Utiliser des jumelles pour explorer rapidement les graphes triangulés

Jérémie Chalopin, Emmanuel Godard, Antoine Naudin

Utiliser des jumelles pour explorer rapidement les graphes triangulés

Jérémie Chalopin, Emmanuel Godard et Antoine Naudin

LIF, Université Aix-Marseille et CNRS, FRANCE

Nous nous intéressons à l'exploration de graphes par un agent mobile. Il est connu que sans information "globale" sur le graphe (taille, diamètre,...), un agent peut explorer et s'arrêter uniquement si le graphe est un arbre. Afin d'explorer plus de graphes avec arrêt sans information "globale" sur le graphe exploré, l'agent est équipé de jumelles lui permettant de voir le graphe induit par les sommets voisins du sommet courant. Avec des jumelles, il est possible d'explorer une très large famille maximale de graphes avec arrêt ([CGN15]). Cependant, tout algorithme d'exploration universel pour cette famille a une complexité non calculable. Nous considérons donc dans cette étude une sous famille de ces graphes, les graphes de Weetman, qui sont une généralisation de la famille des graphes triangulés. Nous présentons un algorithme d'exploration pour cette famille ne nécessitant aucune information globale sur le graphe exploré. Cet algorithme est efficace dans le sens où même si les graphes de Weetman peuvent être très denses (un nombre quadratique d'arêtes), le nombre de mouvements effectués par l'agent restera linéaire en le nombre de sommets. De plus, notre algorithme permet également de construire une carte du graphe.

Mots-clefs : Agent mobile, Exploration de graphes, Graphes anonymes, Revêtement, Graphes Triangulés

1 Introduction

Le Modèle. Cet article continue l'étude du modèle d'agent avec jumelles introduit dans [CGN15]. Ce modèle est une extension du modèle standard d'agent mobile (voir [Das13]). Un agent mobile est une entité mobile naviguant dans un graphe $G = (V,E)$ de sommet en sommet via les arêtes. Chaque sommet $v \in V(G)$ dispose d'une étiquette $label(v)$. De plus, les graphes sont anonymes, c.à.d., l'étiquette $label(v)$ ne permet pas d'identifier le sommet v de manière unique. Afin de permettre à l'agent de se déplacer dans le graphe, les sommets sont dotés d'un étiquetage des ports localement injectif $\delta = \{\delta_v\}_{v\in V(G)}$ donnant pour un sommet $v \in V(G)$, une identité unique à chacun de ses voisins. On note $\delta_u(v)$, le numéro de port en u désignant l'arête uv. Nous noterons v_0 le sommet base de l'agent, i.e., sa position initiale dans le graphe G. Un agent visitant un sommet du graphe peut utiliser des jumelles lui permettant de "voir" le graphe induit par le sommet visité et ses voisins ainsi que les numéros de ports présents dans ce graphe. De manière équivalente, nous supposons que chaque sommet du graphe possède une étiquette, appelée *étiquetage de jumelles*, encodant cette boule de rayon 1 autour du sommet.

Exploration de graphes anonymes. Un algorithme $\mathcal{A}$ est un *algorithme d'Exploration* si pour toute exécution sur un graphe G avec étiquetage de jumelles et pour tout sommet de départ $v_0 \in V(G)$, l'agent visite tous les sommets de G et s'arrête, ou l'agent ne s'arrête jamais (exploration perpétuelle). Un graphe G est *explorable* s'il existe un algorithme d'Exploration qui, pour tout sommet de départ, explore G et s'arrête. Un algorithme $\mathcal{A}$ explore une famille de graphes $\mathcal{F}$ si $\mathcal{A}$ explore tout graphe $G \in \mathcal{F}$ et s'arrête et pour tout $G \notin \mathcal{F}$, si $\mathcal{A}$ s'arrête alors G est exploré. Il est connu que dans un modèle "sans jumelles" (et sans pouvoir marquer les noeuds), un agent sans information "globale" sur le graphe (taille, diamètre,...) peut explorer et s'arrêter uniquement si le graphe est un arbre. Dans [CGN15], nous avons caractérisé la famille $\mathcal{F}^*$ maximale de graphes anonymes explorables avec jumelles. C'est à dire, nous montrons que tout graphe explorable est dans $\mathcal{F}^*$ et nous donnons un algorithme universel d'exploration pour tout graphe de $\mathcal{F}^*$. Notons que $\mathcal{F}^*$ est bien plus vaste que les arbres ; elle contient, entre autres, tous les graphes

ayant un complexe de clique simplement connexe (graphes triangulés, triangulations planaires ou encore les triangulations du plan projectif). Cependant, nous avons montré que pour tout algorithme explorant $\mathcal{F}^*$, on ne peut pas borner la longueur des exécutions par une fonction calculable.

