Université Claude Bernard Lyon 1
M2 Intelligence Artificielle

Projet multi-agents

Thomas Ranvier *Nicolas Désilets*

Encadrant
Nadia Kabachi

07 octobre, 2019

Pour ce projet nous avons choisi de considérer un scénario dans lequel un vaisseau se trouve sur une planète inconnue. L'objectif est de localiser et récolter des gisements de pierres précieuses réparties sur la surface de la planète. Pour cela, le vaisseau va déployer une flotte de robots qui se chargera d'explorer la planète, et de récolter les pierres ainsi découvertes.
Notre objectif lors du développement de ce projet fut d'essayer de trouver une solution qui permettent aux robots d'être plus efficient et donc de faire le moins de déplacements possibles avant de ramener toutes les pierres au vaisseau.

Règles :

- Chaque robot ne peut transporter qu'une pierre à la fois, ils doivent donc aller la déposer au vaisseau avant de pouvoir en récolter une nouvelle.
- Les pierres sont présentent sous forme de gisement, comprenant au moins une vingtaine d'éléments.
- Chaque robot possède sa propre représentation du monde, initialement vierge, qu'il met à jour de feux façons : directement en explorant son environnement, ou en combinant sa représentation à celle du vaisseau lorsqu'il y revient.
- Le vaisseau possède une représentation du monde, initialement vierge, qui est mise à jour à chaque fois qu'un robot lui envoie sa propre map.
- Si un robot n'a aucun gisement visible sur sa représentation interne, il explore son environnement.
- Si un robot à détecté plusieurs gisements sur sa représentation interne, il se dirige en priorité vers le plus proche.

Nous avons développé plusieurs variantes de ce système entre lesquelles nous pouvons activer ou non au choix par l'activation de flags :

- *communicationActivated = false.* Les communications entre les différents agents sont désactivées.
- *communicationActivated = true.* Les communications sont activées.

- *localGoalActivated = false.* Les robots explorent leur environnement en se déplaçant aléatoirement.
- *localGoalActivated = true.* Les robots explorent leur environnement en sélectionnant aléatoirement une cellule inconnue pour s'y rendre et ainsi explorer les zones dans cette direction.

- *interBotCommunication = false.* Ce flag n'est effectif que si les communications sont activées. Lorsque les robots reviennent au vaisseau ils lui envoie leur carte interne via les *ACLMessages* de Jade. Le vaisseau va ensuite combiner la carte du vaisseau à sa propre représentation interne. Ensuite le vaisseau envoie la nouvelle carte au robot qui peut ensuite repartir.
- *interBotCommunication = true.* Dans cette version lorsque les robots reviennent au vaisseau ils ne partagent pas leur carte, en revanche dès lors qu'un robot découvre un nouveau gisement il envoie sa carte au vaisseau. Lorsque le vaisseau la reçoit il la retransmet directement à tous les autres robots. Lorsque que les robots reçoivent la carte ils la combine avec leurs représentations personnels. Ainsi, les robots prennent connaissances des nouveaux gisement sans avoir besoin de revenir au vaisseau, ce qui permet d'économiser des mouvements.

1 Structure du projet

Notre projet est constitué des classes suivantes :

- Une classe *World* dont le rôle est d'initialiser l'environnement *Jade* et les différents agents. C'est ici qu'est stockée la version complète est continuellement à jour de la carte du monde, à laquelle les agents n'ont pas accès.
- Une classe *Spaceship* qui représente le vaisseau servant de base.
- Une classe *Bot* qui représente les robots qui explorent et ramènent les pierres.
- Une classe *Utils* qui comporte toutes les fonctions utilitaires.
- Une classe *Renderer* dont le rôle est de permettre une visualisation simultanée du point de vue du monde, du point de vue du vaisseau et du point de vue d'un des robots.

2 Détails technique des différentes variantes du projet

Ce projet a été réalisé selon une architecture BDI. Les croyances, désirs, et intentions des agents varient grandement selon la version du projet.

Chaque version du projet propose différentes approches de la résolution du problème initial, qui seront explicitées dans les parties suivantes. Les différentes approches seront ensuite comparées au sein de différents environnements, pour démontrer leur efficacité.

