Exploration et couverture par stigmergie d'un environnement inconnu avec une flotte de robots autonomes réactifs

Nicolas Gauville, François Charpillet

Exploration et couverture par stigmergie d'un environnement inconnu avec une flotte de robots autonomes réactifs

Nicolas Gauville[a, b] nicolas.gauville@loria.fr
François Charpillet[a] francois.charpillet@inria.fr

[a] Universite de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy

[b] Safran Electronics & Defense

Résumé

L'exploration autonome d'un environnement inconnu peut être envisagée de différentes manières. On peut notamment citer les approches par frontières, où des robots sont affectés à des zones inexplorées de la carte. Ces dernières méthodes sont efficaces mais nécessitent de partager une carte, globaliser les décisions d'affectation. Les approches Brick and Mortar, quant à elles, utilisent un marquage au sol avec une prise de décision locale, mais donnent des performances beaucoup moins intéressantes. L'algorithme présenté ici est un compromis entre ces deux approches, permettant une prise de décision locale et, de façon surprenante, des performances proche des approches par frontières globales. Nous proposons également une étude comparative de la performance des trois différentes approches : Brick & Mortar, frontières globales et frontières locales. Notre algorithme local est également complet pour le problème d'exploration et peut être facilement distribué sur des robots avec une perte de performance mineure.

Mots-clés : *Multi-robot, stigmergie, exploration*

Abstract

Different approaches exist for multi-robot autonomous exploration. These include frontier approaches, where robots are assigned to unexplored areas of the map, which provide good performance but require sharing the map and centralizing decision-making. The Brick and Mortar approaches, on the other hand, use a ground marking with local decision-making, but give much lower performance. The algorithm presented here is a trade-off between these two approaches, allowing local decision-making and, surprisingly, performances are closed to centralized frontier approaches. We also propose a comparative study of the performance of the three different approaches : Brick & Mortar, Global Frontiers and Local Frontiers. Our local algorithm is also complete for the exploration problem and can be easily distributed on robots with a minor loss of performance.

Keywords: *Multi-robot, stigmergy, exploration*

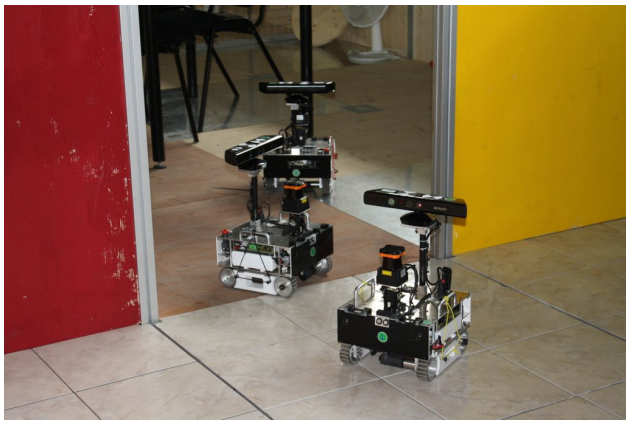

FIGURE 1 – Robot Minirex (utilisé pour le projet Cart-O-Matic) explorant une zone test.

1 Introduction

L'exploration autonome d'un environnement inconnu est un défi important en robotique mobile. L'exploration peut être une fin en soi : des robots de nettoyage (aspirateur, tondeuse à gazon, etc.) explorent systématiquement chaque zone à couvrir. Dans les applications plus sophistiquées, il s'agit principalement d'acquérir des informations : cartographier l'environnement [2], rechercher un intrus, localiser une source sonore ou olfactive [14, 17], ou encore rechercher d'éventuelles personnes à secourir dans un bâtiment en feu, . . .

L'objectif de cet article est de proposer une approche par frontières *locales* pour aborder le problème de l'exploration multi-robot. L'objec-

tif principal de ce travail est de proposer une approche évolutive et efficace, même dans un environnement où la communication est restreinte, pour résoudre le problème de l'exploration avec un grand nombre de robots. Nous avons également essayé de fournir un algorithme aussi simple que possible, exécutable avec des ressources limitées.

Notre approche, bien qu'inspirée des approches par frontières globales, aborde différemment le problème de l'exploration : dans le cas de l'approche globale, c'est avant tout un problème de planification, où l'algorithme doit décider de l'affectation des robots aux frontières connues. Dans notre version locale, les robots ne connaissent pas le nombre ou la position des autres robots et ne s'en soucient pas. Ils réagissent uniquement aux informations locales. La collaboration entre robots se fait par stigmergie, dans notre cas simulée à l'aide d'une carte qui est partagée entre robots à intervalles réguliers, mais d'autres méthodes pourraient être envisagées pour limiter au mieux la quantité et la fréquence des informations partagées [11].

2 État de l'art

L'exploration multi-robot vise à explorer un environnement inconnu le plus rapidement possible pour une équipe homogène de robots mobiles. C'est une question proche du problème de couverture [10], mais avec une carte inconnue.

Différentes approches ont été développées en fonction des contraintes. On y retrouve notamment les approches de type *Brick & Mortar* [8, 9, 1] qui peuvent être classées dans les « algorithmes fourmis », où les robots partagent une carte commune pour *marquer* l'environnement avec des traces similaires aux phéromones des insectes sociaux (à la différence que ces traces ne changent pas avec le temps). Les décisions sont alors prises localement, mais le partage d'une carte globale entre robots est nécessaire pour partager les traces virtuelles laissées sur la carte.

