RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
Ministre de l'Enseignement Supérieur et de la Recherche Scientifique
Université d'El-Oued

Faculté des Sciences et Technologie
Département de Mathématiques et d'Informatique

Order №:
№ Series:

Mémoire de Licence LMD

Option : Fondamentale d'Informatique

Simulation des comportements d'agent autonome, implémentation sur plate-forme NetLogo

Réalisé par:
KIRAM Younès

Soutenu le 05 juin 2013

Membres du jury :

Examinateur 1	**:**	**Mlle. KHOLADI Nadjoua Houda**	**Université d'El-Oued**
Examinateur 2	**:**	**Mr. BEN ALI AbdelKamel**	**Université d'El-Oued**
Encadreur	**:**	**Mme. Gueia Sana Sahar**	**Université d'El-Oued**

2012/2013

Remerciement

Je remercie, tout d'abord, Dieu le tout puissant, pour avoir guidé vers un avenir inchallah promoteur et m'a donné le pouvoir et la patience pour terminer ce modeste travail.

Allah soit loué

Je tiens tout particulièrement à exprimer mes profonds remerciements et mes respects gratitudes à ma chère promotrice ***M^{me}. Gueia Sana Sahar*** *qui a été vraiment patiente avec moi et qui m'a fourni de précieux conseils durant mon travail afin de mener à bien ce projet.*

*Je tiens à adresser mes vifs et sincères remerciements aux membres de jury (****Mlle. KHOLADI Nadjoua Houda*** *et* ***Mr. BEN ALI AbdelKamel****) qui ont bien voulu me faire l'honneur d'être avec moi pour évaluer ma mémoire.*

Comme je ne voudrai pas manquer de remercier ***Mr. LEDJDAL Ibrahim*** *qui m'a proposé de prendre ce thème de mémoire au moment de la sélection de thème de mémoire.*

Enfin, j'adresse mes plus sincères remerciements à toute les enseignants et toute qui ont contribué de près ou du loin à moi amener à ce stade.

Dédicace

Tout d'abord je dédie cette mémoire à mes parents :

- ma mère qui m'a aidé tout le long de mon étude et me dit toujours «vas-y en avant ! ne recule pas ! les jours passent, si tu t'arrêtes ça veut dire, tu es en retrait »

- mon père qui m'a encouragé par tous les moyens, et me dit toujours : «dans ce monde il n'y a pas de facile, reste debout ».

- tous mes frères (Yahya et Sadok) et mes sœurs (Anissa, Amira, Assila).

- Mes cousins

- Mes chers amis (Hamoud, Tarek, Zaki, Hamid, Hamza, Tedjani, Rida, ...).

Résumé

La simulation est devenu un outil très important dans la vie et après cela s'est avéré être le meilleur moyen de faire des expériences sans avoir à détruire ou de changer quelque chose dans la réalité.

L'objectif de l'achèvement de cette mémoire est d'essayer d'accomplir l'une de ces simulations et d'apprendre à configurer et terminé à une expérience très complète et ce afin d'accroître les connaissances sur ce sujet et de comprendre son importance pour l'humanité dans divers domaines, en utilisant un programme de simulations nommé NetLogo par citation sur le teint fourmis collective qui sera la base simulations.

Mots clefs:

- Simulation
- Fourmi
- Netlogo
- Comportments
- Outil de simulation

Abstract

Simulation has become a very important tool in life and after that proved to be the best way to do experiments without having to destroy or change anything in reality.

The objective of the completion of this note is to try to accomplish one of these Simulations and learn how to configure and completed to a very complete and experience and this in order to increase knowledge of this subject and understand its importance to mankind in various magazines, using a program Simulations Net Joe and citation on complexions collective ants that will be the basis Simulations.

Keywords:

- Simulation
- Ant
- NetLogo
- Behaviors
- Simulation tool

ملخص

اصبحت المحاكاة وسيلة مهمة جدا في الحياة وذلك بعد أن أثبتت أنها أحسن وسيلة للقيام بالتجارب دون الإضطرار لإتلاف أو تغيير أي شئ في الواقع

الهدف من إتمام هذه المذكرة هو محاولة إنجاز احد هذه المحاكاة و التعرف على كيفية تهيئتها وإنجازها الى غاية إتمامها و تجربتها وهذا بغرض زيادة المعرفة بهذا الموضوع وإدراك أهميتها للبشرية في شتى المجلات، وذلك بالإستعانة ببرنامج المحاكاة نت لوجو و الإستناد على السلوكيات الجماعية للنمل التي ستكون أساس المحاكاة.

كلمات مفتاحية:

- محاكاة
- نملة
- نت لوجو
- السلوكيات
- برنامج المحاكاة

Table des matières

Table des illustrations

Introduction Générale

Nous avons réalisé dans ce mémoire la simulation de comportement d'agents autonomes à la recherche d'objets spécifiques dans l'environnement, exactement la simulation de comportement de fourmis à la recherche de nourriture. Le comportement des fourmis est un comportement collectif. Chaque fourmi a pour priorité le bien être de la communauté.

En observant une colonie de fourmis à la recherche de nourriture dans les environs du nid, on s'aperçoit qu'elle résout des problèmes tels que celui de la recherche du plus court chemin. Les fourmis résolvent des problèmes complexes par des mécanismes assez simples à modéliser .Il est ainsi assez simple de simuler leur comportement par des algorithmes.

Pour simuler le comportement de colonie de fourmis, on utilise le système multi agents, les systèmes multi-agents apportent une solution radicalement nouvelle au concept même de modèle et de simulation dans les sciences de l'environnement, en offrant la possibilité de représenter directement les individus, leurs comportements et leurs interactions.

La simulation multi-agents est fondée sur l'idée qu'il est possible de représenter le comportement d'un individu par un processus informatique, c'est à dire par un élément de programme disposant de sa propre autonomie opératoire.

Chaque individu peut être alors représenté informatiquement sous la forme d'un agent, c'est à dire un processus informatiques autonome, capable de percevoir et de réagir aux transformations de l'environnement, son comportement étant défini pour toutes les phases de son évolution (naissance, recherche de nourriture et de partenaire, reproduction et mort) avec tous les détails nécessaires.

L’application a été réalisée sur la plate-forme Netlogo qui est un langage de simulation multi-agents, c’est un langage simple et qui permet à la fois de créer des choses assez complexes avec une interface graphique adaptée.

Plant de travail

Ce mémoire se compose de 3 chapitres :

Chapitre 1 : Le premier chapitre est consacré à l'étude des systèmes multi-agents et a pour objectif de situer son concept, en particulier les systèmes multi-agents réactifs qui permettent de modéliser des phénomènes collectifs.

Chapitre 2 : Les fourmis résolvent des problèmes complexes par des mécanismes assez simples à modéliser. Dans ce chapitre est étudié le comportement des fourmis en commençant par les fourmis réelles et en arrivant aux fourmis artificielles.

Chapitre 3 : Ce chapitre est destiné à la modélisation de l'application, en se basant sur les principes et les concepts vus dans les chapitres précédents, ensuite on présente l'implémentation proprement dite par la description de l'application réalisée, la réalisation a été faite sur la plate-forme Netlogo, qui est une plate-forme dédiée à la simulation multi-agents.

Chapitre 1

Agents et Système multi-agents

Introduction

De nos jours, le mot «agent» est utilisé dans plusieurs domaines et, de ce fait, plusieurs sens lui sont attachés. D'ailleurs, même à l'intérieur du domaine de l'informatique, plusieurs chercheurs ont défini le concept d'agent de manières différentes.

Tout d'abord, nous tenons à mentionner que les agents peuvent être conçus de plusieurs manières différentes. Ce sont les caractéristiques de l'environnement qui influencent le choix de conception. Si les agents sont dans un environnement en constants changements et qu'ils doivent, par conséquent, réagir très vite aux événements de l'environnement, alors une architecture réactive est appropriée. Ce type d'agent réagit très vite, car il ne fait qu'appliquer des règles prédéfinies pour choisir ses actions.

Toutefois, si l'environnement exige que l'agent raisonne pour atteindre son but, alors une architecture délibérative est plus appropriée. Les agents délibératifs peuvent raisonner sur leur but ou à l'aide d'une certaine fonction d'utilité pour choisir l'action qui les satisfait le plus.

Nous pouvons même avoir des architectures qui mélangent ces deux extrêmes pour tenter de retirer les avantages de chacune d'elles. Ces architectures hybrides permettent à

l'agent d'avoir un comportement réactif lorsque les situations l'exigent et un comportement délibératif dans d'autres circonstances.