Dans chacune des approches, les robots n'ont aucune connaissance du monde qui les entoure (croyance initiale nulle), et leur unique désir initial est l'exploration. Les croyances de chacun des agents seront mises à jour au fur et à mesure de leur exploration (mise à jour de leur carte interne), tandis que leurs désirs et intentions varient selon la version du projet.

2.1 Découverte de l'environnement par les robots

Il a d'abord fallu s'intéresser au problème d'exploration de la planète. La première approche que nous avons développée est très simple, aléatoire, tandis que la seconde est un peu plus complexe et permet d'obtenir de meilleurs résultats.

2.1.1 Découverte aléatoire

Lorsque les robots ne portent pas de pierre et n'en détectent pas, ils errent aléatoirement dans l'espoir de découvrir leur environnement.

Ce déplacement aléatoire est rendu légèrement plus consistent par l'ajout de variables internes qui mémorisent la dernière direction que le robot a suivie. Lors du prochain mouvement, le robot aura environ 1 chance sur 3 d'effectuer à nouveau un mouvement dans cette direction.
Cela permet ainsi d'explorer des zones plus vastes. Dans le cas contraire, les robots auraient tendance à toujours rester proche de leur point de départ.

2.1.2 Découverte basée sur des objectifs locaux

Dans cette version, lorsque les robots ne transportent pas de pierre et n'en détectent pas, ils désignent une cellule inconnue aléatoire en tant qu'objectif intermédiaire et s'y rendent.
Nous avons légèrement modifié l'algorithme A* pour que les robots ne considèrent pas les cellules inconnues comme des obstacles (de poids infini) mais comme des cellules ayant un poids plus faible que les cellules vides. Ainsi les robots favorisent l'exploration de zones inconnues pour découvrir leur environnement.

L'algorithme de recherche de chemin A* n'est sensé donner un chemin que si la cible est directement accessible. Or, dans notre cas nous voulons atteindre une cellule inconnue, il faut donc considérer les cellules inconnues comme étant une voie praticable afin d'obtenir un chemin.
Une fois un chemin obtenu le robot va le suivre, il va alors découvrir la carte devant lui. Il continuera à suivre le chemin tant que les cellules sur lesquelles il se rend ne sont pas des obstacles et il s'arrêtera dès qu'il aura atteint son objectif ou un mur. Lorsqu'il s'arrête il va analyser sa représentation interne, s'il y détecte un gisement de pierre non vide il s'y rends, sinon il sélectionne un nouvel objectif local.

2.2 Communications entre les agents

La communication permet d'introduire un aspect social à notre système, formalisant ainsi son caractère multi-agents.

Nous avons développé deux façons de communiquer, une restrictive, et l'autre plus permissive. Ces deux modèles permettent d'envisager d'autres contraintes liées à l'environnement, qui sont peu intéressantes dans le cadre de ce projet, mais qui seraient cruciales dans le cas d'une application réelle.

2.2.1 Partage de la carte lors du retour au vaisseau

La première modélisation, restrictive, permet une communication différée entre les robots. Ceux-ci ne peuvent pas communiquer entre eux, et peuvent seulement communiquer avec le vaisseau pour mettre à jour leur carte interne.

Lorsqu'un robot rentre au vaisseau après avoir trouvé une pierre, il va fusionner sa carte interne avec celle du vaisseau. Les autres robots pourront alors avoir accès une version plus complète de la carte lorsqu'il rentreront, à leur tour, au vaisseau.

Ce modèle permet d'envisager des contraintes liées à la communication en milieu hostile, car on suppose qu'il n'est pas toujours possible de communiquer librement avec les autres agents, qui sont possiblement très éloignés les uns des autres.

2.2.2 Partage de la carte lors de la découverte d'un nouveau gisement

Le seconde modélisation est beaucoup plus permissive. Elle permet aux robots de communiquer leur représentation interne dès lors qu'ils trouve un nouveau gisement de pierres.

Lorsqu'un robot découvre un nouveau gisement de pierre il va envoyer sa carte au vaisseau. Le vaisseau va ensuite pouvoir envoyer cette carte à tous les autres robots. Lorsque les robots reçoivent une carte, ils la combinent avec la leur et peuvent ainsi aider à l'exploitation de ce nouveau gisement.