Nos Résultats. Dans cette étude, nous nous intéressons à l'exploration efficace, c'est à dire, où l'agent doit explorer le graphe en un nombre de mouvements linéaire en fonction du nombre de sommets. Au vu des précédents résultats, nous montrons qu'une sous-famille des graphes explorables avec jumelles, appelée graphes de Weetman, peut être explorée efficacement. Les graphes de Weetman sont une extension des graphes triangulés introduite par Weetman [Wee94]. Cette famille contient un nombre infini de graphes ayant une taille arbitrairement grande. De plus, les graphes de Weetman peuvent avoir un nombre quadratique d'arêtes.

2 Définitions et Notations

Graphes. Nous considérons des graphes sans boucle, ni arête multiple. Soit $p = (v_0, \dots, v_k)$ un chemin dans un graphe G munis d'un étiquetage des ports δ. L'étiquetage induit par $p \subset G$, noté $\lambda(p) = \{\delta_{v_i}(v_{i+1}) \mid 1 \leq i < k\}$, est la séquence de numéros de ports suivi par p dans G. Soit $N_G(v) = \{w \mid vw \in E(G)\}$. Soit $B_G(v,k)$, le graphe induit par l'ensemble de sommets à distance au plus k de v. Nous supposons que l'étiquette $label(v)$ donnée au sommet $v \in V(G)$ correspond à l'étiquetage de jumelles, i.e., $label(v)$ encode un graphe isomorphe à $B_G(v,1)$ (préservant l'étiquetage des ports). Un revêtement de graphe $\varphi : G \to H$ est une fonction de $V(G)$ dans $V(H)$ tel que $\forall v \in V(G)$, $N(v)$ et $N(\varphi(v))$ sont en bijection. Ces définitions sont étendues aux graphes étiquetés. Par la suite, nous noterons G pour un graphe $(G, label)$ étiquetés avec son étiquetage de jumelles $label$ et nous considérerons toujours des revêtements sur les graphes étiquetés $(G, label)$.

Cycles contractibles. Un cycle c dans un graphe G est dit contractible si, $\forall v \in V(G)$, il existe une séquence de cycles $\{c = c_0, ..., c_m = \{v\}\}$ telle que $\forall\ h \leq m$, $c_h = \{u^0, ..., u^{i-1}, u^i, u^{i+1}, ..., u^k\}$ et $c_{h+1} = \{u^0, ..., u^{i-1}, u^{i+1}, ..., u^k\}$ ou $c_h = \{u^0, ..., u^{i-1}, u^{i+1}, ..., u^k\}$ et $c_{h+1} = \{u^0, ..., u^{i-1}, u^i, u^{i+1}, ..., u^k\}$, et soit $u^i = u^{i+1}$, soit $u^{i-1} = u^{i+1}$ ou soit $u^{i-1}u^{i+1} \in E(G)$. Un graphe est simplement connexe [†] si tous ses cycles sont contractibles.

Clusters. Soit $S^i(v_0) = B_G(v_0, i) \setminus B_G(v_0, i-1)$ où $v_0 \in V(G)$ est le sommet base de l'agent. Les *clusters* sont les composantes connexes des $S^i(v_0) \subset G$, $i \in \mathbb{N}$. Le lemme 2.1 se base sur le fait qu'un cycle dans le graphe d'adjacence des clusters de G implique un cycle non contractible dans G.

Lemme 2.1. *Le graphe d'adjacence des clusters d'un graphe simplement connexe est un arbre.*

3 Exploration efficace des graphes de Weetman

Graphes de Weetman [Wee94]. Un graphe G est *Weetman* si pour tous $v_0 \in V(G)$, $\forall i \in \mathbb{N}$,

TRI Pour tout arête $uv \in E(S^i(v_0))$, il existe un sommet $w \in V(S^{i-1}(v_0))$ t.q. uvw est un triangle.

INT Pour tout sommet $w \in V(S^i(v_0))$ et pour tous $u, v \in V(S^{i-1}(v_0))$ tels que $uw, vw \in E(G)$, il existe un chemin $\{u = s_1, ..., s_k = v\} \subset S^{i-1}(v_0)$ tel que $s_h w$ est une arête de G, $\forall 1 \leq h \leq k$.

