SMA - Tri collectif multi-agents

par Antoine Ganne et Alicia Rocchia

Approche voyelle

Nous avons commencé par réaliser l'approche voyelle du projet :

Agents
[As] Agent
[Aa] agent réactif
(règles réflexes : si libre et objet, tentative de prise ; si occupé, tentative de dépôt)

Environnement
[Es] Grille de taille nxm
[Ea] perception : cases libres, objet sur la case?
action : prendre objet, déposer objet, se déplacer

Interactions
[Is] Moyens d'interaction :
- perception/action

Situation d'interaction :
- buts → compatibles
- ressources → suffisantes
- compétences → suffisantes

⇒ Indépendance

Choix

Nous avons fait le choix de travailler sur un monde "rond". Par exemple, pour une grille de taille 5x5, l'agent peut passer de la case (3,5) à la case (3,1) en utilisant la direction E.

Nous avons aussi admis que le voisinage d'un agent correspond aux cases de direction N, O, E et S autour de l'agent. Et nous avons aussi ajouté un paramètre "rayon perception" qui permettrait d'agrandir le voisinage. Par exemple, s'il vaut deux, alors le voisinage de l'agent comprendra 8 cases au lieu de 4.

KPI

On définit deux indicateurs de qualité du tri :
- le nombre moyen d'objets de même type dans le voisinage des objets
- le ratio d'objets isolés (sur le total des objets posés), c'est à dire qui n'ont pas d'objets de même type dans leur voisinage.

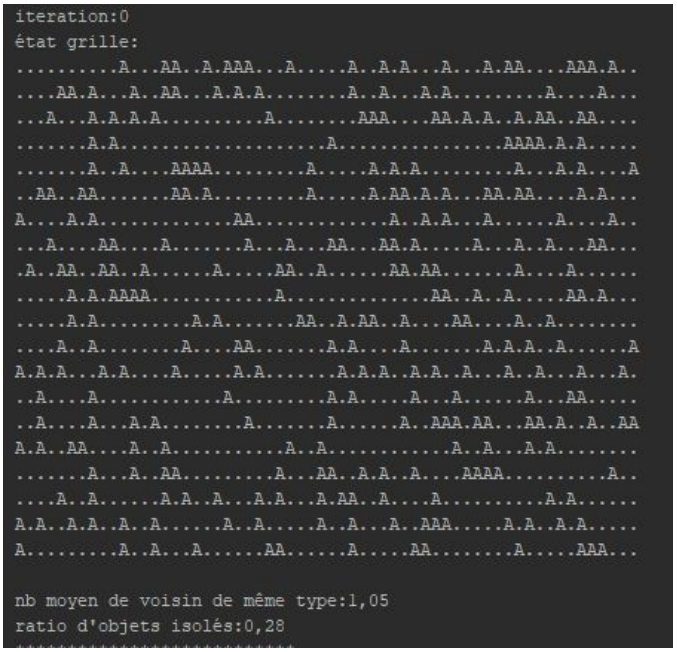
Le voisinage d'un objet est défini tel que celui des agents, c'est à dire que ce sont les 4 cases adjacentes de l'agent (dans les directions N, O, E et S).

Exemple d'exécution

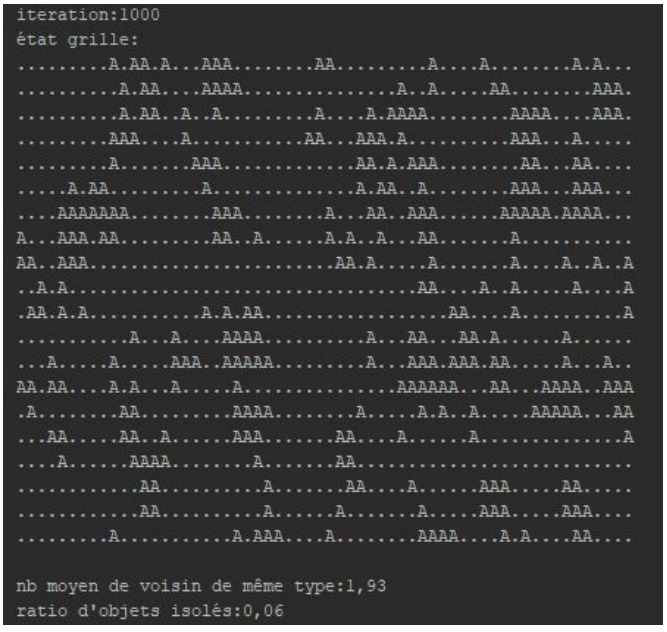
Voici un exemple d’exécution de notre système qui se rapproche de ce qui est montré dans l’article scientifique. On a une grille de taille 20x60, 64 agents, 300 objets de type A (0 objets de type B), k+ à 0.1 et k- à 0.3.