Des approches « purement fourmis » existent aussi, utilisant des phéromones dispersées par l'environnement pour répartir les robots [6]. Ces approches sont souvent plus difficiles à mettre en place que les approches *Brick & Mortar*, car il faut également faire évoluer les phéromones déposées (évaporation, dispersion, . . .). Garnier et al. ont par exemple proposé un système de projection de points avec des robots équipés de capteurs de lumière pour simuler ces phéromones partagées [11, 15], mais cela nécessite d'avoir un appareil pouvant projeter des points sur l'ensemble de la surface à explorer, ce qui n'est pas envisageable dans le cas de l'exploration d'un bâtiment inconnu. Nous préférerons donc ici les approches *Brick & Mortar* où il suffit de partager les traces laissées.

L'exploration peut également être considérée comme un problème d'ordonnancement ; l'approche par frontières [18, 7, 5] consiste à attribuer à chaque robot une frontière, c'est-à-dire une limite entre les parties explorées et inexplorées de la carte. Dans ce cas, un consensus global doit également être assuré.

Si les approches locales sont efficaces, elles restent beaucoup plus lentes que les approches par frontières en termes de temps d'exploration : les robots doivent effectuer de nombreux marquages pour ne pas se bloquer mutuellement ou « oublier » une zone à explorer. D'autre part, les approches par frontières sont plus efficaces et basées sur des algorithmes plus simples, mais nécessitent une coordination des décisions entre robots qui peut poser différents problèmes dans des situations réelles où une communication constante ne peut être assurée [12]. De plus, l'algorithme d'assignation des frontières utilisé nécessite souvent un temps de calcul non linéaire en fonction du nombre de robots et de la taille de la grille, ce qui peut la rendre inapplicable dans une situation réelle si la carte est trop grande ou s'il y a trop de robots. Cependant, il existe différentes méthodes pour optimiser l'affectation et les données échangées, par exemple en partageant uniquement les frontières et non la carte entière [13]. Enfin, notre approche locale peut facilement être distribuée, et fonctionne également avec des communications limitées (voir section 5).

L'algorithme proposé ici est un compromis entre l'approche *Brick & Mortar* et l'assignation de frontières : c'est une approche de frontières locales, où les robots partagent une carte, mais avec une prise de décision locale pour chaque robot. Notre approche est donc moins sensible à l'asynchronisme et réduit significativement les calculs effectués par chaque robot, car le nombre de cellules prises en compte à un moment donné est beaucoup moins important.

Ce travail fait suite au projet *Cart-O-Matic* [3]. Cart-O-Matic était l'un des cinq projets fondés par l'Agence Nationale de la Recherche (ANR) pour sa participation au concours de robotique organisé par la *Délégation générale pour l'ar-*

mement (DGA). Ce concours intitulé « Defi *CAROTTE* » avait pour objectif de définir un système robotique capable d'explorer un environnement intérieur inconnu et identifier des objets localisés dans cet environnement. Notre projet Cart-O-Matic a choisi de déployer un système multi-robot. Cart-O-Matic a remporté le concours final en 2012.

Toutes les approches considérées ici utilisent des grilles d'occupation pour représenter le monde, et considèrent que chaque robot est équipé d'un Lidar principalement utilisé pour la localisation et la cartographie simultanées (SLAM). Cependant, notre approche, comme celle par frontières globales, peut être adaptée à d'autres représentations du monde [16].

2.1 Frontières globales

Dans le cas de l'approche par frontières globales, chaque robot possède une grille d'occupation pouvant prendre trois états : *inexploré*, *exploré* et *obstacle*, initialisée à *inexploré*. À chaque itération, chaque robot marque l'ensemble des cellules dans son rayon de perception comme *exploré* ou *obstacle*. Les frontières entre les zones explorées et inexplorées de la carte peuvent alors être calculées en détectant toutes les cellules contiguës qui ont à la fois des voisines immédiates explorées et inexplorées.

Une fois toutes les frontières de la carte détectées, l'algorithme d'assignation (*MinPos* [4]) est utilisé toutes les 4 itérations pour assigner une frontière à chaque robot, et ceux-ci peuvent alors se diriger vers la frontière qui leur a été assigné. La stratégie d'assignation *MinPos* est décrite en 4.1.

2.2 Algorithme *BMILRV*

L'algorithme *BMILRV*[1] utilise de nombreux marquages sur la grille d'occupation : un état parmi un ensemble d'états { *inexploré, exploré, fermé, obstacle, rendez-vous* }, une valeur indiquant si un robot contrôle une cellule donnée, la direction prise lors du dernier passage sur une cellule, et une variable booléenne indiquant si un robot donné est déjà passé sur une cellule. Les robots peuvent également être dans différents modes parmis { *détection de boucle, contrôle de boucle, fermeture de boucle, nettoyage, pause, arrêt* }.

De manière informelle, l'algorithme *BMILRV* consiste à maintenir une liste de cellules ouvertes permettant d'accéder à toutes les zones inexplorées de la carte, tout en fermant progressivement toutes les cellules qui peuvent l'être (qui ne bloqueront pas le passage vers une zone inexplorée), évitant ainsi de repasser dans les zones déjà visitées.

Lorsque des boucles sont présentes dans le chemin constitué par les cellules explorées, les robots peuvent les fermer en plusieurs étapes permettant d'éviter de bloquer un robot. Le robot détectant la boucle (entrant dans une cellule explorée sur laquelle il est déjà passé en venant de la même direction), va alors passer en mode « *contrôle de boucle* » et parcourir cette boucle pour la marquer comme contrôlée avec son identifiant, vérifiant ainsi qu'il est le seul à chercher à la fermer à ce moment. S'il parvient à marquer la totalité de la boucle, il effectue alors un second passage en mode « *fermeture* » ou il va fermer les cellules de la boucle jusqu'à la première intersection, et enfin parcourir la boucle une troisième fois en sens inverse en mode « *nettoyage* » pour enlever les traces indiquant qu'il contrôle la boucle. S'il ne parvient pas à contrôler la boucle, alors il passe directement en mode « *nettoyage* » si le robot qui la contrôle a une priorité supérieure, ou en mode « *pause* » si il est prioritaire, en attendant que l'autre robot ait nettoyé ses traces sur la boucle.

