Rapport TP 2 SMA Tri-collectif

I – Présentation

Le TP consiste à réaliser un algorithme simple de tri-collectif. Cet **algorithme est inspiré du comportement de certaines colonies de fourmis** triant leurs larves. Les agents se déplacent au hasard, ne **communiquent pas**, n'ont pas d'organisation hiérarchique, pas de représentation globale et ils ne peuvent percevoir que des objets juste devant eux. Ils peuvent cependant distinguer les différents types d'objets avec un certain degré d'erreur. La probabilité qu'ils ramassent ou déposent un objet est modulée en fonction du nombre d'objets identiques qu'ils ont rencontrés dans un passé récent. Cela crée un feed-back positif suffisant pour coordonner l'action des robots. Bien que moins efficace qu'un tri hiérarchisé, cette organisation décentralisée offre des avantages de simplicité, de flexibilité et de robustesse.

II – Mise en place

Soit un environnement représenté par une grille sur lequel on dispose initialement aléatoirement, des objets identifiés par des lettres A ou B et un certain nombre d'agents. Le déplacement des agents est aléatoire et la prise ou le dépôt d'un objet dans l'environnement sont conditionnés respectivement par une probabilité de prise et une probabilité de dépôt qui s'exprime respectivement par : **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 rencontrés qu'il a gardé dans une mémoire à court terme. Cette mémoire à court terme fonctionne comme une pile FIFO qui stocke les objet rencontrés lors des n derniers pas. Si l'agent a une mémoire des 10 derniers pas et que lors de ces 10 derniers pas le robot a rencontrés 3 objets A et 2 objets B alors fa = 3/10 et fb = 2/10. Le fonctionnement de l'agent se résume de manière simple par une boucle de perception de son environnement puis d'action, rien de plus.
Le tri sera évalué en réalisant regardant la proportion d'objets de même type dans un voisinage direct de chaque objet. Chaque case en dehors des bords et des coins a 8 voisins directs. Si sur la case il y a un objet A et dans son voisinage direct 4 objets A et 2 objets B alors, le score associé à cet objet sera de (4-2)/8. On calculera la moyenne pour les objets B et pour les objets A. La valeur maximum théorique est très proche de 1 si tous les objets B sont regroupés entre eux, seuls les objets aux bords ayant pour voisins de objets B et du vide ne vont pas avoir un score de 1.

III – Architecture Voyelle

A, Environnement

- **Grille** où se déplace les agents et où sont placés les objets.
- **Objets A, B, C**, ces objets font partis de l'environnement et leurs méthodes sont gérées par l'environnement.

B, Agents

- **Agents réactifs** : agents qui agissent se déplacent au hasard ou suivent le gradient de phéromone si la trace est assez forte. Ils sont munis de fonctions de **perception** (regard au sol) **déplacement**, de **prise et de dépôt d'un objet.** Ces seules fonctionnalités leurs permettent de triés efficacement les tas d'objets ne nécessitant pas d'interaction (A et B). Pour trié les objets C d'autres fonctionnalités leurs seront implémentés (cf partie IV).

C, Interactions

- Pas d'interaction dans la première partie.

D, Organisation

- **Organisation émergente** : Sans que ces comportements soient codés, les agents vont s'auto-organiser pour triés les objets.

IV – Introduction d'objets C

On souhaite ajouter un nouveau comportement collectif aux robots. Pour cela on ajoute nC objets de type C, qui nécessitent la collaboration de 2 robots pour être portés. Dans ce cas, quand un robot tombe sur un objet C dans son environnement, il émet un phéromone (appel à l'aide) qui se code par une information propagée dans son voisinage (8 cases autour) sur une distance de diffusion **DS** (ex : les 24 cases autour de l'agent si ds=2 dans ce cas pour un agent se trouvant à l'emplacement (i,j,) on aura toutes les cases se trouvant à i+k, j+l, avec 0<=k<=2, et 0<=k<=2.). Il faut noter que l'intensité du phéromone se réduit au fur et à mesure qu'on s'éloigne de la case où se trouve l'agent. Sur la case de l'agent, l'intensité du phéromone est de 1, puis de 1/2 pour les cases à distance 1, puis 1/3 pour les cases à distance 2 etc, ... (1/Norme infini(pos1-pos2)). La phéromone s'évapore d'un taux **R** à chaque itération. Le robot qui émet des phéromones peut réémettre ou abandonner la tâche au bout d'un certain temps **T**, pour éviter des situations d'inter-blocage (les robots sont bloqués à attendre de l'aide, alors qu'il pourraient débloquer la situation en s'aidant mutuellement l'un après l'autre).

Pour pouvoir atteindre les zones où la quantité de phéromone est la plus importante, il faut munir les agents de capacités sensorielles. On ajoute à la fonction de perception (initialement elle ne permet que de voir les objets immédiatement au sol) une capacité de percevoir les taux de phéromones dans les 8 cases directement voisines. Lors de son déplacement, si il ne transporte pas d'objets, l'agent va suivre la direction où la quantité de phéromone est le plus fort avec une probabilité **p=(1-exp(-pheromon))**, cela signifie que si la quantité de phéromone sur un case est faible, cette probabilité sera très proche de 0 alors que si la quantité de phéromone est grande, la probabilité de suivre ce chemin sera très grande. Dans le cas où l'agent ne se dirige pas dans la direction où la quantité de phéromone est la plus importante, il continue de suivre un comportement aléatoire.

Lorsqu'un agent arrive sur une case présentant un objet C, il vérifie par la même occasion si un autre agent se trouvant sur la case serait susceptible de l'aider (agent ne transportant pas d'autres objets). Si c'est le cas, il va y avoir **coordination** des 2 agents pour pouvoir transporter l'objet. L'agent qui vient d'arriver sur la case va être désigné comme le **chef de la relation bilatérale** entre les 2 agents et va régir le comportement de l'association. Le couple d'agent se déplacera selon les bons vouloir de l'agent chef et l'association ne prendra fin que lorsque l'agent chef aura décidé de déposer l'objet C. A partir de ce moment là, les 2 agents seront de nouveaux libre de leurs actions. Si l'agent ne trouve pas d'aide sur la case, il appel à l'aide et attend T itérations jusqu'à ce que quelqu'un susceptible de l'aider arrive. Au bout de T itérations, il s'en va.