La plupart du temps, un agent n'est pas seul dans son environnement, il y a d'autres agents présents autour de lui. Les agents doivent, par conséquent, être capables d'interagir entre eux. Ils peuvent soit coexister, coopérer ou être en compétition. S'ils ne font que coexister, alors chaque agent ne fait que considérer les autres agents comme des composantes de l'environnement. S'ils coopèrent, alors les agents doivent pouvoir communiquer et se coordonner pour agir efficacement ensemble. S'ils sont en compétition, alors les agents doivent être en mesure de négocier. Un système où évoluent plusieurs agents est appelé système multi-agent et il possède généralement plusieurs caractéristiques intéressantes, comme le parallélisme, la robustesse et l'extensibilité.

Ce chapitre se veut une introduction au concept d'agent pour établir les bases de ce mémoire. Il débute par une définition du concept d'agent. Il présente par la suite quelques architectures d'agents. Il introduit finalement les systèmes multi-agents brièvement, car ils feront l'objet d'une présentation plus approfondie dans les chapitres suivants.

1.1 C'est quoi un agent

Dans les dernières années, le concept d'agent a été utilisé et étudié dans plusieurs domaines. Toutefois, il n'y a encore aucun consensus, entre les différents chercheurs, quant à la définition même du mot «agent». Selon Nwana [1], il y a au moins deux raisons qui permettent d'expliquer cette difficulté.

La première réside dans le fait que les chercheurs, dans le domaine des agents, ne sont pas à l'origine de ce terme comme l'ont été, par exemple, les chercheurs dans le domaine de la logique floue. En effet, le terme agent a été et continue d'être utilisé dans la vie de tous les jours par des personnes travaillant dans des domaines très différents. Par exemple, on parle d'agent de voyage, d'agent immobilier, d'agent d'assurance, etc.

La deuxième raison est que même dans la communauté des chercheurs sur les agents logiciels, le mot «agent» est utilisé pour décrire des systèmes très différents les uns des autres. Pour ajouter à la confusion, les chercheurs sont allés même jusqu'à inventer plusieurs synonymes au mot «agent». Ils ont ainsi inventé, par exemple, «knowbots» (robots à base de connaissances), «softbots» (robots logiciel), «taskbots» (robots à base de tâche), «userbots» (robots pour utilisateur), robots, agents personnels, agents autonomes, assistants personnels, etc. Il est vrai qu'une telle prolifération de termes trouve sa justification dans le fait que les agents peuvent prendre différentes formes physiques (robot ou agent logiciel) et qu'ils peuvent aussi jouer plusieurs rôles.

Cela dit, il est tout de même important de s'entendre sur une définition du terme agent pour que les exposés qui suivent dans ce mémoire aient un sens. La définition que nous avons adoptée, et qui semble couvrir les caractéristiques des agents que nous avons développés, est celle proposée par Jennings, Sycara et Wooldridge [2] :

Un agent est un système informatique, situé dans un environnement, qui agit d'une façon autonome et flexible pour atteindre les objectifs pour lesquels il a été conçu.

Il y a trois concepts clés présents dans cette définition :

- Situé signifie que l'agent peut recevoir des entrées sensorielles provenant de son environnement et qu'il peut effectuer des actions qui sont susceptibles de changer cet environnement. Le monde réel et l'Internet sont des exemples d'environnements où les agents peuvent être situés.

- Autonome signifie que l'agent est capable d'agir sans l'intervention directe d'un humain (ou d'un autre agent) et qu'il a le contrôle de ses actions et de son état interne.

- Flexible signifie que l'agent est :

    - Capable de répondre à temps : il peut percevoir son environnement et répondre rapidement aux changements qui s'y produisent.

- Proactif : il n'agit pas simplement en réponse à son environnement, il est également capable d'avoir un comportement opportuniste, dirigé par ses buts ou sa fonction d'utilité, et prendre des initiatives au moment approprié.

- Social : il est capable d'interagir avec les autres agents (artificiels ou humains) afin de compléter ses tâches ou aider les autres à compléter leurs tâches.

Bien entendu, certains agents auront des caractéristiques additionnelles et, pour certains types d'applications, certains attributs seront plus importants que d'autres. Par contre, c'est la présence de tous ces attributs dans une seule entité logicielle qui procure la force au paradigme agent et qui le distingue des autres paradigmes logiciels tels que les systèmes orientés objets, les systèmes distribués et les systèmes experts.

1.2 Historique

La majeure partie des composantes d'un agent autonome provient de l'intelligence artificielle. On serait donc porté à croire que la construction d'un agent autonome serait au centre des problèmes étudiés par l'intelligence artificielle (IA). Ce n'est cependant pas le cas. En effet, avant les années 80, la recherche en intelligence artificielle s'est plutôt concentrée sur les diverses composantes d'un agent, en supposant qu'on pourrait un jour assembler ces différentes parties afin de construire un agent de manière assez directe. Durant cette période, le champ d'intérêt qui a le plus en commun avec les agents autonome est celui de la planification.

1.3 Qu'elles sont les architectures d'agent ?

Il existe plusieurs manières de concevoir des agents, mais peu importe l'architecture adoptée, un agent peut toujours être vu comme une fonction liant ses perceptions à ses actions (voir figure 1.1). Plus précisément, un agent perçoit l'environnement à l'aide de ses capteurs et il agit sur son environnement à l'aide de ses effecteurs. Ce qui différencie les

différentes architectures d'agents, c'est la manière dont les perceptions sont liées aux actions.

Les architectures d'agents peuvent être regroupées en agents réactifs et agents délibératifs comme suit :

- les agents réactifs :
  - les agents à réflexes simples
  - les agents conservant une trace du monde
- les agents délibératifs :
  - les agents ayant des buts
  - les agents utilisant une fonction d'utilité
  - les agents du type BDI «Beliefs-Desires-Intentions»

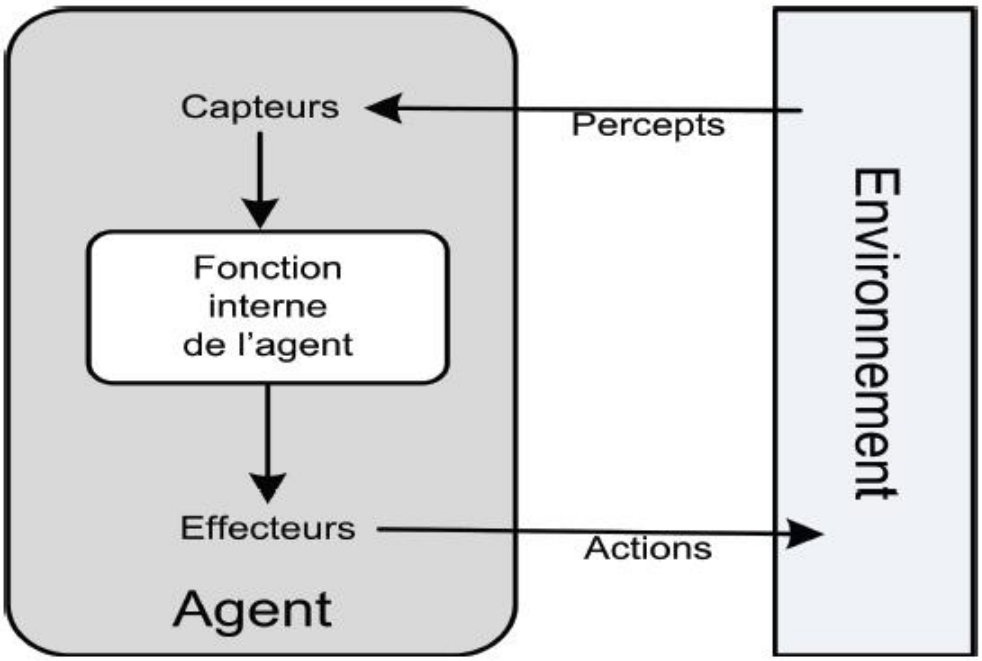

Figure 1-1-Interaction d'un agent avec son environnement.

1.3.1 Agents réactifs

La Comme son nom l'indique, un agent réactif ne fait que réagir aux changements qui surviennent dans l'environnement. Autrement dit, un tel agent ne fait ni délibération ni planification, il se contente simplement d'acquérir des perceptions et de réagir à celles-ci en appliquant certaines règles prédéfinies. Étant donné qu'il n'y a pratiquement pas de raisonnement, ces agents peuvent agir et réagir très rapidement.

Il n'est pas nécessaire que les agents réactifs soient intelligents individuellement pour que le système ait un comportement global intelligent. La structure du système émerge des comportements et non d'une volonté d'organisation.

