On n'explore que deux fois

Quentin Bramas, Toshimitsu Masuzawa, Sébastien Tixeuil

On n'explore que deux fois

Quentin Bramas[1] et Toshimitsu Masuzawa[2] et Sébastien Tixeuil[3]

[1] *Université de Strasbourg, ICube, CNRS, France*
[2] *Université d'Osaka, Japon*
[3] *Sorbonne Université, CNRS, LIP6, IUF, France*

Cet article considère l'exploration d'un graphe pondéré par deux agents mobiles, où le coût énergétique de la traversée d'une arête est égal au poids de l'arête. Les agents situés à la même position (potentiellement sur une arête) peuvent librement transférer de l'énergie, mais l'un d'entre eux peut tomber en panne de manière imprévisible et cesser d'opérer. Deux configurations sont envisagées : asynchrone, sans limite sur la vitesse relative des agents, et synchrone, avec des horloges synchronisées et des vitesses égales. L'étude se concentre sur les réseaux en anneau et examine les conditions d'une exploration complète des arêtes en fonction des niveaux d'énergie initiaux des agents.

Mots-clefs : Exploration, Tolérance aux pannes, Partage d'énergie

1 Introduction

La robotique en essaim a conduit à de nombreuses études sur les capacités de groupes de robots mobiles autonomes, ou agents, aux capacités individuelles limitées. Ces agents travaillent ensemble pour accomplir des tâches complexes telles que la formation de motifs, le regroupement et l'assemblage d'objets, la recherche et l'exploration. La collaboration offre plusieurs avantages, notamment une réalisation plus rapide des tâches, la possibilité de tolérance aux pannes, et des coûts de construction et une efficacité énergétique inférieurs par rapport à des agents plus grands et plus complexes. Cet article examine l'exploration collective d'un graphe pondéré connu par des agents mobiles initialement placés sur des nœuds arbitraires. L'objectif est que chaque arête soit explorée par les agents, les poids des arêtes représentant leurs longueurs. Chaque agent dispose d'une batterie avec un niveau d'énergie initial, qui peut varier entre les agents. Se déplacer sur une distance de x épuise la batterie d'un agent de x. Un mécanisme récemment exploré pour la collaboration entre agents est la capacité à partager de l'énergie, permettant à un agent de transférer de l'énergie à un autre lorsqu'ils se rencontrent. Les capacités de partage d'énergie permettent l'exploration de graphes dans des situations où cela serait autrement impossible. D'un autre coté, un algorithme d'exploration avec partage d'énergie doit attribuer des trajectoires aux agents pour explorer collectivement l'ensemble du graphe et planifier des transferts d'énergie réalisables, et est donc plus complexe à concevoir.

Cet article introduit la possibilité qu'un agent tombe en panne, c'est-à-dire cesse de fonctionner indéfiniment et de manière imprévisible. Un algorithme d'exploration doit maintenant considérer non seulement la capacité du partage d'énergie, mais aussi prendre en compte la possibilité qu'un agent tombe en panne à n'importe quel moment de l'exécution. †

Travaux connexes. Le transfert d'énergie par des agents mobiles a été introduit par Czyzowicz et al. [CDMR16]. Les auteurs examinent des problèmes de communication où les informations détenues par certains nœuds doivent être communiquées à d'autres nœuds ou agents. Avec l'échange d'énergie, les problèmes considérés par Czyzowicz et al. [CDMR16] ont des solutions en temps linéaire sur les arbres. Pour les graphes non orientés et orientés généraux, ces problèmes sont NP-complets. Les travaux les plus proches de notre article sont ceux de Czyzowicz et al.[CDK+21] et de Sun et al.[SKIM23]. D'une part, Czyzowicz et al.[CDK+21] étudient l'exploration collective d'un graphe pondéré connu à n nœuds par k agents mobiles avec une énergie limitée et une capacité de transfert d'énergie. L'objectif est que chaque arête soit

