



Master I Informatique



Année 2005 / 2006

Rapport de projet
Intelligence Artificielle distribuée

Modèle de ségrégation de Schelling

Guillaume Blondel
Ludwig David

Professeur : Serge Stinckwich

Sommaire

Introduction.....	3
I. L'implémentation.....	4
1)La structure de la simulation.....	4
1.1)Structure de l'environnement.....	4
1.2)Structure des agents.....	4
2) Comportement des agents.....	4
2.1) Calcul de satisfaction.....	5
2.2) Déplacement.....	6
3) Interaction avec l'environnement.....	7
3.1) Indiquer sa présence aux voisins.....	7
3.2) Marquer la position.....	8
II. Étude des comportements.....	9
1) Variation de Tolérance.....	9
2)A faible occupation du terrain.....	11
3)A occupation très élevée tu terrain.....	11
4)En créant une minorité.....	12
Conclusion.....	13

Introduction

Le sujet que l'on avait choisit de traiter dans un premier temps pour ce devoir était "Algorithmes de l'informatique amorphe (amorphus computing)". Cependant l'implémentation de ce projet n'était pas réalisable avec Kedama tel qu'il est fait actuellement.

En effet, pour pouvoir faire ce sujet il nous aurait fallu modifier les sources de la plateforme, de manière à pouvoir permettre la communication de messages entre les différents agents. Ces modifications auraient pris beaucoup trop de temps et nous n'aurions pas pu les réaliser dans le court délai qui nous était donné pour réaliser ce projet. Nous avons donc décidé de mettre en oeuvre un autre projet, et avons opté pour le modèle de ségrégation de Schelling, celui-ci pouvant être réalisé en exploitant la plateforme de Kedama tel quel.

Le modèle de ségrégation de Schelling a pour but de montrer que des groupes "ségréationnistes" pouvaient apparaître, même si les préférences des habitants étaient compatibles avec la mixité de la population (ce qui est le cas avec des préférences faibles). Le but d'un agent sera par exemple : " j'accepte d'habiter avec un voisinage majoritairement différent de moi, sauf si je suis trop minoritaire". Cela se traduit par chaque agent acceptant un voisinage majoritairement différent pour peu qu'il y au moins un certain pourcentage de voisin semblable à lui même.

I. L'implémentation

Ici nous décrivons comment a été implémenté le projet. Nous allons tout d'abord expliquer le comportement des agents, puis comment ceux-ci interagissent avec l'environnement.

1) La structure de la simulation

1.1) Structure de l'environnement

La simulation possède plusieurs variables servant de paramètre pour modifier le comportement des agents. On retrouve ainsi:

- Le nombre d'agents de chaque couleurs
- La tolérance des agents de chaque couleurs
- Le nombre total d'agents insatisfait

De plus, la simulation regroupe 3 patchs servant à la détermination des positions des agents. Nous verront par la suite l'utilité des ces patchs.

1.2) Structure des agents

Les agents ne possèdent qu'une seule variable représentant leur état de satisfaction. Les autres variables ne sont que des variables temporaires permettant la bonne exécution des scripts. Il existe 2 types d'agents: Les agents « rouge » et les agents « bleu ».

2) Comportement des agents

A chaque tour d'exécution, les agents exécutent 3 actions principales:

- Le calcul de la satisfaction
- Si l'agent n'est pas satisfait, celui ci se déplace.
- L'agent indique sa présence.