Ce modèle permet une exploration et une collecte des pierres plus efficace en terme de déplacements totaux des robots.

3 Analyse des performances

Pour cette partie, nous allons faire varier plusieurs paramètres pour mesurer l'efficacité de notre simulation, et comparer les résultats de nos deux versions.
Les paramètres modifiables seront les suivants :

- La taille de l'environnement. La planète est représentée sous la forme d'une grille de dimension W*H.
- La fréquence des gisements et des obstacles.
- La taille des gisements.

– Le nombre de robots.

– L'activation ou non des communications.

– L'activation ou non de la découverte par des objectifs locaux.

– La communication lors du retour au vaisseau ou lors de la découverte d'un nouveau gisement.

Pour comparer les différentes versions nous allons nous baser sur le nombre de déplacements total nécessaires aux robots pour ramener toutes les pierres au vaisseau. En effet, il est impossible de comparer le temps d'exécution de chaque versions puisque les communications des robots demandent un certain temps. Les versions sans communications seraient donc avantagées.

Pour une comparaison ayant plus de sens, tous les essais prennent place dans une carte de dimensions 40x40 au sein de laquelle sont répartis 9 gisements de pierres. Tous les obstacles et pierres sont toujours placés aux mêmes emplacements afin de s'assurer qu'aucune variante ne soit avantagée par rapport aux autres. Nous avons récoltés les données de 10 essais pour chaque configuration.

3.1 Données pour 10 robots

Nous allons ici varier les 3 paramètres suivant :

– Communication activées (com=1) ou non (com=0).

– Exploration aléatoire (loc=0) ou par objectifs locaux (loc=1).

– Communication lors du retour au vaisseau (int=0) ou lors de la découverte d'un nouveau gisement (int=1).

Dans un premier temps nous pouvons comparer les résultats avec une exploration aléatoire sur les boîtes à moustaches ci dessous.

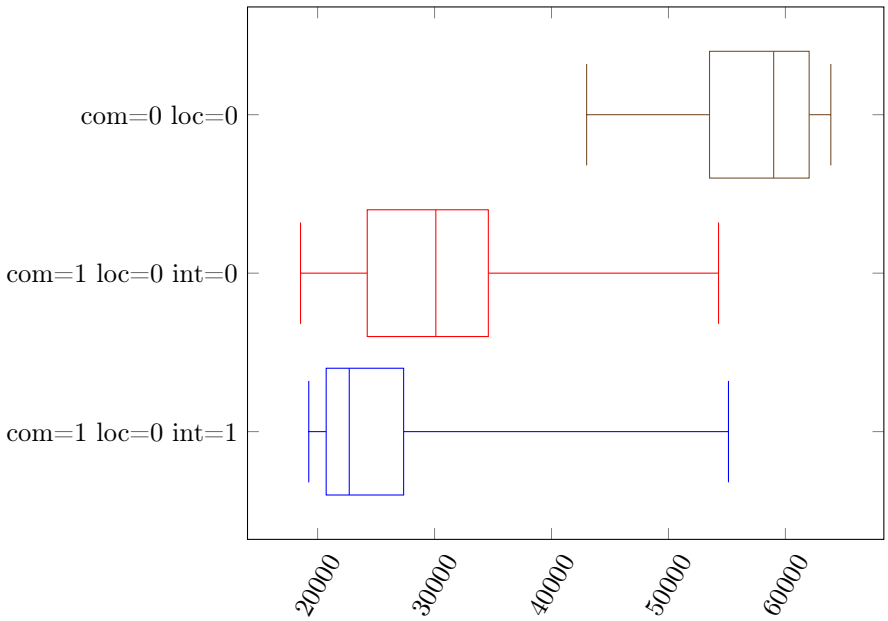

Figure 1: Boîtes à moustache représentant le nombre total de mouvements des robots

On peut voir que sans communications activées les robots nécessitent environ 2 à 3 fois plus de mouvements pour ramener la totalité des pierres au vaisseau.
La médiane est de 58996.5 mouvements lorsque les communications ne sont pas activées, de 30108 lorsque les robots partagent leur carte à chaque retour au vaisseau et de 22706 lorsqu'ils partagent leur carte dès qu'ils trouvent un nouveau gisement.
Ces résultats tendent à montrer que la seconde voie de communication serait plus efficiente.

