Université Claude Bernard Lyon 1

Multi-Agents and Self-* Systems
TP

Architecture BDI

Élèves :
Nathan Etourneau
Paul Flagel

Enseignant :
Nadia Kabachi

20 novembre 2021

Table des matières

Introduction

L'objectif de ce TP est de construire un système multi-agents implémentés suivant une architecture BDI (Belief Desire Intention). Dans notre cas, les agents sont des robots dont l'objectif est de rapporter des pierres rares localisées sur une planète distante. Nous ne savons pas où se trouvent ces pierres, mais nous savons qu'elles y sont en tas. Les robots n'ont pas de carte de cette planète et en raison du terrain très accidenté, ils ne pourront pas communiquer directement entre eux. Enfin, ils devront être capables de retourner à leur base pendant et après avoir accompli leur mission.

1 Scénario et choix techniques

Comme expliqué précédemment, nous avons choisi le scénario suivant : des robots sont situés sur la planète Mars, et cherchent à ramasser des roches situées dans des gisements placés aléatoirement sur la planète.

Ces robots possèdent une batterie limitée qui décroît avec le temps suivant une certaine probabilité, ainsi que des capacités de communication faible. En outre, le relief de la planète est accidenté, ce qui altère considérablement le champ de vision du robot.

Enfin, la base des robots est en mesure d'émettre un très puissant signal, que ces derniers sont en mesure de capter à tout instant. Les robots ont donc constamment accès à l'information du cap et de la distance les menant à leur base.

En accord avec ce scénario, nous avons défini deux rayons de perception utiles : le premier est le rayon de vision directe, qui est très faible (40 unités). Le second est le rayon de communication, qui est bien plus important (150 unités).

Ce TP devait initialement être réalisé au moyen du logiciel JADE, permettant une communication asynchrone entre les différents agents. Toutefois, n'ayant jamais pratiqué Java, nous avons eu l'aimable autorisation de réaliser ce projet en Python que nous maîtrisons mieux. Nous nous sommes donc inspirés du fonctionnement de JADE, et avons choisi d'implémenter notre simulation de façon multithreadée. C'est à dire que chaque robot évolue dans son propre thread et actualise son état seul, sans être soumis à une boucle itérative et sans avoir besoin que les autres agents aient effectués leur tâche pour fonctionner. Nous avons également implémenté un protocole de communication entre agents leur permettant de partager la position d'un gisement lorsque l'un d'eux en trouve un.

Nous avons implémenté une interface graphique permettant de visualiser l'évolution de la simulation (figure 1). Les robots sont représentés par des carrés dont la couleur varie en fonction du niveau de batterie (vert lorsque la batterie est haute, rouge lorsqu'elle est basse). Les gisements de pierres sont représentés par des cercles de différents rayons (en fonction de la richesse du gisement) qui diminuent à mesure qu'ils sont minés. Enfin, la base est représentée par un carré bleu. Lorsque les robots rapportent des roches à la base, ils sont représentés avec un cercle noir en leur centre. Lorsqu'ils n'ont plus de batterie, ils sont représentés avec une croix noire.

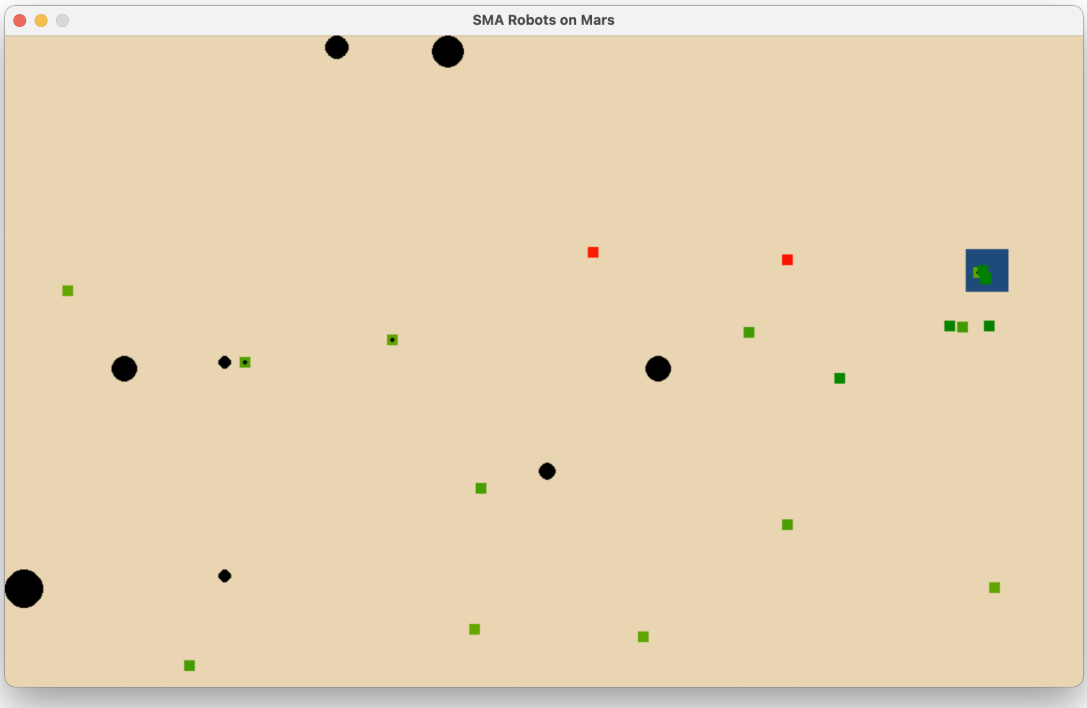

FIGURE 1 – Interface graphique de la simulation

2 Architecture du projet

2.1 Architecture voyelle

L'architecture voyelle du projet est définie de la manière suivante :