Il convient de remarquer que les humains aussi utilisent cette manière d'agir. Dans plusieurs situations, il est souvent préférable de ne pas penser et de réagir immédiatement. Par exemple, lorsqu'une personne met la main sur une plaque très chaude, elle ne commence pas à se demander si c'est chaud, si ça fait mal et s'il faut ou non qu'elle retire sa main. Dans ce cas, elle retire sa main immédiatement, sans réfléchir, et c'est cette rapidité de réaction qui lui permet de diminuer les blessures. Cet exemple montre bien que ce type de comportement réflexe est essentiel pour les êtres humains. De la même manière, il est aussi essentiel pour les agents s'ils veulent pouvoir agir dans le monde réel.

Les deux sous-sections suivantes décrivent deux modèles qui peuvent servir à la conception d'agents réactifs.

1.3.1.1 *Agents à réflexes simples*

Ce type d'agent agit uniquement en se basant sur ses perceptions courantes. Il utilise un ensemble de règles prédéfinies, du type Si condition alors action, pour choisir ses actions. Par exemple, pour un agent en charge du contrôle de la défense d'une frégate, on pourrait avoir la règle suivante :

Si missile-en-direction-de-la-frégate alors lance -intercepteur-de-missile

Comme on peut le constater, ces règles permettent d'avoir un lien direct entre les perceptions de l'agent et ses actions. Le comportement de l'agent est donc très rapide, mais peu réfléchi. À chaque fois, l'agent ne fait qu'exécuter l'action correspondant à la règle activée par ses perceptions. Si on se réfère à la figure 1.1, la fonction interne de l'agent (FIA) pourrait dans ce cas évaluer le percept en vue d'évaluer l'état du monde environnant et de déterminer, partant de là, l'action qu'il convient d'effectuer. Une approche plus générale et plus souple consiste à construire un interpréteur généraliste des règles du type [condition-action] et à créer ensuite des ensembles de règles pour des environnements de tâches spécifiques. Dès lors, il revient à la FIA d'unifier le percept actuel à une paire condition-action et d'enclencher ensuite l'action conséquente.

Ce type d'agent présente l'avantage d'être fort simple mais en pratique il est très limité. En effet, un tel agent ne peut fonctionner correctement que s'il peut prendre la bonne décision sur la base du seul percept courant détecté, c'est à dire seulement si l'environnement est entièrement observable. Le moindre aspect incertain à propos de cette observation entière risque de poser de sérieux problèmes. Par exemple, la règle précédente à propos de la survie d'une frégate, peut (i) ne jamais lancer, (ii) lancer continuellement, (iii) lancer sans nécessité des missiles d'interception, et ce dans le cas où la perception du missile en direction de la frégate peut être évalué avec une certaine marge d'erreur.

1.3.1.2 Agents conservant une trace du monde

Le type d'agent qui a été décrit précédemment, ne peut fonctionner que si un tel agent peut choisir ses actions en se basant uniquement sur sa perception actuelle. Par exemple, si le radar de la frégate détecte un missile et que l'instant d'après, il le perd de vue, dû à un obstacle, cela ne signifie nullement qu'il n'y a plus de missile. Dès lors, l'agent en charge du contrôle de la frégate doit tenir compte de ce missile dans le choix de ses actions et ce, même si le radar ne détecte plus le missile. Un tel problème survient parce que les capteurs de l'agent ne fournissent pas une vue complète du monde. Pour régler ce problème, l'agent doit maintenir des informations internes sur l'état du monde dans le but d'être capable de distinguer deux situations qui ont des perceptions identiques, mais qui, en réalité, sont différentes. L'agent doit pouvoir faire la différence entre un état où il n'y a pas de missile et

un état où le missile est caché, même si ses capteurs lui fournissent exactement les mêmes informations dans les deux cas.

Pour que l'agent puisse faire évoluer ses informations internes sur l'état du monde, il a besoin de deux types d'information. Tout d'abord, il doit avoir des informations sur la manière dont le monde évolue, indépendamment de l'agent. Par exemple, il doit savoir que si un missile avance à une vitesse de 300 m/s, alors 5 secondes plus tard, il aura parcouru 1500 mètres.

L'agent doit avoir ensuite des informations sur la manière dont ses propres actions affectent le monde autour de lui. Si la frégate tourne, l'agent doit savoir que tout ce qui l'environne tourne aussi. Il doit donc mettre à jour la position relative de tous les objets autour de la frégate.

Dès lors, la structure d'un tel agent pourrait être vue comme une amélioration de l'agent à réflexes simples précédent et elle viserait à améliorer le processus d'évaluation de l'environnement à l'instant t par :

- une trace du monde reflétant l'instant t – 1.
- une dynamique qui indiquerait comment le monde évolue.
- une fonction d'évaluation de l'impact des actions de l'agent.

1.3.2 Agents délibératifs

Les agents délibératifs sont des agents qui effectuent une certaine délibération pour choisir leurs actions. Une telle délibération peut se faire en se basant sur les buts de l'agent ou sur une certaine fonction d'utilité. Elle peut prendre la forme d'un plan (ou d'une politique) qui reflète la suite d'actions que l'agent doit effectuer en vue de (i) réaliser son but ou (ii) maximiser l'espérance de son utilité sur un certain horizon qui pourrait être fini ou infini.

1.3.2.1 Agents ayant des buts

Dans la section précédente, les agents utilisaient leurs connaissances sur l'état actuel de l'environnement pour choisir leurs actions. Toutefois, dans plusieurs situations, cela peut s'avérer insuffisant pour prendre une décision sur l'action à effectuer. Par exemple, on ne peut pas se fier uniquement à l'état actuel de l'environnement pour déterminer la direction que la frégate doit prendre, tout simplement parce que cela dépend aussi de l'endroit où on veut se rendre. Donc, l'agent a besoin, en plus de la description de l'état actuel de son environnement, de certaines informations décrivant ses buts. Lesquels buts peuvent être vus comme des situations désirables pour l'agent, par exemple, l'arrivée au port d'Halifax pour l'exemple de la frégate.

Par la suite, l'agent peut combiner les informations sur ses buts avec les informations sur les résultats de ses actions pour choisir les actions qui vont lui permettre d'atteindre ses buts. Cela peut être facile lorsque le but peut être satisfait en exécutant seulement une action, mais cela peut aussi être beaucoup plus complexe si l'agent doit considérer une longue séquence d'actions avant d'atteindre son but. Dans ce dernier cas, il doit utiliser des techniques de planification pour prévoir les actions devant l'amener à son but.

Contrairement aux agents réactifs, les agents délibératifs, qui raisonnent sur les buts, tiennent compte d'une certaine projection dans le futur. Ils se posent des questions comme «Qu'est-ce qui va arriver si je fais telle ou telle action ?» et «Est-ce que je serai satisfait si cela se produit ? ». Bien entendu, l'agent raisonnant sur ses buts prend, en général, beaucoup plus de temps à agir qu'un agent réactif. Il offre en revanche beaucoup plus de flexibilité. Par exemple, si nous voudrions changer de destination, il faudrait changer toutes les règles de l'agent réactif, tandis que pour l'agent ayant des buts, nous ne changerions que le but.

La figure 1.2 montre la structure d'un agent basé sur les buts. Comme on peut le constater, il est identique à l'agent réactif gardant une trace de l'environnement, sauf qu'il se projette dans le futur pour voir l'impact de ses actions et qu'il choisit ses actions en se basant sur ses buts, contrairement à l'agent réactif qui ne faisait qu'appliquer des règles prédéfinies pour relier ses perceptions à ses actions.

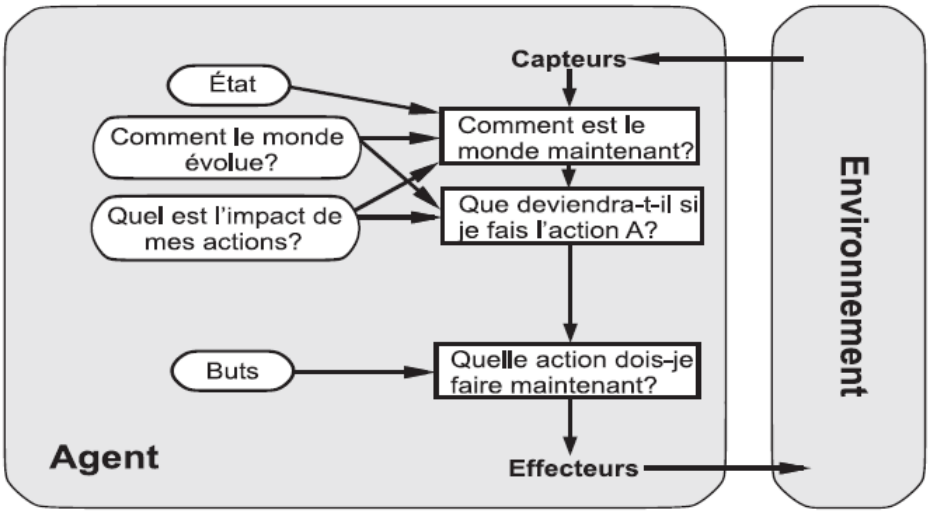

Figure 1-2 -Schéma d'un agent ayant des buts [1].