†. Les résultats présentés ici ont été acceptés à la conférence SSS2023 et la version longue est disponible en ligne : `https://bramas.fr/static/crash_tolerant_sss2023.pdf`

parcourue par au moins un agent. Pour un arbre à n nœuds avec ℓ feuilles, ils fournissent un algorithme en temps $O(n + \ell k^2)$ pour trouver une stratégie d'exploration s'il en existe une. Pour les graphes généraux, décider si l'exploration est possible par des agents partageant l'énergie est NP-difficile, même pour les graphes 3-réguliers. Cependant, il est toujours possible de trouver une stratégie d'exploration si l'énergie totale des agents est au moins deux fois le poids total des arêtes ; cela est asymptotiquement optimal. Sun et al. [SKIM23] examinent l'exploration par des agents partageant l'énergie sur un graphe arbitraire. Ils présentent une condition énergétique nécessaire et suffisante pour l'exploration et un algorithme pour trouver une stratégie d'exploration s'il en existe une.

Notre Contribution. Nous considérons le problème d'explorer chaque arête pondérée d'un graphe donné par une équipe de deux agents mobiles. À la différence des approches précédentes, nous introduisons la possibilité pour un agent de tomber en panne de manière imprévisible et de cesser de fonctionner pour toujours (c'est-à-dire, de subir un crash).

Dans ce contexte, nous considérons deux paramètres : *asynchrone*, où aucune limite sur la vitesse relative des agents n'est connue (donc si des agents se donnent rendez-vous à un noeud et temps donnés, il est impossible de distinguer un agent qui pose un lapin, *i.e.*, est tombé en panne, d'un agent très lent), et *synchrone*, où les deux agents ont des horloges synchronisées et se déplacent à exactement la même vitesse. Nous considérons des réseaux en forme d'anneau et étudions les conditions nécessaires et suffisantes pour la solvabilité complète de l'exploration des arêtes, en fonction des énergies initiales allouées à chaque agent.

2 Modèle

Notre modèle étend celui de Czyzowicz et al. [CDK+21] pour prendre en compte les pannes des agents. Nous avons un graphe pondéré $G = (V, E)$ où V est un ensemble de n nœuds, E est un ensemble de m arêtes, et chaque arête $e_i \in E$ est associée à un entier positif $w_i \in \mathbb{N}^+$, représentant son poids. Nous disposons de k agents mobiles $r_0, r_1, \ldots, r_{k-1}$ placés respectivement sur certains des nœuds $s_0, s_1, \ldots s_{k-1}$ du graphe. Plusieurs agents peuvent être situés au même endroit. Chaque agent r_i possède initialement une quantité spécifique d'énergie égale à en_i.

Un agent a la capacité de se déplacer le long des arêtes du graphe G dans n'importe quelle direction. Il peut mettre en pause son mouvement si nécessaire et peut changer de direction à un nœud ou le long d'une arête. L'énergie consommée par un agent en mouvement est égale à la distance x parcourue. Un agent ne peut se déplacer que si son énergie est strictement supérieure à zéro. La distance entre deux points est la plus petite quantité d'énergie nécessaire pour aller d'un point à l'autre.

Nous supposons que les agents peuvent partager de l'énergie. Lorsque deux agents, r_i et r_j, se rencontrent (éventuellement sur une arête) dans le graphe, r_i peut prendre une partie de l'énergie de r_j. Si leurs niveaux d'énergie au moment de la rencontre sont en'_i et en'_j, alors r_i peut prendre une quantité d'énergie $0 < en \leq en'_j$ de r_j. Après le transfert, leurs niveaux d'énergie seront respectivement $en'_i + en$ et $en'_j - en$.