Une des choses à prendre en compte dans la modélisation présentée jusque la est que si un agent abandonne, il va tout de même être tenté de revenir vers la case où il se trouvait de part la forte quantité de phéromone qui si trouve et ceux malgré le phénomène d'évaporation. Pour prendre en compte ce phénomène, il faut donc faire en sorte que les agents ignorent les informations concernant les phéromones pendant une durée **D** après avoir abandonné la prise d'un objet C.

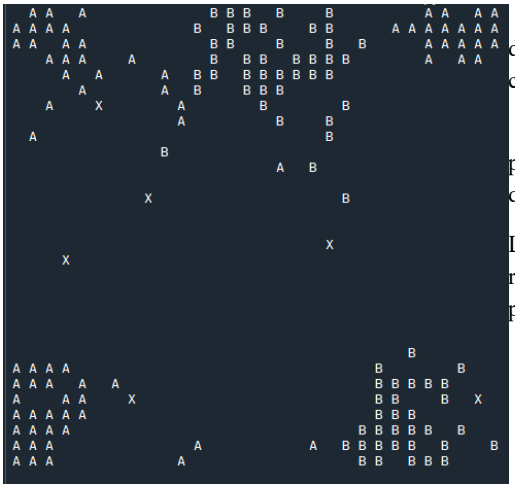
V – Résultats

A, Premier Tri-collectif

Première configuration : On utilise cette première configuration pour réduire les temps de calcul des différents test réalisés
NA = 75 , NB = 75, NR = 10 / Soit 75 objets de type A, 75 de type B et 10 Robots
HEIGHT = 30, LENGHT = 30 / Soit un environnement de taille 30*30

Kt = 0.1, Kp = 0.3, ER = 0 / Constante de prise d'un objet égale à 0,1, / de dépôt égale à 0,3 et pas d'erreur de reconnaissance

IT = 100000 / 100000 itérations, chaque robot agit 1 fois par itération



Les objets A sont représentés par des A, les objets B par des B et les robots par de X. Si un robot porte un objet alors il sera affiché sur la représentation comme cet objet.

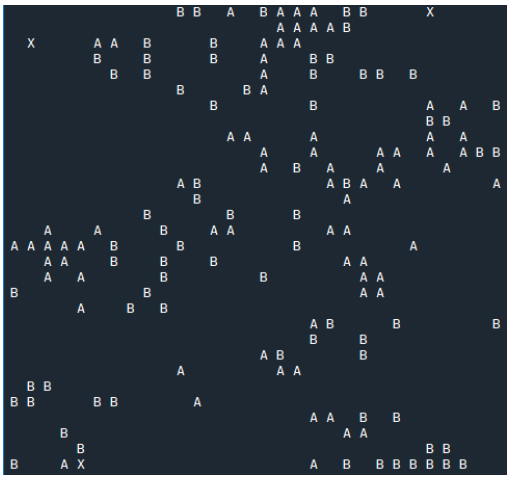
On observe qu'avec ces paramètres l'algorithme de tri-collectif fonctionne plutôt bien. On retrouve bien différents tas d'objets identiques. Il y a 3 tas d'objets A et 2 tas d'objets B.

Les résultats de l'évaluation donnent score A = 0,58 et score B = 0,50. Ces résultats sont plutôt bons au vu du la fonction d'évaluation qui favorise les tas placés dans les coins et sur les bords.

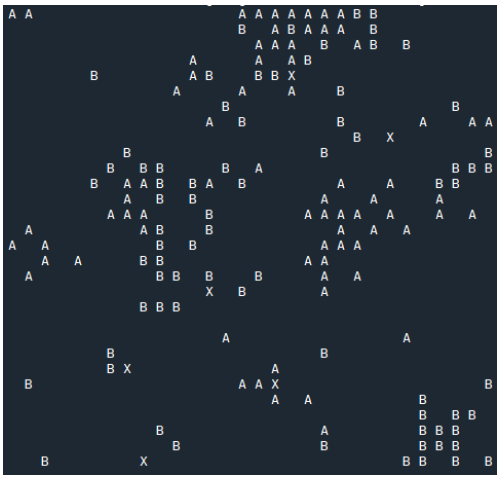
B, Étude de l'influence du nombre d'itérations

Pour étudier cette influence, on va jouer sur le nombre d'itérations en gardant notre première configuration. On va réaliser une génération et retourner un rendu à différentes itérations. Il faut savoir que les comportements des robots introduit une grande part d'aléatoire et qu'il n'y a pas de reproductibilité des résultats (même en plaçant une seed à savoir pourquoi). C'est pourquoi les conclusions faites à partir de ces exemples seront biaisées.

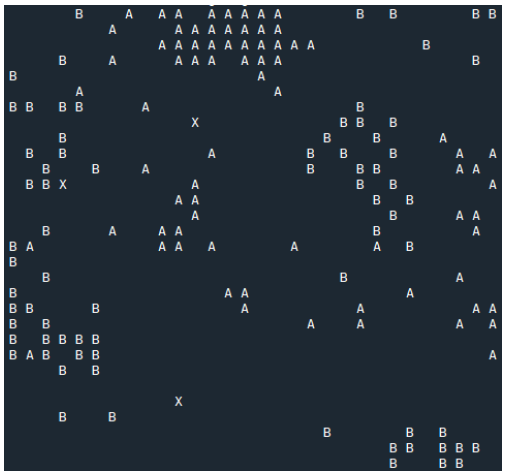
10 000 Itérations (0.21 / 0.11)



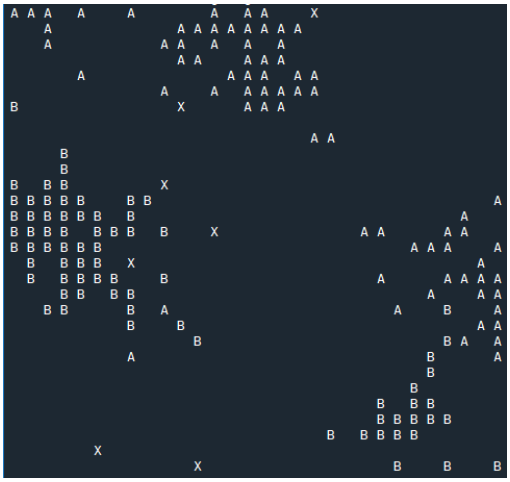
20 000 Itérations (0.23 / 0.15)



