Système Multi-agent

Tri collectif

Marion Vertessen et Dane Badiel

Introduction

Ce rapport se rapporte à un travail pratique sur le tri collectif réalisé dans le cadre de nos études à l'université Lyon 1. Il consiste à détailler la simulation d'un tri collectif avec des agents réactifs comme évoqué dans l'article de J.L Denebourg & al. dans *The Dynacis of Collective sorting Robot-Like Ant and Ant-Like Robot.* Pour cela, on possède deux types d'objets à trier (les objets A et B) qui sont répartis aléatoirement en quantité n_a et n_b sur une grille de taille $N \times M$. Des agents déplacent les objets pour les trier selon les caractéristiques suivantes :

- Ils se déplacent aléatoirement dans 8 directions d'un pas $i \geq 1$.
- Ils prennent un objet avec une probabilité : $P_{prise} = \left(\frac{k^+}{k^+ + f}\right)^2$ avec k^+ une constante.
- Ils déposent un objet avec une probabilité : $P_{dépôt} = \left(\frac{f}{k^- + f}\right)^2$ avec k^- une constante.
- On définit une mémoire à court terme des objets que l'agent a déjà rencontré sur les derniers pas pour calculer la valeur de f dans les formules précédentes. De fait, f correspond à la proportion d'objets de même type dans l'environnement immédiat.

Dans un premier temps, on fixe ces différents paramètres selon la *figure 1* ci-dessous :

Taille de la grille	Pas de déplacement	Nombre d'agents	k^+	k^-	Nombre d'objets de type A	Nombre d'objets de type B	Taille de la mémoire
50×50	$i = 1$	20	0,1	0.3	200	200	10

figure 1 : Tableau des paramètres utilisés pour modéliser le tri

I. Implémentation

A. Structure du projet

Pour définir la problématique présentée ci-avant, on utilise le langage Python ainsi que la librairie PyQt5 afin de générer l'interface graphique. On définit donc un agent comme une classe. Cet agent est ici considéré comme réactif : il scanne son environnement et agit en conséquence. On peut donc modéliser le comportement d'un agent comme décrit ci-après :

- L'agent scanne son environnement pour déterminer si un objet se trouve sur la case et sa position sur la grille.
- Il met à jour sa mémoire avec l'élément se trouvant sur la case.
- S'il tient un objet, il le dépose sur la case avec une probabilité $P_{dépôt}$ si aucun objet n'est présent sur la case. Sinon, il prend l'objet, s'il y en a un, avec une probabilité P_{prise}.
- L'agent envoie son souhait de se déplacer à l'environnement dans une direction aléatoire.

L'environnement a les caractéristiques suivantes :

- Une grille de taille 50×50 sur laquelle se trouve les objets A et les objets B
- La liste des agents et de leur position
- Il donne la parole à un agent à chaque itération. La parole est donnée aléatoirement.

B. Nombre de clusters

Afin de déterminer le nombre de clusters, on utilise l'algorithme K-means. En effet, il s'agit d'un algorithme de clustering très répandu. Il permet d'analyser des données caractérisées afin de les regrouper en clusters. Cependant, cet algorithme prend nécessairement un nombre k en entrée qui détermine en combien de clusters notre jeu de données va être séparé.

Dans notre cas, on ne connaît pas ce nombre et on cherche à le déterminer. On a donc choisi de tester l'algorithme du K-means avec différentes valeurs de k. Ensuite, on calcule la silhouette pour chaque valeur de k et on garde le nombre de clusters associé à la meilleure valeur. La silhouette est une mesure permettant de déterminer la validité d'un clustering, elle utilise la distance moyenne avec des individus du même groupe que lui et la distance moyenne avec les individus des autres groupes.

De plus, on sépare notre jeu de données en deux : les objets A et les objets B puis on construit un modèle sur les deux. On obtient ainsi deux nombres de clusters qu'il suffit d'ajouter pour obtenir le nombre de clusters total.