Chaque agent suit une trajectoire préétablie jusqu'à rencontrer un autre agent ou terminer l'exploration. Lors d'une rencontre, l'agent détermine s'il acquiert de l'énergie et calcule sa trajectoire ultérieure. La définition d'une trajectoire dépend du modèle de synchronie :

- **Dans le modèle synchrone**, une trajectoire est une séquence de paires $((u_0, t_0), (u_1, t_1), \ldots)$, où u_i est un nœud et t_i indique le moment où l'agent doit atteindre u_i. Pour chaque $i \geq 0$, $t_i < t_{i+1}$, et u_{i+1} est soit égal à u_i (l'agent attend à u_i entre t_i et t_{i+1}), soit adjacent à u_i (l'agent quitte u_i à l'instant t_i et arrive à u_{i+1} à l'instant t_{i+1}). Pour simplifier, nous supposons dans nos algorithmes que la vitesse de déplacement est toujours égale à un.
- **Dans le modèle asynchrone**, une trajectoire est simplement une séquence de nœuds $(u_0, u_1, u_2, \ldots)$, u_{i+1} étant adjacent à u_i pour chaque $i \geq 0$, et les moments auxquels il atteint les nœuds sont déterminés par un adversaire.

En d'autres termes, dans le modèle synchrone, l'agent contrôle sa vitesse et son temps d'attente aux nœuds. Le calcul de la trajectoire et la décision d'échanger de l'énergie reposent sur un algorithme localisé exécuté par l'agent. Le temps peut être divisé en rondes discrètes qui correspondent aux instants où au moins deux agents se rencontrent. Dans une exécution donnée, la configuration à la ronde t est notée C_t.

Algorithme localisé. Un *algorithme localisé* f_i exécuté par un agent r_i à l'instant t prend en entrée le *passé* de r_i et de ses agents colocalisés, et renvoie *(i)* une trajectoire et *(ii)* la quantité d'énergie prise à chaque agent colocalisé r_j. Le passé de l'agent r_i à la ronde t correspond au chemin déjà parcouru par r_i union le passé de tous les agents précédemment rencontrés (au moment de leur rencontre).

Un ensemble d'algorithmes localisés est *valide* pour une configuration initiale donnée c, si pour toute exécution à partir de c, les agents devant se déplacer ont suffisamment d'énergie pour le faire, et lorsqu'un agent r_i prend de l'énergie à un agent r_j à la ronde t, alors r_j ne prend pas aussi de l'énergie à r_i.

À n'importe quel moment de l'exécution, un agent r_i peut tomber en panne et cesser de fonctionner pour toujours. Cependant, si $en'_i > 0$, alors d'autres agents rencontrant r_i peuvent prendre de l'énergie à r_i pour n'importe quelle raison. Maintenant, un ensemble d'algorithmes localisés est t-tolérant aux pannes s'il est valide même dans des exécutions où au plus t agents tombent en panne. Nous nous intéressons à la résolution du problème général suivant de l'exploration collaborative tolérante aux pannes :

Exploration collaborative tolérante aux pannes. Étant donné un graphe pondéré $G = (V, E)$ et k agents mobiles $r_0, r_1, \ldots, r_{k-1}$ avec leurs énergies initiales respectives $en_0, \ldots, en_{k-1}$ et positions $s_0, \ldots, s_{k-1}$ dans le graphe, trouver un ensemble valide d'algorithmes localisés qui explore (ou couvre) toutes les arêtes du graphe malgré les pannes inattendues d'au plus $t < k$ agents. Il est important de noter que les algorithmes localisés dépendent de la configuration initiale. Cependant, une fois l'exécution débutée, ils s'exécutent de manière distribuée et localisée.

3 Nos algorithmes pour deux agents sur un anneau

Nos algorithmes sont présentés sous forme de règles. Chaque règle est composée d'une condition et d'une suite d'actions à exécuter lorsque la condition est satisfaite. Une action peut être un déplacement ; une alternance d'actions, selon une condition booléenne ; ou une instruction d'attente (dans le cadre synchrone). Il existe deux déplacements possibles vers un point cible p, l'un pour se déplacer dans le sens horaire ($\circlearrowright(p)$) et l'autre pour se déplacer dans le sens antihoraire ($(p)\circlearrowleft$). La condition booléenne $p_1 \prec p_2$ est vraie si et seulement si le point p_1 est plus proche que le point p_2 de l'agent qui évalue la condition.

Soit $G = (V, E)$ un graphe en forme d'anneau. ℓ représente le poids total de G, et d, respectivement m, le poids du plus court, respectivement du plus long, chemin de r_0 à r_1. Supposons que les agents r_0 et r_1 sont initialement situés aux nœuds s_0 et s_1 respectivement. Sans perte de généralité, nous pouvons supposer que le chemin le plus court de r_0 à r_1 est dans le sens antihoraire (l'orientation est connue par les agents).