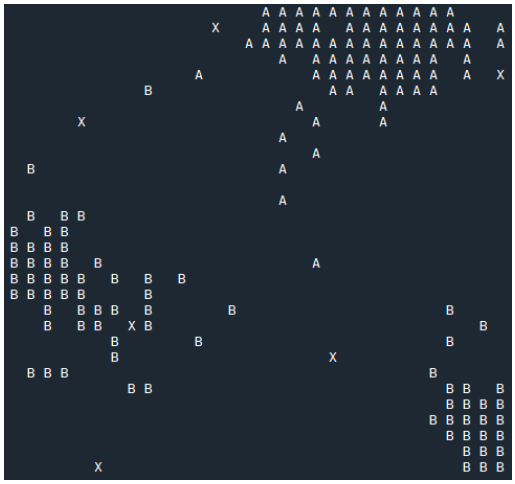
80 000 Itérations (0.39 / 0.23)



150 000 Itérations (0.42 / 0.52)



500 000 itérations (0.69 / 0.59)

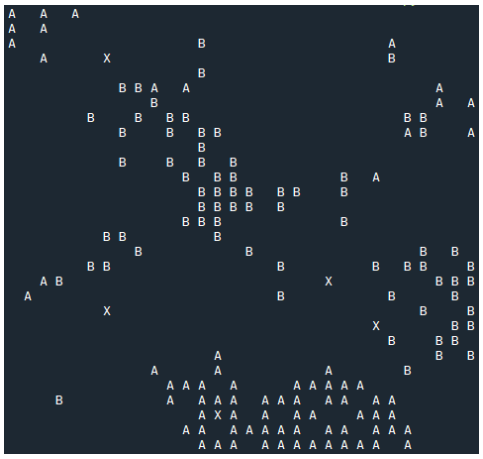


On observe que en partant d'un environnement complètement non trié, de petits tas d'objets identiques commencent à se former assez rapidement. A 10 000 itérations on retrouve par exemple un tas de B en bas à droite et un tas de A tout à gauche, l'évaluation de tri donne des résultats assez médiocres 0,21 pour les objets A et 0,11 pour les objets B. En effet les objets A semblent légèrement mieux triés. Dès 20 000 itérations, les tas se dessinent plus nettement et ils sont moins nombreux, il reste cependant une grande zone au milieu où tous les objets sont un peu mélangés. A 80 000 itérations c'est très net, on a 3 tas de A et 3 tas de B relativement bien dessinés et les évaluations s'améliorent nettement (0.39 / 0.23). A 1500 itérations on a plus que 2 tas de B et 2 tas de A, en bas à droite les 2 tas de A et B sont assez proche ce que l'on ne retrouve plus a 500 000 itérations. 3 tas sont très nettement dessinés et les évaluations sont très bonnes 0.69 pour A et 0.59 pour B. On a bien une convergence de l'algorithme de tri en temps long.

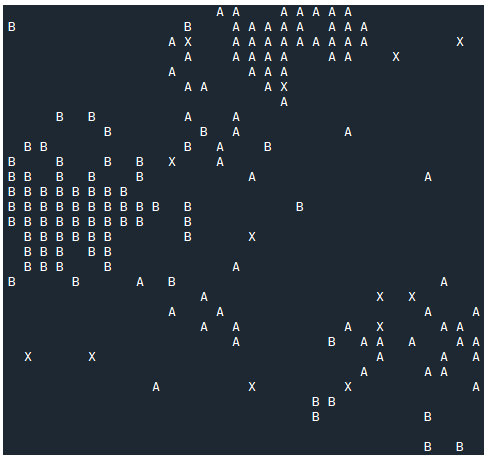
C, Étude de l'influence du nombre de robots

Instinctivement on pourrait penser qu'augmenter le nombre de robots revient un peu à la même chose que d'augmenter le nombre d'itération dans un tel environnement où il y a assez de place pour que tout le monde puisse se déplacer sans trop se gêner. On va donc tester cela avec 50 000 itérations à chaque fois.

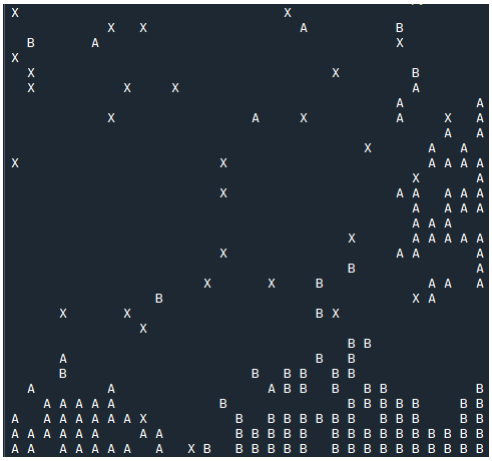
10 Robots (0.44 / 0.28)



20 Robots (0.42 / 0.52)



50 Robots (0.49 / 0.70)

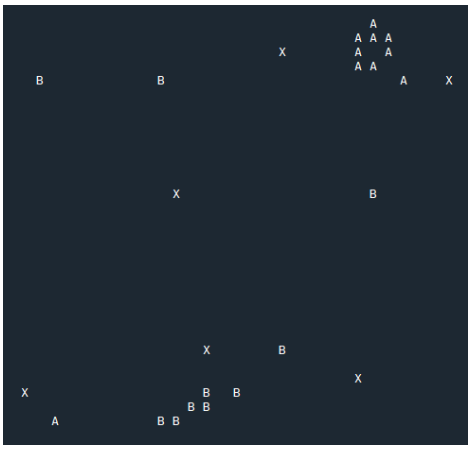


On observe effectivement qu'avec le même nombre d'itérations plus il y a de robots, mieux l'environnement est trié. Avec 10 Robots, les objets A sont assez bien triés avec un gros tas en bas, mais les objets B sont principalement regroupés en bas à droite et au milieu, zone assez peu dense et avec quelques objets A intercalés. Le score est de 0.44 pour les objets A et de 0.28 pour les B. Avec 20 robots, on retrouve un environnement mieux trié, un peu ce que l'on pourrait avoir au bout de 100 000 itérations en utilisant uniquement 10 robots. Idem avec 50 robots on a une presque convergence de l'algorithme de tri score de 0.49 pour les objets A et de 0.70 pour les objets B. On observe tout de même quelques objets ''perdus'' dans le vide, cela est du au déplacement aléatoire des robots. On peut conclure qu'augmenter le nombre de robots accélère la vitesse de convergence, il serait tentant de dire que pour déterminer l'état de tri de l'environnement c'est le nombre d'actions total qui compte ie nombre de robots * nombre d'itérations mais cela resterait à démonter.

