Benchmarking des performances de systèmes multirobots
Application à l'exploration

Zhi Yan [1], **Luc Fabresse** [1], **Jannik Laval** [2], **Noury Bouraqadi** [1]

1. *Département Informatique et Automatique, École des Mines de Douai, France*
   prenom.nom@mines-douai.fr
2. *Laboratoire DISP, Université de Lyon, France*
   jannik.laval@univ-lyon2.fr

RÉSUMÉ. *Le benchmarking de performance est un thème important dans la robotique. C'est un moyen essentiel de comparer des solutions dans des conditions différentes. Nous nous intéressons à l'analyse comparative des performances des systèmes multirobots dans le cadre de l'exploration et la cartographie d'espaces inconnus. Nous proposons une sélection de métriques pour comparer objectivement les algorithmes de coordination de systèmes multi-robots pour l'exploration. Ce travail est une démarche concrète pour résoudre le problème général de l'évaluation quantitative de différents algorithmes de coordination multirobot. Nous identifions des paramètres qui influent sur la performance de flottes robotiques. En faisant varier ces paramètres, nous arrivons à identifier les atouts et les limites d'un algorithme. Nous illustrons ces contributions avec des simulations réalistes d'une stratégie d'exploration basée sur les frontières. Ces simulations ont été mises en œuvre à l'aide de l'intergiciel ROS (*Robot Operating System*).*

ABSTRACT. *Performance benchmarking has become an important topic within robotics. It is indeed, a critical way to compare different solutions under different conditions. In this paper, we focus on performance benchmarking of multi-robot systems which explore and map unknown terrains. We summarize metrics to objectively compare different algorithms that can be applied to collaborative multi-robot exploration. This work is also a first concrete step to address the general problem of objectively comparing different multi-robot coordination algorithms. We also identify parameters that impact robotic fleet performances. By varying these parameters, we can identify strengths and limits of an algorithm. We illustrate these contributions with realistic benchmark simulations of the frontier-based exploration strategy. The simulations were implemented in ROS (*Robot Operating System*).*

MOTS-CLÉS : *benchmarking de performances, systèmes multirobots, exploration collaborative.*

KEYWORDS: *performance benchmarking, multi-robot systems, collaborative exploration.*

DOI:10.3166/RIA.30.211-236 © 2016 Lavoisier

1. Introduction

De nombreuses applications robotiques peuvent bénéficier de l'utilisation d'une flotte de plusieurs robots au lieu de compter sur un seul robot (Parker, 2008). En effet, avoir plusieurs robots signifie un accroissement de la robustesse grâce à la redondance. En outre, plusieurs robots peuvent effectuer des tâches en parallèle et donc accélérer la résolution (la vitesse d'exécution des actions des robots reste la même), ce qui permet d'améliorer les applications telles que la recherche et le sauvetage suite à une catastrophe naturelle, la recherche des foyers d'incendie à l'intérieur des bâtiments, l'exploration minière, ou encore le déminage.

Cependant, l'utilisation des systèmes multirobots soulève le défi de la coordination (Yan *et al*., 2013). Pour vraiment profiter du parallélisme potentiel d'une flotte robotique, nous devons adopter des stratégies d'organisation de l'activité des robots qui garantissent la plus haute performance. A titre d'exemple, considérons l'exploration d'un environnement inconnu (Stachniss, 2009) qui est une tâche commune à de nombreuses applications. Une stratégie de coordination doit attribuer à chaque robot un ensemble de zones à explorer de façon à réduire à la fois le temps nécessaire pour construire une carte complète, et l'énergie totale consommée par la flotte robotique (Yan *et al*., 2014). Cependant, la définition d'une stratégie optimale ou quasi-optimale pour la coordination n'est pas facile (Stachniss, 2009 ; Doniec *et al*., 2009 ; Howard, 2006 ; Zlot *et al*., 2002 ; Burgard *et al*., 2000 ; Yamauchi, 1998).

La multitude d'algorithmes d'exploration multirobot est en soi un problème quand on doit choisir la solution la plus appropriée à une situation donnée. Les auteurs évaluent leurs solutions avec différents robots ou simulateurs, dans différents environnements et conditions. Par conséquent, les résultats présentés ne peuvent pas être comparés facilement. De plus, la reproductibilité des expériences est presque impossible.

Une évaluation mathématique des algorithmes, comme l'analyse de la complexité, est pratiquement infaisable, à cause des nombreux paramètres des systèmes multirobots et de leurs environnements qui peuvent influer sur les performances. Des exemples de ces paramètres sont : le nombre de robots, la puissance de calcul disponible, la taille de la mémoire sur chaque robot, ou encore l'homogénéité ou hétérogénéité de la flotte.

Dans cet article, nous adoptons une approche alternative qui est une évaluation empirique. Elle consiste au *benchmarking* des algorithmes qui peuvent être appliqués à l'exploration multirobot (Bonsignorio *et al*., 2014 ; Pobil, 2006). Pour comparer efficacement les différents algorithmes, nous avons d'abord identifié des paramètres qui influent sur la performance d'exploration d'un système multirobot. Ces paramètres visent à faciliter la reproductibilité des expériences. Ils favorisent la définition des environnements standard et des plans d'expérience qui peuvent être partagés par la communauté, et faciliter le benchmarking des résultats obtenus par différents chercheurs.

Nous avons ensuite sélectionné cinq métriques qui permettent une évaluation quantitative de la performance des solutions d'exploration robotique. Elles permettent de mesurer le temps, le coût et l'efficacité de l'exploration, ainsi que la complétude et la qualité des cartes construites. Pour chaque expérimentation, nous définissons le plan d'expérience comme l'ensemble des paramètres et des métriques utilisés. Cet ensemble doit être explicitement et clairement défini pour permettre à l'expérience d'être reproduite par d'autres chercheurs.

Ensuite, nous décrivons l'architecture de notre banc d'essai pour le benchmarking des performances de systèmes multirobots. Nous détaillons les différents choix de structure qui nous ont permis d'utiliser la grappe de serveurs de l'École des Mines de Douai. Nous discutons les compromis et les impacts de ces choix.

Enfin, nous illustrons nos métriques et paramètres en utilisant une simulation 3D réaliste (voir figure 1) avec des robots basés sur l'intergiciel ROS (abréviation anglaise de *Robot Operating System*). Le choix de ROS est justifié par le fait qu'il est un standard *de facto* utilisé par de nombreuses institutions. Nos simulations utilisent la stratégie de Yamauchi (Yamauchi, 1998) pour l'exploration multirobot. Nous montrons que nos métriques peuvent sensiblement refléter l'impact des différents algorithmes de fusion de carte sur la performance du système. Pour montrer l'influence des paramètres, nous avons mesuré les performances de différentes tailles de flotte sur différents terrains.

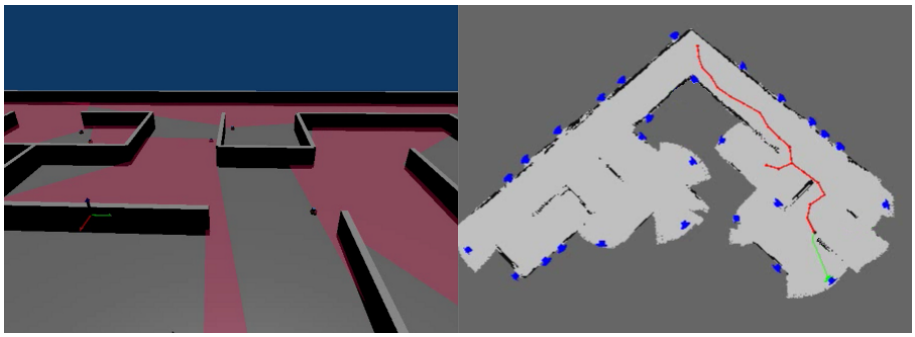


Figure 1. Le simulateur MORSE 3D (gauche) et la carte générée (droite). *Dans la partie gauche, la zone rouge représente le scan laser. Dans la partie droite, les blocs bleus représentent les destinations potentielles, le bloc vert indique la destination actuelle, la ligne rouge indique le trajet déjà effectué, et la ligne verte représente la trajectoire pour atteindre la destination actuelle*

La section 2 donne un aperçu des travaux connexes. La section 3 présente notre processus de benchmarking, et introduit la notion clé que représente le plan d'expérience. Puis, dans la section 4, nous présentons la liste des paramètres à considérer pour les plans d'expériences d'explorations multirobots, ainsi qu'une sélection de métriques pour mesurer les performances des stratégies de coordination. La section 5 décrit notre architecture matérielle et logicielle de benchmarking. La section 6 décrit les simulations réalisées pour comparer deux algorithmes de fusion de carte, ainsi

que les résultats que nous avons obtenus. Le papier se termine avec nos conlusions données dans la section 7.