1.3.2.2 Agents utilisant une fonction d'utilité

Dans plusieurs situations, les buts ne sont pas suffisants pour générer un comportement de haute qualité. Par exemple, s'il y a plusieurs chemins possibles pour atteindre le port, certains seront plus rapides et d'autres plus dangereux. Dans cette situation, l'agent raisonnant uniquement sur ses buts n'a pas de moyens pour choisir le meilleur chemin, son seul but étant de se rendre à destination. En effet, les buts ne procurent qu'une simple distinction entre les états où l'agent est satisfait et les états où il ne l'est pas. En fait, l'agent doit plutôt s'appuyer sur une manière plus fine d'évaluer les états pour être en mesure de reconnaître pour chacun des états son degré de satisfaction. Pour cela, on dit que l'agent va préférer un état à un autre si son utilité est plus grande dans le premier état que dans le deuxième.

Généralement, l'utilité est une fonction qui attribue une valeur numérique à chacun des états. Plus l'état a une grande valeur, plus il est désirable pour l'agent. Dès lors, la spécification d'une fonction d'utilité permet à l'agent de prendre des décisions rationnelles dans deux types de situations où le raisonnement sur les buts échoue. Ainsi, par exemple, lorsqu'il y a des buts conflictuels qui ne peuvent pas être satisfaits en même temps (par

exemple, la vitesse et la sécurité), la fonction d'utilité spécifie le compromis approprié entre les différents buts. De même, lorsqu'il y a plusieurs buts possibles, mais qu'aucun d'eux ne peut être atteint avec certitude, la fonction d'utilité permet de pondérer la chance de succès avec l'importance de chacun des buts.

1.3.2.3 Agents BDI

L'architecture BDI est une autre approche utilisée dans la conception des agents délibératifs. BDI est un acronyme qui signifie, en anglais, Belief, Desire, Intentions. Ce qui se traduit en français par croyances, désirs et intentions. Les agents se basent donc sur ces trois aspects pour choisir leurs actions. Dans ce cadre, Wooldrige [2] a proposé, partant des idées de Bratman [3], une architecture ayant sept composantes, telles que présentées sur la figure1.3 :

- Un ensemble de croyances courantes, représentant les informations que l'agent possède à propos de son environnement courant.
- Une fonction de révision des croyances (fonction brf()), qui prend les entrées des capteurs et les croyances actuelles de l'agent et qui détermine un nouvel ensemble de croyances.
- Une fonction de génération des options (fonction options()), qui détermine les options disponibles pour l'agent (i.e. ses désirs), en se basant sur les croyances courantes de l'agent à propos de son environnement et sur ses intentions courantes.
- Un ensemble de désirs, représentant les options disponibles à l'agent.
- Une fonction de filtre (fonction filtre()), qui représente le processus de délibération de l'agent et qui détermine les intentions de l'agent en se basant sur ses croyances, ses désirs et ses intentions courantes.
- Un ensemble d'intentions structurées (fonction plan()) courantes, représentant le centre d'attention actuel de l'agent, c'est-à-dire les buts envers lesquels il s'est engagé et envers lesquels il a engagé des ressources.

- Une fonction de sélection des actions, qui détermine l'action à effectuer en se basant sur les intentions courantes de l'agent.

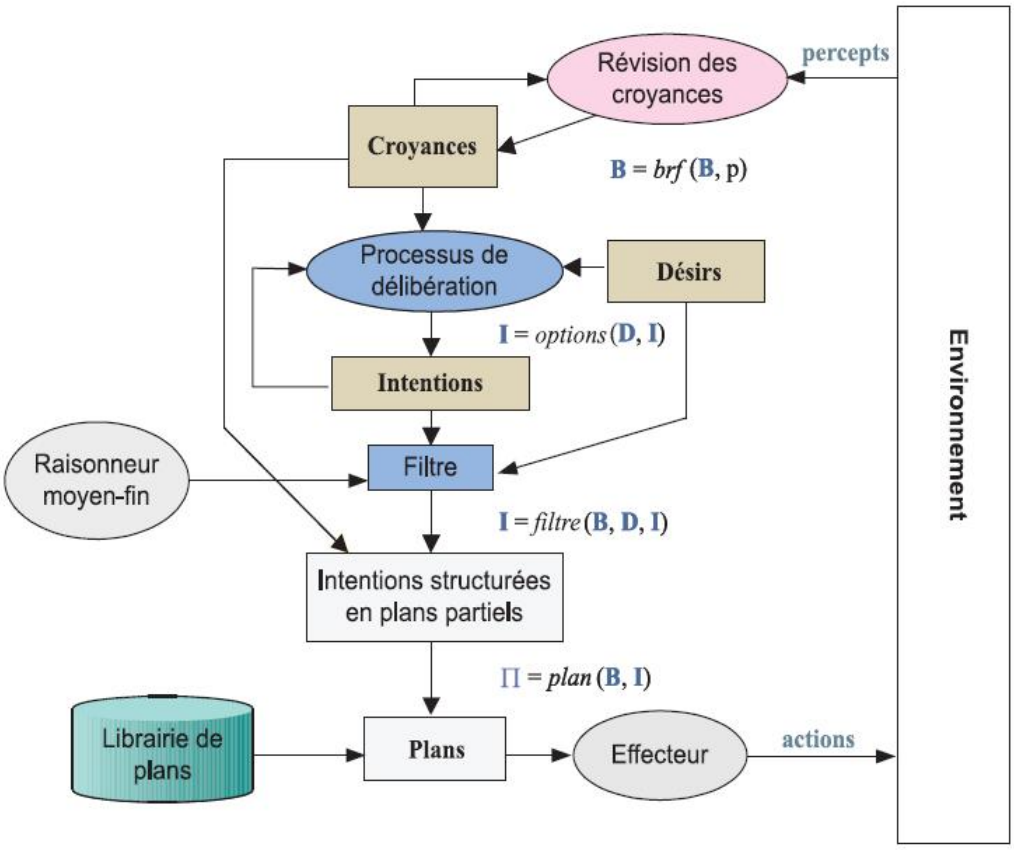

Figure 1-3 -Architecture d'agent BDI [2].

En résumé, un agent BDI doit donc mettre à jour ses croyances avec les informations qui lui proviennent de son environnement et décider quelles options lui sont offertes, filtrer ces options afin de déterminer de nouvelles intentions et poser ses actions en se basant sur ses intentions. Ainsi, ce type d'architecture fait intervenir une analyse du type «moyens-fins» en mettant sur la balance les différentes alternatives dans le but de décider quoi faire et comment le faire.

Il convient de remarquer qu'à l'inverse des architecture délibératives précédentes faisant intervenir buts et/ou utilité et qui sont plus orientées «action rationnelle», l'architecture BDI met l'emphase beaucoup plus sur les croyances et leur révision et est donc plus orientée «pensée rationnelle».

1.3.3 Agents hybrides

Les sections précédentes ont présenté deux types d'architectures : réactive et délibérative. Chacune de ces architectures est appropriée pour un certain type de problème. Pour la majorité des problèmes cependant, ni une architecture complètement réactive, ni une architecture complètement délibérative n'est appropriée. Comme pour les humains, les agents doivent pouvoir réagir très rapidement dans certaines situations (comportement réflexe), tandis que dans d'autres, ils doivent avoir un comportement plus réfléchi.

Dans ce cas, une architecture conciliant à la fois des aspects réactifs et délibératifs est requise. On parle alors d'architecture hybride, dans laquelle on retrouve généralement plusieurs couches logicielles. Les couches peuvent être arrangées verticalement (seulement une couche a accès aux capteurs et aux effecteurs) ou horizontalement (toutes les couches ont accès aux entrées et aux sorties), voir figure 1.4.