Les agents ne sont pas affichés car cela apporte peu d’information sur le tri et complique la compréhension du visuel.

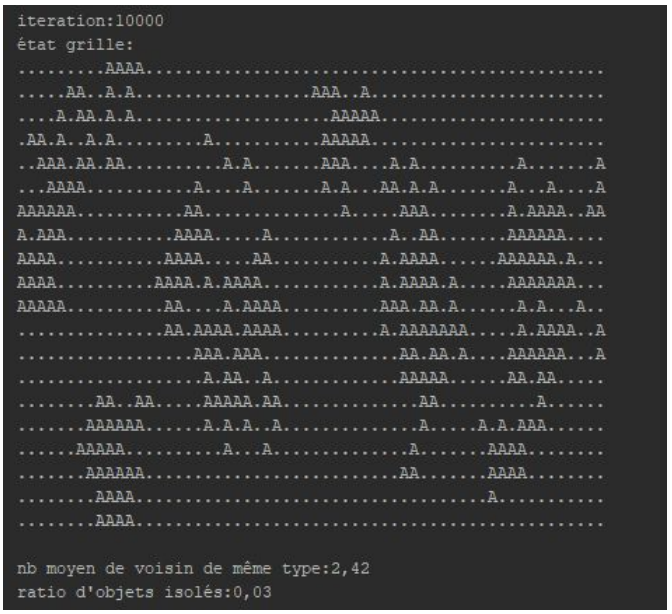
On part d’une répartition aléatoire des objets A.



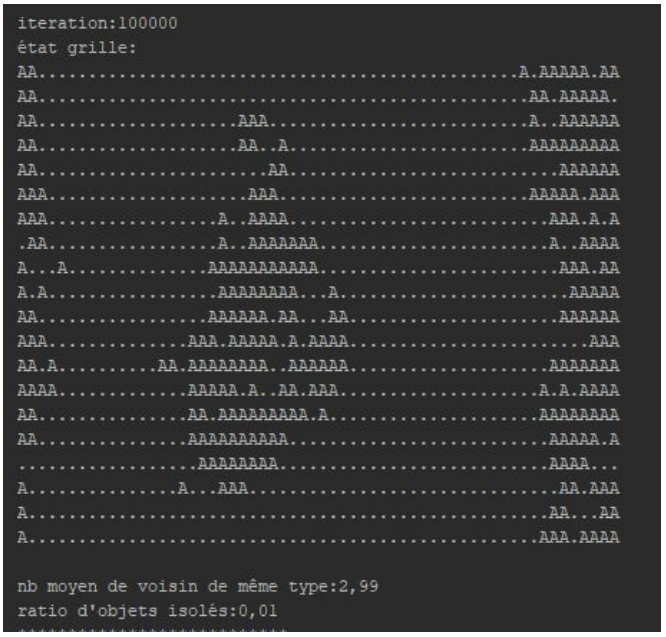
Itération 1000 :



Itération 10 000 :



Itération 100 000 :