2. État de l'art

(Bautin *et al*., 2012) ont comparé leur algorithme d'affectation des frontières (nommé MinPos) pour l'exploration multirobot avec deux algorithmes bien connus dans l'état de l'art : celui de (Yamauchi, 1998) et celui de (Burgard *et al*., 2000), dans trois environnements différents. Ils ont mesuré le temps d'exploration donné par les étapes de simulation, tout en faisant varier le nombre de robots. Le benchmarking a été éffectuée sur deux simulateurs développés par eux-mêmes. L'un est discret, l'autre est plus réaliste et permet d'utiliser exactement le même code source en simulation et sur les vrais robots.

(Faigl *et al*., 2014) ont proposé un framework d'évaluation pour étudier les stratégies d'exploration. Ils comparent cinq algorithmes de répartition des tâches dans les scénarios d'exploration multirobot. Ils ont réalisé des simulations discrètes dans quatre environnements représentant des espaces ouverts et des bureaux. La métrique de performance utilisée est le nombre de pas de simulation nécessaires pour explorer tout l'environnement.

(Couceiro *et al*., 2014) ont présenté plusieurs expériences de simulation pour évaluer cinq algorithmes d'exploration et de cartographie multirobot. Ils ont utilisé deux métriques de performance : 1) le rapport d'exploration d'environnement au fil du temps, calculé en réalisant la division de la surface de la carte construite par les robots à un instant donné par la surface totale du terrain ; 2) l'aire sous la courbe (*area under the curve – AUC*) qui est obtenue en calculant la moyenne de 500 itérations du rapport d'exploration. Dans leurs expériences, le temps est représenté par le nombre d'itérations nécessaires à l'exploration. Les expériences ont été effectuées en utilisant le simulateur discret basé sur Matlab, MRSim [1].

(Frank *et al*., 2010) ont comparé quatre stratégies différentes de sélection de frontières pour l'exploration multirobot et analysé la performance en termes de temps nécessaire pour explorer tout l'environnement compté en nombre de pas de simulation, ainsi qu'en termes de zone explorée par pas de temps. Les expériences sont réalisées dans un simulateur discret et idéal : elles négligent les problèmes de localisation et seulement les environnements convexes sans obstacle sont considérés. Les expériences de simulation ont été réalisées en Matlab.

(Amigoni, 2008) a expérimentalement comparé quatre stratégies d'exploration afin d'évaluer leurs forces et leurs faiblesses. Les expériences ont été effectuées sur un simulateur discret avec un seul robot. Deux métriques ont été prises en compte pour comparer les performances des stratégies d'exploration : le nombre d'opérations de

1. http://www.mathworks.com/matlabcentral/fileexchange/38409-mrsim-multi-robot-simulator--v1-0-

détection nécessaires pour réaliser l'exploration et la distance totale parcourue par le robot pendant l'exploration.

(Scrapper *et al*., 2009) se sont concentrés sur le développement de méthodes et techniques de tests standards pour évaluer quantitativement la performance des systèmes de cartographie robotique par rapport aux exigences définies par l'utilisateur. Ils ont défini la qualité de la carte comme le rapport entre le nombre de points caractéristiques valables trouvés sur la carte construite par le robot et le nombre de points caractéristiques dans la véritable carte. Toutefois, cette métrique n'évalue pas si les caractéristiques présentent la même forme.

(Lass *et al*., 2009) ont examiné plusieurs métriques d'évaluation pour les systèmes multiagents. Ils ont classé les métriques suivant deux axes : la performance et les types de données. Les métriques de performance quantifient la consommation des ressources du système, telles que l'utilisation de bande passante, la consommation d'énergie, et le temps d'exécution. Les types de données correspondent en effet aux quatre niveaux de mesure dans les statistiques, incluant le nominal, l'ordinal, l'intervalle, et le rapport.

Au travers de l'état de l'art, nous pouvons constater que chaque proposition est validée de manière différente des autres. Cette différence touche un élément aussi central que les métriques utilisées pour évaluer les performances. Les terrains explorés sont également très différents au niveau taille ou type et densité des obstacles. Les simulateurs mis en œuvre sont également très différents. Certains sont discrets et idéaux, alors que d'autres sont plus réalistes. Ces différences rendent difficile le benchmarking de diverses propositions. Il est donc nécessaire de disposer d'un ensemble de métriques de référence qui pour évaluer les différentes facettes du problème d'exploration multirobot. Par ailleurs, nous devons également disposer de plans d'expériences les plus complets possible, afin de permettre d'identifier les cadres précis dans lesquels sont effectuées les évaluations ou *benchmarking*.

3. Benchmarking et plan d'expérience

Le processus de benchmarking que nous proposons est représenté par la figure 2. Il commence par la conception de l'expérience par l'utilisateur au travers de la définition d'un *plan d'expérience*. Puis, les différentes expérimentations sont réalisées indépendamment les unes des autres. Ces expérimentations peuvent être réalisées sur de vrais robots ou en simulation. Elles peuvent être exécutées de manière séquentielle ou en parallèle en fonction des ressources disponibles. Enfin, les données recueillies sont utilisées pour calculer les résultats sur la base de métriques.

L'élément central de notre processus de benchmarking est le plan d'expérience. Il spécifie les valeurs des paramètres de l'expérience et les données à collecter. Un exemple de plan d'expérience est fourni par le tableau 1. En explicitant ces informations, il permet la répétabilité des expérimentations.

Dans le cadre d'applications multirobots, un plan d'expérience doit comporter les information suivantes :

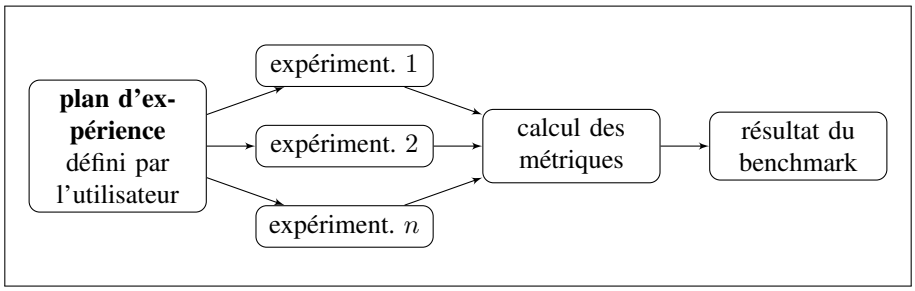


Figure 2. Processus de benchmark : du plan d'expérience au calcul des métriques

Paramètres abstraits. Il s'agit d'une liste de paramètres (par exemple nombre de robots ou la taille du terrain) qui potentiellement peuvent avoir un impact sur les performances de la flotte robotique. Cet impact est hypothétique tant qu'il n'a pas été validé par l'expérience. Pour chaque paramètre, l'utilisateur fournit une ou plusieurs valeurs (par exemple nombre de robots dans l'intervalle [1, 30]) qui seront utilisées dans les différentes expérimentations. Par la suite nous utilisons"paramètre" pour désigner un "paramètre abstrait" quand il n'y a pas d'ambiguïté.

Vecteurs de paramètres concrets. Un vecteur de paramètres concrets (ou plus simplement vecteur de paramètres [2]) contient une valeur unique pour chacun des paramètres abstraits précédemment identifiés (par exemple <3 robots, terrain de 30 x 30 m>). Chaque expérimentation (simulation ou exécution sur de vrais robots) utilise un seul vecteur de paramètres. L'utilisateur peut soit lister tous les vecteurs de paramètres qu'il veut explorer, soit créer une fonction qui les calcule (par exemple, on veut toutes les combinaisons possibles entre nombres de robots et tailles de terrains).

Itérations. C'est le nombre de fois que chaque expérimentation est répétée. Les expérimentations robotiques sont en général non déterministe. Cela peut provenir de phénomènes physiques externes comme la luminosité qui peut varier ou d'éléments liés aux robots comme les marges d'erreur sur les valeurs des composants ou le bruit au niveau des capteurs ou encore du fait du recours à des algorithmes non déterministes. Il est donc nécessaire de réaliser chaque expérimentation plusieurs fois pour valider statistiquement les résultats.

Conditions d'arrêt. On distingue deux types de conditions d'arrêt pour une expérimentation : celles où l'expérimentation se termine correctement, et l'on peut calculer les métriques associées, et celles où l'expérimentation n'est pas en mesure de terminer correctement. Ce deuxième type de condition d'arrêt correspond aux boucles infinies ou aux situations de blocage.

Mesures. Ce sont les données à collecter pour chaque simulation. L'utilisateur doit spécifier à quel moment la mesure doit être réalisée en spécifiant la fréquence

2. Nous utilisons cette dénomination raccourcie par la suite.

pour une mesure périodique ou une condition de déclenchement pour les données à récolter lors d'évènements particuliers.

Métriques. Elles permettent de calculer les résultats de l'expérimentation à partir des mesures récoltées pour comparer quantitativement (et objectivement) les différents paramètres (cf. section 4.2).

Tableau 1. Exemple d'un plan d'expérience

Paramètres	Robot : Pioneer 3-DX
	Nombre de robots : $[1, 30]$
	...
Vecteurs de paramètres	Toutes les combinaisons possibles
Itérations	5
Conditions d'arrêt	99 % du terrain exploré
	2000 secondes écoulées
Mesures	espace exploré toutes les secondes
	% processeur utilisé pour chaque robot toutes les 5sec
	...
Métriques	Temps d'exploration
	Coût d'exploration
	...