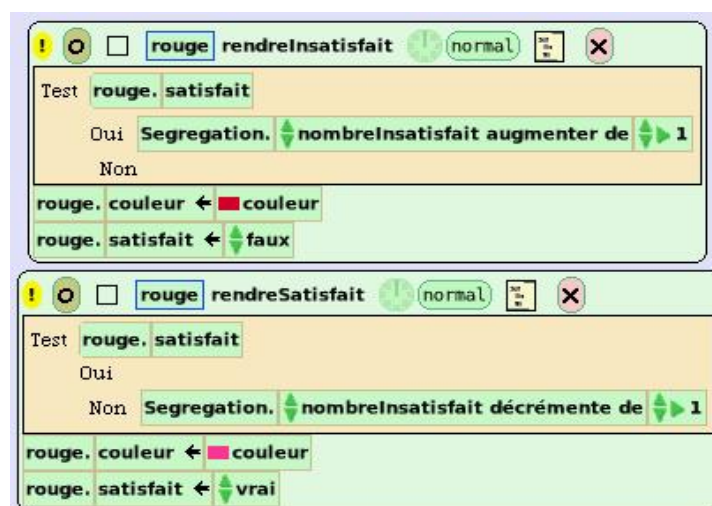
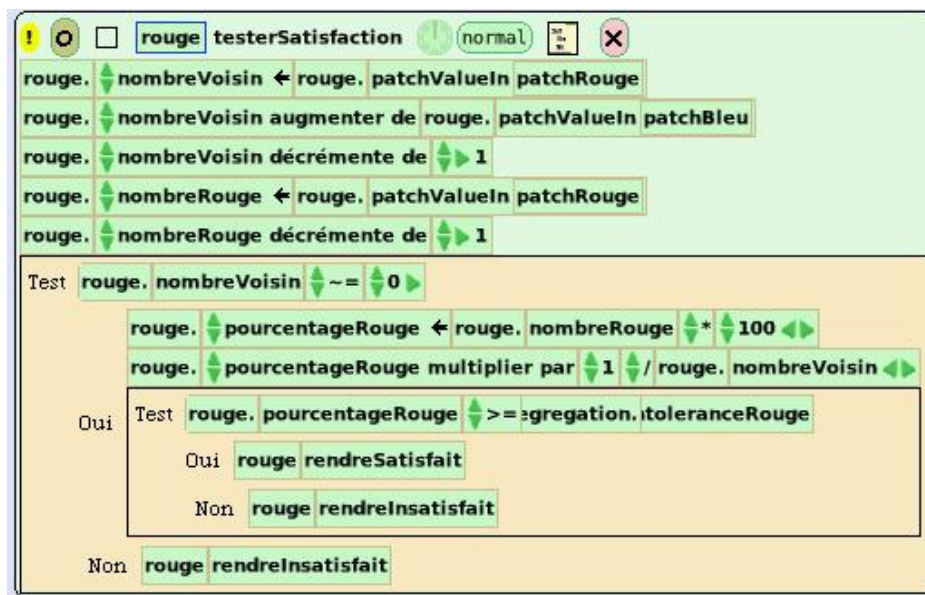
2.1) Calcule de satisfaction

Les agents possèdent une tolérance propre. L'agent comptabilise de nombre de voisins de sa couleur et le nombre total de voisins. Il calcule ensuite le pourcentage de voisin de sa couleur par rapport au nombre total.

Si ce pourcentage est supérieur ou égale à sa tolérance, celui-ci est satisfait, dans le cas contraire il devient insatisfait.

Lorsque l'agent n'a pas de voisin, il est considéré comme insatisfait.

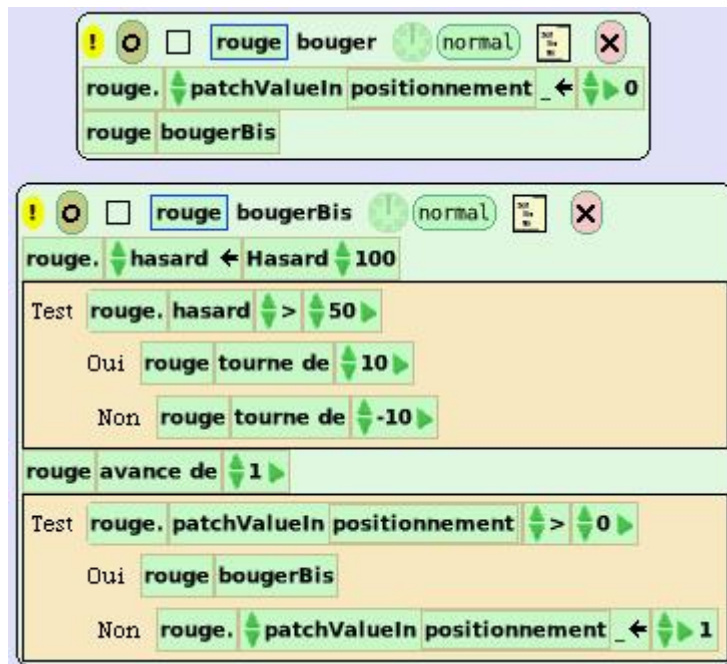
Lorsque l'agent change d'état, il change de aussi de couleur. Ce qui permet de distinguer dans la simulation quels sont les agents satisfait ou non. De plus, il incrémente et décrémente un compteur de la simulation permettant de connaître le nombre exacte d'agent insatisfait.