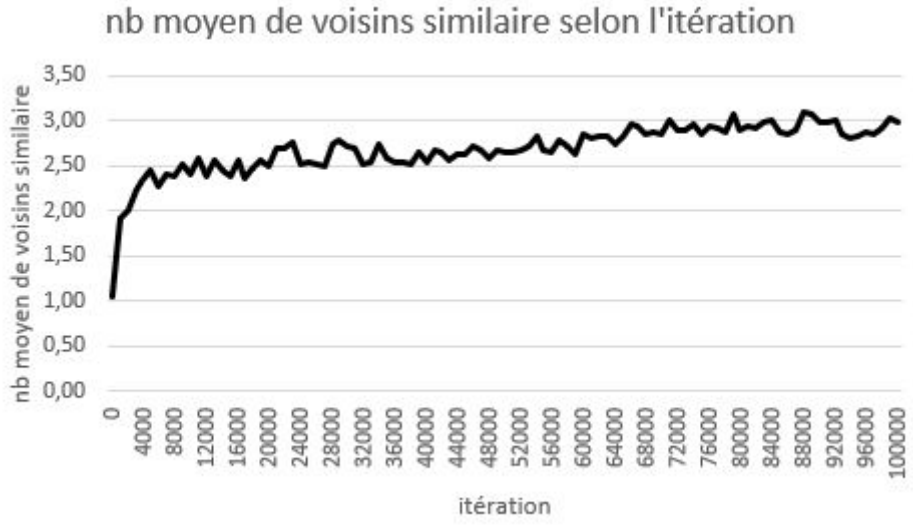


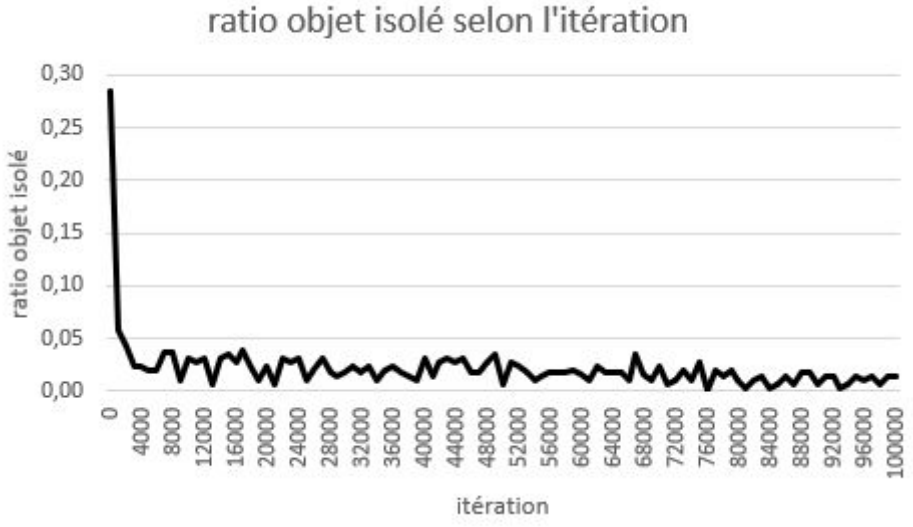
Nous observons bien le comportement montré dans l'article scientifique. En effet, les objets, d'abord répartis aléatoirement sur la grille sont regroupés en tas de plus en plus grand. On rappelle que l'on simule un monde rond et que donc les deux tas à gauche et à droite correspondent à un unique tas.

Les graphiques ci-dessous montrent le nombre moyen de voisins similaires selon l'itération, ainsi que le ratio d'objets isolés selon l'itération.

Nous constatons que le nombre moyen de voisin double vers la 4000ème itération, puis continue à augmenter petit à petit. Ceci montre bien que les objets A sont rapprochés.

De même, le ratio d'objets isolés diminue fortement jusqu'à la 4000ème itération, puis se rapproche de plus en plus de 0. Les objets sont bien triés.





Mise en place de tests unitaires

Nous avons mis en place des tests unitaires afin de vérifier le fonctionnement de la mémoire, de la grille, de la perception, de l'environnement ainsi que du système.
Cela nous permet d'assurer le bon fonctionnement de notre code.

Influence des paramètres

Soient différents paramètres qui influent sur le système :

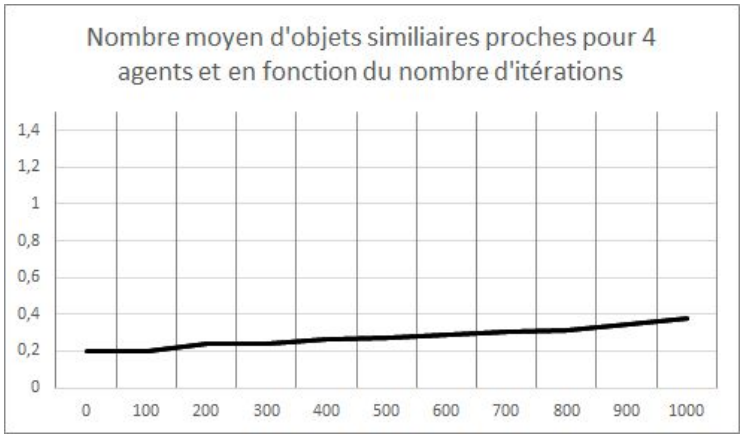
- la taille de la grille : n x m
- le nombre de déplacement d'un agent : i
- le nombre d'agents : nbAgents
- le nombre d'objets de type A : nA
- le nombre d'objets de type B : nB
- la constante servant à calculer la probabilité de prise d'un objet par un agent : k+
- la constante servant à calculer la probabilité de dépôt d'un objet par un agent : k-

Nous allons réaliser quelques tests pour voir à quel point certains de ces paramètres peuvent influencer le système, et afin de décider quelles valeurs idéales il faut choisir pour avoir les meilleurs résultats. Pour cela, nous avons créé des fichiers Excel et fait des graphiques.