4. Plans d'expériences pour l'exploration multirobot

Cette section décrit les concepts nécessaires à la création de plans d'expériences dans le contexte de l'exploration et de la cartographie d'environnements inconnus par des systèmes multirobots. Notre objectif est de fournir à la communauté un cadre de référence pour le benchmarking de stratégies de coordination multirobot. Idéalement, ce cadre devrait permettre de constituer une bibliothèque de plans d'expériences que les chercheurs utiliseraient pour évaluer leurs propositions. Les évaluations des différents travaux devraient être ainsi facilitées du fait du recours à un référentiel commun. On retrouve cette idée dans d'autres contextes, par exemple la robotique de sauvetage avec *RoboCup Rescue* [3]. Cependant, on ne trouve que des bibliothèques d'environnements types ou des relevés de capteurs, mais pas les plans des expériences réalisées.

4.1. Paramètres

Nous avons identifié trois familles de paramètres qui peuvent avoir un impact sur les performances des algorithmes d'exploration multirobot : *Robot*, *Flotte*, et *Environnement*. La suite de cette section présente les paramètres contenus dans ces trois familles.

3. http://wiki.robocup.org/wiki/Robot_League#Field_Description

4.1.1. Robot

– Propriétés de locomotion. Elles couvrent les caractéristiques du robot telles que le modèle de mouvement (holonome [4] ou non).

– Capacités de calcul : CPU, RAM, fréquence d'horloge. Le choix d'un algorithme d'exploration doit prendre en compte les ressources disponibles sur le robot.

– La précision, la fréquence et la portée des capteurs. Les caractéristiques des capteurs influent sur la localisation, la construction de carte, et peuvent avoir un impact sur la qualité de la carte construite et sur le temps de construction.

4.1.2. Flotte

– Nombre de robots. Intuitivement on pourrait penser qu'avoir plus de robots peut conduire à une exploration plus rapide. Mais, cela dépend en fait de la stratégie de coordination et du terrain. Les robots d'une même flotte peuvent se gêner par exemple dans un environnement avec beaucoup de couloirs.

– Homogénéité de la flotte. L'utilisation d'une flotte hétérogène (e.g. robots aériens et robots terrestres) peut améliorer la performance d'exploration.

– Positions initiales des robots. Selon l'environnement et les obstacles, la performance d'exploration peut être significativement affectée par la position des robots au démarrage de l'exploration (Yan *et al*., 2014).

– Bande passante des canaux de communication. Certains algorithmes nécessitent que les robots échangent de grandes quantités de données. Leur performance pourrait baisser considérablement lors de l'utilisation des robots avec des interfaces réseau qui offrent une bande passante limitée.

– Portée de communication. L'exploration multirobot nécessite souvent une communication par des réseaux sans fil. Selon le réseau sans fil utilisé, la portée de communication peut varier. Cette portée influence la collaboration et la performance d'exploration. En effet, sur de grands terrains ou en fonction de la nature des obstacles, les transmissions sans fil peuvent être ralenties. Les robots pourraient être déconnectés et incapables de communiquer ou de coopérer. Toutefois, cette question peut être atténuée par exemple par la prise en compte de la connectivité du réseau dans la planification des déplacements (Doniec *et al*., 2009 ; Le *et al*., 2009).

4.1.3. Environnement

– Taille du terrain. A nombre de robots constants, l'exploration d'un grand terrain nécessite généralement plus de temps que pour un petit terrain.

– Densité et forme des obstacles. Dans un environnement avec beaucoup d'obstacles, il y a moins d'espace à explorer. Cependant, la navigation peut être plus compliquée, surtout avec des obstacles concaves où les blocages peuvent survenir lorsque plusieurs robots sont situés dans la même zone.

4. Capable de se déplacer dans toutes les directions.

– Morphologie du terrain. L'exploration d'un terrain avec un seul grand espace prend sans doute moins de temps qu'un environnement décomposé en plusieurs espaces ouverts reliés avec des couloirs étroits. Dans ce dernier, il est probable que les robots se gênent dans leurs déplacements.

– Dynamicité. Si l'environnement change (*e.g.* effondrements de construction) ou s'il y a d'autres entités mobiles (*e.g.* sauveteurs humains ou d'autres robots), le temps d'exploration et les coûts associés peuvent varier entre différentes expérimentations.

4.2. Métriques

La mesure de performance et l'évaluation quantitative sont nécessaires pour l'exploration robotique notamment pour des applications critiques comme les opérations de sauvetage après une catastrophe naturelle. Une évaluation quantitative de plusieurs méthodes d'exploration permet de choisir la méthode la plus efficace à une situation donnée.

Dans cette section, nous présentons cinq métriques de performance que nous avons sélectionnées pour quantifier les résultats d'exploration et donnons clairement leurs définitions. Elles sont le *temps d'exploration*, le *coût d'exploration*, l'*efficacité d'exploration*, la *complétude de carte*, et la *qualité de carte*.

Elles présentent l'avantage d'être applicables à une large variété de problèmes d'exploration avec des flottes robotiques de caractéristiques différentes qui opèrent dans différents types d'environnements.

4.2.1. Temps d'exploration

L'un des objectifs concernant l'optimisation de l'exploration multirobot est de minimiser le temps global d'exploration (Yan *et al.*, 2014 ; Stachniss, 2009 ; Burgard *et al.*, 2000 ; Yamauchi, 1998). Le défi est de faire que chaque robot se déplace vers une direction différente afin de maximiser la surface découverte de l'environnement et de réduire ainsi le nombre de zones visitées plusieurs fois.

Nous définissons le temps d'exploration comme le temps nécessaire pour achever une mission d'exploration pour une flotte de robots. Dans notre définition, le décompte commence lorsqu'au moins un robot de la flotte commence l'exploration, et se termine lorsqu'une proportion significative a été explorée par au moins un robot (par exemple, 99 % de l'ensemble du terrain). La durée de l'exploration est mesurée en unité universelle du temps (secondes).

4.2.2. Coût d'exploration

La définition du coût d'exploration dépend fortement des besoins des utilisateurs. Il peut être défini comme l'énergie consommée par les actionneurs et les ressources de calcul (par exemple, CPU, RAM, et bande passante de réseau), ou le prix des robots, ou encore leurs coûts de manutention et d'entretien.

Cependant, la dépense énergétique est la seule à être directement influencée par les algorithmes d'exploration. Par ailleurs, l'énergie consommée pour le calcul peut être négligée face à celle consommée par les moteurs pour les déplacements. La distance parcourue par les robots est une bonne approximation du coût énergétique des moteurs. Elle est aussi simple à mesurer, ce qui explique sa large utilisation dans la littérature (Yan *et al.*, 2014 ; Stachniss, 2009 ; Zlot *et al.*, 2002 ; Burgard *et al.*, 2000).

Ainsi, nous définissons le coût d'exploration comme la somme des distances parcourues par chaque robot de la flotte :

$$coutDExploration(n) = \sum_{i=1}^{n} d_i \tag{1}$$

où n est le nombre de robots dans la flotte, et d_i est la distance parcourue par le robot i mesuré en mètre.

4.2.3. Efficacité d'exploration

L'efficacité d'exploration est directement proportionnelle à la quantité d'information extraite de l'environnement, et est inversement proportionnelle aux coûts d'exploration de la flotte de robot (Zlot *et al.*, 2002) :

$$EfficaciteDExploration(n) = \frac{M}{coutDExploration(n)} \tag{2}$$

où n est le nombre de robots dans la flotte, et M est la surface de la zone explorée totale en mètres carrés. La surface de la carte issue de l'exploration constitue une bonne approximation de la zone effectivement explorée.

Rappelons que le coût d'exploration est mesuré en mètres parcourus. Ainsi, si la valeur de l'efficacité d'exploration est $1,6$, cela signifie qu'un robot qui se déplace d'$1m$ découvre en moyenne $1,6m^2$ du terrain. Inspiré de l'analyse de rapport coûts-avantages en économie, les utilisateurs peuvent considérer qu'un algorithme est digne d'être utilisé si son efficacité d'exploration a une valeur supérieure ou égale à 1. Ceci est bien évidemment vrai dans le cas d'un terrain dont les distances entre obstacles sont supérieures à 1 m.

4.2.4. Complétude de carte

La construction de carte est une tâche étroitement couplée à l'exploration. La complétude de la carte est l'un des problèmes principaux (Couceiro *et al.*, 2014 ; Frank *et al.*, 2010 ; Scrapper *et al.*, 2009). Mesurer la complétude nécessite une connaissance préalable du terrain exploré et cartographié. Nous définissons la complétude de carte comme le rapport entre la superficie de la carte résultat de l'exploration M et la superficie réelle du terrain P :

$$completudeDeCarte = \frac{M}{P} \tag{3}$$

4.2.5. Qualité de carte

Construire une carte exacte par des robots autonomes est encore un problème ouvert. Les raisons des erreurs d'une carte peuvent être la précision des capteurs ou les algorithmes de SLAM (*Simultaneous Localization And Mapping*). Pour identifier ces erreurs, nous avons besoin d'une carte-vérité [5] du terrain.

