCHI, N. SANTORO, « Time-varying graphs and dynamic networks », *CoRR*, n° 1012.0009, 2010.

[3] C. COOPER, A. M. FRIEZE, « Crawling on simple models of web graphs », *Internet Mathematics*, vol. 1(1), 2003.

[4] A. DESSMARK, A. PELC, « Optimal graph exploration without good maps », *Theor. Comput. Sci.*, vol. 326(1-3), p. 343–362, 2004.

[5] P. FLOCCHINI, M. KELLETT, P. C. MASON, N. SANTORO, « Mapping an unfriendly subway system », *5th Intl Conference on Fun with Algorithms (FUN)*, vol. LNCS 6099, p. 190–201, 2010.

[6] P. FLOCCHINI, B. MANS, N. SANTORO, « Exploration of periodically varying graphs », *20th Intl Symposium on Algorithms and Computation (ISAAC)*, vol. LNCS 5878, p. 534–543, 2009.

[8] P. PANAITE, A. PELC, « Exploring Unknown Undirected Graphs », *J. Algorithms*, vol. 33(2), p. 281–295, 1999.

[9] O. REINGOLD, « Undirected st-connectivity in log-space », *37th ACM Symposium on Theory of Computing (STOC)*, p. 376–385, 2005.

[10] C. E. SHANNON, « Presentation of a maze-solving machine », *8th Conf. of the Josiah Macy Jr. Found. (Cybernetics)*, p. 173-180, 1951.