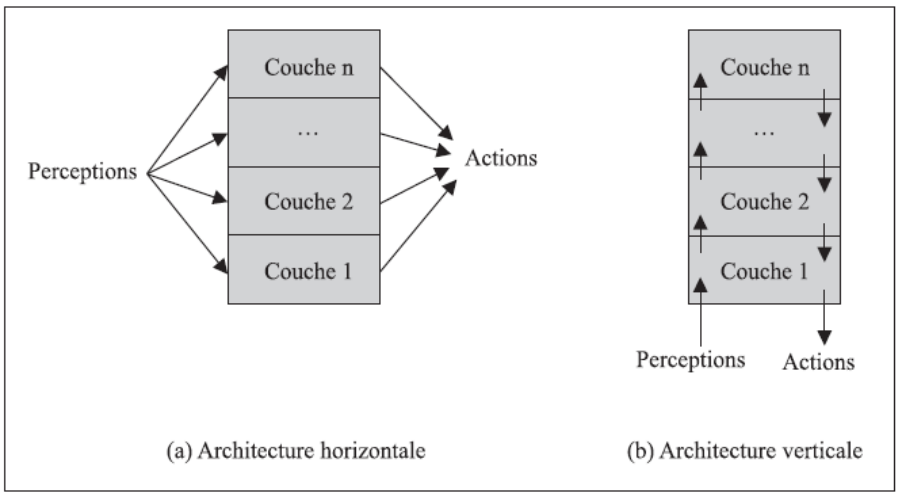

Figure 1-4 -Architectures d'agents en couches [4].

Dans ce type d'architecture, les couches sont arrangées de manière hiérarchique. Les différents niveaux de la hiérarchie traitent les informations provenant de l'environnement à différents niveaux d'abstraction. La plupart des architectures considèrent que trois couches suffisent amplement [5]. Ainsi, au plus bas niveau de l'architecture, on retrouve habituellement une couche purement réactive, qui prend ses décisions en se basant sur des données brutes en provenance des capteurs. La couche intermédiaire fait abstraction des

données brutes et travaille plutôt avec une vision qui se situe au niveau des connaissances de l'environnement. Finalement, la couche supérieure se charge des aspects sociaux de l'environnement.

Dans cette dernière couche, on retrouve généralement une représentation des autres agents (leurs buts, leurs croyances, etc.). Pour produire le comportement global de l'agent, ces trois couches interagissent ensemble. Ces interactions varient beaucoup d'une implémentation à une autre, et c'est la raison pour laquelle elles ne sont pas décrites ici.

1.4 Agents et apprentissage

L'idée derrière l'apprentissage, c'est que les perceptions de l'agent ne devraient pas être utilisées uniquement pour choisir des actions, mais également pour améliorer l'habilité de l'agent à agir dans le futur. Une telle amélioration est une forme d'apprentissage et est très importante car c'est elle qui lui permet d'évoluer, de s'adapter et de s'améliorer.

À la section 1.4.1, nous avons vu un exemple de réflexe inné chez l'être humain, c'est-à-dire retirer notre main lorsque l'on se brûle, mais les êtres humains peuvent aussi apprendre de nouveaux réflexes. Par exemple, on peut penser à la conduite automobile. Au début, la conduite est très difficile, car on doit penser à toutes les actions que l'on fait, mais plus on se pratique, moins on réfléchit et plus la conduite devient un réflexe.

Pour les agents, on peut penser à appliquer la même chose. C'est-à-dire, lorsqu'un agent fait face à une situation pour la première fois, il doit délibérer plus longtemps pour choisir ses actions. Mais, avec un module d'apprentissage, plus l'agent effectue des tâches similaires, plus il devient rapide. Son comportement passe graduellement d'un état délibératif, à un état réactif. L'agent a donc appris à exécuter une tâche. D'un point de vue plus technique, on peut dire que l'agent a, en quelque sorte, «compilé» son raisonnement dans une certaine forme d'ensemble de règles qui lui permettent de diminuer son temps de réflexion. Ce type d'apprentissage peut être très utile pour des agents hybrides.

Ce n'est qu'une façon dont les agents peuvent apprendre, il en existe plusieurs autres. En fait, on considère que toute technique qui permet à un agent d'améliorer son efficacité est une technique d'apprentissage. D'une façon générale en apprentissage machine, on distingue quatre forme d'apprentissage : (i) l'apprentissage supervisé ; (ii) l'apprentissage non supervisé ; (iii) l'apprentissage par renforcement et (iv) l'apprentissage multi-agent.

L'apprentissage supervisé est une technique qui vise à «prédire» une certaine donnée (présentée à l'entrée du processus d'apprentissage supervisé) à partir d'une base de données d'apprentissage contenant des exemples de cas déjà traités. Deux cas peuvent être distingués dans cette forme d'apprentissage. Le premier est du type «régression » et il se produit lorsque la sortie désirée que l'on cherche à associer à une entrée, est une valeur dans un ensemble continu de réels. Le deuxième est du type «classification» et il vise à étiqueter une entrée suivant un ensemble de valeurs de sorties à cardinal fini. Ainsi avec cette méthode d'apprentissage, un agent pourrait «apprendre» (a) des règles du type conditions-action ; (b) associer des percepts à des situations prédéfinies, etc.

L'apprentissage non supervisé consiste, quant à lui, à discerner des motifs au niveau de l'ensemble d'entrées, quand aucune valeur de sortie spécifique n'est donnée. Ainsi cette forme d'apprentissage se distingue de l'apprentissage supervisé par le fait qu'il n'y a pas de sortie à priori. Par contre, il y a un ensemble de données collectées au niveau de l'entrée, traitées en général comme des variables aléatoires. Ainsi par exemple, notre agent en charge de la défense d'une frégate pourrait développer progressivement un concept de «bons coups» et de «mauvais coups» sans qu'on ne lui en donne jamais des exemples spécifiques.

L'apprentissage par renforcement consiste pour sa part, à faire en sorte que l'agent apprenne à partir de son expérience en renforçant les «bonnes actions». Pour cela, on donne à l'agent un certain renforcement après chacune de ses actions. Si le renforcement est positif, cela signifie que l'agent a bien agi, celui-ci tentera donc de refaire les mêmes actions. Si, par contre, le renforcement est négatif, alors l'agent saura qu'il a mal agi et il tentera, à l'avenir, d'éviter les actions donnant un tel renforcement négatif. C'est un apprentissage par essais et erreurs qui permettent à l'agent de s'améliorer avec le temps. Bien entendu, pour utiliser ce type d'apprentissage, on doit être dans un environnement autorisant les erreurs. Par exemple, dans le domaine des frégates, on ne peut se permettre d'erreurs, car on ne voudrait pas qu'une

frégate reçoive dix missiles avant d'apprendre comment les intercepter adéquatement. Dans ce type d'application, une des solutions consiste à avoir recours aux simulations, procurant ainsi un environnement offrant des garanties de sécurité, pour l'apprentissage des agents avant de les implémenter au niveau de l'environnement réel.

Finalement, l'apprentissage multi-agent vise à étudier l'apprentissage au niveau des interactions entre agents. Dans ce cadre, les agents peuvent apprendre à se coordonner, à communiquer et à négocier efficacement. Dans le cas particulier d'agents compétitifs, ils peuvent aussi apprendre comment améliorer leurs gains aux dépens d'autrui et/ou ne pas être exploité par autrui. Nous reviendrons dans la section 2.4 ci-après sur cet aspect.

1.5 Agent Vs Objet

D'après ce que nous avons vu précédemment sur les agents, maintenant on peut déduire une petite comparaison entre un agent et un objet tout facilement, alors:

- Un agent est comme un objet encapsule son état et son comportement mais l'agent à le contrôle sur son comportement contrairement pour l'objet qui n'a pas le contrôle que sur son état (un objet est un agent obéissant).
- Un agent va décider par son propre processus de décision s'il exécute ou non une action requise.
- les agents ont leurs propres buts et ils agissent d'une manière proactive pour atteindre leurs buts.
- les agents sont capables d'un comportement social : ils peuvent s'engager dans des interactions complexes avec d'autres agents.
- Un objet est passif : il ne revient à la vie que lorsqu'il reçoit un message.
- Un objet représente un niveau d'abstraction trop fin pour le comportement.
- Une méthode est un mécanisme trop primitif pour décrire une interaction.

- Seules les organisations de type "est un" et "est composé de" sont disponibles.

1.6 SMA (System Multi-Agent)

Jusqu'ici on a fait état des systèmes où évoluait un seul agent. Dans la plupart des situations réelles toutefois, l'agent n'est pas seul dans son environnement et il y a bien d'autres agents autour de lui. Il nous faut donc aborder des systèmes où plusieurs agents doivent interagir entre eux. De tels systèmes sont appelés «systèmes multi-agents» (SMA) et ils possèdent les caractéristiques principales suivantes [46]:

- chaque agent a des informations ou des capacités de résolution de problèmes incomplètes, donc chaque agent a un point de vue limité.
- il n'y a pas d'entité centrale qui assure le contrôle global.
- les données sont distribuées.
- les traitements au niveau des agents sont asynchrones.