Comme la grille d'occupation [6] est largement utilisée pour représenter l'espace inconnu, occupé et libre dans l'exploration, nous définissons l'erreur de carte comme le nombre de cellules c dans la carte explorée qui ont une valeur différente de la cellule p correspondante dans la carte réelle :

$$erreurDeCarte = \sum_i c_i, c_i \neq p_i \tag{4}$$

où c_i et p_i sont les cellules avec l'indice i dans la carte 2D.

Les résultats obtenus de cette formule sont affectés par la résolution de la carte. En utilisant les mêmes capteurs et algorithmes, l'erreur est susceptible d'être plus importante dans une carte à haute résolution que dans une carte à faible résolution. Une bonne performance d'exploration doit afficher un compromis entre l'erreur de carte et sa résolution.

Nous pouvons calculer ensuite la qualité de carte comme le rapport entre d'une part la superficie de l'intersection de la carte obtenue et la carte-vérité de terrain, avec d'autre part la superficie de la carte-vérité de terrain P :

$$qualiteDeCarte = \frac{M - A(erreurDeCarte)}{P} \tag{5}$$

où M est la surface de la zone explorée totale en mètre carré, et $A(erreurDeCarte)$ est la superficie occupée par les cellules d'erreur.

5. Banc d'essai

Dans cette section, nous présentons notre banc d'essai à travers son modèle d'architecture. Nous détaillons ensuite l'infrastructure construite pour le mettre en œuvre ainsi que la façon dont nous automatisons le déploiement des contrôleurs de robot.

5.1. Architecture

La figure 3 illustre l'architecture de notre banc d'essai composée de trois parties inter-connectée à l'aide de l'intergiciel ROS : le simulateur, le moniteur, et un ensemble de contrôleurs de robot.

---

5. Traduction de *ground truth*
6. Traduction de *Occupancy grid map*

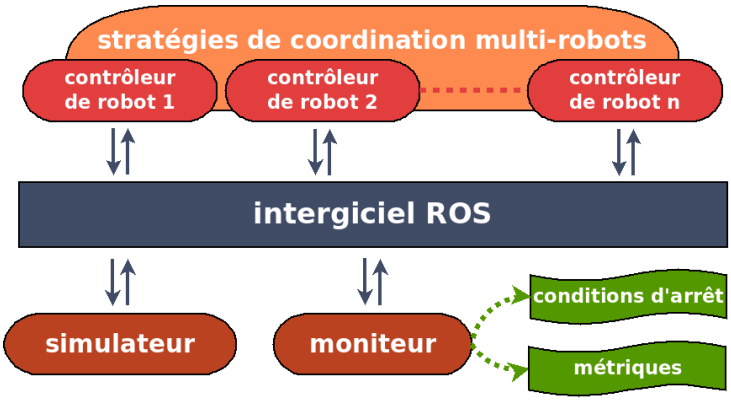


Figure 3. L'architecture du banc d'essai

Nous utilisons le simulateur réaliste 3D MORSE encapsulé dans un nœud ROS. Le moniteur, également encapsulé dans un nœud ROS, nous permet de contrôler et d'arrêter la simulation lorsqu'une condition d'arrêt survient. Il stocke également les données nécessaires au calcul des métriques.

Chaque contrôleur de robot se matérialise comme un ensemble de nœuds ROS. Le composant de stratégies de coordination représenté dans la figure 3 est transversal aux contrôleurs de robot, car nous sommes intéressés par des algorithmes de coordination distribués. Cela signifie que, chaque contrôleur de robot contient des nœuds ROS dédiés à la coordination.

5.2. Infrastructure

Le simulateur MORSE et le moniteur système sont déployés sur un poste de travail avec 8 processeurs, 8 Go de RAM, une carte graphique GeForce 770 GTX et un adaptateur Ethernet Gigabit. Les contrôleurs de robot sont déployés sur une grappe de serveurs afin de répondre aux exigences de calcul pour une simulation de plusieurs robots simultanément. Notre grappe se compose de 70 nœuds de calcul qui fournissent les ressources nécessaires pour les applications distribuées de haute performance. Chaque nœud de calcul dispose de 8 processeurs au moins et de 16Go de RAM minimum. La communication entre les contrôleurs de robots ainsi que les communications entre le simulateur et les contrôleurs de robots sont effectuées via le réseau en utilisant un mécanisme *publish-subscribe*.

Les grappes de calcul sont généralement partagées et imposent des contraintes quant au système d'exploitation. Par exemple, la grappe de notre université exécute une version de CentOS qui n'est pas supportée par l'intergiciel ROS. Nous avons résolu ce problème en utilisant des machines virtuelles (VMs) libres VirtualBox [7]. C'est

7. http://virtualbox.org

aussi un choix permettant de maximiser la réutilisabilité de notre banc d'essai. Donc pour chaque robot, nous avons utilisé une VM avec Ubuntu 12.04.1 LTS et la version Groovy de ROS. Ainsi, nous pouvons utiliser le même logiciel en simulation qu'avec de vrais robots. Comme tous les robots fonctionnent sous la même infrastructure, le ralentissement potentiel est uniformément réparti. En conséquence, les mesures de différents algorithmes seront uniformément influencées et donc comparables.

Les communications en réseau dans notre banc d'essai peuvent être divisées en trois parties :

– Les communications entre les robots et le moniteur. Le moniteur reçoit des données de mesure telles que les cartes d'exploration et les distances parcourues de robots. La bande passante du réseau consommée peut dépendre principalement de la taille du terrain à explorer. Par exemple, elle est environ de 5.0Mo/s pour un terrain de $80m \times 80m$ pour chaque robot. Ce terrain est représenté sur la figure 4d.

– Les communications entre le simulateur et les robots. Le simulateur envoie aux robots des informations de capteurs tels que les balayages de laser et les données d'odométrie [8], et reçoit des informations de commande de mouvement des robots. Évidemment, la quantité de données transmises et la bande passante réseau requise augmentent avec le nombre de robots.

– Les communications entre les robots. Cette partie des données est principalement due à la stratégie de coordination mise en place comme l'échange de cartes. Le taux de transmission entre les robots peut aussi dépendre de la taille du terrain à explorer et augmenter avec le nombre de robots.

Le tableau 2 rassemble les valeurs de la bande passante moyenne consommée par les communications entre 2 robots, ainsi que celles entre les robots et le simulateur. Ces valeurs ont été obtenues avec le terrain représenté dans la figure 4d.

Tableau 2. Bande passante utilisée par les différents types de communication

	simulateur $\downarrow$ **1 robot**	**simulateur** $\uparrow$ **1 robot**	**simulateur** $\updownarrow$ **tous les robots**	**1 robot** $\updownarrow$ **1 robot**
2 robots	$\approx$ 3Mo/s	$\approx$ 380Ko/s	$\approx$ 7Mo/s	$\approx$ 5.0Mo/s
4 robots	$\approx$ 5Mo/s	$\approx$ 460Ko/s	$\approx$ 22Mo/s	$\approx$ 5.0Mo/s
8 robots	$\approx$ 10Mo/s	$\approx$ 600Ko/s	$\approx$ 85Mo/s	$\approx$ 3.5Mo/s
16 robots	$\approx$ 17Mo/s	$\approx$ 650Ko/s	$\approx$ 282Mo/s	$\approx$ 2.2Mo/s

5.3. Automatisation de déploiement

Comme l'automatisation est une caractéristique importante pour un banc d'essai, nous avons entièrement automatisé le travail de déploiement en scriptant toutes les

8. Estimation de la position du robot basée typiquement sur des compte-tours situés sur les roues.

manipulations de fichiers, les exécutions de programmes, et les impressions de texte. L'algorithme 1 résume le processus de déploiement.

Algorithme 1 Déploiement de simulation multirobot

1: Obtenir l'ensemble O des nœuds de calcul attribués par l'ordonnanceur de grappe.
2: Déployer autant de contrôleurs de robots que $taille\_equipe\_max$ (nous avons donc jusqu'à $robots\_par\_noeud$ robots sur chaque nœud $o \in O$).
3: **for** $n = taille\_equipe\_min$ **to** $taille\_equipe\_max$ **do**
4: **for** $essai = 1$ **to** $essais\_max$ **do**
5: Exécuter le simulateur MORSE et le moniteur sur le poste de travail.
6: Lancer n contrôleurs de robots sur p nœuds où $p \subseteq O$.
7: **end for**
8: **end for**

La planification des tâches sur la grappe de serveurs est effectuée par un *Portable Batch System (PBS)*, qui peut répartir les tâches de calcul parmi les ressources informatiques disponibles. Le nombre de nœuds de calcul demandé (noté $noeuds$) peut être calculé à partir du nombre maximum de robots dans une flotte (noté $taille\_equipe\_max$) et le nombre de robots qui peut être déployé sur chaque nœud (noté $robots\_par\_noeud$) :

$$noeuds = \lceil taille\_equipe\_max \div robots\_par\_noeud \rceil \tag{6}$$

Nous avons utilisé 2 processeurs par contrôleur de robot. En fait, ce choix est un compromis entre la performance de la VM et la ressource disponible de la grappe de serveurs pour la simulation d'exploration multirobot. Nous avons mené plusieurs expériences avec différentes configurations de matériel émulé. Nous avons fixé le nombre de robots à 8 et la RAM de chaque VM à 2 Go, en variant le nombre de processeurs par VM de 1 à 8 et le nombre de nœuds de calcul également de 1 à 8. Le tableau 3 montre les résultats expérimentaux pour explorer le terrain représenté dans la figure 4d, dans lequel le temps de simulation est la valeur moyenne de 5 essais identiques.