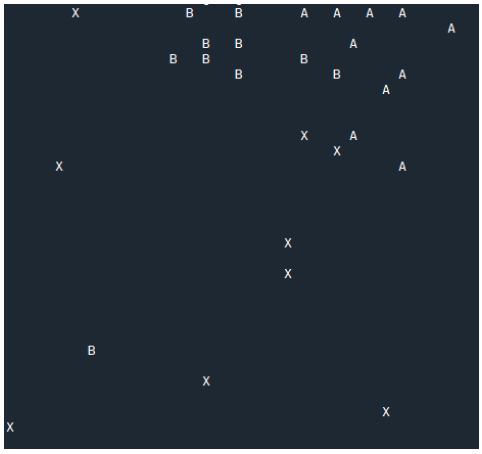
D, Étude de l'influence du nombre d'objets

Encore une fois on peut intuitivement se dire que si il y a plus d'objets à trier alors le temps de tri sera plus élevé. Cependant il est intéressant d'étudier les cas limites où l'environnement est saturé en objets et celui où il en est presque dépourvu pour vérifier si l'algorithme arrive à converger dans ces cas. Dans le premier cas il y aura 10 objets de chaque types et dans le second 300 ce qui signifie que 66 % de l'espace sera occupé par des objets.

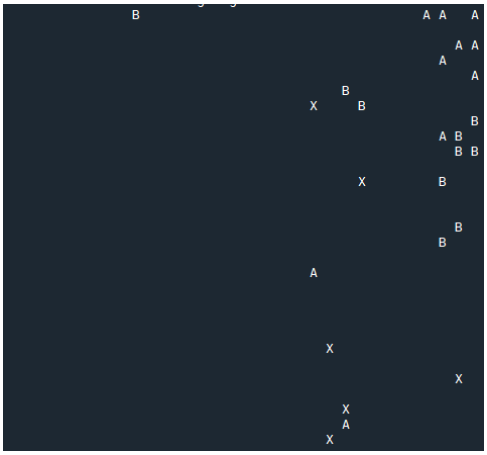
10 000 Itérations (0.32 / 0.16)



50 000 Itérations (0.02 / 0.03)

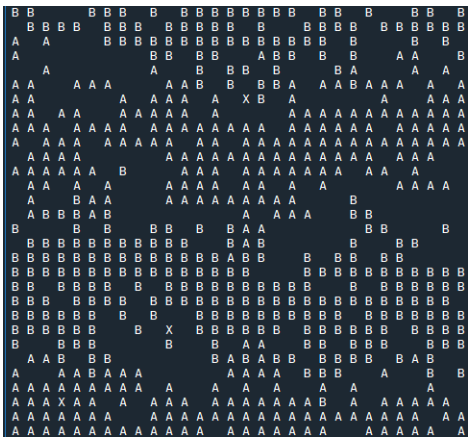


100 000 itérations (0.12 / 0.22)

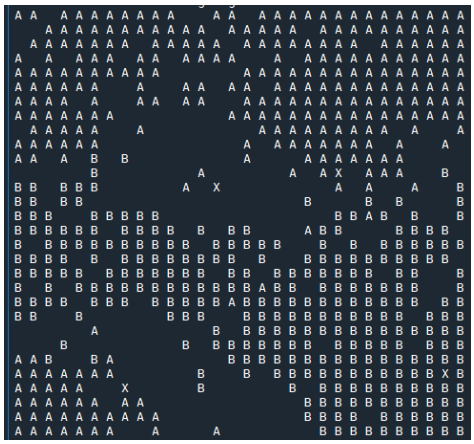


On observe que même avec une très faible densité d'objets le tri-collectif peut se réaliser. Tous les objets sont regroupés dans une même zone et non uniformément éparpillés dans l'espace même si localement le tri entre les 2 types d'objets n'est pas forcément très bon. Dès 10 000 itérations on a un bon résultat de tri les objets de Type A et B forment 2 groupes bien distincts et le score des objets de type A est de 0.32 et 0.16 pour les objets B. Cependant ce tri peut se défaire au fil du temps du fait des déplacements aléatoires des robots et que la variable de répartition f soit presque souvent faible étant donné la faible densité. En effet les objets en bordure de tas sont plus souvent ramassés par les robots étant donné que la probabilité qu'a le robot de passer sur cette case sans passer sur une autre case contenant le même type d'objet est plus grande que pour les objets au cœur du tas. Et étant donné la faible densité, tous les objets sont des objets de bordure de tas, cela pousse le robot à prendre en permanence des objets même si le tas avait l'air bien trié à l'œil nu. C'est pour cela qu'a 50 000 itérations les tas sont un peu défaits. Mais il se reformeront comme à 100 000 itérations.

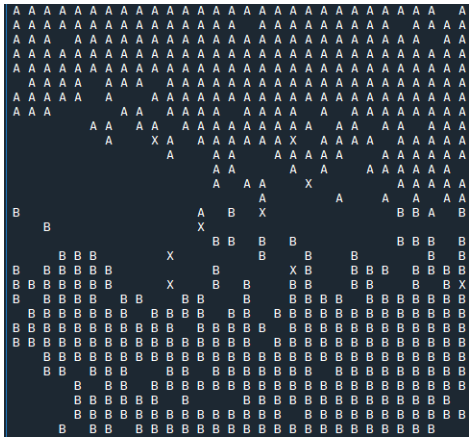
20 000 Itérations (0.62 / 0.63)



200 000 Itérations (0.77 / 0.76)



500 000 Itérations (0.85 / 0.79)



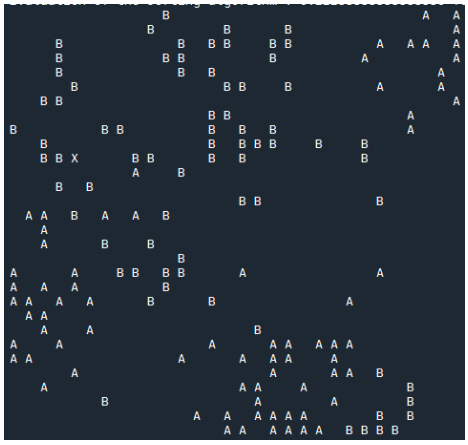
On observe que même avec une très forte densité d'objets l'algorithme fonctionne très bien. Premièrement le temps de convergence semble plus élevé qu'avec une plus faible densité d'objets, il aura fallu attendre 500 000 Itérations pour obtenir 2 paquets très net. Dès 20 000 itérations on observe un bon résultat de tri, les objets de Type A et B forment plusieurs tas biens distincts et le score est de 0.62 pour les objets A et 0.63 pour les B. A 200 000 itérations on ne distingue déjà plus que 2 tas A et un tas B. On pourrait penser que le tas A en bas à gauche est coincé et ne pourra pas sortir étant donné la forte densité de ce tas mais, tous les robots arrivant du tas B n'ont enregistré que des objets B dans leur mémoire et arrivant sur un objet A ils vont s'en saisir. Cela va permettre d'aboutir à la figure de 500 000 itérations où les objets A et B sont très distinctement séparés par un espace vide. Une forte densité d'objet permet d'aboutir à de très bons résultats.