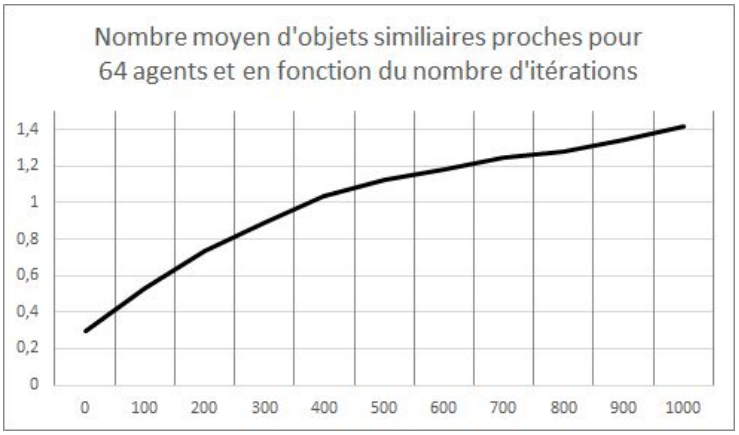
Nombre d'agents (nbAgents)

Pour commencer, on va tester le nombre d'agents. Pour cela, on choisit (selon la question 1 du TP) une grille de taille 50x50, un pas i=1, k+ = 0.1, k- = 0.3, 200 objets A et 200 objets B.

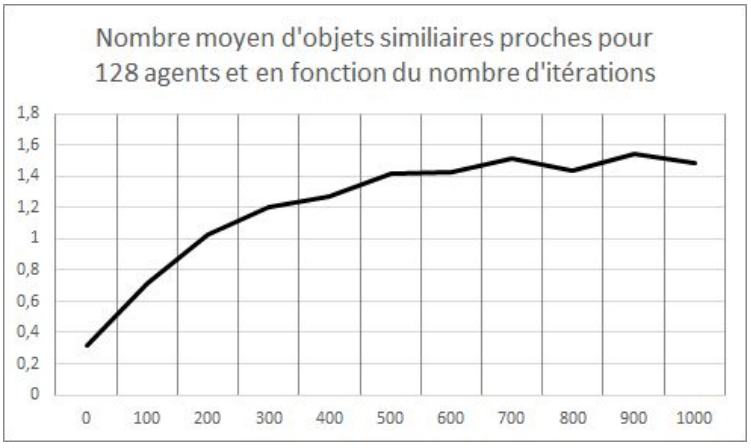
Si on a 4 agents, on constate que le nombre moyen d'objets similaires (A ou B) proches (dans le voisinage) est très faible, même au bout de 1000 itérations, alors qu'on voudrait un nombre moyen le plus grand possible. Ainsi, il nous faudrait beaucoup plus d'agents.



On choisit maintenant un nombre de 64 agents. On constate que plus le nombre d'itérations augmente, plus le nombre moyen d'objets similaires proches va augmenter.



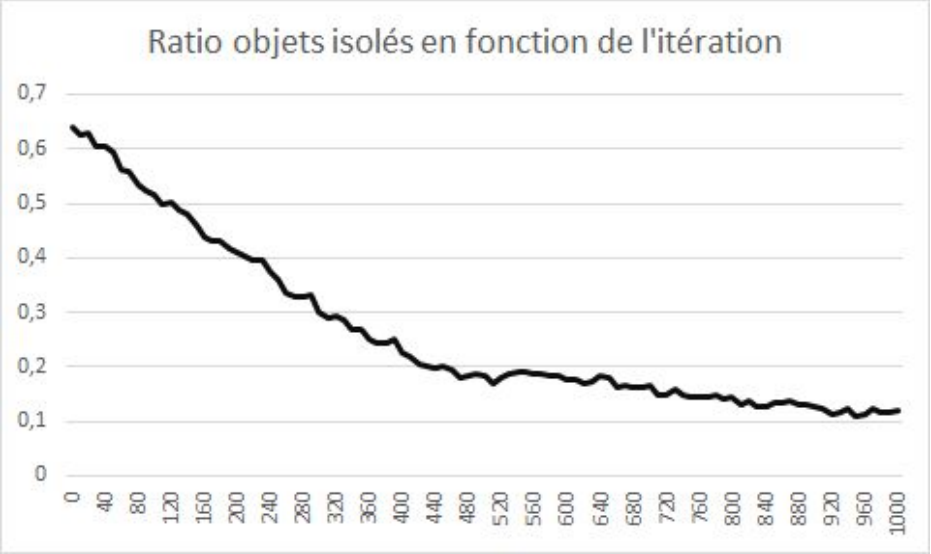
Par contre, si on choisit un nombre d'agent trop grand, alors le nombre moyen d'objets similaires proches va augmenter très vite, mais va rapidement ne plus augmenter et va plutôt stagner. Par exemple, avec les paramètres précédent, prendre 128 agents permet surtout d'améliorer la convergence de l'algorithme et donc diminuer le nombre d'itérations nécessaires à l'obtention d'un tri intéressant.