Par exemple, les graphes triangulés et les graphes de Johnson sont des graphes de Weetman. Comme les conditions TRI et INT nous permettent toujours de contracter un cycle, on a :

Lemme 3.1. *Tout graphe de Weetman est simplement connexe et ses clusters forment un arbre.*

Algorithme. Le principe de notre algorithme est d'effectuer un parcours en profondeur de l'arbre des clusters pour construire une carte, nommée $\mathcal{M}$. Cette carte lui permet de calculer des plus courts chemins pour explorer un cluster et pour se rendre d'un cluster à un autre. L'algorithme s'exécute par phase. Durant chaque phase, l'agent explore un cluster puis met à jour sa carte $\mathcal{M}$ avec les nouveaux sommets et arêtes découverts. L'exécution se termine quand tous les sommets de la carte ont été explorés. Grâce aux jumelles, l'agent est en mesure de cartographier le voisinage des sommets explorés. L'agent est donc en

†. Un graphe est simplement connexe si tous les cycles de son complexe de clique sont contractibles (voir [LS77])

mesure de cartographier tous les clusters "fils" adjacents lors de l'exploration d'un cluster, rendant ainsi le parcours en profondeur des clusters possible. Soit $n_0 \in V(\mathcal{M})$ le sommet correspondant au sommet base $v_0 \in V(G)$. Durant une exécution, l'agent positionné sur un sommet u de G connaît le sommet n dans sa carte correspondant à u, i.e., si $p : v_0 \to u$ est le chemin parcouru pour rejoindre le sommet u depuis le début de l'exécution, alors $n = \texttt{DEST}_{\mathcal{M}}(n_0, \lambda(p))$ est le sommet de $\mathcal{M}$ accessible depuis n_0 en suivant la suite de port $\lambda(p)$.

Lors de l'exploration d'un cluster (lignes 2 à 7), pour tout sommet u de G exploré correspondant à un sommet n de $\mathcal{M}$, l'agent fera appel à une primitive `getBino()` pour accéder à l'étiquetage de jumelles du sommet u (Ligne 4) . De plus, l'étiquetage de jumelles de u sera stocké dans une structure $\mathcal{B}$ indexée par n, i.e., notée $\mathcal{B}[n]$. Soit $\psi(n) = u \in V(\mathcal{B}[n])$, le sommet de $\mathcal{B}[n]$ correspondant à n. Un sommet v de $\mathcal{B}[n]$ est *connu* si une arête nm telle que $\delta_n(m) = \delta_u(v)$ est déjà présente dans la carte $\mathcal{M}$ ($nm \in E(\mathcal{M})$). Sinon le sommet v est *découvert* (non présent dans la carte). Trois structures sont utilisées lors de l'exploration d'un cluster pour mettre à jour la carte $\mathcal{M}$. La structure `PRE-VERT` encode tout sommet découvert lors de l'exploration du cluster (les pré-sommets à ce stade de l'exécution). Un pré-sommet (n, p, q) sera ajoutée à `PRE-VERT` ligne 5 si il y a une arête $uv \in E(\mathcal{B}[n])$ telle que $u = \psi(n)$ et v est découvert (i.e. $\texttt{DEST}_{\mathcal{M}}(n, p) \notin V(\mathcal{M})$). Notons que `PRE-VERT` encode en même temps toutes les arêtes verticales, i.e., les arêtes joignant un sommet connu et un sommet découvert. On définit une relation $\equiv$ entre les pré-sommets que l'on stocke dans une structure $\texttt{R}_{\equiv}$. La relation $\equiv$ identifie les couples de pré-sommets correspondant à un même sommet de G du point de vue de l'agent, i.e., si il y a un triangle $uvw \in \mathcal{B}[n]$ tel que $\psi(n) = u$, v est connu (et $nm \in E(\mathcal{M})$) et w est découvert alors le couple de pré-sommets $((n, p, q), (m, p', q'))$ est ajouté à $\texttt{R}_{\equiv}$ ligne 6 (i.e. $(n, p, q) \equiv (m, p', q')$) car $\texttt{DEST}_{\mathcal{B}[n]}(n, p) = \texttt{DEST}_{\mathcal{B}[n]}(m, p')$. Notons que la clôture transitive et réflexive de $\equiv$, nommée $\equiv^*$, est une relation d'équivalence pour les pré-sommets de `PRE-VERT`. Nous désignerons par $[n, p]$ la classe d'équivalence du pré-sommet (n, p). La structure `HOR` stocke les pré-sommets "adjacents", i.e., les arêtes "horizontales" liant deux sommets découverts. Si dans le triangle $uvw \in \mathcal{B}[n]$, v et w sont découverts et $\psi(n) = u$, alors une entrée $(n, p, p', (r, s))$ sera ajoutée à `HOR` ligne 7 telle que $(n, p), (n, p') \in$ `PRE-VERT` et (r, s) est l'étiquetage de l'arête liant les sommets $v = \texttt{DEST}_{\mathcal{B}[n]}(u, p)$ et $w = \texttt{DEST}_{\mathcal{B}[n]}(u, q)$ de $\mathcal{B}[n]$.

