GASSION Axel INFO-5A
GAUFRETEAU Simon

SMA - TP1

On peut mettre une vidéo de chaque exec en détail (en console)
On ajoute ce rapport dans le git, avec les vidéos et on lui donnera le git

1ère Partie

Pour cette première partie, nous avons implémenté des agents qui ne sont pas en capacité de communiquer entre eux.
Un agent perçoit l'agent du dessus (ou le vide), l'agent du dessous (ou la table), et peut savoir s'il est poussé.

En partant d'une configuration aléatoire, l'objectif est le suivant (peu importe la table) :

4		
3		
2		
1		
Table 1	Table 2	Table 3

Ainsi, à chaque itération de l'algorithme, nous sélectionnons un agent aléatoire.
Dans le cas où notre agent est poussé, ou mal placé (c'est à dire sur le mauvais puisqu'il peut uniquement percevoir ce qu'il y a en dessous et au dessus de lui), nous allons procéder comme suit :
Si cet agent peut se déplacer (c'est à dire qu'il n'a rien au-dessus), il va être déplacé par l'Environnement sur une autre pile aléatoire.
S'il ne peut pas, il va essayer de pousser l'Agent au-dessus.
S'il n'est pas poussé et bien placé, alors il ne se déplacera pas.

Comme il n'y a pas de communication, chaque agent va tenter de bien se placer sans savoir réellement si son déplacement est bon, ce qui peut amener à un très grand nombre d'itérations.

Résultats

Un paramètre important dans cette partie est le nombre d'itérations maximum. En effet l'aléatoire va souvent amener les agents à boucler sur des positions pendant de nombreuses étapes. Nous avons donc effectué quelques tests pour voir la réussite de cette stratégie sur plusieurs nombre d'itération maximum.
Sur 10 000 environnements aléatoires, nous avons les résultats suivants :

Itérations max	Réussi	Echec	Part de réussi	Moyenne d'itérations
1000	6687	3313	66.87%	244
2000	8660	1340	86.6%	522
3000	9441	559	94.41%	694
4000	9761	239	97.61%	810
5000	9876	124	98.76%	881
7500	9987	13	99.87%	933
10000	9998	2	99.98%	954

2è Partie

Pour rendre la planification plus performante, nous avons choisi une nouvelle stratégie basée sur la communication.
Comparé à la première partie, les agents peuvent désormais s'échanger des messages (que nous avons simplifié dans le code par des appels de fonctions) qui leur permettent d'avoir des informations supplémentaires sur l'environnement et de prendre des décisions plus optimales.

Ainsi, nous avons ajouté deux principaux axes pour améliorer le système :

- La destination de l'agent qui se déplace n'est plus totalement aléatoire.
- L'agent qui se déplace n'est plus sélectionné aléatoirement.

Avant de discuter de la manière dont nous avons implémenté chacun avec des exemples, résumons notre nouvelle stratégie dans son ensemble

1. Stratégie globale

Choix de l'agent à déplacer :

- Mise à jour des perceptions des agents
- Demander à chaque agent si le déplacer sera optimal
- Si aucun agent ne répond à l'étape précédente, choisir un agent aléatoirement

Est-ce que mon déplacement sera optimal ?

- Suis-je sur ma cible ?
- Est-ce que ma pile est correcte ? (i.e. tous les agents sont-ils sur leur cible ?)

Choix de la destination :

- Demander aux autres agents s' ils sont sur le dessus d'une pile
- Parmi ces agents, si il y a ma cible, le choisir comme destination
- Sinon prendre un agent aléatoire (autre que moi-même) comme destination

2. Destination

Comme les agents communiquent entre eux, il est, à chaque itération, maintenant possible pour la conscience collective d'avoir une idée de l'organisation dans l'espace (au moins partielle).
En effet, à chaque tour, nous connaissons ce qui est au-dessus et en dessous de chaque agent.

Prenons la situation suivante, où 1 va se déplacer

1		
3		
2		4
Table 1	Table 2	Table 3

1 sait qu'il peut se déplacer.
Il doit aller soit sur la table 2, soit s'empiler sur 4 (Table 3).
Dans un premier temps, il communique avec les autres agents pour savoir le nombre de piles vides (ici 1) ainsi que les agents sur lesquels il peut s'empiler (ici le 4).

Comme 1 a pour objectif la Table en ce qui concerne ce qui doit être sous lui, il va donc choisir d'aller sur l'emplacement vide (sans réellement savoir sa position mais seulement qu'il sera à la bonne place s'il effectue ce déplacement).

Au final, grâce à la perception de l'environnement collective créée par la communication entre les agents, ces derniers vont pouvoir savoir si un déplacement vers leur agent cible est possible, et le faire si c'est le cas.

Comme on a pu le voir dans la présentation générale, si on ne trouve aucun agent cible parmi les têtes de piles, on prend un agent aléatoire comme destination.
Pour reprendre l'exemple précédent, si la tête de pile n'était pas 1 mais 3, il choisira aléatoirement soit 4 soit la pile vide comme destination.

3. Choix de l'agent

Auparavant, un agent était choisi aléatoirement et se déplaçait s'il était mal positionné (ou alors il poussait s'il était impossible de se mouvoir).

Maintenant, on peut à la fois connaître tous les agents qui peuvent se déplacer, mais aussi savoir exactement si chaque agent est bien placé :

3		
2	4	1
Table 1	Table 2	Table 3

Dans la situation ci-dessus, 3, 4 et 1 peuvent se déplacer.
On regarde en premier la situation de 3.
Si on considère qu'il a uniquement accès à ses informations, il est bien placé et ne va donc pas se déplacer.

Cependant, il a accès aux informations des autres agents qui communiquent avec lui et sait donc que 2 est sur la Table, et sous 3.
2 est donc mal placé, ce qui fait que 3 est aussi mal placé, et va donc se déplacer.

Résultats

Sur 10 millions d'environnements aléatoires, nous avons en moyenne **11.2 étapes** pour avoir un tri correct avec la méthode que nous avons proposée.

Ce résultat est tout à fait correct pour une méthode qui ne comprend que très peu de règles mais peut être améliorée via différentes optimisations. Si nous reprenons l'exemple de la partie 3 sur le choix de l'agent, deux déplacements sont possibles pour l'agent 3 :

- Se déplacer sur 4
- Se déplacer sur 1

Dans notre cas, nous avons choisi de conserver une méthode la plus simple possible pour montrer qu'il est possible d'avoir de bon résultats sans avoir de règles très complexes mais un humain remarque facilement que le choix aléatoire n'est pas optimal ici.
En effet le 3 se situe sur le 2 qui lui doit aller sur le 1. Si le 3 se place sur le 1, il "bloquera" la position pendant quelques étapes alors que s'il se déplaçait sur le 4, le 2 pourrait se déplacer sur le 1 et le tri sera fini en moins d'étapes.