E, Étude de l'influence des constantes Kt et Kp

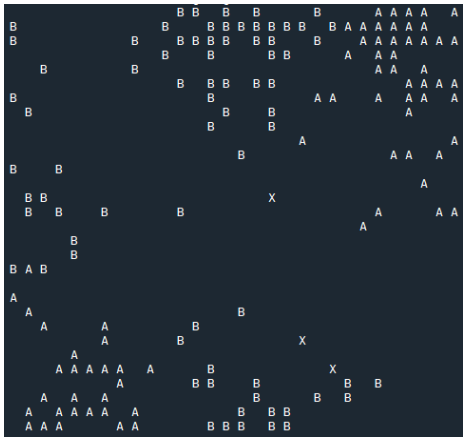
On souhaite étudier l'influence de Kt et Kp sur les résultats de l'algorithme de tri.

Sur les 2 premiers exemples on étudie les l'influence du paramètre Kt. Avec Kt = 0.8 pour les 3 premiers graphiques et Kt = 0.01 pour les 3 suivants.

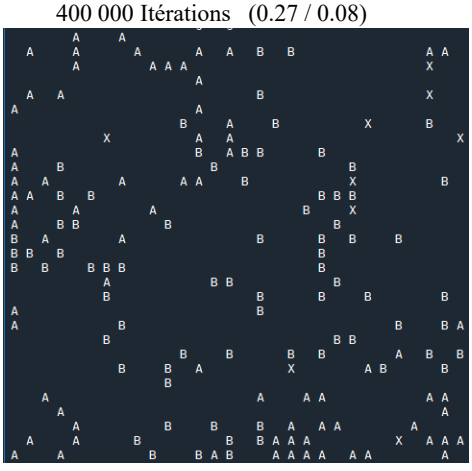
Kt=0.8 / 20 000 Itérations (0.22 / 0.17)



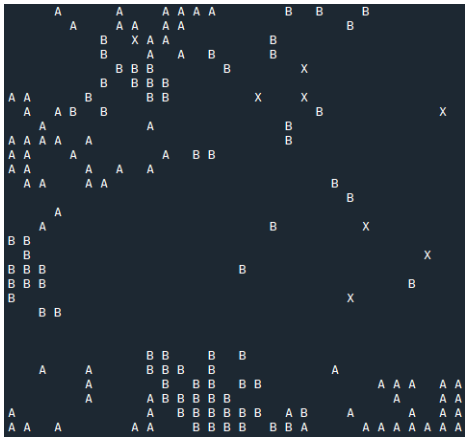
200 000 Itérations (0.38 / 0.27)



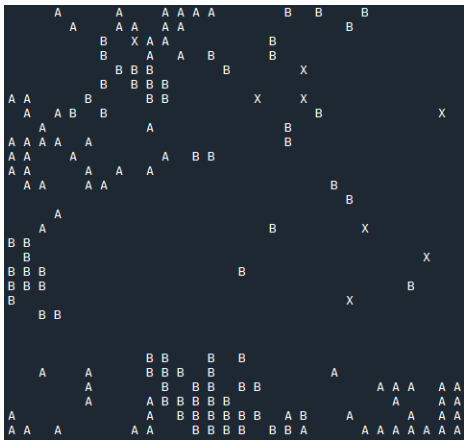
400 000 Itérations (0.27 / 0.08)



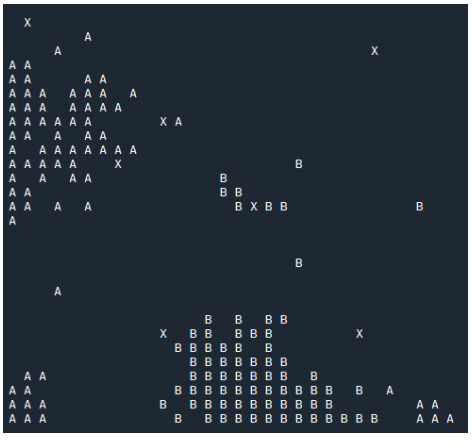
Kt=0.1 / 400 000 Itérations (0.22 / 0.27)



4 000 000 Itérations (0,38 / 0,39)



5 000 000 Itérations (0.61 / 0.69)

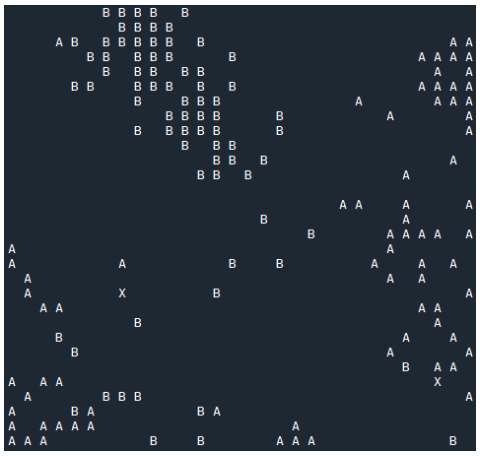


Sachant que Pprise= (Kt (Kt + f))2, le fait que Kt = 0.8 entraîne que min(Pprise = 0.8/(0.8+1)) = 0.45 donc même si le robot passe ses 10 derniers pas sur le tas d'un même type d'objet il aura une probabilité p de ramasser l'objet alors que l'objet est très probablement bien placé vue que le robot a passé ses 10 derniers pas sur le même type d'objets. Ce qui va faire que le robot va avoir tendance a déranger des tas déjà bien constitués. On observe qu'a 20 000 Itérations plusieurs petits tas sont déjà constitués et c'est encore plus net à 200 000 itérations. Mais au bout de 400 000 itérations les tas relativement bien triés sont dé-triés et les scores des objets A et B sont bien moins bons.