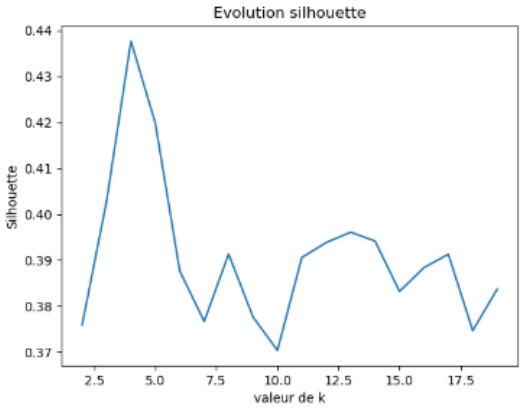

Figure 2 : la silhouette en fonction du nombre cluster k pour les objets A à t = 0 itérations

Sur la *figure 2*, on retrouve la silhouette en fonction du nombre de clusters k. Cette solution comporte deux limites :

- La figure 2 a été tracée à t = 0 itérations, on remarque que le nombre de clusters retenu pour l'objet A est de 4. Cependant, à t = 0, les objets ont été placés aléatoirement sur la grille. On peut donc considérer que le nombre de clusters est beaucoup plus élevé que 4. Cette erreur est causée par l'organisation chaotique des objets au début de la simulation.
- Si on s'intéresse au cas où les objets sont répartis en deux groupes, on remarque que le nombre de clusters déterminé par l'algorithme est de 4. En effet, la méthode k-means détermine les clusters pour $k \geq 2$. On ne peut donc pas tester les valeurs pour un cluster de 1. En effet, cela poserait des problèmes d'évaluation, car si on essaye de positionner tous les points dans un seul cluster on aura nécessairement un score élevé. Cependant, en réalisant un nombre d'itérations raisonnable en temps, on remarque que l'on arrive le plus souvent à deux clusters par objet.

Ainsi, on remarque que ce système possède des limites pour des grandes valeurs de clusters et pour de très petites. Cependant, il reste utilisable dans notre cas.

C. Simulation

On réalise une simulation qui affiche l'état de la grille à $t = 1\,000\,000$, $t = 2\,000\,000$, $t = 3\,000\,000$, $t = 5\,000\,000$, $t = 10\,000\,000$, $t = 15\,000\,000$ et $t = 20\,000\,000$, avec t le nombre d'itérations. Comme on peut le voir sur les figures 3,4,5,6,7,8,9 et 10, on observe une modification de son état.