Nous pouvons maintenant comparer les résultats avec une exploration basée sur des objectifs locaux.

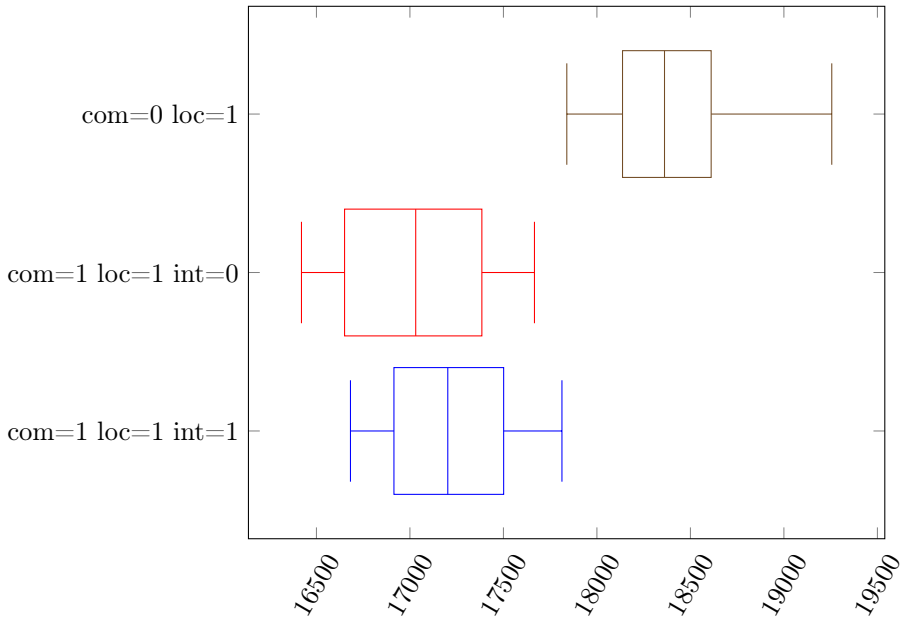

Figure 2: Boîtes à moustache représentant le nombre total de mouvements des robots

la première chose à noter est que l'ordre de grandeur des mouvements effectués est bien plus bas qu'avec une exploration aléatoire. On peut voir que même sans communications les robots sont ici capable de ramener toutes les pierres au vaisseau avec une médiane de 18361.5 mouvements, ce qui est bien inférieur aux meilleurs scores des configurations précédentes.
On peut donc d'ores et déjà conclure que l'exploration de l'environnement par objectif locaux est une fonctionnalité qui améliore grandement les performances globales de notre système.

En revanche, nous pouvons voir qu'avec cette manière d'explorer l'environnement, la méthode de communication la plus efficace semble être celle permettant aux robots de partager leur carte lorsqu'ils retournent au vaisseau.
Nous avons fait les tests à plusieurs reprises afin de s'assurer que ces résultats soient consistants et c'est bien le cas. Pour essayer de mieux appréhender la raison de ces résultats nous avons décidé de tester les mêmes configurations en changeant seulement le nombre de robots.

3.2 Données pour 4 robots

Comme précédemment, la figure ci-dessous montre les résultats pour la méthode de déplacement aléatoire.