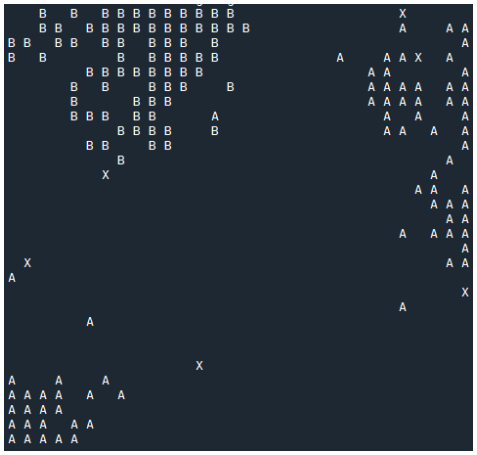
Dans le cas où Kt = 0.1 alors min(Pprise = 0.1/(0.1+1)) = 0.09 la probabilité de ramasser un objet si on est déjà tombé sur 10 objets du même type est donc assez faible ce qui est plutôt bien. De plus les tas constitués vont toujours avoir une bonne densité. Cependant de manière générale, les probabilités de ramasser les objets est assez faible ce qui va ralentir la vitesse de convergence. On observe que même à 5 000 000 itérations la convergence n'est pas aboutie.

Étudions l'influence du paramètre Kp. Kp = 0.8 pour les 3 premiers graphes et Kp = 0.05 pour le suivant.

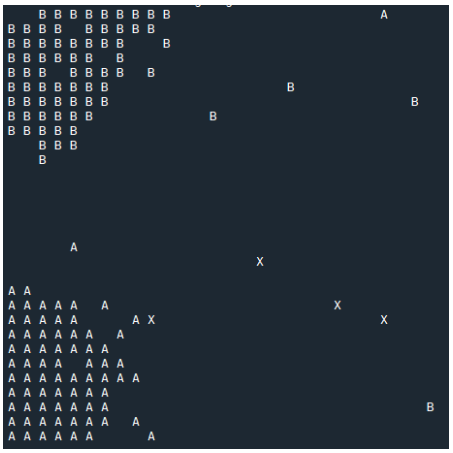
Kp=0.8 / 20 000 Itérations (0.18 / 0.39)



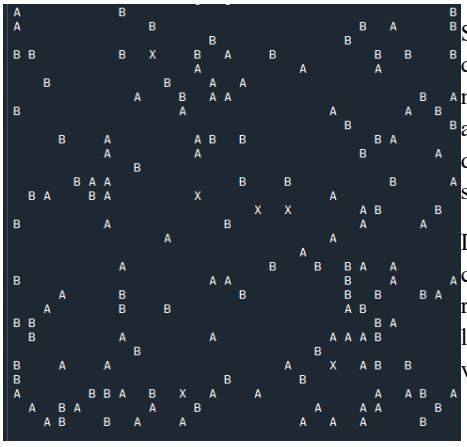
80 000 Itérations (0,52 / 0,62)



200 000 Itérations (0.81 / 0.76)



Kp = 0.05 / 2 000 000 Itérations (0.25 / 0.25)

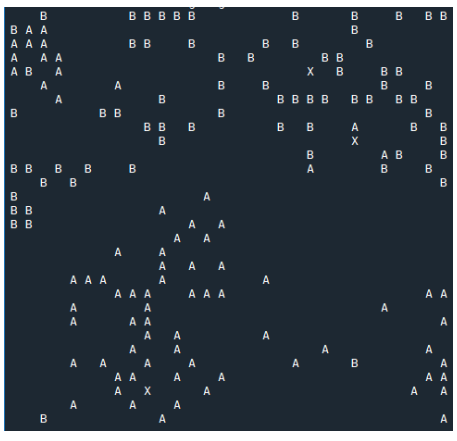


Sachant que Pdépôt= (f/(Kp +f))2, le fait que Kp = 0.8 fait que Pdépot va être plus faible qu'avec la valeur de base de Kp = 0.3. L'agent va avoir tendance à garder les objets sur lui et ne les déposer que lorsqu'il est tombé de nombreuses fois sur le même type d'objets auparavant. On observe que dès 20 000 itérations les tas sont assez nettement définis, la vitesse de convergence semble plus rapide qu'avec la valeur de base. A 200 000 itérations les 2 tas sont distinctement formés et séparés par une espace vide.

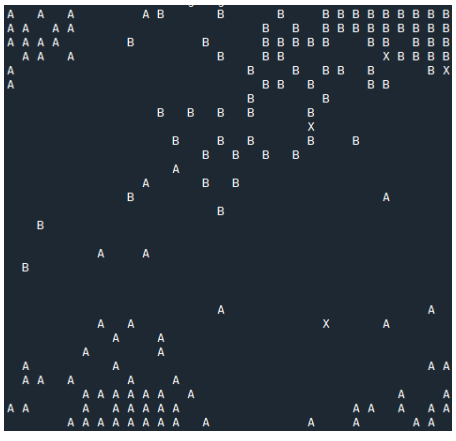
Dans le cas où Kp est très faible, Pdépôt va presque toujours être égal à 1 si l'agent à le type de cet objet dans sa mémoire à court termes ce qui va souvent être le cas étant donné qu'avant de ramasser l'objet, il l'analyse sur le sol et l'ajoute à sa mémoire. Avec une telle valeur, l'algorithme ne converge pas. Voici l'état de l'environnement après 2 000 000 d'itérations. La valeur limite de Kp pour faire converger l'algorithme semble expérimentalement être 0.12.