Tableau 3. Test de performance avec différentes configurations matérielles de robots

Processeur(s) par VM	1	2	4	8
Nœud(s) de calcul	1	2	4	8
Robot(s) simulé(s) par nœud	8	4	2	1
Temps de simulation (en secondes)	489s	**349s**	350s	384s
Bande passante de réseau (simulateur $\leftrightarrow$ robots)	$\approx$ 85Mo/s			
RAM utilisée par chaque VM	$\approx$ 710Mo			

En regardant le tableau 3, nous pouvons voir qu'une équipe de robots avec 2 processeurs chacun obtient un temps de simulation le plus court. Ensuite, la bande passante nécessaire pour la communication entre le simulateur et les robots est la même

que pour les autres configurations. C'est pourquoi nous avons choisi cette configuration pour notre banc d'essai. Ainsi, comme chaque nœud de calcul dispose de 8 processeurs, alors $robots\_par\_noeud = 4$. Supposons que $taille\_equipe\_max = 30$, alors $noeuds = 8$.

Une autre possibilité serait d'utiliser 4 processeurs par robot, puisque nous avons obtenu presque le même temps de simulation. Cependant, ceci est un gaspillage inutile de ressources. En outre, comme nous utilisons une grappe de serveurs partagée, plus nous demandons de ressources de calcul [9], plus nous devons attendre avant d'avoir un créneau de calcul alloué pour exécuter nos simulations.

Notre algorithme permet de lancer séquentiellement plusieurs expériences en prenant en compte les différentes tailles d'équipe et le nombre d'essais pour chaque taille d'équipe. En outre, plusieurs essais pour une taille d'équipe nous permettent d'obtenir des résultats objectifs du point de vue statistique.

Comme indiqué dans la section 5.2, chaque robot simulé est encapsulé dans une VM. Les VM prennent beaucoup d'espace de stockage (6.8Go dans notre cas) et le temps nécessaire pour les déployer peut être très long. Le temps de déploiement est proportionnel au nombre de robots.

Pour réduire le temps de copie, nous proposons ici une stratégie de déploiement efficace, décrite dans l'algorithme 2. Nous avons d'abord construit une VM correspondante à un contrôleur de robot générique appelée un *prototype*. Ce prototype est ensuite copié dans une zone de stockage temporaire de chaque nœud de calcul. Ensuite, nous le clonons localement autant de fois que nécessaire, et attribuons des adresses IP différentes à chaque VM clonée.

Algorithme 2 Déploiement des VMs de contrôleurs de robots

1: Copier le prototype de robot (VM) dans une zone de stockage temporaire du système de fichiers de chaque nœud $o \in O$.
2: $nombre\_vm\_deployees = 0$.
3: **while** $nombre\_vm\_deployees < taille\_equipe\_max$ **do**
4: Cloner le prototype et attribuer une adresse IP à la VM clonée.
5: $nombre\_vm\_deployees + +$
6: **end while**

6. Illustration avec des simulations

Pour illustrer l'utilisation de nos métriques et nos paramètres de benchmarking, nous avons effectué une série de simulations en utilisant la stratégie d'exploration multirobot de Yamauchi basée sur les frontières (Yamauchi, 1998). Dans cette stratégie décentralisée, chaque robot décide de manière autonome où aller en fonction de

9. Il s'agit d'une requête au logiciel gestion et d'ordonnancement de la grappe de calcul.

sa carte locale. L'échange de la carte est la seule tâche coopérative. La carte locale est mise à jour par une fusion avec les cartes envoyées par les autres robots. Une fois qu'un robot met à jour sa carte, il sélectionne la frontière la plus proche et se déplace vers elle. Nous évaluons l'impact de deux algorithmes de fusion de carte sur la performance d'exploration.

6.1. Plan d'expérience

Le tableau 4 donne le plan d'expérience d'une exploration multirobot basée sur les frontières. Certains paramètres ont des valeurs fixées. C'est à dire, la même valeur est utilisée dans toutes les simulations. Conformément à la Section 4, le tableau fournit toutes les informations nécessaires pour reproduire l'expérience.

Un ensemble homogène de robots Pioneer 3-DX avec 2 processeurs et 2 Go de mémoire est simulé. Chaque robot est équipé d'un laser SICK LMS500 qui fournit 180 points d'échantillonnage avec 180 degrés de champ de vision et une portée maximale de 30 mètres. Conformément au robot réel, la vitesse maximale de chaque robot simulé a été fixée à 1,2 mètre par seconde pour un mouvement linéaire et 5,24 radians par seconde pour un mouvement de rotation. Un bruit blanc gaussien a été ajouté aux données d'odométrie. L'écart type est de 0,022 mètre pour le bruit de position (x et y) et 0,02 radian pour le bruit de rotation. Ce bruit est proche de celui du vrai robot Pioneer 3-DX.

Les robots sont initialement placés le long d'une ligne verticale à partir du coin en haut à gauche vers le coin en bas à gauche du terrain. La distance entre les positions initiales des robots est réglée à 2 mètres. Les robots communiquent entre eux à travers un réseau dont le débit est 1 Gigabit par seconde. La portée maximale de communication entre les robots est fixée à 200 mètres. L'impact des obstacles sur la communication est ignoré. Ce réglage n'a pas d'impact sur notre évaluation, car nous évaluons des algorithmes de fusion de carte. Néanmoins, nous avons prévu d'aborder ce point important dans nos futurs travaux et ajouter différents modèles de communication à notre banc d'essai.

Tous les terrains font une taille de 80 mètres de long et 80 mètres de large. La hauteur des obstacles est réglée à 2 mètres. Ainsi, tous les obstacles sont plus haut que les robots. La distance minimale entre obstacles est quant à elle fixée à 8 mètres. Les robots peuvent donc passer par toutes les ouvertures.

L'expérimentation est réalisée avec différentes tailles de flotte allant de 1 à 30 robots, et quatre terrains présentés dans la figure 4. Ces terrains sont inspirés de ceux de la *RoboCup Rescue*. Ils ont la même taille, mais sont structurés différemment :

1. Le terrain *loop* correspond à une boucle.

2. Le terrain *cross* correspond à un carrefour. Il contient cinq croisements de routes, mais la densité d'obstacles est faible.

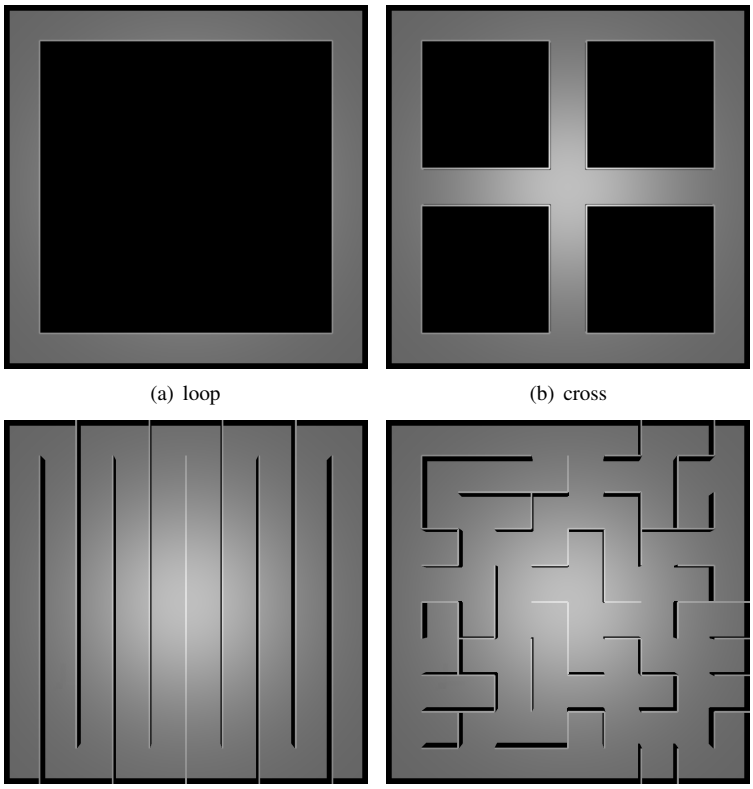
3. Le terrain *zigzag* n'a pas de croisement, mais contient plus d'obstacles. En outre, le robot doit planifier un long chemin.