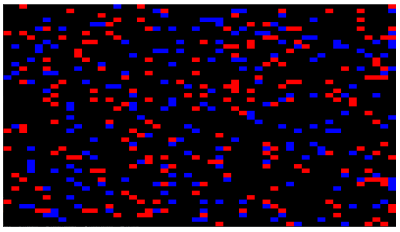

figure 3 : 0 itération

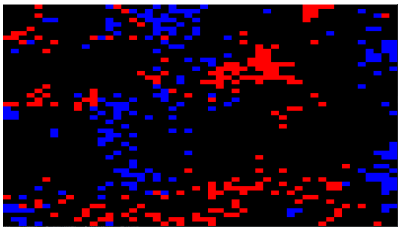

figure 4 : 1 000 000 itérations

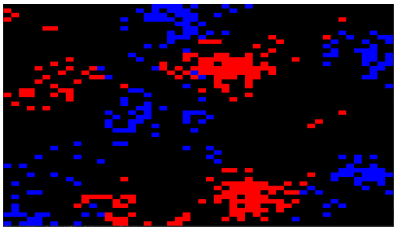

figure 5 : 2 000 000 itérations

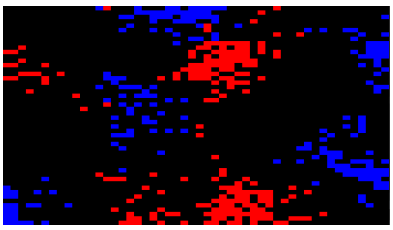

figure 6 : 3 000 000 itérations

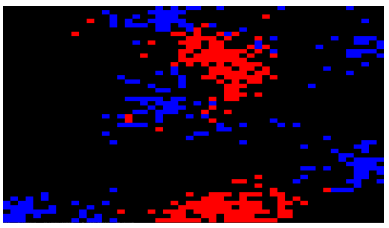

figure 7 : 5 000 000 itérations

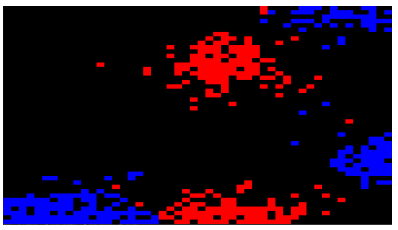

figure 8 : 10 000 000 itérations

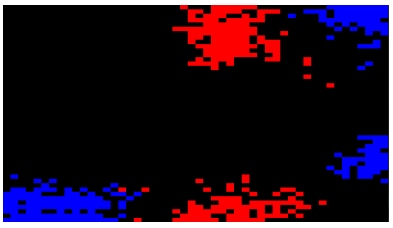

figure 9 : 15 000 000 itérations

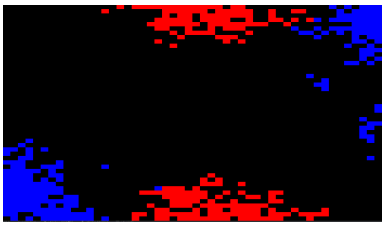

figure 10 : 20 000 000 itérations

De fait, on remarque que le tri commence à vraiment être significatif lorsque l'on atteint les 10 000 000 d'itérations. En effet, on remarque sur la *figure 8* cinq gros clusters d'objets (trois clusters de l'objet A et deux clusters de l'objet B). De fait, on remarque que l'algorithme semble fonctionner. Cependant, on n'obtient pas deux clusters (un de l'objet A et un de l'objet B) comme on pourrait s'y attendre. Cela semble provenir du déplacement aléatoire des agents ainsi que de la position initiale des objets A et B.

De plus, les objets se regroupent sur les bords de la grille. Cela est dû à la limitation de la possibilité de dépôt sur les bords ainsi qu'à la limitation dans le choix de déplacement des agents.

On retrouve sur la figure 11, l'évolution du nombre de clusters en fonction du nombre d'itérations. On calcule le nombre de clusters toutes les 500 000 itérations ce qui correspond à une moyenne de 25000 déplacements pour chaque agent.

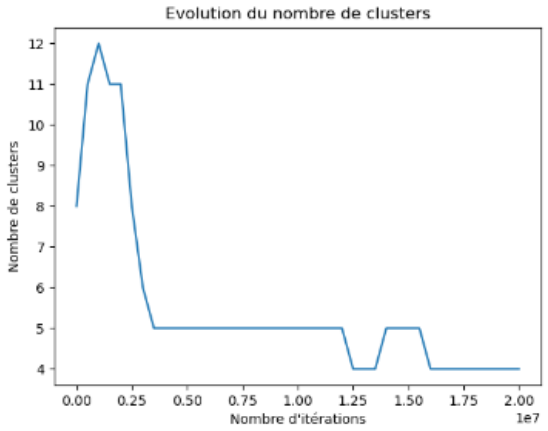

Figure 11 : Nombre de clusters en fonction du nombre d'itérations

On remarque sur cette figure que le nombre de clusters diminue et commence à atteindre au maximum quatre clusters vers 15 000 000 d'itérations, c'est-à-dire 750 000 déplacement de chaque agent. De fait, le modèle est assez long à converger. Cela est principalement dû au déplacement aléatoire des agents et à leur probabilité de prise et dépôt.

Dans la suite, nous allons étudier la modification de certains paramètres et leur impact sur la convergence du modèle.

II. Ajout dun pourcentage d'erreur

Dans cette partie, on s'intéresse à l'impact de l'ajout d'une erreur e dans la reconnaissance des objets. Dans un premier temps, on teste pour $e = 0.3$. De fait, le système a une grande chance de se tromper sur le type d'un objet. On obtient le résultat présent sur le *figure 12 et 13*.