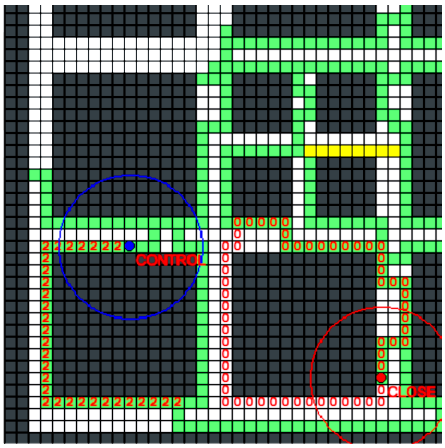

FIGURE 2 – Représentation de la grille lors d'une exploration par l'algorithme *BMILRV*. Les cellules fermées sont en gris, ouvertes en vert, point de rendez vous en jaune et inexplorées en blanc. Les chiffres dans les cellules correspondent aux numéros des robots contrôlant ces cellules, l'état des robots est indiqué en rouge.

Les différentes étapes de l'algorithme BMILRV ne sont pas triviales ; elles nécessitent de nom-

breuses vérifications pour fermer correctement les cellules, et un décalage ou asynchronisme dans la procédure de marquage des cellules pourra bloquer les robots. Pour des raisons de place, nous n'entrerons pas dans les détails du fonctionnement de l'algorithme ici.

3 Suivi de frontières locales

3.1 Exploration exhaustive

De façon informelle, l'algorithme local consiste à marquer tout chemin parcouru comme visité, à se déplacer le plus loin possible vers des zones inconnues, et revenir sur ses pas lorsqu'il n'y a plus de zone inconnue visible, jusqu'à ce que l'on en retrouve une.

Lorsque le robot est revenu à son point de départ sans avoir vu aucune zone inexplorée, toute la carte a été explorée.

Dans le cas de plusieurs robots, selon les conditions initiales et la topologie de la carte, un ou plusieurs robots peuvent être revenus à leur point de départ avant la fin de l'exploration, ce qui entraîne une mauvaise distribution de l'exploration et réduit la performance de l'algorithme. Pour réduire cette perte de performance, on peut envisager d'utiliser dans ce cas les traces des autres robots pour reprendre l'exploration une fois qu'ils sont revenus au point de départ, si tous les autres robots ne se sont pas arrêtés. Cette piste d'amélioration est implémentée dans la partie 6.

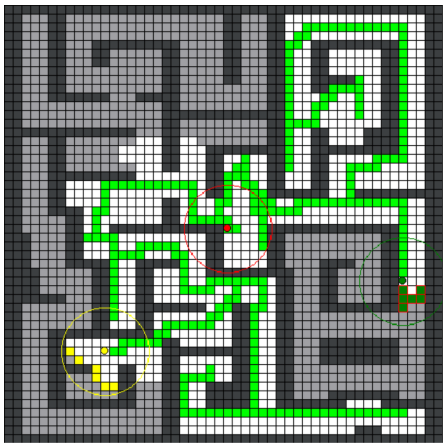

FIGURE 3 – Représentation de la grille lors d'une exploration par l'algorithme par frontières locales (cellules explorées en blanc, traces en vert ; les cellules vert foncé et jaune correspondent aux frontières locales des robots vert foncé et jaune).

3.2 Formalisation (frontières locales)

Nous considérons une flotte homogène de m robots mobiles $R = \{r_1, \dots r_m\}$ équipés de capteurs leur permettant de détecter des obstacles dans un rayon de perception ρ. Les robots utilisent une grille d'occupation $\mathcal{O}$ représentant la carte, où chaque cellule peut être dans trois états : *inexploré*, *obstacle* ou *exploré*, ainsi qu'une grille $\mathcal{P}$ pour assigner un booléen à chaque cellule, indiquant si une cellule est visitée ou non (c'est à dire si un robot a été sur cette cellule), initialisée à *faux*. Ces cartes sont échangées à intervalles réguliers entre les robots (voir section 5). Enfin, chaque robot peut définir une cellule objectif (représentant un point à atteindre sur la carte), et possède une trace $\mathcal{G}r_i$ correspondant à la distance maximale parcourue depuis le point de départ.

Pour chaque déplacement (d'une cellule), chaque robot connaît l'état des cellules dans son rayon de perception et sa position sur la grille (x, y). On notera $\mathcal{V}_{r_i}^t$ l'ensemble des cellules visibles perçues par le robot à un moment donné t qui ne sont pas des obstacles.

Les cartes locales ($\mathcal{G}$ et $\mathcal{P}$) sont partagées entre les robots toutes les d_{sync}itérations avec la procédure décrite dans la partie 5.

Le paramètre d_{max} est utilisé pour forcer les robots à réassigner la cellule objectif à intervalles réguliers, évitant ainsi de choisir une cible maintenue trop longtemps si le rayon de perception est grand.

Chaque robot r_i applique l'algorithme suivant :

1. Marquer toutes les cellules visibles comme explorées :
   $\forall (x, y) \in \mathcal{V}_{r_i}^t, \mathcal{O}_{r_i}(x, y) = \textit{exploré}$
2. Si aucune cellule objectif n'est définie, déterminez les frontières locales $\mathcal{F}$, c'est-à-dire toutes les cellule de $\mathcal{V}_{r_i}^t$ ayant des voisins inexplorés.
   (a) Si $|\mathcal{F}| > 0$, choisir la cellule frontière la plus éloignée de toute cellule visitée (dans la grille $\mathcal{P}_{r_i}$) à une distance maximale d_{max} de la cellule comme nouvel objectif.
   (b) Si $|\mathcal{F}| = 0$ et si on est de retour à la cellule de départ, s'arrêter.
   (c) Sinon, retournez un pas en arrière sur la trace G_{r_i} : choisir la cellule, parmi les 4 cellules adjacentes, ayant la valeur $\mathcal{G}_{r_i}(x, y)$ la plus basse.