Bien que les SMA offrent de nombreux avantages potentiels pour les applications qui s'y prêtent, ils doivent aussi relever beaucoup de défis.

Parmi ces défis, il convient de citer:

- Comment décomposer les tâches et les allouer aux agents ?
- Comment faire communiquer efficacement les agents ?
- Comment faire agir efficacement les agents ? Comment leur assurer une coordination profitable ?
- Comment reconnaître et réconcilier les conflits entre agents ? Comme les faire négocier et trouver le meilleur compromis entre eux ?

- Comment permettre aux agents de représenter et raisonner sur les actions, politiques ou stratégies d'autrui ?
- Comment maximiser la satisfaction (ou l'utilité) de chacun des agents dans un environnement compétitif ?
- Comment un agent pourrait maximiser sa propre satisfaction tout en contribuant à la réussite du groupe dans un environnement d'agents coopératifs ?

Par le passé, plusieurs livres ont été élaborés sur les systèmes multi-agents. Parmi les livres les plus récents on peut citer, entre autres, les livres de Shoham [6] et de Vlassis [7].

1.7 Historique

En 1980, un groupe de chercheurs s'est réuni pour discuter des défis concernant la résolution "intelligente" de problèmes dans un système comportant plusieurs résolveurs de problèmes. Lors de cette réunion, il a été décidé que l'intelligence artificielle distribuée n'axerait pas ses travaux sur les détails de bas-niveau de la parallélisation ni sur comment paralléliser les algorithmes centralisés mais plutôt sur le fait de savoir comment un groupe de solutionneurs de problèmes pourrait coordonner ses efforts afin de résoudre des problèmes de manière efficace.

On peut dire que les SMA ont vu le jour avec l'avènement de l'intelligence artificielle distribuée (IAD). ٥ ses début toutefois, l'IAD ne s'intéressait qu'à la coopération entre résolveurs de problèmes afin de contribuer à résoudre un but commun. Pour y parvenir, on divisait en général, un problème en sous problèmes, et on allouait ces sous-problèmes à différents résolveurs qui sont appelés à coopérer pour élaborer des solutions partielles. Celles-ci sont finalement synthétiser en une réponse globale au problème de départ. Ainsi donc, l'IAD au départ privilégiait le "problème à résoudre" tout en mettant l'accent sur la résolution d'un tel problème par de multiples entités intelligentes. Dans les SMA d'aujourd'hui, les agents sont (entre autres) autonomes, possiblement préexistants et généralement hétérogènes. Dans ce cas, l'accent est plutôt mis sur le fait de savoir comment

les agents vont coordonner leurs connaissances, buts et plans pour agir et résoudre des problèmes.

1.8 Utilité des systèmes multi-agents

Certains domaines sont géographiquement ou fonctionnellement distribués et par conséquent les systèmes multi-agents constituent la façon adéquate de les modéliser. C'est le cas, par exemple, des applications distribuées comme la gestion de plusieurs senseurs, robots, véhicules, frégates, ou bien le contrôle aérien, les bases de données coopératives distribuées, etc. C'est le cas également des réseaux complexes comme l'Internet, le commerce électronique, les réseaux d'eau, de transport, de télécommunication, etc.

Les systèmes multi-agents sont également requis dans les situations où lorsque les différents systèmes et les données qui s'y rattachent appartiennent à des organisations indépendantes qui veulent garder leurs informations privées et sécurisées pour des raisons concurrentielles. Par exemple, la majorité des missions maritimes se font maintenant en collaboration avec plusieurs pays. Donc, il y a plusieurs bateaux de pays différents qui doivent agir ensemble. Pour concevoir un système qui permettrait aux bateaux de se coordonner, il faudrait avoir des informations sur toutes les caractéristiques de chacun des bateaux. Bien entendu, aucun pays ne voudra donner ces informations puisqu'elles sont considérées comme des secrets militaires. De plus, les façons de faire diffèrent d'un pays à l'autre. Une solution à ce problème consiste à permettre à chaque pays de concevoir ses propres agents qui représenteront adéquatement ses buts et ses intérêts. Par la suite, ces agents pourront communiquer ensemble pour coordonner les informations nécessaires à une bonne coordination des bateaux.

Même si le domaine ne requiert pas l'utilisation des systèmes multi-agents, il existe tout de même de bons avantages à les utiliser dans bien des cas. Ainsi, ils peuvent s'avérer bien utiles pour des problèmes possédant de multiples méthodes de résolution, de multiples perspectives et/ou de multiples résolveurs. En particulier, les systèmes multi-agents sont très utiles pour modéliser le raisonnement humain à l'intérieur de grandes simulations de combats aériens [8].

Ils possèdent également les avantages traditionnels de la résolution distribuée et concurrente de problèmes, en particulier la modularité, le parallélisme et la fiabilité [5]. La modularité rend en général la programmation beaucoup plus facile tout en permettant aux systèmes multi-agents d'être facilement extensibles, parce qu'il est plus facile d'ajouter de nouveaux agents à un système multi-agent que d'ajouter de nouvelles capacités à un système monolithique. Le parallélisme croit la vitesse de calcul dans la mesure où plusieurs agents peuvent travailler en même temps pour la résolution d'un problème. Finalement la fiabilité pourrait également être atteinte, dans la mesure où le contrôle et les responsabilités étant partagés entre les différents agents, le système peut tolérer la défaillance d'un ou de plusieurs agents. Si une seule entité contrôle tout, alors une seule défaillance de cette entité fera en sorte que tout le système tombera en panne.

1.9 Interactions entre agents

Les systèmes multi-agents ont surtout l'avantage de faire intervenir des schémas d'interaction sophistiqués. Ils peuvent ainsi coexister, être en compétition ou coopérer.

S'ils ne font que coexister, alors chaque agent ne considère les autres agents que comme des composantes de l'environnement, au même titre que toutes les autres composantes. Si les agents ont une représentation physique, les autres agents ne seront vus que comme des obstacles que l'agent doit éviter. Il s'ensuit qu'il n'y a aucune communication directe entre les agents. En fait, il peut y avoir une certaine forme de communication indirecte parce que les agents peuvent se percevoir les uns les autres. Le but visé n'est toutefois pas de communiquer avec l'autre. Ces informations ne servent qu'à mieux éviter les autres agents. Par exemple, si l'on considère une personne marchant dans une foule d'étrangers, elle communique avec les autres à l'aide de gestes ou de mouvements, mais uniquement dans le but de pouvoir circuler sans accrocher tout le monde.

S'ils sont en compétition, alors le but de chaque agent est de maximiser sa propre satisfaction, ce qui se fait généralement aux dépens des autres agents. La situation de compétition la plus fréquente se produit lorsque plusieurs agents veulent utiliser ou acquérir la même ressource. Les agents doivent donc pouvoir communiquer entre eux pour résoudre

le conflit. Cette communication prend habituellement la forme d'une négociation. Les agents se transmettent des propositions et des contre-propositions jusqu'à ce qu'ils arrivent à une entente ou qu'ils se rendent compte qu'une entente est impossible. Ce type de communication demande un protocole de négociation et un langage de communication de haut niveau du type de FIPA-ACL [9] pour permettre une certaine structure dans la négociation.

S'ils sont en coopération, alors le but des agents n'est plus seulement de maximiser sa propre satisfaction mais aussi de contribuer à la réussite du groupe. Les agents travaillent ensemble à la résolution d'un problème commun. Dans ce type de système, les agents communiquent ensemble, à l'aide de messages plus ou moins sophistiqués, dans le but d'améliorer la performance du groupe. Ils peuvent s'échanger des informations sur l'environnement pour augmenter leurs perceptions individuelles, ou bien se transmettre leurs buts pour que les agents puissent avoir une idée de ce que les autres font. Somme toute, les communications servent aux agents à améliorer leur coordination, c'est-à-dire à organiser la résolution du problème de telle sorte que les interactions nuisibles soient évitées et/ou que les interactions bénéfiques soient exploitées.

1.10 Systèmes multi-agents réactifs

L'école réactive prétend qu'un système globalement intelligent ne nécessite nullement l'intelligence individuelle de ses agents. C'est un système multi-agents dont les agents ne disposent pas de mémoire, de représentation globale du système, de la tache à effectuer et des autres agents. De simples mécanismes de réaction aux évènements, ne prenant en compte ni explicitation des buts, ni mécanismes de planification, Leurs prises de décision sont simples, de type stimulus réponse et se fondent uniquement sur des perceptions locales. Ces mécanismes peuvent en effet résoudre des problèmes qualifiés de complexes.