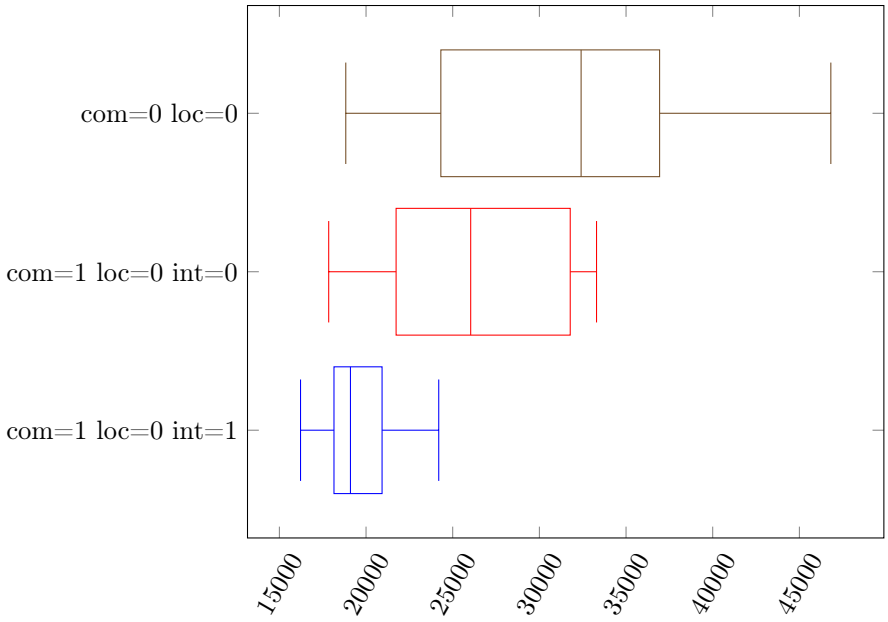

Figure 3: Boîtes à moustache représentant le nombre total de mouvements des robots

On peut voir que le nombre total de mouvements nécessaires est plus bas lorsque le nombre de robots l'est également.
On voit également que, comme précédemment, les deux versions avec communications sont plus efficaces. On peut, en revanche, noter que le résultat de ces deux dernières ne sont pas 2 à 3 fois plus bas comme précédemment.
Cela peut s'expliquer simplement par le fait que si le nombre de robots au sein du système diminue alors l'impact des communications diminue également.

Nous allons maintenant analyser les résultats avec l'exploration basée sur objectifs locaux.

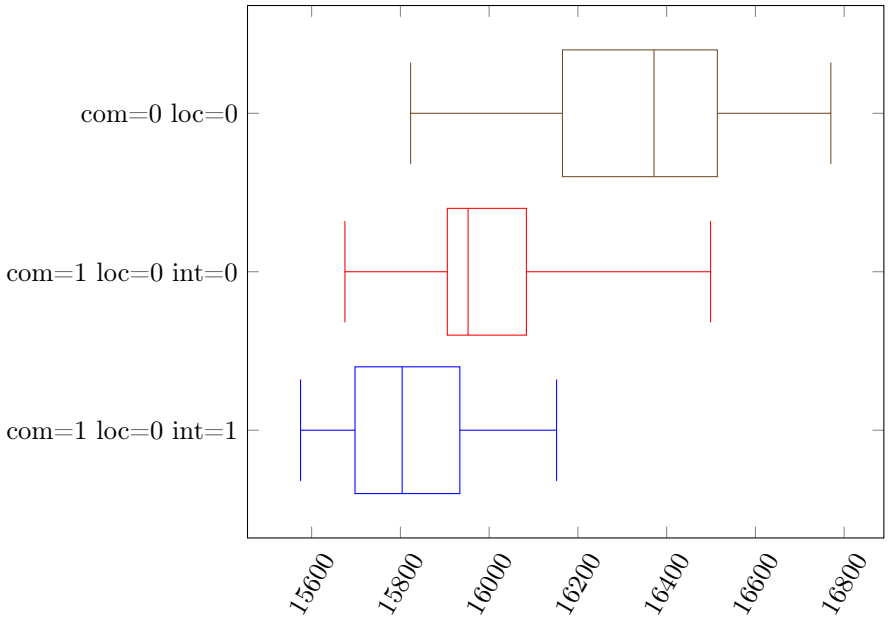

Figure 4: Boîtes à moustache représentant nombre total de mouvements des robots

On peut à nouveau noter l'efficacité de cette méthode d'exploration par rapport à l'aléatoire.

On voit que cette fois la technique de communication la plus efficace est la seconde, il semble donc que la seconde méthode soit plus adaptée à une population de robots faible.
Il s'avère qu'après plus de tests nous avons pu déterminé que plus le nombre de robots augmente et moins la seconde méthode de communication n'est efficace en comparaison à la première, ce qui confirme cette théorie.