Tableau 4. Plan d'expérience d'une exploration multirobot basée sur les frontières

Paramètres	**Robot**	Modèle : Pioneer 3-DX
		Capacités de calcul : 2 processeurs, 2 Go de RAM
		Vitesse maximale : 1.2m/s, 5.24rad/s
		Télémètre laser : SICK LMS500
	Flotte	Nombre de robots : $[1, 30]$
		Homogénéité : Homogène (Pioneer 3-DX)
		Positions initiales de robots : à gauche, de haut en bas, tous les 2 mètres
		Réseau de communication : Gigabit Ethernet câblé
		Portée de communication : 200m
	Environnement	Terrains : $loop, cross, zigzag, maze$
		Taille de terrain : 80m $\times$ 80m
		Hauteur d'obstacle : 2m
		Largeur de couloir : 8m
Vecteurs de paramètre	Toutes les combinaisons possibles	
Itérations	5	
Conditions d'arrêt	99 % du terrain exploré	
	2000 secondes écoulées	
Mesures	Zone explorée par pas de 1 seconde	
	% de CPU utilisé par chaque robot toutes les 5 secondes	
	% de RAM utilisée par chaque robot toutes les de 5 secondes	
	ko reçus par chaque robot toutes les 5 secondes	
	ko envoyés par chaque robot toutes les 5 secondes	
Métriques	Temps d'exploration	
	Coût d'exploration	
	Efficacité d'exploration	
	Qualité de carte	

4. Le terrain *maze*, plus complexe, contient de nombreux obstacles et impasses.

Comme (1) il existe des composants non-déterministes dans l'environnement simulé, tels que les bruits de scan de laser et d'odométrie ; (2) l'algorithme mis en oeuvre pour le SLAM est un algorithme probabiliste ; (3) nous utilisons une grappe de serveurs partagée avec une bande passante variable suivant la charge ; nous avons effectué cinq essais pour chaque taille de flotte, et affiché la valeur moyenne de ces essais. Cinq essais sont suffisants pour démontrer le processus de benchmarking proposé dans cet article. Ce choix est pragmatique du fait du nombre important d'expérimentations et de leurs durées (plus de 30 min dans certaines configurations). Néanmoins, plus d'essais permettraient d'améliorer la valeur statistique des résultats. Un moniteur



(a) loop (b) cross

(c) zigzag (d) maze

Figure 4. Terrains créés pour le benchmarking de performance

est déployé sur une station de travail. Il termine chaque essai lorsque 99 % du terrain est découvert (essai réussi) ou lorsque le temps d'exploration dépasse 2000 secondes (essai échoué). Des vidéos extraites de nos simulations sont disponibles sur notre site web à l'adresse http://car.mines-douai.fr/author/zhi/.

6.2. Logiciel de contrôle de robot

Nous utilisons l'intergiciel ROS pour construire le logiciel de contrôle des robots dans notre banc d'essai. L'architecture de ce logiciel est détaillée dans la figure 5. Chaque contrôleur de robot intègre des composants pré-existants. Les plus importants sont représentés dans la figure 5 et décrits ci-dessous :

– *gmapping* : Il exécute un algorithme de SLAM à partir des données laser (Grisetti *et al*., 2007). Nous l'utilisons pour extraire la position du robot, qui alimente le package *explore*.

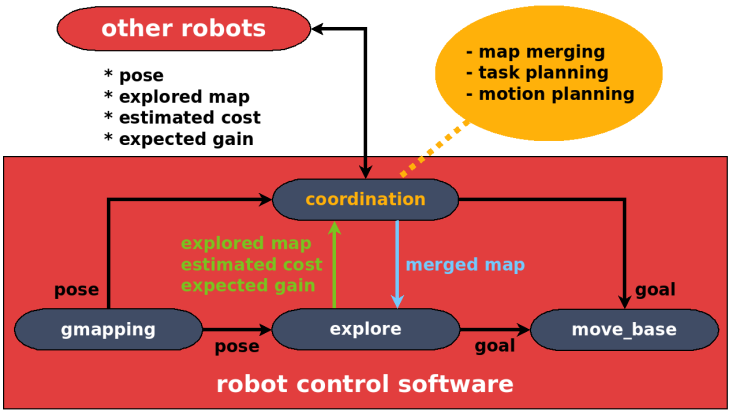


Figure 5. Architecture du logiciel de contrôle de robot

– *explore* : Il met en œuvre la stratégie de Yamauchi pour l'exploration basée sur les frontières.

– *coordination* : Nous avons construit ce paquetage comme une interface pour le banc d'essai. Ce paquetage fournit un framework, dans lequel les utilisateurs peuvent mettre en œuvre leurs propres stratégies de coordination tels que la fusion de carte, la planification des tâches et la planification de mouvement. Fondamentalement, il peut recevoir la position de robots, la carte explorée, le coût estimé et le gain d'information attendue des autres robots. Il peut fournir la carte fusionnée au paquetage *explore* et l'emplacement de destination au paquetage *move_base*. L'emplacement de destination peut être le résultat de la planification des tâches ou la planification de mouvement.

– *move_base* : Il met en œuvre l'algorithme de Dijkstra pour la planification de chemin.

La conception de cette architecture logicielle de robot offre un système robuste. Quand un robot ne peut pas se coordonner avec l'équipe à cause de pertes de communication, il peut toujours prendre ses propres décisions.

Nous avons publié nos packages développés sur notre site web. Les paquetages sont *open-source* avec l'intention de fournir à la communauté un système reproductible, afin d'accélérer les comparaisons de résultats. Nous avons également évalué notre système basé sur ROS avec deux véritables robots robuLAB-10 pour l'exploration intérieure. Une vidéo avec ces deux robots est également disponible sur notre site web.

6.3. Algorithmes de fusion de cartes

Nous nous sommes intéressés à l'impact sur la performance d'exploration de deux algorithmes de fusion de cartes respectivement utilisés dans (Yamauchi, 1998) et (Burgard *et al.*, 2000). Le premier (Yamauchi, 1998) (algorithme 3) est un algorithme

glouton qui se concentre sur l'espace inconnu. Le deuxième (Burgard *et al.*, 2000) (algorithme 4) est un algorithme probabiliste dans lequel le robot construit la carte fusionnée en calculant la probabilité d'occupation de chaque cellule de la carte explorée. Ces deux algorithmes nécessitent la connaissance des positions initiales des robots.

Algorithme 3 Algo de Fusion des Cartes Glouton (de [Yamauchi, 1998])

$\delta \leftarrow PoseInitialeRelative(robot_a, robot_b)$
for all $cellule$ in $robot_a.grille\_exploree$ **do**
  if $cellule == ESPACE\_INCONNU$ **then**
    $cellule \leftarrow robot_b.grille\_exploree(coordonnee(cellule) + \delta)$
  end if
end for

Algorithme 4 Algo de Fusion des Cartes Probabiliste (de [Burgard *et al.*, 2000])

$\delta \leftarrow PoseInitialeRelative(robot_a, robot_b)$
for all $cellule$ in $robot_a.grille\_exploree$ **do**
  $Probabilite(cellule) \leftarrow Probabilite(cellule \bullet$
    $robot_b.grille\_exploree(coordonnee(cellule) + \delta))$
  if $cellule \geq LIMITE\_DE\_CONFIANCE$ **then**
    $cellule \leftarrow ESPACE\_OCCUPE$
  else
    $cellule \leftarrow ESPACE\_LIBRE$
  end if
end for

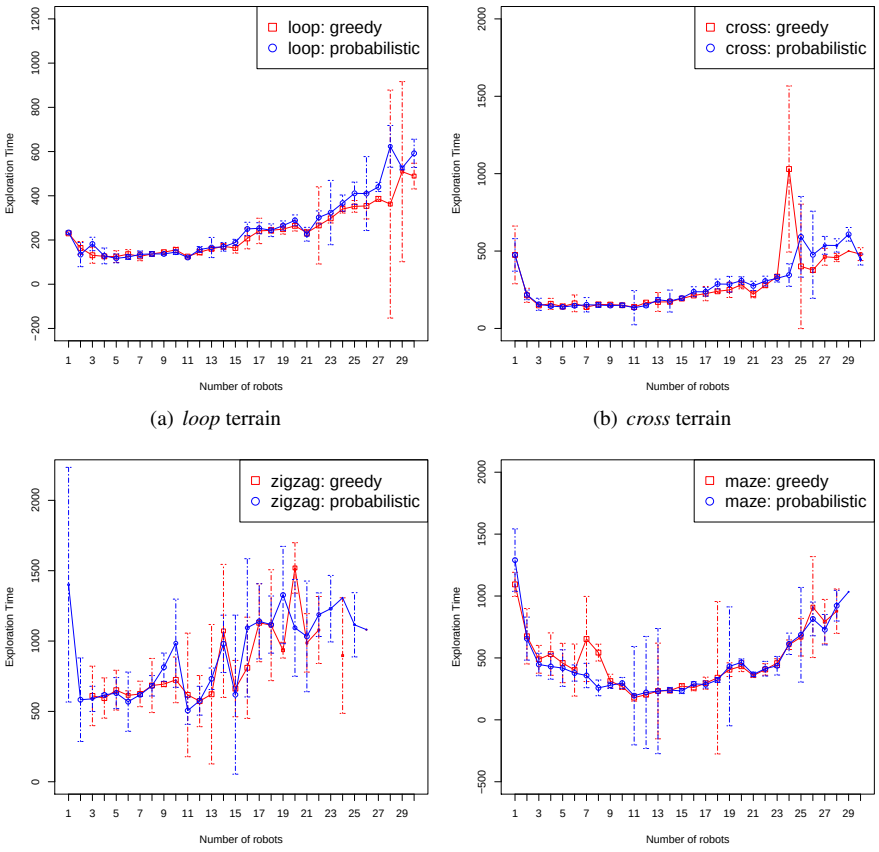
6.4. Résultats

Les résultats sont représentés sur les figures 6, 7, 8 et 9. Chaque figure contient un graphe pour chacun des quatre terrains et une métrique donnée. Dans chaque graphe, l'abscisse représente le nombre de robots dans la flotte, l'ordonnée indique les mesures de métriques, et la barre d'erreur indique l'intervalle de confiance correspondant à chaque mesure avec un niveau de confiance de 0.95. Le carré rouge représente l'algorithme glouton, et le cercle bleu représente l'algorithme probabiliste.