La résolution de la tâche est produite des couplages des comportements des agents. Ces couplages peuvent être la conséquence d'interactions méditées par l'environnement comme dans les approches stigmergiques. L'environnement est modifié par les actions effectuées par les agents et ces modifications ont en retour une influence sur les comportements des

autres agents. De la même manière la position d'un agent dans l'environnement peut être perçu par les autres agents et influencer leurs comportements.

L'un des exemples les plus frappants étant celui d'une fourmilière artificielle : Alors que toutes les fourmis se situent sur un même pied d'égalité, et en l'absence d'une quelconque autorité stricte, les actions des agents se coordonnent de telle manière que la colonie survie et se développe. L'entité collective est en mesure de résoudre des problèmes complexes tels que la recherche de nourriture, les soins à donner aux œufs, et aux larves, la construction de nid, la reproduction, etc.

L'idée clés est que le comportement intelligent peut être généré sans représentation explicite, sans raisonnement abstrait explicite et que l'intelligence est une propriété émergente de certains systèmes complexes dut aux **interactions** c'est-à-dire que le comportement intelligent de l'agent n'est pas séparé, mais c'est un produit de l'interaction que l'agent maintient avec son environnement.

1.11 Systèmes multi-agents et apprentissage

L'apprentissage est une composante importante des systèmes multi-agents. Ces systèmes évoluent généralement dans des environnements complexes (c'est-à-dire larges, ouverts, dynamiques et non prévisibles) [10]. Pour de tels environnements, c'est très difficile et même quelquefois impossible de définir correctement et complètement les systèmes à priori, c'est-à-dire lors de la phase de conception, bien avant leur utilisation. Ceci exigerait de connaître à l'avance toutes les conditions environnementales qui vont survenir dans le futur, quels agents seront disponibles à ce moment et comment les agents disponibles devront réagir et interagir en réponse à ces conditions. Une manière de gérer ces difficultés est de donner à chaque agent l'habileté d'améliorer ses propres performances, ainsi que celles du groupe auquel il appartient.

Il est à noter que l'apprentissage dans un système multi-agent comprend l'apprentissage dans un système mono-agent parce qu'un agent peut apprendre en solitaire et de façon complètement indépendante des autres agents. Mais aussi, il l'étend bien au delà dans la

mesure où les activités d'apprentissage d'un agent peuvent être influencées considérablement par les autres agents et que plusieurs agents peuvent apprendre de manière distribuée et interactive comme une seule entité cohérente.

Dans un environnement multi-agent, les agents peuvent apprendre grâce aux autres. Par exemple, un agent A, qui voudrait savoir comment se rendre à un certain endroit, pourrait demander à un autre agent B s'il connaît un bon chemin. Si B connaît un bon chemin, il peut le transmettre à A, permettant ainsi à l'agent A d'apprendre un bon chemin grâce à l'agent B.

D'un autre côté, les agents peuvent aussi apprendre à propos des autres. Par exemple, un agent peut regarder un autre agent agir dans certaines situations et, à l'aide de ces informations, il pourrait construire un modèle du comportement de l'autre agent. Ce modèle pourrait lui servir pour prédire les actions de l'autre agent dans le futur. Cette information pourrait l'aider à mieux se coordonner ou à mieux collaborer avec l'autre agent.

Une autre facette importante est l'apprentissage des interactions (coordination, collaboration, communication, etc.) entre les agents. Par exemple, si une frégate US doit travailler avec un bateau de guerre anglais, elle doit pouvoir apprendre une manière efficace de se coordonner avec cet allié en tenant compte de ses capacités. En fait, elle doit pouvoir le faire pour toutes les possibilités, c'est-à-dire avec n'importe quel nombre de bateaux provenant de n'importe quel pays et dans n'importe quelle situation. Les agents doivent donc pouvoir adapter leurs mécanismes de coordination et de communication pour que l'ensemble des bateaux puissent agir de manière cohérente et ainsi se défendre le plus efficacement possible. Cette adaptabilité est une forme d'apprentissage de vie sociale.

1.12 Conclusion

Apres ce chapitre j'ai riche d'information sur les agents (ces types, proprietes, ..) et sma aussi, j'ai trouvé que ce domaine est presque nouveau ce qu'il me faire un défi de le comprendre plus de plus surtout les agent réactifs qui je les utilise dans les chapitres suivants.

Chapitre 2

Fourmis réelles & Fourmis artificielles

Introduction

Les fourmis sont des insectes sociaux qui forment des colonies contiens dizaines d'individus et travaillent collectivement avec des méthodes très organisés pour achèvent des objectives nécessaires pour qu'elles vivent, donc si on les observe on trouve que les comportements des fourmis est un comportement collectif, que chaque fourmi (Agent) a pour priorité le bien être de la communauté et que Chaque individu de la colonie est à priori indépendant et n'est pas supervisé d'une manière ou d'une autre.

Alors La colonie est auto contrôlé (Autonomie) par le biais de mécanismes relativement simples à étudier.

2.1 Les insectes sociaux

Les insectes sociaux sont des insectes vivant et s'organisant en colonies et démontrant une intelligence collective leur permettant de retirer un bénéfice de leur instinct grégaire.

Parmi les insectes sociaux, on compte :

- toutes les espèces de fourmis.

- toutes les espèces de termites.
- certaines espèces d'abeilles.
- certaines espèces de guêpes.

Ces espèces présentent en général une structure sociale d'un type particulier baptisée eu-socialité qui se manifeste par une division du travail, la présence d'une caste stérile (ouvrières) et une absence de séparation franche entre les différentes générations.

Le fait que certains individus (« ouvrier » ou « ouvrière ») de la colonie ne se reproduisent pas s'explique par le fait que le bénéfice n'est pas donné à leur progéniture mais à celle d'individus qui leur sont étroitement apparentés.

2.2 Les Fourmis et leur colonie

Les fourmis représentent un univers à elles seules, Elles occupent en effet [11] presque tous les habitats terrestres où leur biomasse est considérable. Par exemple, en forêt amazonienne, la biomasse des fourmis est quatre fois supérieure à la biomasse des vertébrés terrestres [12]. Aujourd'hui on compte presque 40.000 espèces différentes et il en reste peut-être encore autant à découvrir, surtout en milieu tropical très mal connu. Elles ont inventé l'élevage des pucerons (c'est un vrai bétail qui est mis à l'abri en hiver, transporté lors des déménagements, et défendu contre les prédateurs), la culture de champignons, l'esclavage (il s'agit d'esclavagisme interspécifique: une espèce comme la fourmi amazone Polyergus utilise une autre espèce pour subvenir à ses propres besoins), la vie en parasite, la guerre, la division du travail, le nomadisme, des antibiotiques et des poisons. Elles mesurent de moins d'un millimètre à 4 cm. Leurs colonies comportent de quelques individus à plusieurs dizaines de millions (plus de 20 millions chez les fourmis légionnaires nomades d'Afrique tropicale, plus de 300 millions dans les super-colonies de fourmis rousses des bois au Japon ou dans le Jura suisse, avec un million de reines et 45 000 nids sur quelques km2). Elles nichent dans le sol jusqu'en haut des plus hauts arbres de la canopée. Comment expliquer ce succès ?

Justement par leur vie sociale très développée dont on va présenter les principales caractéristiques, mais d'abord comment reconnaître une fourmi ?

Ce que l'on appelle fourmi est en général une ouvrière. Il s'agit d'un insecte, caractérisé par un exosquelette (littéralement « squelette extérieur », c'est-à-dire carapace, aussi appelée cuticule), six pattes, des yeux et deux antennes. Elle a un corps en trois parties : la tête, le thorax et l'abdomen (figure 2.1). Tout cela fait que le faciès fourmi est très caractéristique. Tout le monde reconnaît facilement une fourmi de loin, même si elle n'a que quatre pattes comme dans les dessins animés de Walt Disney. L'anatomie de la fourmi se caractérise classiquement par un tube digestif, un appareil circulatoire, un système nerveux et de nombreuses glandes, qui sont de véritables usines chimiques. Le tube digestif comporte une poche spéciale, le jabot, où est stockée la nourriture liquide. L'ouvrière est stérile. La reine est le seul individu pondeur, il peut d'ailleurs y avoir plusieurs reines au sein d'une même colonie (chez les fourmis envahissantes comme la fourmi d'Argentine sur la Côte d'Azur, il y a des milliers de reines). Elles sont en général beaucoup plus grosses que les ouvrières et ce sont souvent de véritables machines à pondre (deux œufs à la minute, en période de ponte, chez les fourmis légionnaires).

Lorsque l'on observe une fourmilière ou des fourmis dans un nid artificiel en laboratoire, on est surpris par la richesse du répertoire comportemental de ces animaux.