Anneaux Asynchrones. Nous montrons qu'un algorithme existe si et seulement si la condition suivante est satisfaite : $c_1 : (en_0 \geq d) \wedge (en_1 \geq d) \wedge (en_0 + en_1 \geq 2\ell)$. Dans ce cas, les algorithmes localisés sont les suivants : $r_0 :\ (r_1)\circlearrowleft ;\ (s_0)\circlearrowleft \qquad r_1 :\ \circlearrowright(r_0)\ ;\ \circlearrowright(s_1)$

Dans les algorithmes ci-dessus, lorsque deux agents se rencontrent, si un agent à moins de $l - x$ (où x est la distance déjà parcourue), alors il prend de l'énergie à l'autre agent pour atteindre $l - x$.

Lemme 1 *Si la condition c_1 est vérifiée, alors si au plus un agent tombe en panne, deux agents asynchrones exécutant les algorithmes localisés prescrits ci-dessus explorent l'ensemble de l'anneau. Sinon, si c_1 n'est pas vérifiée, le problème n'est pas soluble.*

Intuition de la preuve. Les agents r_0 et r_1 explorent d'abord le chemin le plus court entre eux pour se rencontrer. Puisque $en_0 \geq d$ et $en_1 \geq d$, ils ont tous deux suffisamment d'énergie pour atteindre l'autre, même si l'autre tombe en panne. Puisqu'au plus l'un d'eux tombe en panne, la rencontre se produit. Lorsqu'ils se rencontrent, une énergie d a été dépensée au total. Plus précisément, l'agent r_0 a dépensé x, et l'agent r_1 a dépensé $d - x$, où x est la distance de s_0 au point de rencontre. Maintenant, puisque $en_0 + en_1 \geq 2\ell$, r_0, resp. r_1, peut s'assurer d'avoir $\ell - x$ d'énergie, resp. $\ell - (d - x)$ d'énergie, ce qui est suffisant pour que chaque agent termine son exploration. Comme chaque agent explore totalement l'anneau, ce dernier est exploré même dans le cas d'une panne. Si la condition c_1 n'est pas satisfaite, l'examen exhaustif de tous les algorithmes possibles montre que l'exploration est impossible.

Anneaux Synchrones. Lorsque les agents sont synchrones, nous montrons qu'une au moins des conditions suivantes doit être satisfaite :

c_1 : $(en_0 \geq d) \wedge (en_1 \geq d) \wedge (en_0 + en_1 \geq \ell + m)$

c_2 : $(en_0 \geq m) \wedge (en_1 \geq m) \wedge (en_0 + en_1 \geq \max\left(\ell + d, \ell + \frac{m}{2}\right))$

Pour chaque condition, les actions correspondantes sont les suivantes :

Action a_1

r_0 :	$(r_1)\,\}\,;\ (s_0)\,\}$
r_1 :	**attendre** *d unités de temps* r_0 **et faire** $\mid\ (s_0)\,\}$ **si absent, faire** $\mid\ \{\,(r_0)\,;$ **si** $s_0 \prec s_1$ **alors** $\{\,(s_0)\,;\ \{\,(s_1)$ $\mid$ **sinon** $(s_1)\,\}\,;\ (s_0)\,\}$ **fin**

Action a_2 :

r_0 :	$\{\,(r_1)\ ;\ \{\,(s_0)$
r_1 :	**attendre** *m unités de temps* r_0 **et faire** $\mid\ \{\,(s_0)$ **si absent, faire** $\mid\ (r_0)\,\}\,;$ **si** $s_0 \prec s_1$ **alors** $(s_0)\,\}\,;\ (s_1)$ $\mid\ \}$ **sinon** $\{\,(s_1)\,;\ \{\,(s_0)$ **fin**

Dans les algorithmes ci-dessus, lorsque deux agents se rencontrent, si l'autre agent n'est pas encore en panne, les agents partagent l'énergie de sorte que chacun ait la même quantité. Sinon, si un agent r est en panne (c'est détectable dans le cas synchrone), alors l'autre agent prend toute l'énergie restante de r.