Notons aussi que lorsqu'il y a un nombre important d'agent alors il se crée de l'encombrement qui peut détériorer les performances de l'algo car les agents ont du mal à se déplacer (en plus de demander plus de performances de calcul).

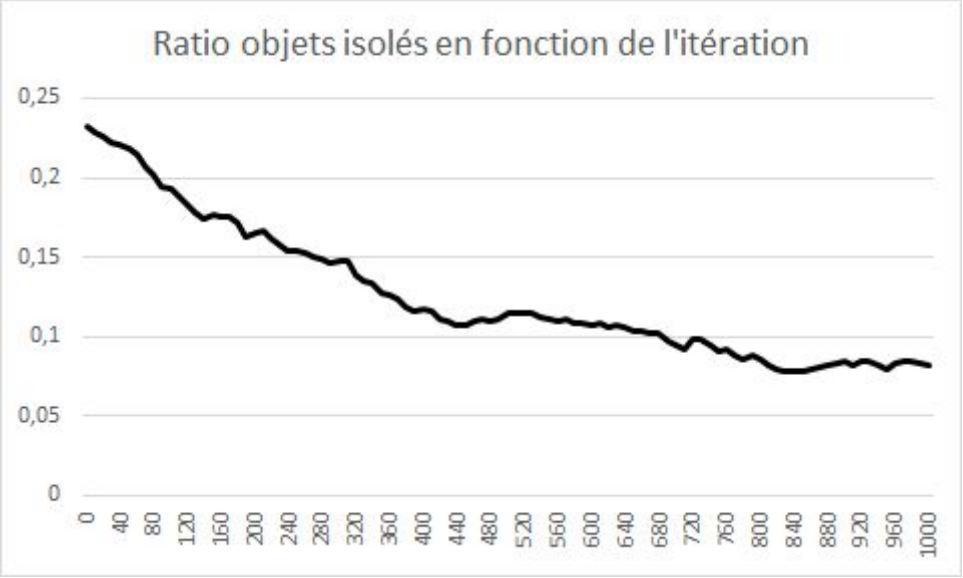
Nombre d'objets de type A et de type B (nA, nB)
Nous décidons de garder 60 agents, et nous allons maintenant tester le nombre d'objets de type A ou B (on prendra toujours le même nombre d'objets pour les deux types).

On opte dans un premier temps pour avoir 200 objets de chaque type. En ce qui concerne le graphique du nombre moyen d'objets similaires proches pour les agents, il se rapproche de celui créé précédemment pour 64 agents. Nous calculons le ratio d'objets isolés en fonction de l'itération, et on obtient le graphique suivant :



Nous constatons qu'il y a très peu d'objets isolés lorsqu'on se rapproche des 1000 itérations. On voit aussi que l'on part d'un ratio d'objets isolés important (plus de 60%) et que l'algorithme fait baisser ce chiffre à 10%.

Si on prend un nombre important d'objets (on choisit 800 objets de chaque type) sur le même nombre d'itérations, alors le ratio d'objets isolés selon l'itération est le suivant :



Le ratio passe de 23% (il est déjà faible puisqu'il y a beaucoup d'objets sur la grille et donc la probabilité pour un objet d'être à côté d'un objet de même type est importante) à un peu moins de 10% à la 1000 ème itération. Si on augmente le nombre d'itérations, alors les agents vont parvenir à trier les objets.

Il est donc logique de conclure que plus le nombre d'objets sur la grille est grand, plus les agents vont avoir besoin de temps pour les trier.

Probabilité de prise / dépose (k+ / k-)
Etant donné que la perception de nos agents est la même que celle de l'article alors on utilise les mêmes paramètres k : k+ = 0.1 et k- = 0.3.