F, Étude de l'influence de la longueur de la séquence mémoire des agents

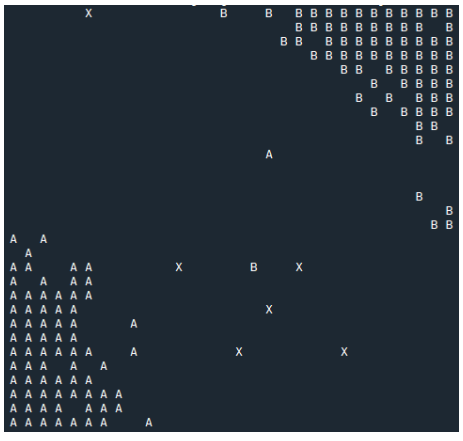
M_size=50 / 20 000 Itérations (0.20 / 0.18)

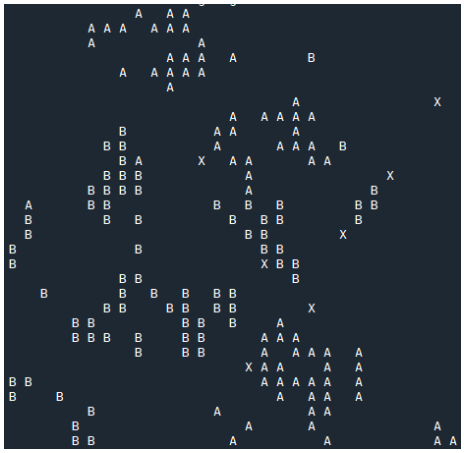


50 000 Itérations (0.41 / 0.42)

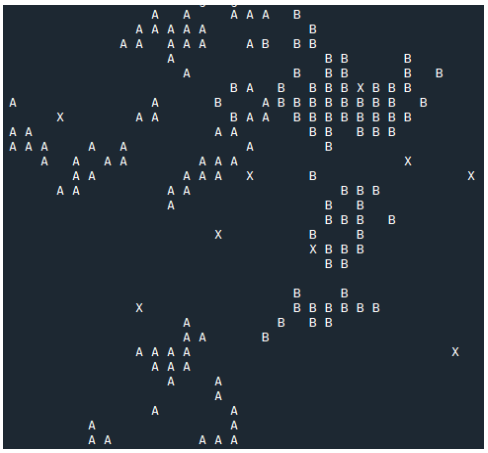


300 000 Itérations (0.75 / 0.74)





M_size=2 / 500 000 Itérations (0.14 / 0.12)



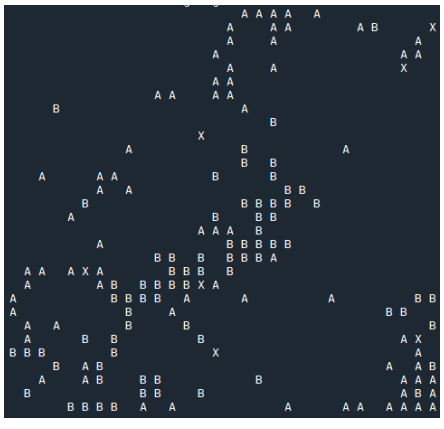
5 000 000 Itérations (0.24 / 0.36)

La taille de la mémoire va avoir une influence sur l'échelle d'action de chaque agent, une grande mémoire va permettre à un agent d'avoir une action plus précise sur un plus grand périmètre étant donnée qu'il aura une représentation récente des X dernières cases sur lequel il est passé même si il ne sait pas dans quelle direction il se déplace. Alors qu'avec une petite mémoire l'agent oublie presque immédiatement son environnement. On peut voir qu'avec une mémoire de taille 50, l'algorithme converge assez rapidement ,plus rapidement qu'avec une mémoire de taille 10 même si cela peut être un effet de variance. A 300 000 itérations le tri est parfait les 2 groupes d'objets sont au 2 extrémités de l'environnement. Avec un mémoire de taille 2, les tas se multiplient mais restent du même type. A 5 000 000 d'itérations les tas deviennent plus massifs et plus denses. L'algorithme semble converger mais la vitesse de convergence est très faible. Le cas limite d'une mémoire de taille 0 serait un agent sans mémoire et son comportement serait alors complètement aléatoire.

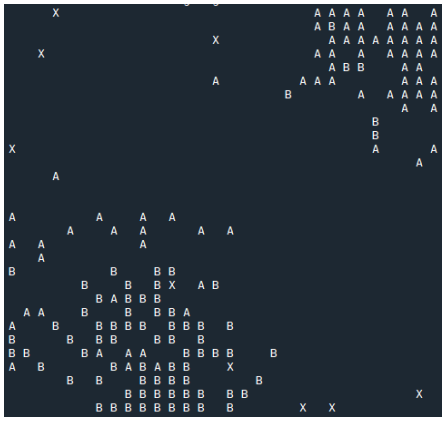
G, Étude de l'influence du taux d'erreur d'identification

En introduisant une erreur dans le taux d'identification, les agents peuvent confondre les types des objets. Si le taux d'erreur est de 0.5 alors les agents ne savent plus faire la différence entre les objets A et les objets B. Si le taux d'erreur est de 100 %. Ils confondent les objets A avec les objets B et inversement mais le tri se fera tout de même très bien étant donné que l'agent n'a pas de notion d'objet A et d'objet B, il sait seulement que ce sont 2 types distincts, il y a une certaine symétrie. On observe que si le taux d'erreur est trop proche de 0.5, 0.3 pour les 3 derniers exemples, le tri va se faire mais un seul tas va être créer, l'agent a trop de confusion entre les objets A et B. Dans le cas où le taux d'erreur reste faible, le tri fini par converger. Mais en y regardant de plus près, on voit qu'avant d'atteindre cet état de convergence, les tas sont composés majoritairement d'un type d'objet mais que certains objets de l'autre type se cachent parmi ces tas. Par Exemple à 300 000 itérations le tas en haut à gauche est bien isolé et il contient majoritairement des objets A et quelques objet B. Le temps de convergence est néanmoins nettement plus lent. L'algorithme converge expérimentalement jusqu'à un taux d'erreur de 0,18.

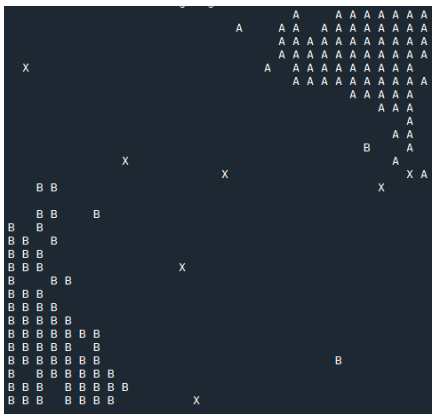
Er=0.1 / 100 000 Itérations (0.16 / 0.23)



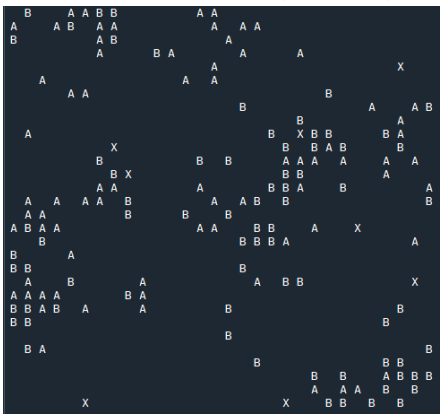
300 000 Itérations (0.24 / 0.34)



