Corre Valentin CORV03119201
Amamou Houssem (AMAH10029004)

8INF844
–
Systèmes multi-agents
-
TP 1

3 Février 2015

Dans ce rapport, nous allons décrire l'architecture utilisée afin d'implémenter notre système multi-agent. Nous allons aussi décrire les actions des agents ainsi que leurs interactions avec leur environnement afin de réaliser leurs buts.

L'application décrite dans ce rapport comporte trois classes. Ces classes permettent de modéliser les différents éléments composants le système boursier.

En premier lieu, la classe « Template » permet de modéliser un agent. Notre système va contenir trois agents autonomes, ces agents vont avoir des buts a atteindre. Les buts de chaque agent sont modéliser par les variables « pref1, pref2, pref3 » qui représentent les indices pour lesquels l'agent va battre avec les autres agents afin d'essayer d'augmenter ses indices. L'augmentation des indices voulu ce fait de façon aléatoire avec la fonction « randAddPoint ». La mesure de performance de chaque agent est représenté par la valeur de l'indice que l'agent doit maximiser. Lorsqu'un agent veut modifier la valeur d'un indice, cet indice doit en premier lieu être libre, c'est à dire qu'aucun autre agent n'est en train de le modifier. En le verrouillant, il s'assure qu'il est le seul à pouvoir le modifier. Les indices que l'agent va sélectionner sont choisis de façon aléatoire grâce à la méthode « preference ». Chaque instanciation d un agent va créer un thread. C'est dans ce thread que l agent va modifier les valeurs des indices qui l intéresse.
La communication entre les agents ne se fait pas directement mais à travers les annonces faites dans l'environnement. C'est à dire que chaque modification d'un indice vers le haut ou le bas va créer une annonce qui va être entendu par tous les agents qui écoutent l'environnement à ce moment là.
De ce fait, chaque agent à une vue complete de l'environnement et peut à chaque instant avoir une idée sur la valeur d'un indice en particulier.

La deuxième classe qui a été implémenté est la classe « Environnement ». L'environnement que nous avons mis en place est complètement observable par tous les agents, il est déterministe car l'environnement est complètement déterminé par son état courant et l'action de l'agent, il est épisodique car chaque épisode de l'environnement dépend du précédent, il est dynamique car les valeurs des différents indices peuvent changer lors de la délibération. Cette classe va regrouper toutes les propriétés de l'environnement dans lequel les agents vont exécuter leurs actions. Pour cela, les trois agents de notre système ainsi que les différents indices sont présents dans cet environnement. Les indices présents dans l'environnement sont créés en double pour pouvoir comparer les indices entre le temps t et le temps t+1. Ainsi on a une idée sur la fluctuation de chaque indices. Lors de l instanciation d un environnement, un thread est créer et va utiliser la finction « balance ». La fonction « balance » permet de simuler les changement qui peuvent subvenir dans le système, et plus précisément dans les valeurs des indices, dues à des actions externes. Cela peut s'expliquer par le fait que l'augmentation d'un indice peut engendrer la diminution des autres indices. Lorsqu un indice est changé par un agent, les autres indices sont impacté. Ces changement se font de façon aléatoires grâce à la fonction « randSubPoint ». On remarque le cas où deux ou plusieurs agents ont la même préference pour un indice, un seul de ces agents va augmenter cet indice car les autres remarquent que cet indice est entrain de grimper donc ce n'est la peine de l'augmenter.

L'interface graphique a éé implementé en utilisant la bibliothèque JavaFX et a été directment codé dans le main de notre application en utilisant les différents fonctions offertes par cette API