Sujet TP 2 SMA

Élève :
Abdellaoui Anas

Enseignants :
HASSAS Salima

Problématique

On modélise note environnement par une grille ou sont disposés aléatoirement N_A objet de type A, N_B Objet de type B et N_Agent agents, avec une condition : sur la même case on ne peut pas avoir ni deux agents ni deux objets. Les agents se déplacent aléatoirement sur les 8 directions disponibles et peuvent ramasser ou déposer un objet avec une probabilité dépendant de la proportion d'objets vus dans leur mémoire. Pprise= (k+ /(k+ + f))2 et Pdépôt= (f/(k- +f))2 avec : k+ et k- des constantes et f représentant la proportion d'objet de même type A ou B dans l'environnement immédiat (voisinage de l'agent). Le voisinage de l'agent est défini par les cases atteignables en 1 pas de temps par l'agent dans les 8 directions. Le but des agents est de trier la grille en regroupant les objets de même type.

Implémentation

Après la présentation du but de TP, je vais aborder la partie de réalisation alors j'ai créé plusieurs classes :

- **Cell :** qui représente une case de la grille. Elle est caractérisée par ses coordonnées et si oui ou non un agent/objet est sur elle.
- Objet: qui représente les objets, ils sont caractérisés par leur type (A ou B)
- Agent : qui représente les agents. Ils sont caractérisés par la cellule où ils se situent, l'environnement où il se situe, 2 constantes permettant de calculer les probabilités de prise/dépot d'objet, une mémoire des dernières cases parcourues et un taux d'erreur de discernement de l'objet. Les agents fonctionnent comme suit : 1) Il se déplace de manière aléatoire sur une des cases voisines libre. 2) S'il possède un objet et que la cellule n'a pas d'objet, il pose l'objet avec une probabilité. 3) S'il ne possède pas d'objet et que la case en possède un, il le ramasse avec une probabilité.
- Environnement : correspondant à l'environnement extérieur. Il est caractérisé par la grille. C'est cette classe qui possède la fonction principale `run` qui fait agir les agents.

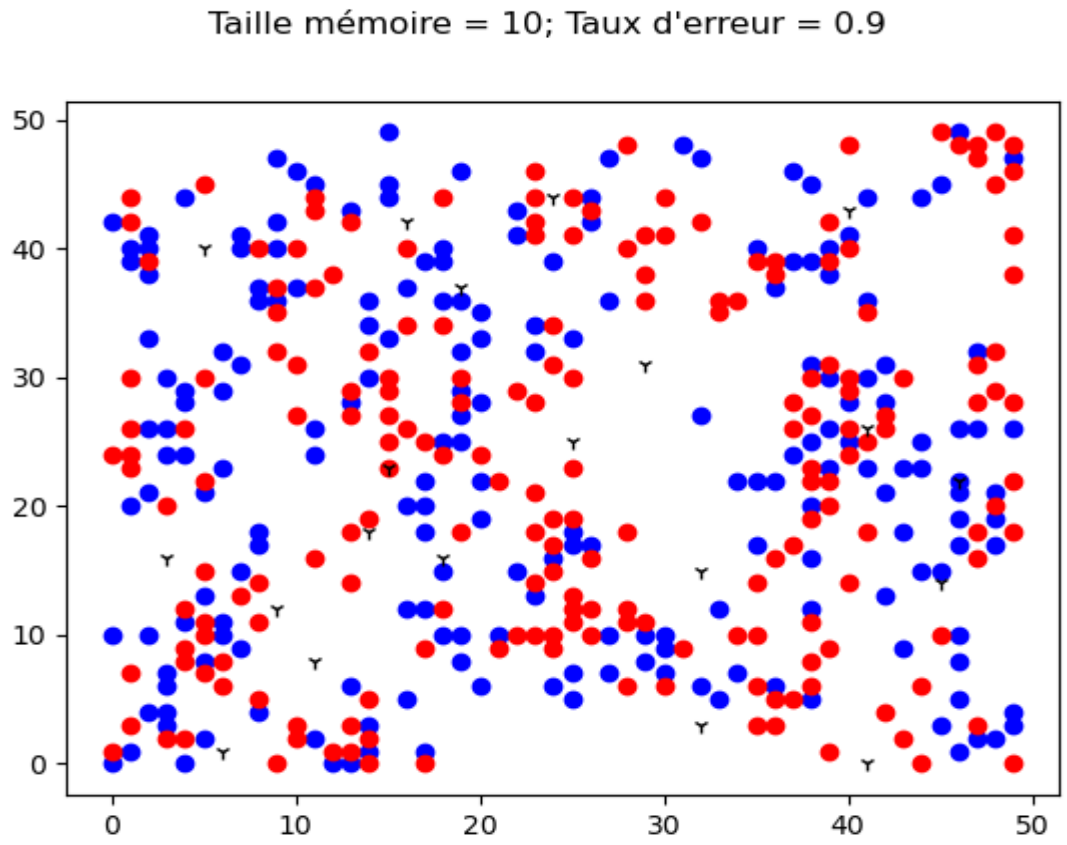
J'ai implémenté deux fonctions `run` pour prévoir deux manières de générer l'animation.