La valeur moyenne n'est pas affichée si les cinq essais ont échoué. Ces situations se produisent par exemple lorsque la taille de la flotte est supérieure à 26 robots dans le terrain *zigzag*. L'échec peut provenir de deux causes :

– l'utilisation de contrôleurs de robots basés sur ROS : le système est basé sur une architecture distribuée encore en développement pour la partie multirobot. La perte de localisation de robot et l'échec de planification de chemin de longue distance sont occasionnellement apparus dans nos expériences.

– l'incertitude de trafic réseau : comme mentionné précédemment, la grappe de serveurs est partagée. Lorsque l'expérience est exécutée pendant les périodes de pointe



(a) *loop* terrain

(b) *cross* terrain

(c) *zigzag* terrain

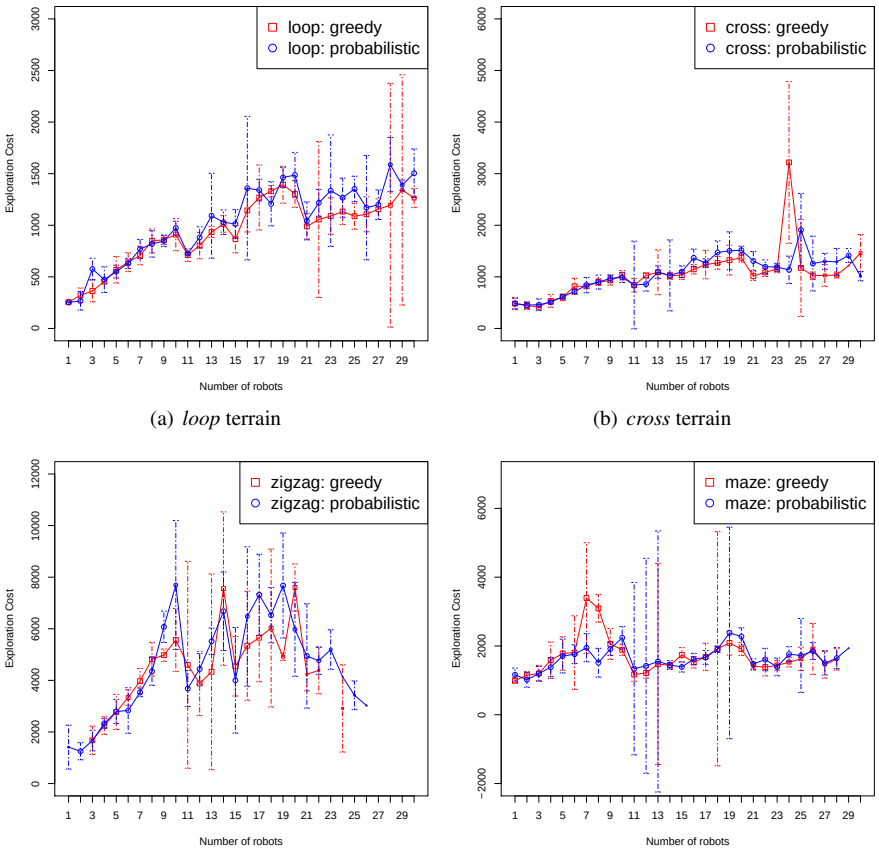
(d) *maze* terrain

Figure 6. Résultats avec le métrique temps d'exploration

du réseau, un essai peut échouer, ou déboucher sur des valeurs aberrantes (*cf.* 24 robots dans le terrain *cross* mettant en œuvre l'algorithme glouton en termes de *temps d'exploration*, *coût d'exploration*, et *efficacité d'exploration*).

Les figures montrent que les différences de résultats entre l'algorithme glouton et l'algorithme probabiliste ne sont pas significatives en termes de *temps d'exploration*, *coût d'exploration*, et *efficacité d'exploration*.

Dans un premier temps, le *temps d'exploration* décroit quand le nombre de robots augmente, et ce jusqu'à un optimum. Puis il croît lorsque le nombre de robots continue à augmenter. Le *coût d'exploration* augmente en général, et l'*efficacité d'exploration* diminue lorsque le nombre de robots augmente, sauf pour le terrain *maze*. En effet, plus il y a de robots, plus il leur faut du *temps* pour s'éviter i.e. replanifier leur trajectoire plus fréquemment ce qui augmente le *coût d'exploration*.



(a) *loop* terrain

(b) *cross* terrain

(c) *zigzag* terrain

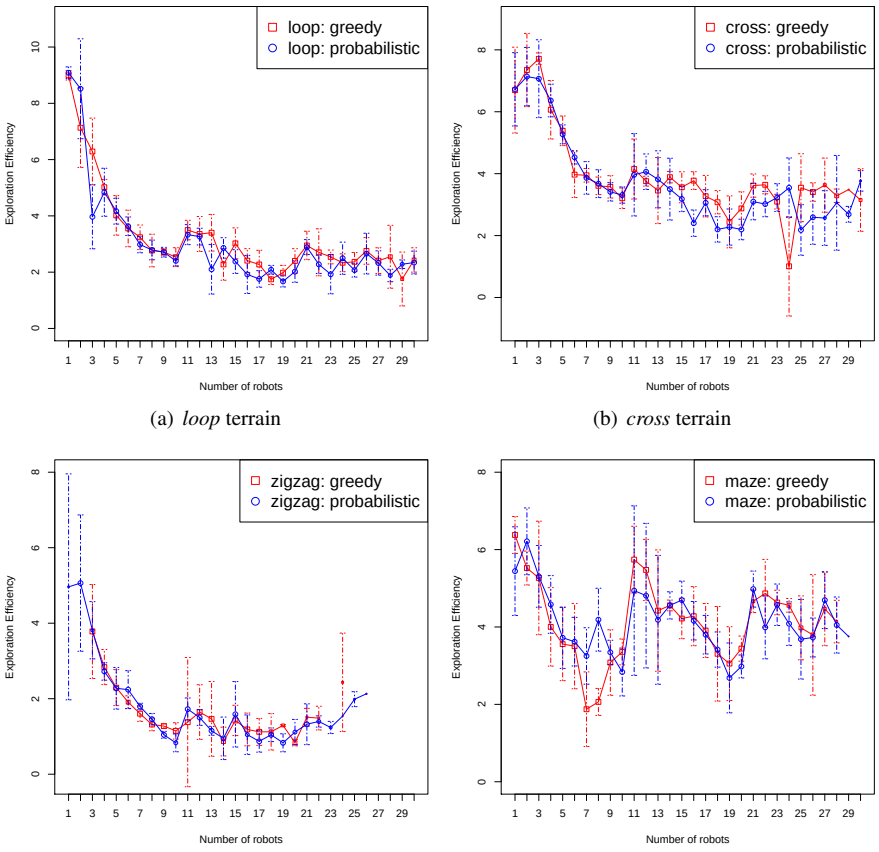
(d) *maze* terrain

Figure 7. Résultats avec le métrique coût d'exploration

Ces résultats permettent aussi de déterminer la taille optimale de la flotte de robot pour un terrain donné. Par exemple, lors de l'exploration du terrain *maze*, la flotte idéale est de 11 robots, car cette taille minimise le *temps d'exploration* et le *coût d'exploration* tout en assurant une bonne *efficacité d'exploration*.

La figure 9 montre que la *qualité de la carte* est meilleure avec l'algorithme glouton plutôt qu'avec l'algorithme probabiliste alors que ces deux algorithmes n'impactent pas significativement les autres métriques.

En outre, les résultats montrent que, avec le terrain *zigzag*, la position initiale des robots influence largement les mesures de performance. Les simulations ont montré que l'exploration est principalement effectuée par un seul robot. En effet, il n'y a qu'une seule frontière et elle est toujours proche du même robot.