4 Visualisation

Afin de pouvoir visualiser la découverte de la planète nous avons créé une classe dans son propre thread qui affiche trois cartes :

- La première carte affichée est celle du monde, on y voit en rouge les gisements, en blanc les cellules praticables et en noir les obstacles. Les points verts sont les robots et le point bleu est le vaisseau (souvent masqué par des robots).
- La seconde carte est celle du vaisseau, on y voit l'état actuel de la carte interne du vaisseau. Le code couleur est le même, les cellules grises sont les zones inconnues. Lorsque les communications ne sont pas activées cette carte reste entièrement grise puisqu'elle n'est pas mise à jour.
- La dernière est la carte interne du robot numéro 1, elle permet ainsi de mettre évidence le partage des cartes lorsque les communications sont activées.

Ci-dessous est un exemple de visualisation avec les communications activées.

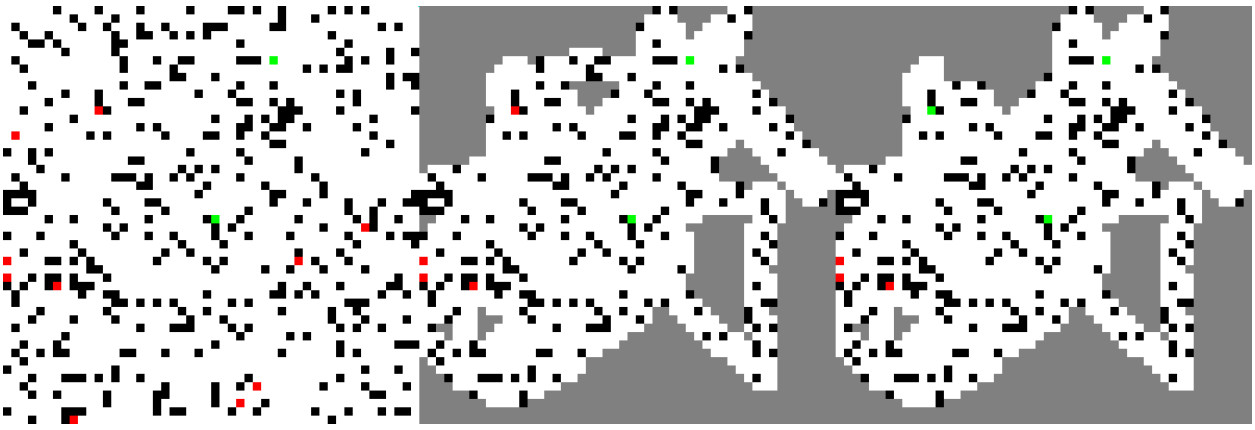

Figure 5: Gauche : carte du monde, milieu : carte du vaisseau, droite : carte d'un robot

5 Difficultés rencontrées

Initialement, les versions avec communications activées ne performaient pas mieux que celles sans communications et nous avons mis du temps avant de réussir à remédier à cette situation.
Cela était principalement dû au fait que notre implémentation d'A* ne considérait pas les cellules inconnues comme praticables, ainsi lorsqu'un robot se rendait vers une pierre ou au vaisseau il n'explorait pas de nouvelles zone. En modifiant A* pour favoriser l'exploration de l'environnement nous avons rendu les versions utilisant les communications bien plus efficaces que les autres.

6 Exécution du code

Pour exécuter notre code vous pouvez utiliser un IDE pour avoir accès au code source et pouvoir modifier les paramètres à votre guise, ces derniers se trouvent au sein de la classe main.
Sinon vous pouvez lancer l'archive *mas_project.jar* comme suit :

```
> java -jar mas_project.jar x y z
```

Où « x », « y » et « z » sont des paramètres obligatoires :

- X à 0 les communications sont désactivées, à 1 elles sont activées.
- Y à 0 l'exploration est aléatoire, à 1 l'exploration utilise des objectifs locaux.
- Z à 0 les communications ont lieus lors des retours des robots au vaisseau, à 1 elles ont lieu lors de la découverte d'un nouveau gisement.

Dans cette version la carte est de dimensions 50x50, elle change à chaque exécution et le taux de gisements est de 8%. Lorsque l'exécution est terminée, le nombre total de mouvements des robots est affiché avant le message « The end ».