- `run_without_saving` : qui après chaque tour des agents actualise le graphe, moins gourmant en mémoire et permet l'affichage du graphique avant la fin d'execution de tous les tous. Il ne permet cependant pas de sauvegarder la figure et les images par seconde sont limitées.
- `run_with_saving` : qui garde en mémoire toutes les positions de tous les agents et tous les objets à chaque tour, puis génère une animation avec. Gourmant en mémoire et prend du temps à se lancer car il faut tout calculer avant de pouvoir afficher. La video est cependant sauvegardable et les images par secondes sont très bien.

Attention : il faut supprimer/renommer l'ancienne animation si vous souhaitez en sauvegarder une autre.

On exécute le programme (main.py) on peut distinguer plusieurs cas en changent les paramètres d'entrés :

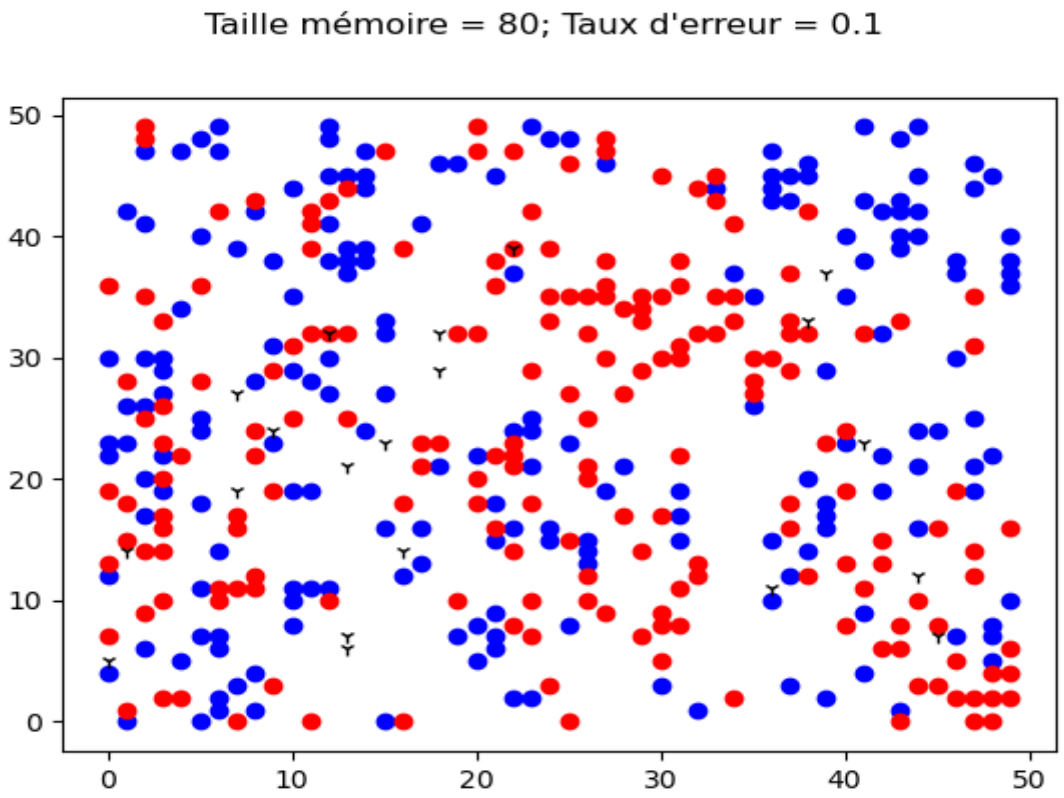
- **Taux d'erreur très élevé**



Si l'erreur est élevé, on remarque que plus l'erreur est grande, plus les agents regroupent les objets entre-eux mais pas forcément du même type. En effet, lorsque l'erreur est élevée, l'agent n'arrive plus à distinguer le type de l'objet, en revanche il arrive quand même à distinguer s'il y a ou non un objet.

- **Taille mémoire élevé**

Plus la taille de la mémoire est grande, moins les clusters d'objets sont nombreux et plus ils sont étendus (moins denses). C'est l'inverse lorsque la taille de la mémoire est faible. En effet, plus la mémoire est grande, plus les agents ont une vision étendue. Ce qui signifie qu'ils essaient d'homogénéiser de plus grandes zones.



On a chaque agent à une probabilité P_prise = (k_plus/(k_plus + f_x))² et P_depot = (f_x/(k_moins+f_x))² de prendre/déposer un objet de type x, avec f_x la proportion d'objet de type x dans la mémoire de l'agent (ie si la mémoire est "AOOBBAOOB" f_a = 2/9) et k_moins, k_plus sont des constantes.

On remarque alors 2 choses :

- Plus f_x augmente, plus P_prise diminue et P_depot augmente
- Plus f_x diminue, plus P_prise augmente et P_depot diminue

Donc plus la proportion d'objet de type X dans la mémoire de l'agent augmente plus il est susceptible de déposer un objet du même type et de ramasser un objet du type opposé. De plus les agents bougent de manière aléatoire, il y a donc beaucoup de chance qu'ils tournent en rond (car la probabilité d'aller d'un coté est la même que d'aller du coté opposé), donc leur mémoire correspond à une zone uniforme autour de l'agent. La mémoire de l'agent est en fait une densité à un endroit donné.