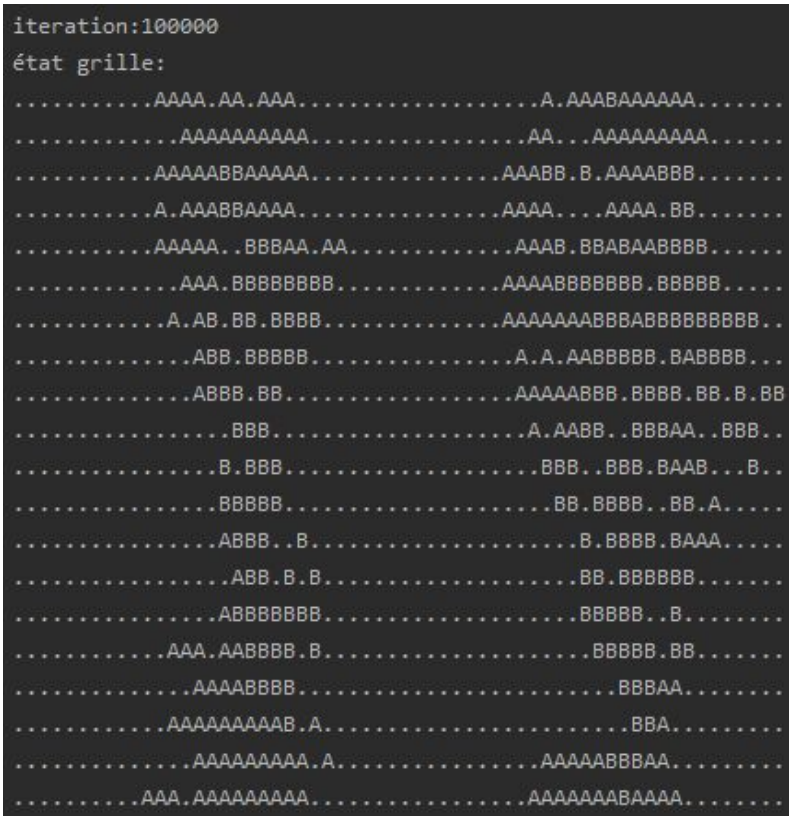
Nous avons implémenté le rayon de perception d'un agent, c'est à dire le nombre de cases que l'agent peut percevoir vers les directions N, O, E et S, ce qui va donc modifier son voisinage. Pour l'intégralité des tests du rapport, nous avons gardé un rayon égal à 1.

Cependant, nous avons observé que lorsque l'on change le rayon de perception, et donc la taille du voisinage, alors il faut penser à adapter les paramètres k car cela a pour effet de diminuer la valeur de proportion d'objets dans le voisinage. Par exemple, si le voisinage est constitué de 10 cases alors la présence d'un unique objet correspond à une proportion de 0.1 tandis qu'il est de 0.4 dans notre cas (voisinage de 4 cases).

Pourcentage d'erreur (e)
Par la suite, nous introduisant un pourcentage d'erreur e dans la reconnaissance des objets, qui va donc influencer la probabilité de prise d'un objet par un agent.

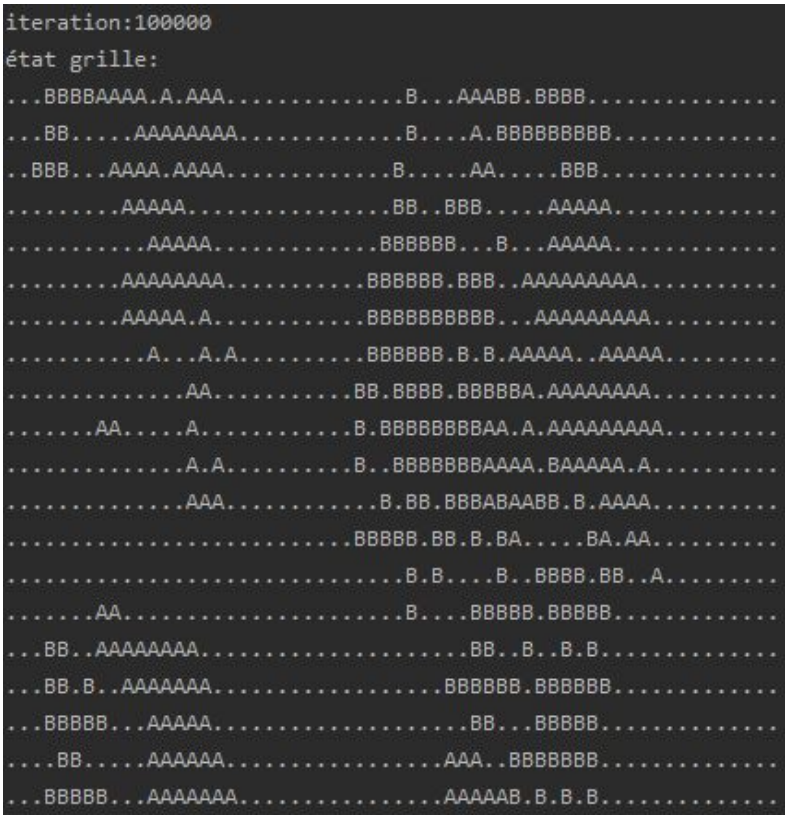
Pour nos tests, on garde une grille de taille 20x60, 200 objets de chaque type et 60 agents.