3. Sinon, si une cellule objectif est définie
   (a) Si $\mathcal{G}_{r_i}(x,y) = 0$, mettre à jour la trace : $\mathcal{G}_{r_i}(x,y) = 1 + \max(\mathcal{G}_{r_i}(x-1,y), \mathcal{G}_{r_i}(x+1,y), \mathcal{G}_{r_i}(x,y-1), \mathcal{G}_{r_i}(x,y-1), \mathcal{G}_{r_i}(x,y+1))$.
   (b) Mettre à jour la grille : $\mathcal{P}_{r_i}(x,y) = 1$
   (c) Se rapprocher de l'objectif (à l'aide d'un algorithme de *pathfinding* parmi les cellules visibles), puis revenir à l'étape 1.
4. Si plus de d_{sync}itérations ont été effectuées depuis le dernier partage de carte, partagez la carte $(\mathcal{O}_{r_i}, \mathcal{P}_{r_i})$ avec les autres robots (voir section 5).

Dans nos expériences, nous avons utilisé l'algorithme *A** pour rechercher le chemin optimal entre les robots et leurs cellules objectifs.

4 Comparaison des différentes approches

4.1 Algorithmes choisis

Puisque l'algorithme présenté ici est un compromis entre les approches *Brick & Mortar* et par *frontière*, nous avons choisi de le comparer à un algorithme par frontières globales et à un algorithme *Brick & Mortar*.

Les approches par frontières peuvent être mises en œuvre avec différentes stratégies d'affectation et différentes conditions de réaffectation. Nous avons choisi la stratégie d'affectation *MinPos* présentée dans [4], et réassignons les frontières toutes les d_{max} itérations (d_{max} est le paramètre utilisé dans l'algorithme local, et correspond au nombre maximum de déplacement entre chaque changement de cellule objectif, donnant ainsi une condition de réaffectation proche). La stratégie d'affectation *MinPos* consiste à calculer, pour chaque robot i et chaque frontière j, un rang $r_{i,j}$, correspondant au nombre de robots plus proches de la frontière j que le robot i, puis à attribuer à chaque robot la frontière la plus proche pour laquelle son rang est minimal.

Il existe également différentes approches *Brick & Mortar*. Celle que nous avons implémenté ici, *BMILRV* (Brick & Mortar Improved Long Range Vision) [1], permet de considérer un ensemble de cellules dans le rayon de perception du robot, et pas seulement la cellule sur laquelle le robot se trouve, ce qui permet une comparaison plus directe avec notre approche locale, considérant à un instant donné pour un robot le même nombre de cellules voisines.

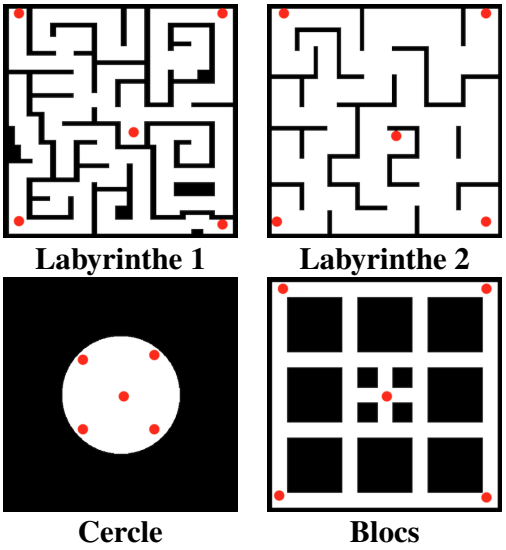

FIGURE 4 – Cartes testées (les points rouges correspondent aux points de départ des robots).

4.2 Conditions expérimentales

Nous allons comparer les trois approches afin d'explorer une carte inconnue. L'approche par frontières globales reproduit l'expérience décrite dans [7] avec réaffectation de l'objectif tous les d_{max} changements de cellule, et la stratégie d'assignation *MinPos*. Comme cette approche ne fournit pas de point de rendez-vous, nous considérerons le nombre total d'itérations jusqu'à ce que la carte soit entièrement visitée. Nous observerons ainsi le pourcentage de cellules explorées par rapport au nombre d'itérations passées. L'approche *BMILRV* est une reproduction de l'algorithme décrit dans [1]. Enfin, l'approche par frontières locales correspond à l'algorithme décrit dans la partie 3.2.

Cette partie présente les résultats expérimentaux réalisés sur simulateur pour 4 cartes différentes présentées figure 4. Les points de départ des robots sont représentés par les points rouges. Dans ces expériences, la carte est partagée à chaque itération ($d_{\text{sync}} = 1$, voir section 5 pour la version distribuée), le rayon de perception ρ est de 5 cellules, et $d_{\text{max}} = 2$.

Le *labyrinthe 2* est celui utilisé dans [1], le *labyrinthe 1* est une variante avec des couloirs plus étroits. La carte *Cercle* permet de tester un environnement avec peu d'obstacles (par rapport au rayon de perception des robots), et la carte *Blocs* correspond à un environnement avec de nombreux obstacles isolés, formant ainsi des boucles

pour les robots.

4.3 Simulateur utilisé

Pour obtenir une première comparaison expérimentale des performances des différents algorithmes, nous avons implémenté un simulateur en Python permettant d'exécuter les algorithmes dans des conditions identiques sur les 4 cartes présentées figure 4.