- **Agents** : les robots mineurs et la base
- **Environnement** : la surface de la planète, modélisée par un plan en 2D, où sont placés des gisements de différentes tailles. Les robots évoluent de manière continue sur le plan
- **Interactions** : les agents n'ayant jamais connaissance de leur position absolue interagissent surtout avec l'environnement qui leur envoie le cap et la distance vers leur objectif. Les agents communiquent également entre eux lorsque l'un d'entre eux trouvent un gisement de pierres. Les autres peuvent alors choisir de se diriger vers ce gisement.
- **Organisation** : Les agents n'ont pas de hiérarchie entre eux, ils sont tous "ouvriers" et doivent uniquement rapporter le plus de pierres possible à la base, en évitant de tomber en panne.

2.2 Architecture BDI

Les agents ont été implémentés suivant l'architecture BDI dont nous détaillons les choix effectués dans la table suivante.

Robot	
Knowledge	- Niveau de batterie - Position relative à la base
Beliefs	- Gisements de pierres dans son champ de vision - Robots sans batterie dans son champ de vision - Cap et distance à un gisement lorsque avertis par un message d'un autre agent
Desire	1. Rapporter le maximum de pierres à la base 2. Ne pas tomber en panne 3. Aider les robots en panne
Intentions	1. Chercher un gisement 2. Ramasser une pierre 3. Déposer la pierre à la base 4. Si batterie faible : recharger à la base 5. Si robot en panne & batterie suffisante : donner la moitié de sa batterie au robot

3 Scénarios d'action des robots

En fonction des croyances et de la connaissance du robot à l'instant t, son action ne sera pas la même. Les différents scénarios d'action sont listés ci-dessous.

Algorithm 1 Scénario d'action du robot s'il a plus de 10% de batterie

```
search rocks
if rocks in sight then
    go to rock
end if
if arrived to rocks then
    mine rock
    send message to robots around
end if
if rocks mined then
    return to base
end if
if arrived to base then
    release rock
end if
search rocks
```

Algorithm 2 Scénario d'action du robot s'il a moins de 10% de batterie

```
if battery < 10% then
    return to base
end if
if arrived to base then
    charge battery
end if
search rocks
```

Algorithm 3 Scénario d'action du robot s'il trouve un robot sans batterie

```
if dead robot in sight and battery > 30% then
    go to dead robot
end if
if arrived to dead robot then
    give dead robot half of my battery
end if
search rocks
```

4 Intéractions entre les classes

Pour résoudre notre problème, nous avons choisi d'implémenter plusieurs classes : `Robot`, `Rocks`, `BaseCamp`, `Environment`, `Simulation` hiérarchisées comme suit :

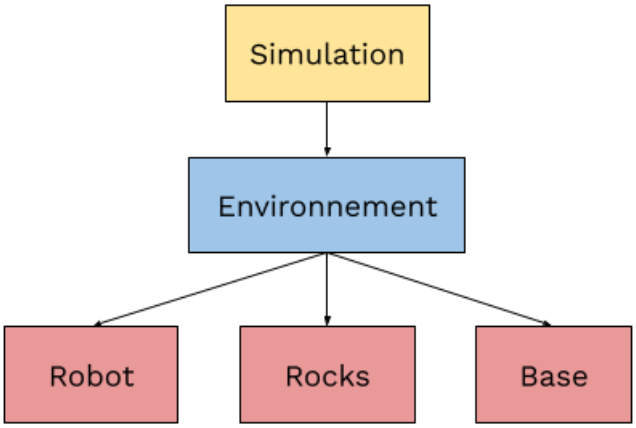

FIGURE 2 – Hiérarchie des classes

A chaque actualisation de leur état, les robot effectuent une `perception` de leur environnement, permettant de réviser leurs croyances. Ils appellent ensuite leur méthode `option` où ils actualisent leurs désirs en fonction de leurs nouvelles croyances, et sélectionnent leur intention. Puis une méthode `update` est appelée où les robots agissent : la vitesse et les attributs du robot sont actualisés.

Les robots n'ayant jamais conscience de leur position absolue, c'est l'environnement qui leur donne le cap et la distance jusqu'au gisement ou au robot en détresse à portée. De même, c'est l'environnement qui gère les échanges de messages entre les robots.

A ce sujet, lorsqu'un robot trouve un gisement, il envoie un message aux robots dans son champ de communication qui calculent par des formules trigonométriques (reliant leur position à la base et la position du robot en train de miner à la base). Ces robots choisissent alors de se diriger vers le gisement s'ils n'ont pas d'autre gisement à proximité ou de robot en détresse à aider.

Conclusion

Ce TP nous a donc permis d'implémenter un système multi-agents en suivant une approche BDI. Ce type de structure d'agent intelligent permet d'émerger une intelligence et une coopération collectives permettant à chaque agent de s'entraider tout en accomplissant ses désirs. Nous pourrions imaginer ce type d'architecture appliqué à une flotte de voitures autonomes qui partagent chacune le réseau routier et qui doivent aller à des endroits distincts.