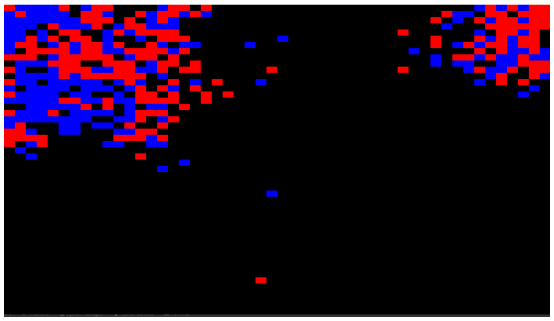

Figure 12 : tri avec $e = 0.3$

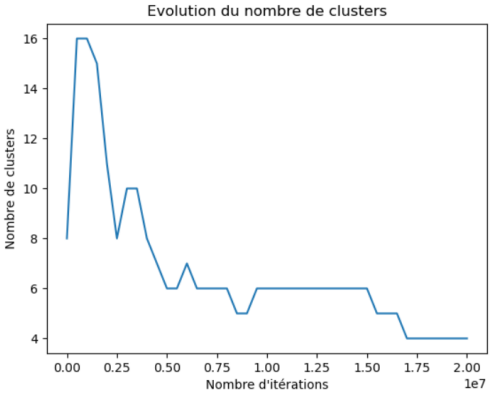

Figure 13 : Evolution du nombre de clusters pour e = 0.3

Sur cette figure, on remarque qu'il n'y a plus de dsicrimination entre les objets A et les objets B. En effet, des clusters sont formés assez rapidement mais mélangent les deux types d'objets. De plus, on remarque sur la *figure 13* que les deux clusters ont été plus lents à se former que sur le modèle utilisé précédemment. On teste maintenant avec une erreur $e = 0.01$ et on obtient les résultats présentés en *figure 13* et en *figure 14*.

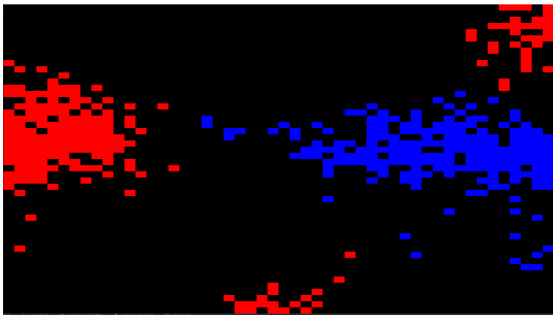

Figure 13 : tri avec $e = 0.01$

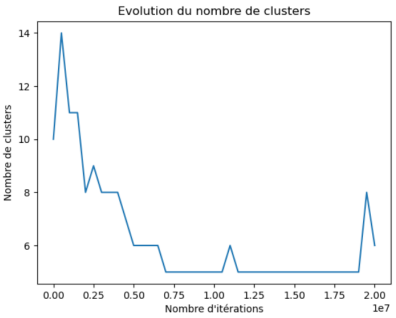

Figure 14 : Evolution du nombre de clusters pour $e = 0.01$

On voit donc que pour un faible pourcentage d'erreurs, on a un modèle qui arrive à former des clusters plutôt efficacement qui rejoint l'étude présentée au I.C.

Les deux valeurs étudiées précédemment peuvent donc être considérées comme les deux valeurs extrêmes de e. L'une donnant des résultats de séparation des objets A et B quasiment nulle et l'autre séparant les données comme si l'erreur n'avait pas été introduite.

Enfin, en réalisant différents tests, on se rend compte que la discrimination des objets A et B restent minime lorsque l'on ne dépasse pas $e = 0.2$. Cette erreur correspond à une erreur de 20% ce qui est assez élevé.

Ainsi, le système semble plutôt bien supporter des perturbations liées à l'erreur de reconnaissance d'objet.

Conclusion

Pour conclure, notre système de tri collectif permet de trier deux types d'objets. On obtient des résultats satisfaisants pour un grand nombre d'itération c'est-à-dire lorsque les agents se sont déplacés 750 000 fois chacun en moyenne. Afin de déterminer la qualité du tri, on s'est focalisé sur l'utilisation de l'algorithme k-means. Ce modèle possède des inconvénients dans notre cas mais cela permet de donner des approximations sur la validité de notre tri.

Enfin, en introduisant une perturbation liée à l'erreur de reconnaissance des objets placés en mémoire, on modifie les probabilités de prise et de dépôt de notre agent. Ces probabilité conditionnent la discrimination des objets lors du tri. On remarque que pour une erreur inférieur à 20%, on obtient des résultats convenables. De fait, notre système semble plutôt robuste face à de telles erreurs.

Afin de déterminer la robustesse de notre système face à d'autre modification, on pourrait également s'intéresser à la modification des valeurs suivantes :

- La valeur du pas de déplacement des agents
- Les constantes k^+ et k^- présentes dans la probabilité de prise et de dépôt
- La taille de la grille
- La vision des agents

Dans une seconde partie de ce projet, nous introduirons des objets C qui ne sont déplaçables que si deux agents collaborent.