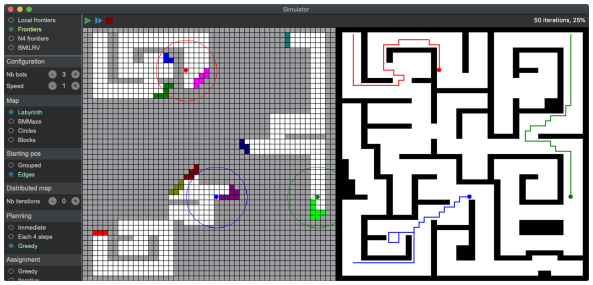

FIGURE 5 – Simulateur utilisé pour les expériences, ici avec l'algorithme par frontières globales. La partie gauche montre la carte des robots (cellules explorées en blanc, inexplorées en gris et frontières en couleurs), et la partie droite le monde « réel » avec les trajectoires des robots. Le menu de gauche permet de sélectionner les différents algorithmes (frontières globales, frontières locales, BMILRV) et les conditions de l'expérience (points de départ robot, cartes, synchronisation).

Notre simulateur (capture d'écran 5) permet une exécution décentralisée des algorithmes des robots. Dans nos expériences, nous supposons que la localisation est parfaite, que les robots ont une taille plus petite que la taille d'une cellule dans la grille d'occupation et que la détection des obstacles est toujours correcte.

4.4 Durée de l'exploration

La figure 6 représente le nombre d'itérations nécessaires pour explorer la carte entière avec $m = 3$ robots pour les différentes approches.

L'approche *BMILRV* est beaucoup plus lente que les approches par frontières locales et globales, surtout lorsqu'il y a des obstacles isolés (forçant l'algorithme *BMILRV* à contrôler les cycles), ce qui est le cas des cartes *Labyrinthe 1* et *Blocs*. Les deux approches frontières donnent des résultats comparables, l'approche globale restant légèrement meilleure.

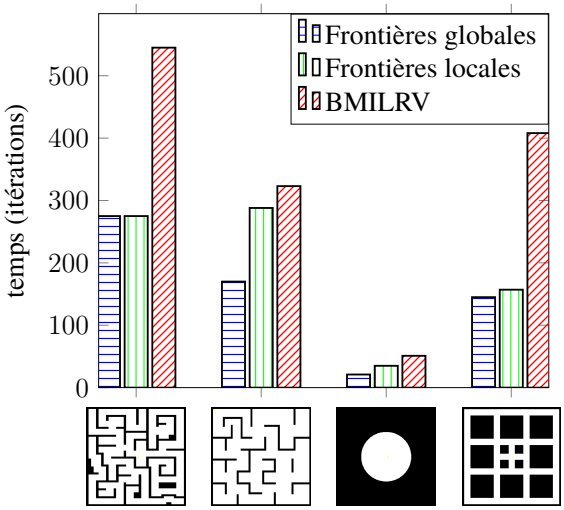

FIGURE 6 – Durée totale de l'exploration pour chaque approche sur les cartes testées avec $m = 3$ robots.

L'approche par frontières locales donne parfois une moins bonne dispersion des robots, et donc de moins bonnes performances, en particulier sur la carte *Labyrinthe 2*. Plusieurs améliorations peuvent être envisagées pour se rapprocher des performances de l'approche globale (voir 7.2).

4.5 Progression de l'exploration

La figure 7 présente l'évolution du pourcentage de cellules vues au cours du temps (en nombre d'itérations) pour les approches par frontières locales et globales, sur les deux premières cartes.

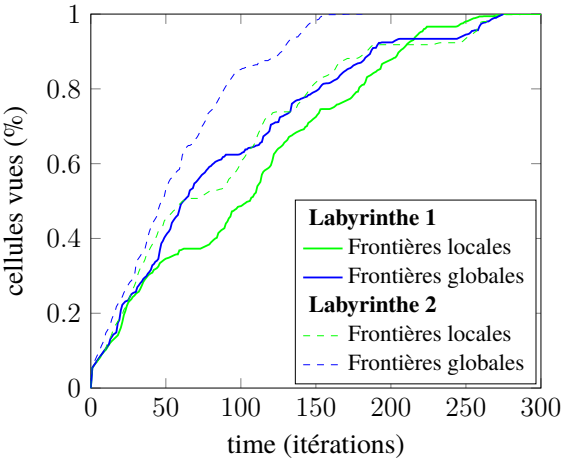

FIGURE 7 – Évolution du pourcentage de cellules vues au cours du temps ($m = 3$ robots) sur les deux premières cartes.

L'algorithme local perd facilement du temps lorsque les robots reviennent en arrière parce qu'ils prennent le même chemin que lors de leur premier passage, contrairement à l'algorithme global qui calcule le meilleur chemin pour atteindre une frontière. Cependant, les deux algorithmes montrent des résultats similaires, les deux courbes se croisent plusieurs fois pour la carte *Labyrinthe 1*.

4.6 Nombre de robots

La figure 8 montre le nombre d'itérations nécessaires pour explorer les 4 cartes pour différents nombres de robots. Sur ces 4 cartes, les approches par frontières locales et globales donnent des résultats comparables.

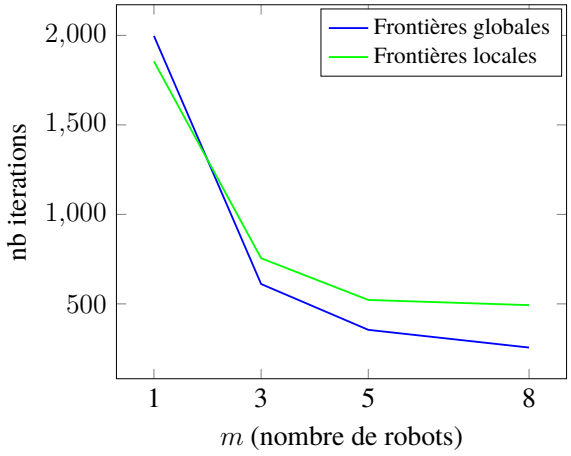

FIGURE 8 – Nombre d'itérations nécessaires pour explorer toutes les cartes pour différents nombres de robots.