Si on prend une erreur très grande (e = 0.9), on constate bien qu'à la 100 000ème itération, les tas ne sont toujours pas homogènes. On observe deux tas comprenant lui même plusieurs petits tas d'objets A ou B :

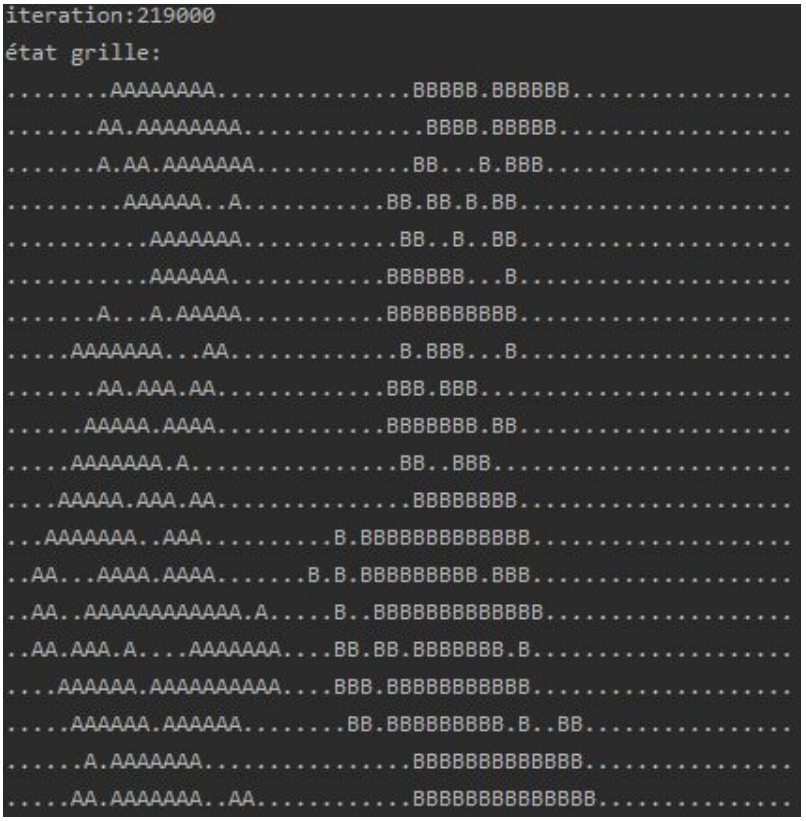


Si on augmente le nombre d'itérations, nous obtenons un seul et unique tas comprenant plusieurs tas homogènes d'objets.

Si on prend une erreur modérée (e = 0.3), on constate qu'au bout de 100 000 itérations, les tas d'objets homogènes sont rapprochés :



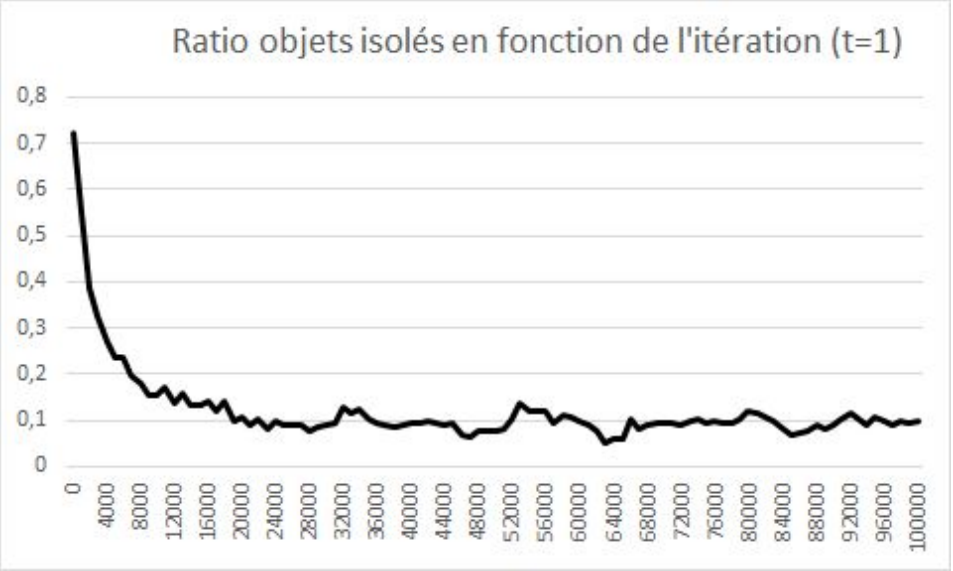
Cependant, en augmentant le nombre d'itérations, on finit par obtenir deux tas distincts d'objets de même type. Ici, on y parvient aux alentours de la 219 000 ème itération.