Une fois le cluster C exploré (ligne 8 à 12), la carte $\mathcal{M}$ est mise à jour. Un sommet $\overline{[\texttt{n}, \texttt{p}]}$ est ajouté à $V(\mathcal{M})$ pour chaque classe d'équivalence $[\texttt{n}, \texttt{p}]$ de pré-sommets obtenu par le quotient de l'ensemble `PRE-VERT` par la relation $\equiv^*$ (ligne 8). Une arête $\texttt{n}\overline{[\texttt{n}, \texttt{p}]}$ est insérée pour chaque pré-sommet $(\texttt{n}, \texttt{p}) \in$ `PRE-VERT` (ligne 9) et une arête $\overline{[\texttt{n}, \texttt{p}][\texttt{n}, \texttt{p}']}$ est insérée pour chaque entrée $\big(n, p, p', (r, s)\big)\big) \in$ `HOR` (ligne 10) représentant les "pré-sommets adjacents". Une fois ces mises à jour faites, si une erreur est détectée ligne 11, i.e., $\mathcal{B}[n] \not\simeq B_{\mathcal{M}}(n, 1)$, $n \in V(C)$, l'agent continue indéfiniment son exploration (ligne 12). Sinon, l'agent débute une nouvelle phase en continuant son parcours en profondeur des clusters du graphe.

Algorithm $\mathcal{A}$: Exploration des graphes de Weetman

1 **Tant que** *il reste des sommets non explorés dans* $\mathcal{M}$ **faire**
2 *Visiter le prochain cluster C de* $\mathcal{M}$ *en suivant un parcours en profondeur*
3 **Pour tout** *sommet* $n \in V(C)$ *exploré* **faire**
4 Appeler `getBino()` et stocker l'étiquetage de jumelles obtenu dans $\mathcal{B}[\texttt{n}]$
5 Extraire les pré-sommets de $\mathcal{B}[\texttt{n}]$ et les ajouter dans `PRE-VERT`
6 Stocker dans $\texttt{R}_{\equiv}$ les couples de pré-sommets de $\mathcal{B}[\texttt{n}]$ satisfaisant $\equiv$, i.e., correspondant à un même sommet de $\mathcal{B}[\texttt{n}]$
7 Stocker dans `HOR` les couples de pré-sommets de $\mathcal{B}[\texttt{n}]$ adjacents
8 Ajouter à $V(\mathcal{M})$ un sommet $\overline{[\texttt{n}, \texttt{p}]}$ pour chaque classe d'équivalence de pré-sommet $[\texttt{n}, \texttt{p}]$
9 Ajouter à $E(\mathcal{M})$ une arête $\texttt{n}\overline{[\texttt{n}, \texttt{p}]}$ étiquetée $(\texttt{p}, \texttt{q})$ pour tout $(\texttt{n}, \texttt{p}, \texttt{q}) \in$ `PRE-VERT`
10 Ajouter à $E(\mathcal{M})$ une arête $\overline{[\texttt{n}, \texttt{p}][\texttt{n}, \texttt{q}]}$ étiquetée $(\texttt{p}', \texttt{q}')$ pour tout $\big(\texttt{n}, \texttt{p}, \texttt{q}, (\texttt{p}', \texttt{q}')\big) \in$ `HOR`
11 **Si** *il existe* $n \in V(C)$ *t.q.* $\mathcal{B}[n] \not\simeq B_{\mathcal{M}}(n, 1)$ **alors**
12 *Faire des aller retour sur une arête indéfiniment*

Correction. Nous prouvons la correction de notre algorithme en montrant que lorsque l'agent a visité tous les sommets de $\mathcal{M}$, i.e., l'agent s'est arrêté, alors il existe un revêtement $\varphi : \mathcal{M} \to G$. Le lemme 3.2 se montre par induction sur les phases d'une exécution. Nous noterons $\mathcal{M}^i, \texttt{PRE-VERT}^i$ pour indiquer le contenu des structures à la phase i. Soit $\partial\mathcal{M}^i = \mathcal{M}^i \setminus \{C^h\}_{h<i}$ avec C^h, le cluster exploré phase $h \leq i$.