Lorsque le nombre de robots augmente, l'algorithme par frontières globales prend l'avantage en offrant une meilleure répartition des robots sur la carte.

5 Version distribuée

Cette partie se concentre sur la version décentralisée de l'algorithme par frontières locales et tente d'évaluer l'impact de la décentralisation sur les performances.

5.1 Procédure de partage de carte

Le partage de la carte est nécessaire pour deux raisons : la grille d'occupation permet aux robots de définir les frontières locales ; si elle n'est pas partagée, les robots ne peuvent pas savoir si autre un robot a déjà exploré une partie donnée de la carte. De plus, la trace est également partagée pour permettre une meilleure répartition des robots, car elle est utilisée dans le choix de la frontière à définir comme objectif.

Toutes les d_{sync}itérations, chaque robot peut envoyer sa grille d'occupation $\mathcal{O}$ et ses cellules visitées $\mathcal{P}$ à tous les autres robots. Lorsqu'un robot r_i reçoit des données d'un autre robot r_j (grille d'occupation $\mathcal{O}_{r_j}$ et cellules visitées $\mathcal{P}_{r_j}$), il met à jour sa grille d'occupation de la façon suivante :

$$\mathcal{P}_{r_i}(x, y) = \mathcal{P}_{r_i}(x, y) \vee \mathcal{P}_{r_j}(x, y)$$

Un *ou* logique est appliqué entre la valeur du robot et la valeur reçue : une cellule est visitée si le robot ou tout autre robot l'a déjà marquée comme visitée.

Il n'est pas nécessaire que les robots soient synchronisés pour partager leurs cartes en même temps, ils peuvent les envoyer ou les recevoir à tout moment.

5.2 Résultats expérimentaux sur l'effet de la décentralisation sur la durée de l'exploration

La figure 9 montre le nombre d'itérations nécessaires pour explorer les 4 cartes avec 3 robots pour différentes valeurs d_{sync}.

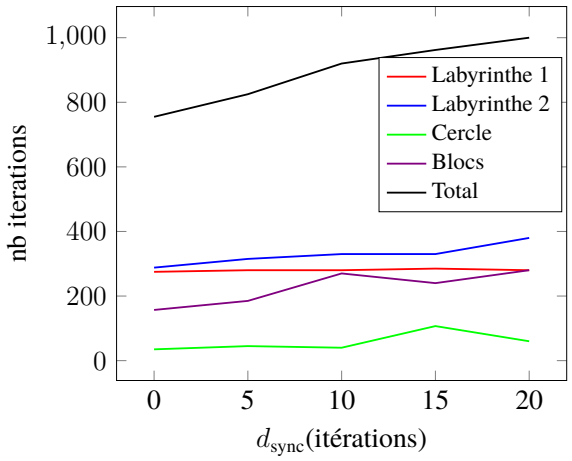

FIGURE 9 – Évolution de la durée de l'exploration des 4 cartes pour différentes valeurs de d_{sync}

L'augmentation de la durée de l'exploration lorsque d_{sync}augmente est due à une moins bonne

dispersion, notamment lorsque plusieurs robots sont proches les uns des autres : ils explorent alors la même zone simultanément. Concrètement, si l'on suppose que deux robots donnés partagent leur carte d'autant plus souvent qu'ils sont proches, cette diminution sera alors limitée la plupart du temps (la présence de boucles peut cependant faire explorer une même zone à deux robots éloignés).

5.3 Perte de communication

Bien qu'augmenter d_{sync}diminue les performances de l'exploration, une perte de communication, ou une communication éparse ne causera pas de problème dans l'exploration (« oubli » d'une zone, robot bloqué,...) et ne peut, dans le pire des cas, que prolonger la durée de l'exploration. Dans des applications concrètes, le partage de cartes entre robots peut être adapté en fonction de leur vitesse et de leur rayon de perception, éventuellement en fonction de la distance entre les robots.

6 Amélioration : « seconde chance »

Il peut arriver lors de l'exploration qu'un ou plusieurs robots retournent à leur point de départ et ne trouvent plus de frontières visibles avant que l'exploration soit terminée, laissant les robots encore actifs terminer seuls. Lorsque cela se produit, l'exploration reste complète, mais les robots revenus à leur point de départ ne participent plus, ce qui peut augmenter la durée totale de l'exploration.

Pour éviter cette perte de performances, nous pouvons envisager un système de « seconde chance », autorisant ces robots à repartir explorer les zones restantes en suivant les traces des autres robots. Nous allons donc modifier l'algorithme décrit dans la partie 3.2.

6.1 Algorithme avec seconde chance

Pour permettre aux robots de repartir, nous allons ajouter une variable *mode*, assigné à « *normal* » par défaut, et qui vaudra « *secondeChance* » lorsqu'un robot repart en suivant les traces des autres robots.

Pour activer le mode seconde chance, nous modifions le point 2. (b) de l'algorithme :

2. (b) Si $|\mathcal{F}| = 0$ et si on est de retour à la cellule de départ :
   i. Si il existe au moins un autre robot qui ne s'est pas arrêté et qui n'est pas en mode seconde chance, passer en mode seconde chance : *mode* $\longleftarrow$ *secondeChance*. Se déplacer vers une cellule voisine déjà visitée par un autre robot, si possible parmi celles que l'on a pas soit-même visité. Si on voit à nouveau une frontière, repasser en mode normal.
   ii. Sinon, s'arrêter.