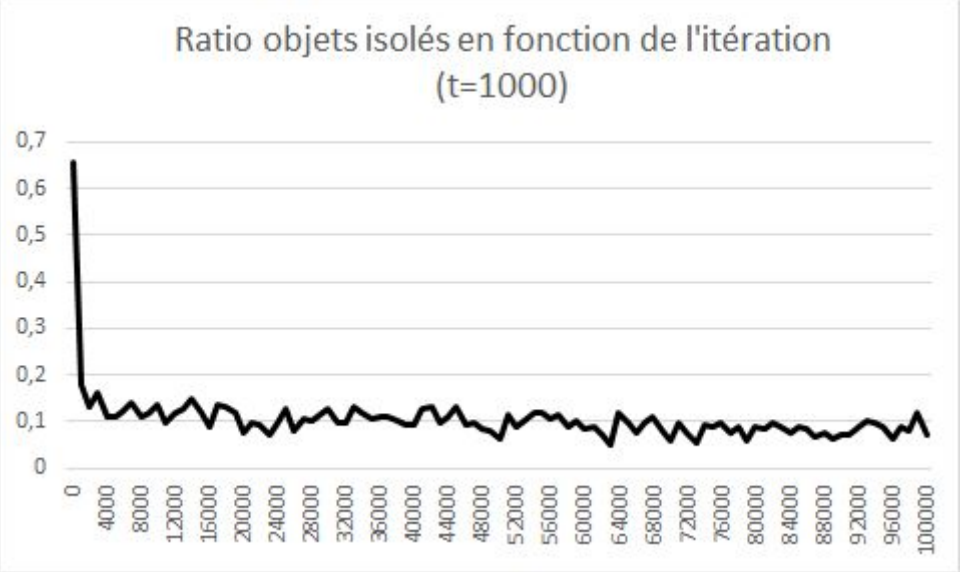


D’après ces résultats, nous pouvons en déduire que plus l’erreur se rapproche de 1, plus les agents vont avoir du mal à faire des tas homogènes distants les uns des autres. Cela induit que la convergence vers des tas homogènes et distincts prendra un plus grand nombre d’itérations.
On note aussi que si l’erreur est à 1 alors l’algorithme ne peut pas faire de tas homogènes car il n’est pas capable de discriminer entre les objets A et B.

Taille de la mémoire

Tous les agents possèdent une mémoire de taille t = 10 dans nos tests. Nous nous sommes demandé les résultats que nous pourrions obtenir si nous modifions cette taille. Nous avons donc testé deux tailles extrêmes : t = 1 et t = 1000.





En comparant les deux graphiques précédent, nous constatons que plus les agents ont une mémoire grande, plus ils vont trier les objets de manière efficace et donc plus rapide. Ce qui

semble logique puisque les agents vont mémoriser les objets plus longtemps et la valeur du f (la proportion d'objet de même type dans l'environnement immédiat) est d'autant plus petite que la taille de la mémoire est grande et donc la probabilité de prise ou de dépose sera plus grande.

La question ici est de savoir si la complexité spatiale introduite est compensée par le gain d'efficacité. Mais c'est une question qui s'éloigne du biomimétisme recherché ici.

Conclusion

Au vu de ces résultats, nous estimons avoir effectué une réplication intéressante de l'algorithme décrit dans l'article scientifique fournis dans ce TP "The dynamics of collective sorting robot-like ants and ant-like robots". Nous n'avions pas accès à une expérimentation avec des fourmis afin de vérifier que ce système reproduit bien le comportement de tri collectif observé dans la nature.
En plus de ça, nous avons mis en place des indicateurs pour mesurer la qualité du tri effectué par l'algorithme.
Finalement nous avons effectué des analyses du comportement de l'algorithme dans le temps et selon différents paramètres afin d'étudier leur influence.