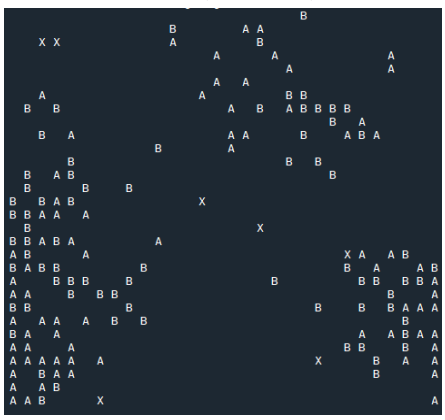
2 000 000 Itérations (0.71 / 0.72)



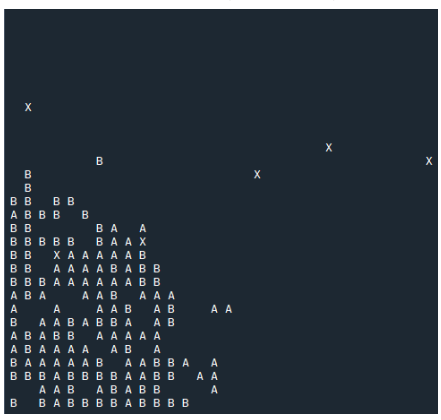
Er=0.3 / 20 000 Itérations (0.04 / 0.02)



100 000 Itérations (0.10 / 0.06)



500 000 Itérations (0.16 / 0.11)

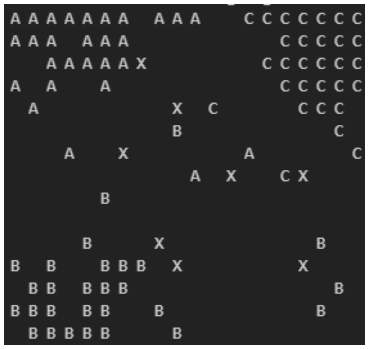


H, Introduction d'objets C

L'introduction des comportement des nouveaux comportement des agents et des objets C augmente le temps nécessaire pour atteindre une situation convenablement triées, on va don réduire la taille de l'environnement pour accélérer les processus. Les paramètres utilisés pour les test de la partie introduisant les objets C sont les suivant :
NA = 30, NB = 30, NC = 30, NR = 20, HEIGHT = 15, LENGHT = 20, Kt = 0,1, Kp = 0,3, ER = 0, M_size = 10.
Les nouveaux paramètres sont SD (Spread distance), R (Evaporation Rate), T (Wait Time), D (Ignore Duration).
Pour cette première introduction des objets C, on utilise SD = 2, R = 0,25, T = 5, D = 5

50 000 Itérations (0,51/0,47/0,32)

300 000 Itérations (0,52/0,43/0,69)



Après l'introduction des objets C, le tri se fait toujours correctement, on remarque néanmoins que le temps pour que les objets C soient triés est beaucoup plus long que pour le objets A et B bien que les nombre d'objets soient les mêmes.

I, Influence de SD

SD est un paramètre qui doit être judicieusement choisi en fonction de la taille de l'environnement, dans un grand environnement, un SD trop petit ne sera pas efficace pour obtenir de l'aide, personne ne tombera sur le signal de phéromone. Et inversement un SD trop grand sur un petit environnement va saturé l'environnement en phéromone bien que l'intensité soit décroissante, trop de robots vont venir aider et cela augmentera le temps nécessaire pour obtenir un environnement trié.

SD=4 100 000 itérations(0,54/0,54/0,49)

SD=10 100 000 itérations(0,18/0,25/0,16)

One remarque que le choix de SD=4 était probablement plus pertinent au vu de l'environnement. Au bout de 100 000 itérations, l'environnement est très bien trié, ce qui n'est pas du tout le cas lorsque SD=10. Néanmoins, ces conclusions sont à nuancer, c'est en réalité le couple DS/R qui est à considérer pour éviter la saturation de l'environnement en phéromone.
On prendra SD=4 pour la suite.

K, Influence de R

R est un facteur qui doit être choisit assez faible. Un R trop grand entraînera une trop forte évaporation des phéromones et l'aide aura du mal à arriver et un R trop faible va progressivement entraîner la saturation de l'environnement en phéromone les agents vont suivre des trajectoires porteuse d'un information obsolète.

R=0,95 100 000 itérations(0,46/0,51/0,53)

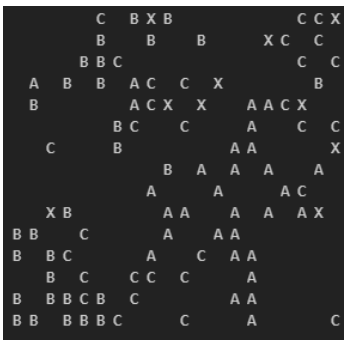
R=0,05 100 000 itérations(0,32/0,36/0,42)

Les graphes présentés à 100 000 itérations ne permettent pas de tirer de conclusions claires étant donné qu'il faut prendre en compte qu'un caractère aléatoire très important est présent. On peut néanmoins supposé que le tri semble plus lent dans le cas où l'environnement est saturé en phéromone avec un R très faible. Le cas où le signal décroît très rapidement aboutit à une situation bien triés, cela est également du à la taille de l'environnement, en l'augmentant ou en réduisant le nombre de robots, l'effet serait probablement beaucoup plus marqué.

L, Influence de T

Le temps d'attente est à déterminer en fonction de la taille de l'environnement et du nombre de robots. En ayant un temps d'attente trop faible, les robots vont avoir du mal à trié les objets C mais si ce temps est trop long, on risque d'aboutir à des situations de blocage ce qui serait catastrophique sur le temps de tri.

T=1000 100 000 itérations(0,24/0,18/0,36)



T=1 100 000 itérations(0,24/0,18/0,36)

En utilisant un temps d'attente choisi artificiellement extrêmement grand, on voit que l'environnement n'est pas très bien trié au bout de 100 000 itérations. Il va souvent y avoir des robots bloqués à attendre et qui ne participeront pas au tri. Dans le cas où T est très faible, les objets A et B vont être rapidement bien triés ce qui ne sera pas le cas des objets C. En effet, les robots ne vont pas avoir la patience d'attendre de l'aide, ils vont partir rapidement après être arrivé sur un objet C et aidé au tri des autres objets.