6.2 Évolution de l'exploration

Nous allons maintenant comparer trois algorithmes : frontières globales (avec *MinPos*), frontières locales sans seconde chance, et frontières locales avec seconde chance. Cette fois, nous choisissons $m = 10$ robots qui partirons tous du centre du labyrinthe.

La figure 10 présente l'évolution de l'exploration pour les trois algorithmes sur le premier labyrinthe. Sur cette carte, où les approches globales et locales donnaient des résultats très proches pour $m = 3$ robots, l'algorithme local prend l'avantage avec la seconde chance, finissant l'exploration avant l'algorithme global.

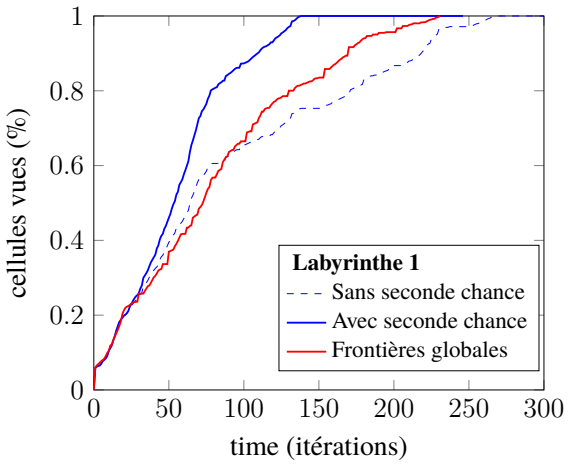

FIGURE 10 – Évolution du pourcentage de cellules vues au cours du temps ($m = 10$ robots) avec et sans seconde chance.

6.3 Durée de l'exploration

Nous allons maintenant nous intéresser à la durée totale de l'exploration des quatre cartes par les trois algorithmes, pour $m \in \{3, 5, 8\}$.

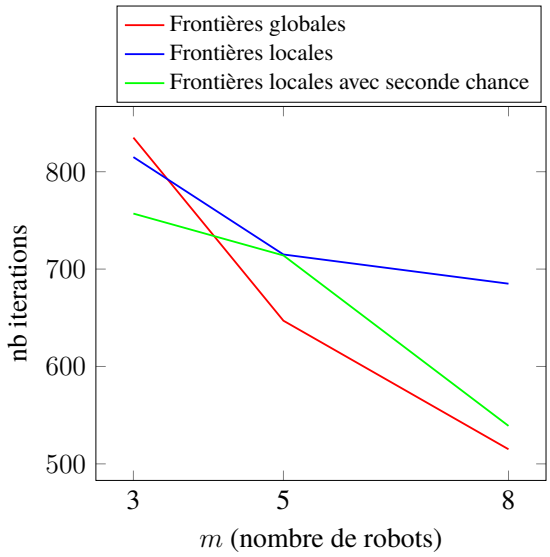

FIGURE 11 – Nombre d'itérations nécessaires pour explorer toutes les cartes pour différents nombres de robots. Les robots partent ici groupés au centre des cartes à explorer.

La figure 11 présente le nombre d'itérations nécessaires pour explorer les quatre cartes avec 3, 5 ou 8 robots. L'approche locale s'avère nettement meilleure avec la seconde chance pour 3 ou 8 robots et s'approche des performances de l'approche globale.

Notons que nous mesurons ici la durée nécessaire pour parcourir tout le labyrinthe. Si nous prenons en compte le temps de retour au point de départ des robots (leur permettant notamment de savoir que l'exploration est terminée), la seconde chance peut dans certains cas allonger l'exploration.

7 Discussion

7.1 Forces et faiblesses des différents algorithmes

L'approche *BMILRV* est locale, mais a un fonctionnement complexe (de nombreux états pour les robots, la fermeture des cellules nécessite de nombreux calculs [1]) et donne de mauvais résultats lorsque des obstacles sont isolés ou qu'il y a des boucles sur la carte, forçant les robots à faire plusieurs passages pour marquer les cellules successivement comme contrôlées, fermées et finalement nettoyées (comme le confirment les expériences sur la figure 6). A notre connaissance, cette approche n'a pas été adaptée pour une mise en œuvre décentralisée.

L'approche par frontières (globales) est à la fois simple et efficace. Elle peut être distribué facilement, mais nécessite une synchronisation fréquente de la carte et des positions du robot. Avec la stratégie d'affectation *MinPos*[4], chaque robot prend en compte la position de tous les autres robots pendant chaque affectation, ce qui implique que la carte des frontières globales et la position du robot sont fréquemment partagés. De plus, le temps de calcul de cette stratégie d'affectation dépend du nombre de robots et du nombre de frontières (les distances entre toutes les paires frontière/robot sont calculées à chaque itération), ce qui peut rendre cette méthode inutilisable si le nombre de robots est trop grand, ou si la carte est trop grande.

Enfin, l'approche par frontières locales présente des résultats proches de l'approche globale dans la plupart des cas, tout en ayant une complexité constante, ne dépendant ni du nombre total de robots ni de la taille de la carte. Il est également facilement distribuable sans contrainte sur la fréquence de partage des cartes, et les robots n'ont pas besoin de connaître le nombre ou la position des autres robots. Sans l'ajout de la « seconde chance », cependant, certains robots peuvent s'arrêter avant la fin de l'exploration, réduisant ainsi la performance de l'exploration.

7.2 Possibilités d'amélioration

La version de l'algorithme décrite ici ramène au point de départ des robots qui ne trouvent plus de frontières à explorer. Selon la carte et les conditions initiales, un robot peut revenir à son point de départ et s'arrêter avant que la carte entière ne soit explorée (laissant les autres robots terminer l'exploration). La distribution entre les robots est alors moins équilibrée et les performances de l'algorithme sont réduites. Pour éviter cela, nous pouvons envisager un système de « seconde chance » (voir partie 6), permettant aux robots qui sont revenus à leur point de départ avant la fin de l'exploration de recommencer en suivant la trace de l'un des autres robots jusqu'à ce qu'il trouve une frontière.