Lemme 2 *Si la condition c_1, resp. c_2, est vérifiée, alors si au plus un agent tombe en panne, deux agents synchrones exécutant les algorithmes localisés prescrits par l'action a_1, resp. a_2, explorent l'ensemble de l'anneau. Si les deux conditions c_1 et c_2 ne sont pas satisfaites, alors le problème n'est pas soluble.*

Intuition de la preuve. Supposons que c_2 est vérifiée (le cas où c_1 est vérifiée est similaire mais les agents commencent par explorer le chemin le plus court au lieu du plus long). Tout d'abord, l'agent r_0 explore le chemin le plus long entre les agents tandis que r_1 attend pendant m instants. Si l'agent r_0 ne tombe pas en panne, alors r_0 a utilisé m d'énergie et r_1 sait que le chemin le plus long est entièrement exploré. Puisque $en_0 + en_1 \geq \ell + d = m + 2d$, après avoir partagé leur énergie, chaque agent a au moins d d'énergie, ce qui est suffisant pour explorer le chemin le plus court entre s_0 et s_1, même si un agent tombe en panne.

Si r_0 tombe en panne, alors r_1 le remarque finalement et se déplace sur le chemin le plus long jusqu'à ce qu'il atteigne r_0. Lorsqu'ils se rencontrent, l'énergie totale dépensée est de m. r_1 récupère toute l'énergie restante de r_0, de sorte que r_1 a au moins $m/2 + d$ d'énergie (car $en_0 + en_1 \geq \ell + m/2 = m + m/2 + d$). Maintenant r_1 se déplace vers le nœud le plus proche s_0 ou s_1, ce qui dépense au plus $m/2$ d'énergie, puis traverse le chemin le plus court entre s_0 et s_1, ce qui dépense d d'énergie. Ainsi, l'exploration est atteinte.

Si aucune des deux conditions n'est satisfaite, l'examen exhaustif de tous les algorithmes possibles montre que l'exploration est impossible.

4 Conclusion

Nous avons caractérisé la solvabilité de l'exploration avec deux agents mobiles à énergie partagée sujets aux pannes dans le cas des topologies en anneau, à la fois dans les configurations synchrones et asynchrones. Des questions ouvertes évidentes incluent la résolution du problème avec plus de deux agents, et la considération d'autres topologies, telles que les arbres et les grilles.

De plus, notre modèle de transfert d'énergie est très simple (toute l'énergie peut être transférée instantanément entre deux agents). Il serait intéressant d'étudier des modèles de batterie non linéaires (où la capacité diminue plus rapidement si un courant instantané plus important est tiré, et où la capacité augmente moins si une charge plus rapide est effectuée) dans ce contexte.

Références

[CDK+21] Jurek Czyzowicz, Stefan Dobrev, Ryan Killick, Evangelos Kranakis, Danny Krizanc, Lata Narayanan, Jaroslav Opatrny, Denis Pankratov, and Sunil M. Shende. Graph exploration by energy-sharing mobile agents. In Tomasz Jurdzinski and Stefan Schmid, editors, *SIROCCO 2021, Wrocław, Poland, June 28 - July 1, 2021, Proceedings*, volume 12810 of *Lecture Notes in Computer Science*, pages 185–203. Springer, 2021.

[CDMR16] Jurek Czyzowicz, Krzysztof Diks, Jean Moussi, and Wojciech Rytter. Communication problems for mobile agents exchanging energy. In Jukka Suomela, editor, *SIROCCO 2016, Helsinki, Finland, July 19-21, 2016, Revised Selected Papers*, volume 9988 of *Lecture Notes in Computer Science*, pages 275–288, 2016.

[SKIM23] X. Sun, N. Kitamura, T. Izumi, and T. Masuzawa. Circulating exploration of an arbitrary graph by energy-sharing agents. In *Proc. of the 2023 IEICE General Conference*, pages 1–2, 2023.