A chaque fois que l'agent est insatisfait, il se déplace.

2.2) Déplacement

Nous avons réalisé une fonction de déplacement qui permet d'éviter que 2 agents se superposent. Le déplacement se fait en deux étapes. Premièrement, l'agent change sa direction. Ce changement s'effectue aléatoirement par une rotation de 10 ou -10. Ensuite l'agent se déplace d'une case. Si celui-ci se trouve sur une case où il y a déjà un autre agent, il effectue un nouveau déplacement.



3) Interaction avec l'environnement

La plupart des interactions faites avec l'environnement se font par le biais de patches.

3.1) Indiquer sa présence aux voisins.

Après avoir étudié le comportement des patches de kedama nous avons remarqué que nous pouvions facilement les exploiter pour compter les voisins de chaque agent. En effet le comportement des patches est simple. Lorsque l'on donne une valeur au patch et que l'on appelle la fonction « diffuse » celui-ci divise la valeur sur le nombre de case de la diffusion.

0	0	0
0	9	0
0	0	0

=>

1	1	1
1	1	1
1	1	1

De plus, l'algorithme de diffusion additionne les valeurs lorsque plusieurs valeurs diffusées sont proches.

0	0	0	0	0
0	9	0	9	0
0	0	0	0	0

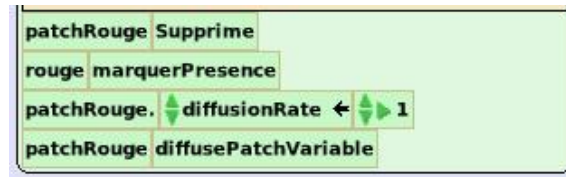
=>

1	1	2	1	1
1	1	2	1	1
1	1	2	1	1

On peut donc grâce au patch déterminer exactement le nombre de voisins. La fonction diffuse étant synchronisée entre tous les agents.

En utilisant un seul patch, il n'est pas possible de connaître le nombre de voisins de chaque couleur. Nous avons donc utilisé 2 patches. Ce qui nous permet de distinguer le nombre d'agents selon la couleur.





3.2) Marquer la position.

Afin d'empêcher les agents de se mettre sur la même case qu'un autre agent, nous avons utilisé un patch afin de marquer leur position. A l'initialisation des agents, ceux-ci commencent par contrôler la valeur du patch de position, si celui-ci a pour valeur 0, l'agent ne bouge pas et change la valeur à 1. Dans le cas contraire, il bouge tant que la valeur n'est pas 0.

Par la suite, avant chaque déplacement, l'agent passe la valeur du patch à 0, puis commence le déplacement. De même, tant que la case sur la quel arrive l'agent ne possède pas un patch de position à 0, celui-ci bouge.

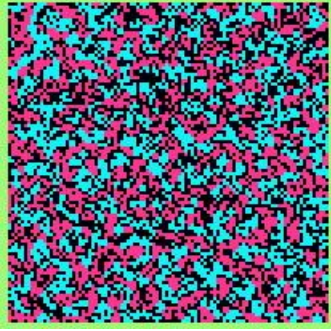


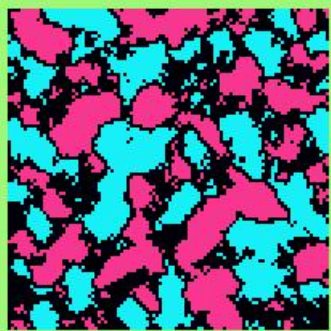
II. Étude des comportements.

Nous avons réalisé un grand nombre de test pour étudier les différents comportements et le temps de stabilisation de la simulation.

1) Variation de Tolérance.

Pour cette première observation, nous avons décider de régler le nombre d'agents a 3000 de chaque couleur, ce qui représente une occupation du terrain de 60%. Cela évite que la stabilisation soit trop longue.