La dispersion des robots est également rudimentaire et ne repose que sur le choix d'une cible aussi loin que possible de toute trace laissée. Lorsque plusieurs robots sont proches (dans leurs rayons de perception respectifs), et en particulier si d_{sync}est élevé, l'ajout d'un système de dispersion supplémentaire pourrait également augmenter de manière significative les performances de l'algorithme sans augmenter consi-

dérablement sa complexité.

7.3 Conclusion

Dans la version proposée ici, l'algorithme donne de bonnes performances (nombre d'étapes pour compléter la couverture) par rapport aux deux autres approches, et au bénéfice d'une complexité constante (en temps et en mémoire) pour un robot donné, ce qui permet d'utiliser un très grand nombre de robots. En effet, dans le cas de l'approche par frontières globales, les stratégies les plus efficaces, telles que *MinPos*, nécessitent de calculer autant de distances que de paires robot/frontière, ce qui peut poser problème s'il y a un trop grand nombre de robots ou si la carte est trop grande. De plus, notre approche locale est plus facilement distribuable au prix d'une perte de performance possible lorsque les robots sont proches, et ne nécessite pas de synchronisation temporelle entre les robots

Ces résultats montrent également qu'il existe des approches intermédiaires entre les algorithmes locaux et les algorithmes globaux, pouvant prendre en compte les cellules de la grille dans un rayon de perception déterminé et garantissant une exploration complète.

Références

[1] M. Andries et F. Charpillet : Multi-robot taboolist exploration of unknown structured environments. *In 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5195–5201, 09 2015.

[2] M. Araya-Lopez, V. Thomas, O. Buffet et F. Charpillet : A closer look at momdps. *In 2010 22nd IEEE International Conference on Tools with Artificial Intelligence*, volume 2, pages 197–204, Oct 2010.

[3] Antoine Bautin, Philippe Lucidarme, Rémy Guyonneau, Olivier Simonin, Sébastien Lagrange, Nicolas Delanoue et François Charpillet : Cart-O-matic project : autonomous and collaborative multi-robot localization, exploration and mapping. 5th Workshop on Planning, Perception and Navigation for Intelligent Vehicles. *In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, page x, Tokyo, Japan, novembre 2013.

[4] Antoine Bautin, Olivier Simonin et François Charpillet : Minpos : A novel frontier allocation algorithm for multi-robot exploration. *In International conference on intelligent robotics and applications*, pages 496–508. Springer, 2012.

[5] W. Burgard, M. Moors, D. Fox, R. Simmons et S. Thrun : Collaborative multi-robot exploration. *In Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 1, pages 476–481 vol.1, 04 2000.

[6] João Paulo Lima Silva de Almeida, Renan Taizo Nakashima, Flávio Neves-Jr et Lúcia Valéria Ramos de Arruda : Bio-inspired on-line path planner for cooperative exploration of unknown environment by a multi-robot system. *Robotics and Autonomous Systems*, 112:32 – 48, 2019.

[7] Jan Faigl, Olivier Simonin et Francois Charpillet : Comparison of task-allocation algorithms in frontier-based multi-robot exploration. *In* Nils Bulling, éditeur : *Multi-Agent Systems*, pages 101–110, Cham, 2015. Springer International Publishing.

[8] Ettore Ferranti, Niki Trigoni et Mark Levene : Brick & mortar : an on-line multi-agent exploration algorithm. *In ICRA*, pages 761–767, 2007.

[9] Ettore Ferranti, Niki Trigoni et Mark Levene : Rapid exploration of unknown areas through dynamic deployment of mobile and stationary sensor nodes. *Autonomous Agents and Multi-Agent Systems*, 19(2):210–243, 10 2009.

[10] Enric Galceran et Marc Carreras : A survey on coverage path planning for robotics. *Robotics and Autonomous Systems*, 61(12):1258 – 1276, 2013.

[11] S. Garnier, F. Tache, M. Combe, A. Grimal et G. Theraulaz : Alice in pheromone land : An experimental setup for the study of ant-like robots. *In 2007 IEEE Swarm Intelligence Symposium*, pages 37–44, April 2007.

[12] Elizabeth A. Jensen et Maria Gini : Effects of communication restriction on online multi-robot exploration in bounded environments. *In* Nikolaus Correll, Mac Schwager et Michael Otte, éditeurs : *Distributed Autonomous Robotic Systems*, pages 469–483, Cham, 2019. Springer International Publishing.

[13] N. Mahdoui, V. Frémont et E. Natalizio : Cooperative frontier-based exploration strategy for multi-robot system. *In 2018 13th Annual Conference on System of Systems Engineering (SoSE)*, pages 203–210, 06 2018.

[14] Q. V. Nguyen, F. Colas, E. Vincent et F. Charpillet : Localizing an intermittent and moving sound source using a mobile robot. *In 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1986–1991, Oct 2016.

[15] O. Simonin, T. Huraux et F. Charpillet : Interactive surface for bio-inspired robotics, re-examining foraging models. *In 2011 IEEE 23rd International Conference on Tools with Artificial Intelligence*, pages 361–368, Nov 2011.

[16] Sebastian Thrun et Arno Bücken : Integrating grid-based and topological maps for mobile robot navigation. *In Proceedings of the National Conference on Artificial Intelligence*, pages 944–951, 1996.

[17] E. Vincent, A. Sini et F. Charpillet : Audio source localization by optimal control of a mobile robot. *In 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5630–5634, April 2015.

[18] B. Yamauchi : A frontier-based approach for autonomous exploration. *In Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation'*, pages 146–151, 1997.