Lemme 3.2. *Pour toute phase i d'une exécution de $\mathcal{A}$ sur un graphe G, si aucune erreur n'a été détectée alors il existe un homomorphisme φ tel que pour tout sommet $n \in V(\mathcal{M}^i)$, $\varphi_{|B_{M^i}(n,1)}$ est injective et pour tout sommet $n \in V(\mathcal{M}^i \setminus \partial\mathcal{M}^i)$, $\varphi_{|B_{M^i}(n,1)}$ est surjective sur $B_G(\varphi(n),1)$.*

Notons que si une exécution se termine lors d'une phase i, alors $\partial\mathcal{M}^i = \emptyset$. Par le lemme 3.2 et comme φ^i préserve les étiquetages de jumelles de $\mathcal{M}^i$ et G, $\mathcal{M}^i$ est un revêtement de G. Notons que comme tout revêtement est surjectif, $|\mathcal{M}| \geq |G|$. Par conséquent, G est exploré.

Proposition 3.3. *[LS77] Soit G' et G deux graphes avec étiquetage de jumelles et un revêtement $\varphi : G' \to G$. Si G est simplement connexe alors G est isomorphe à G'.*

Soit l'agent détecte une erreur et il ne termine jamais son exécution ; soit l'agent ne détecte aucune erreur et, si l'agent s'arrête, le graphe est exploré. $\mathcal{A}$ est donc un algorithme d'Exploration. Pour tout graphe de Weetman G, la condition INT assure que chaque sommet de G correspond à une unique classe d'équivalence. La condition TRI assure que si $\varphi(n)\varphi(m)$ est une arête de G alors, nm est inséré dans $E(\mathcal{M})$. Par le lemme 3.1 et la proposition 3.3, si l'exploration s'arrête sur un graphe G, alors $\mathcal{M}$ et G sont isomorphes. $\mathcal{A}$ explore donc tout graphe de Weetman et termine.

Théorème 1. *$\mathcal{A}$ est un algorithme d'Exploration pour la famille des graphes de Weetman.*

Complexité. Soit $\#C$ le nombre de clusters d'un graphe de Weetman G. Notons que $\mathcal{M} \simeq G$ quand l'exécution s'arrête. Trivialement, comme les clusters de G forment une partition des sommets de G, $\#C \leq |V(G)|$ et $\sum_{C \in G} |C| = |V(G)|$. Comme chaque cluster C est exploré en suivant un arbre couvrant de C, on explore C (et on cartographie tous les clusters fils de C) en $O(|C|)$ mouvements. Pour chaque cluster C, on ordonne les fils de C de telle sorte que le parcours en profondeur des clusters s'effectue en $O(n)$ mouvements (on utilise un arbre couvrant de C). Ainsi, le nombre de mouvements de l'agent lors d'un exécution est linéaire en la taille du graphe.

4 Conclusion

Nous montrons qu'en élargissant "localement" la connaissance de l'agent, il est possible d'explorer efficacement, avec un nombre de mouvements linéaire, de grandes familles de graphes (comme les graphes triangulés) sans aucune connaissance globale "a priori" même si dans le cas général, la complexité est non calculable [CGN15]. Grâce à la spécification de l'Exploration, l'algorithme $\mathcal{A}$ peut être "composé" de façon sûre avec d'autres algorithmes car il ne terminera jamais son exécution sans avoir rempli sa fonction (explorer le graphe). Une piste à explorer serait de caractériser des familles de graphes "métriques" pouvant être explorées en temps "raisonnable" (linéaire ou polynomial).

Références

[CGN15] J. Chalopin, E. Godard, and A. Naudin. Anonymous graph exploration with binoculars. In *Distributed Computing*, volume 9363 of *LNCS*, pages 107–122. Springer Berlin Heidelberg, 2015.

[CGN16] J. Chalopin, E. Godard, and A. Naudin. Using binoculars for fast exploration and map construction in chordal graphs and extensions, 2016. `http://arxiv.org/abs/1604.05915`.

[Das13] Shantanu Das. Mobile agents in distributed computing : Network exploration. *Bulletin of EATCS*, 1(109), Aug 2013.

[LS77] R.C. Lyndon and P.E. Schupp. *Combinatorial group theory*. Ergebnisse der Mathematik und ihrer Grenzgebiete. Springer-Verlag, 1977.

[Wee94] Graham M. Weetman. A construction of locally homogeneous graphs. *Journal of the London Mathematical Society*, 50(1) :68–86, 1994.