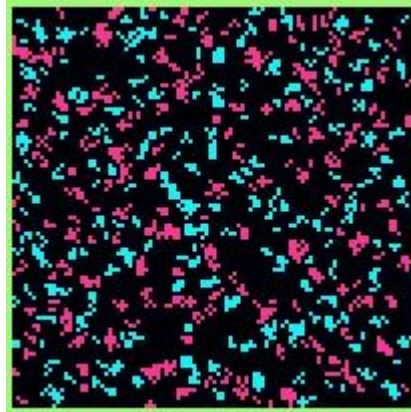
Nous avons pu remarquer que plus on augmente l'intolérance des agents, plus la stabilité est longue à obtenir. De même, les groupes d'agents stables sont de plus en plus gros. On obtient les résultats suivants:

<i>Tolérance des agents</i>	<i>Résultat</i>	<i>Nombre d'itérations</i>
50 %		19
60%		40
70 %		80
80 %		370

A 90%, la simulation ne se stabilise pas, en effet, même si de petit noyaux stable commence a se créer, le simple passage d'un agents de l'autre couleur suffit a rendre les agents insatisfait.

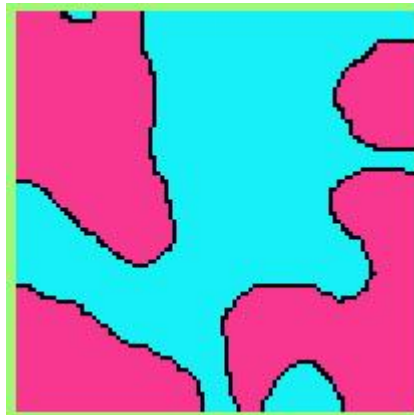
2)A faible occupation du terrain

L'analyse n'est pas très intéressante à faible occupation du terrain, car le système se stabilise assez facilement. On observe des petits groupes d'agent repartit un peu partout. Nous pouvons observer ici le résultat obtenu avec une occupation du terrain de 20%.



3)A occupation très élevée du terrain.

Lorsque l'occupation du terrain est très élevée, la stabilisation est assez longue. Cependant, la simulation met beaucoup de temps à se stabiliser à cause d'un petit nombre d'agents cherchant une place. La plus grande partie forme un grande surface stable assez rapidement. On peut voir le résultat avec 4800 agents de chaque couleur (96% d'occupation):

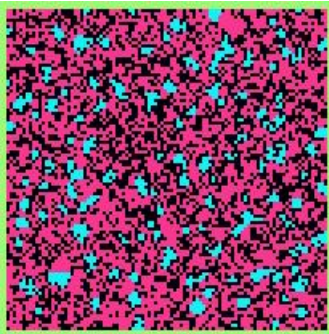

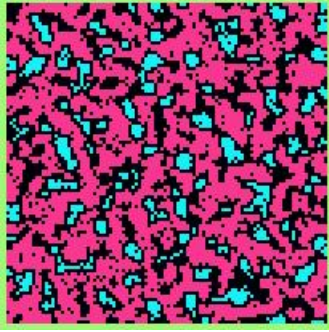
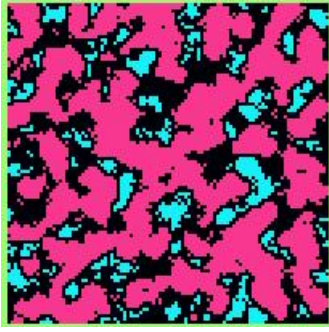


On remarque que les agents forme des groupes très gros. Cependant cette stabilité ne peut s'obtenir avec une tolérance supérieure à 70%.

4) En créant une minorité

Après avoir testé avec un nombre d'agent identique, nous avons réaliser de nouveau test, en partant sur la base d'occupation de terrain à 60%, mais en créant une minorité. Nous avons donc répartis les agents de la sorte: 5000 rouge, 1000 bleus.

Les observation sont les même que précédemment. Le temps de stabilisation et la taille des groupes formés augmente avec la tolérance des agents. A 90% la stabilisation est impossible:

<i>Tolérance des agents</i>	<i>Résultat</i>	<i>Nombre d'itérations</i>
50%		54
60%		90
70%		91
80%		200

Conclusion

Ce projet nous a permis de découvrir le développement sur une plateforme de simulation de système multi-agents. Bien que kedama plus ciblé pour des projets où les agents utilisent beaucoup leurs phéromones pour communiquer et répartir les actions qu'ils doivent accomplir, nous avons réussi à nous adapter et à mettre en œuvre sous kedama un modèle de ségrégation de Schelling.

Nous avons également pu lors de la réalisation de ce devoir apprendre à observer et analyser le comportement des agents. Cela nous a permis de mieux cerner et comprendre la réelle utilité des systèmes multi-agents.