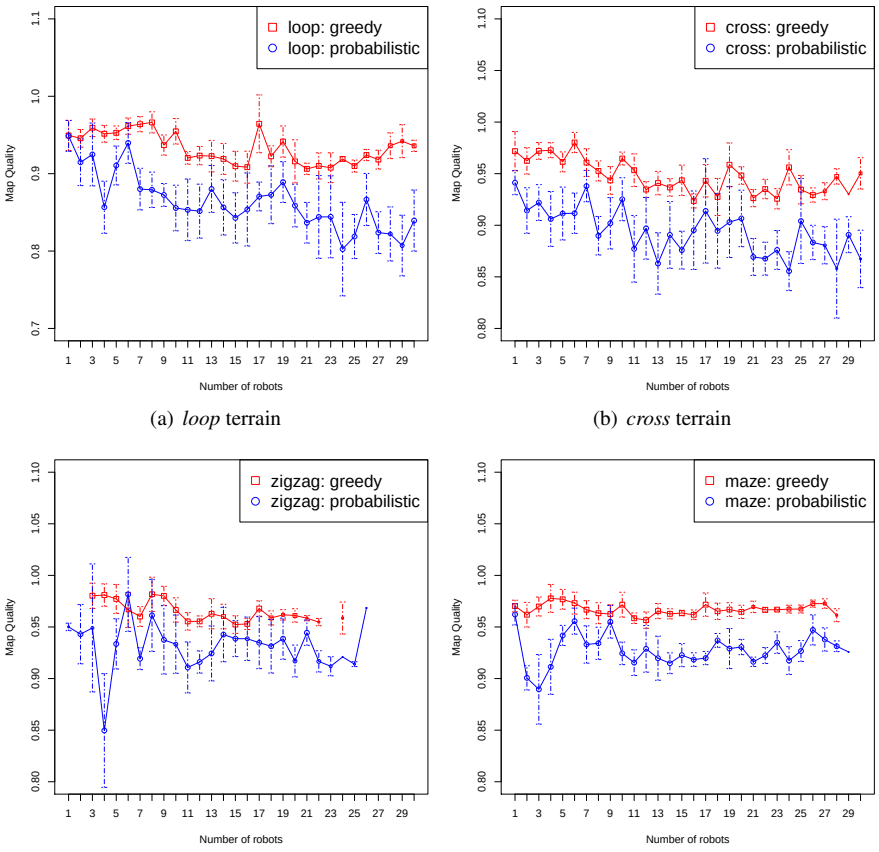


(a) *loop* terrain (b) *cross* terrain

(c) *zigzag* terrain (d) *maze* terrain

Figure 8. Résultats avec la métrique efficacité d'exploration

Basé sur ces résultats expérimentaux, il est clair que la taille de la flotte de robot et la forme de terrain ont un fort impact sur le *temps d'exploration*, le *coût d'exploration* et l'*efficacité d'exploration*, tandis que l'algorithme de fusion de carte a un impact sur la *qualité de carte*. Nous pouvons voir ici que l'algorithme glouton a de meilleurs résultats que l'algorithme probabiliste. Toutefois en pratique, les méthodes probabilistes sont plus adaptées pour faire face aux complexités, aux incertitudes et aux ambiguïtés pour le problème d'exploration et de cartographie multirobot (Thrun *et al.*, 2005). Cela provient selon nous de l'absence de bruit sur les données du laser dans nos simulations car le simulateur MORSE ne le permet pas actuellement. Pour vérifier cette hypothèse, nous projetons d'ajouter du bruit sur les données laser comme nous avons pu le faire pour les données odométriques dans la suite de ce travail.



(a) *loop* terrain (b) *cross* terrain

(c) *zigzag* terrain (d) *maze* terrain

Figure 9. Résultats avec la métrique qualité de carte

7. Conclusion

Dans cet article, nous avons considéré le benchmarking de performance pour l'exploration multirobot. Notre préoccupation est de savoir comment comparer quantitativement différents algorithmes et d'effectuer une évaluation objective sur des paramètres prédéfinis communs. Il n'est pas facile de répondre à cette question en raison de la complexité des systèmes multirobots et des environnements qu'ils explorent.

Pour cela, nous préconisons l'utilisation de plans d'expériences explicitant les paramètres et les métriques à utiliser. Nous avons proposé une liste de paramètres ainsi que cinq métriques permettant de quantifier la performance d'exploration, à savoir : le *temps d'exploration*, le *coût d'exploration*, l'*efficacité d'exploration*, la *complétude de carte*, et la *qualité de carte*. Ces métriques peuvent être utilisées aussi bien en simulation que sur des robots réels grâce à l'utilisation de ROS.

Pour illustrer notre contribution, nous avons comparé deux algorithmes de fusion de cartes lors d'explorations multirobots basées sur les frontières de Yamauchi. Ce benchmarking repose sur des simulations à l'aide du simulateur 3D MORSE et sur des contrôleurs de robot ROS. Nous avons fixé la valeur de la plupart des paramètres et fait varier le nombre de robots et le type de terrain. Les résultats calculés avec les métriques montrent l'impact de ces deux algorithmes sur la performance d'exploration.

La notion de plan d'expérience facilite la répétabilité des expérimentations. Nous pensons que ce concept est un élément clé pour structurer le travail d'une communauté et faciliter les échanges. En effet, une fois que les paramètres et les métriques de référence sont identifiés, il devient possible de définir des plans d'expérience de référence. Ceux-ci constitueraient un référentiel commun utilisé par les chercheurs pour évaluer leurs propositions, facilitant ainsi les comparaisons des différents travaux.

Remerciements

Ce travail fait partie du projet Sucré qui est soutenu par la Région Nord Pas-de-Calais.

Bibliographie

Amigoni F. (2008). Experimental evaluation of some exploration strategies for mobile robots. In *Proceedings of icra'08*, p. 2818–2823.

Bautin A., Simonin O., Charpillet F. (2012). MinPos : A novel frontier allocation algorithm for multi-robot exploration. In *Proceedings of icira'12*, p. 496–508.

Bonsignorio F., Messina E., Pobil A. P. del. (2014, march). Fostering progressin performance evaluation and benchmarking of robotic and automation systems. *Robotics and Automation Magazine*, vol. 21, n° 1, p. 22-25.

Burgard W., Moors M., Fox D., Simmons R. G., Thrun S. (2000, April). Collaborative multi-robot exploration. In *Proc. ICRA'00*, p. 476-481.

Couceiro M. S., Vargas P. A., Rocha R. P., Ferreira N. M. (2014, February). Benchmark of swarm robotics distributed techniques in a search task. *Robotics and Autonomous Systems*, vol. 62, n° 2, p. 200-213.

Doniec A., Bouraqadi N., Defoort M., Le V. T., Stinckwich S. (2009). Distributed constraint reasoning applied to multi-robot exploration. In *Proceedings of ictai'09*, p. 159-166.

Faigl J., Simonin O., Charpillet F. (2014). Comparison of task-allocation algorithms in frontier-based multi-robot exploration. In *Proceedings of eumas'14.*

Frank S., Listmann K., Haumann D., Willert V. (2010). Performance analysis for multi-robot exploration strategies. In *proceedings of simpar'10*, p. 399–410. Springer.

Grisetti G., Stachniss C., Burgard W. (2007). Improved techniques for grid mapping with rao-blackwellized particle filters. *Robotics, IEEE Transactions on*, vol. 23, n° 1, p. 34–46.

Howard A. (2006, December). Multi-robot simultaneous localization and mapping using particle filters. *The International Journal of Robotics Research*, vol. 25, p. 1243-1256.

Lass R. N., Sultanik E. A., Regli W. C. (2009). Metrics for multiagent systems. In *Performance evaluation and benchmarking of intelligent systems*, p. 1–19. Springer.

Le V. T., Bouraqadi N., Stinckwich S., Moraru V., Doniec A. (2009, mai). Making networked robot connectivity-aware. In *Proceedings of icra'09.* Kobe, Japan.

Parker L. E. (2008). Multiple mobile robot systems. *Springer Handbook of Robotics*, p. 921–941.

Pobil A. P. del. (2006). Why do we need benchmarks in robotics research? In *Proceedings of iros'06.* Beijing, China.

Scrapper C., Madhavan R., Lakaemper R., Censi A., Godil A., Wagan A. *et al*. (2009). Quantitative assessment of robot-generated maps. In *Performance evaluation and benchmarking of intelligent systems*, p. 221–248. Springer.

Stachniss C. (2009). *Robotic mapping and exploration*. Springer.

Thrun S., Burgard W., Fox D. (2005). *Probabilistic robotics*. MIT press.

Yamauchi B. (1998). Frontier-based exploration using multiple robots. In *Proceedings of agent'98.*

Yan Z., Fabresse L., Laval J., Bouraqadi N. (2014, October). Team size optimization for multi-robot exploration. In *Proceedings of simpar'14.* Bergamo, Italy.

Yan Z., Jouandeau N., Cherif A. A. (2013, December). A survey and analysis of multi-robot coordination. *International Journal of Advanced Robotic Systems*, vol. 10.

Zlot R., Stentz A., Dias M., Thayer S. (2002). Multi-robot exploration controlled by a market economy. In *Proceedings of icra'02*, vol. 3, p. 3016-3023.