Une telle organisation sociale très sophistiquée ne pourrait fonctionner sans des mécanismes assurant la communication entre individus. Les fourmis utilisent trois types de communication : tactile par les antennes (voir figure 2.2), sonore par des stridulations, et chimique par de nombreuses odeurs. La vision est en général peu importante. La communication tactile intervient surtout pour inviter une congénère à suivre la piste vers une source de nourriture : la recruteuse tambourine la tête de l'autre fourmi pour la stimuler, on parle de comportement d'invitation. Lors de la trophallaxie, on assiste à un véritable ballet entre les antennes des deux fourmis : celle qui reçoit sollicite l'autre en lui balayant la tête ; la donneuse répond de la même manière. Elles s'informent sur leur état de satiété.

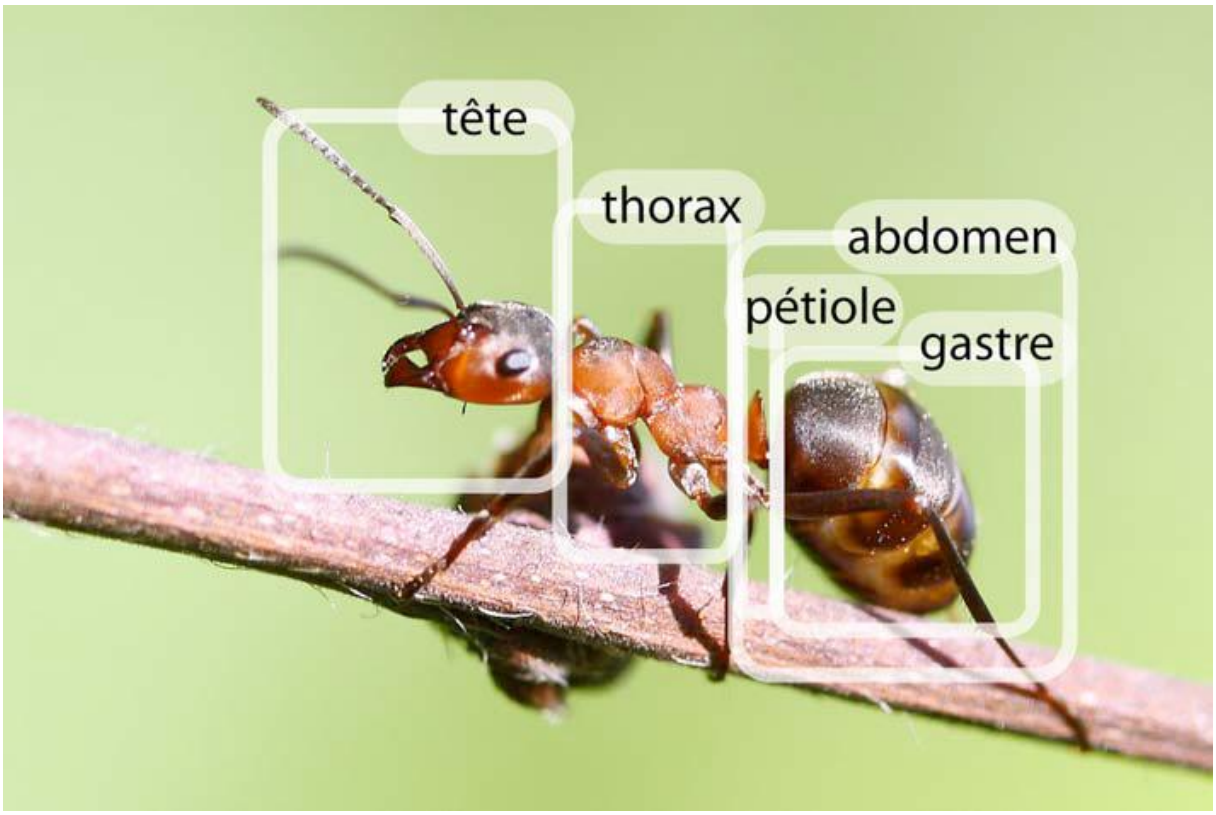

Figure 2-1 -Quelques renseignements anatomiques sur la fourmi (ici, formica polyctena) –photo N. Monmarché.

C'est chez les insectes qu'ont été découvertes, dans les années 1960, les substances chimiques (phéromones) qui permettent à deux individus de la même espèce de communiquer. On dispose à l'heure actuelle de chromatographes en phase gazeuse couplés à un spectromètre de masse qui permettent d'analyser et d'identifier les substances constitutives d'une phéromone à des concentrations très faibles, de l'ordre du nano-gramme (mais soyons modestes, les antennes des insectes ont une sensibilité colossale, à des concentrations que l'on ne sait pas détecter actuellement). Ces phéromones sont sécrétées par de très nombreuses glandes situées partout dans le corps qui est une véritable usine chimique : on connaît déjà 39 glandes chez les fourmis (voir figure 2.4) et plusieurs centaines de substances de catégories chimiques très variées. Elles interviennent dans toutes les activités de la colonie : l'agrégation, le marquage du territoire et de l'entrée du nid, la formation de pistes pour exploiter une source de nourriture ou déménager (changer de nid),

le recrutement de congénères pour défendre la colonie (alarme et défense), l'appel sexuel, la reconnaissance de la reine, des autres membres de la colonie et même des cadavres.

Figure 2-2 -Défendre la colonie est aussi une tâche collective (formicarufa) –photo N. Monmarché

Une autre particularité importante des sociétés d'insectes est qu'elles sont fermées, les individus étrangers à la colonie sont rejetés et parfois tués. On peut assister par exemple au printemps à de véritables « guerres » entre deux colonies voisines qui essaient de repousser leurs frontières au détriment de l'autre, avec de nombreux cadavres entre les deux zones. Certaines espèces tropicales comme les fourmis fileuses oecophylles vivent sur l'arbre qu'elles monopolisent complètement et défendent farouchement contre tout intrus. La fermeture des sociétés d'insectes s'inscrit dans le cadre théorique de la sélection de parentèle, élaboré par W.D. Hamilton [13,14]. En effet, selon la théorie de l'évolution darwinienne, un animal élève sa propre descendance dans le but de transmettre ses gènes. Or, chez les fourmis, les ouvrières stériles ne se reproduisent pas et élèvent leurs sœurs. On les qualifie d'altruistes. Mais cet altruisme est intéressé, elles transmettent indirectement leurs gènes. Alors, elles ne doivent pas élever des larves avec lesquelles elles n'ont pas de lien de parenté. Il faut que la colonie reste une entité fermée au niveau génétique. Même s'il y a de nombreux cas

particuliers et diverses critiques, à l'heure actuelle, la théorie de sélection de parentèle reste un modèle heuristique très intéressant. Par quel mécanisme les fourmis se reconnaissent-elles quand elles se rencontrent et qu'elles échangent des battements antennaires ? Comme l'on pouvait s'en douter, elles se reconnaissent à l'odeur. L'odeur des fourmis est formée de substances très peu volatiles qui sont portées par la carapace de la fourmi et perçues par les antennes. Il s'agit essentiellement d'hydrocarbures qui sont des molécules simples dérivées de lipides, des chaînes de carbone et d'hydrogène. Chaque colonie a un bouquet « d'odeurs » qui la caractérise, composé souvent d'une cinquantaine, voire même d'une centaine de molécules différentes : c'est l'odeur coloniale. C'est un peu un abus de langage de parler d'odeur car cela se rapproche plutôt de la gustation, mais c'est entré dans le langage des myrmécologues. Ces substances sont fabriquées dans le corps par des cellules spécialisées. Chaque fourmi fabrique ses propres hydrocarbures en fonction de son patrimoine génétique, exactement comme les mammifères sécrètent des odeurs liées au complexe majeur d'histocompatibilité (CMH). Les hydrocarbures sont excrétés sur la cuticule, mais sont aussi stockés dans une glande de la tête, la glande post-pharyngienne, dont le contenu est versé avec les liquides régurgités pendant la trophallaxie. Ainsi l'odeur coloniale est-elle échangée constamment entre individus. Si l'on ajoute les léchages interindividuels très fréquents, on constate un brassage permanent et très important des odeurs individuelles. Cela aboutit à une odeur commune partagée entre tous les individus et caractéristique de la colonie. On parle de gestalt [15].

On peut se demander ensuite comment la fourmi s'identifie à sa colonie pour être capable de discriminer les autres. La jeune fourmi apprend, quand elle sort de son cocon, à reconnaître l'odeur de son nid, donc de ses sœurs et elle va mémoriser cette odeur pour toute sa vie. Cet apprentissage commence d'ailleurs parfois même à l'état larvaire. Bien sûr, il reste une flexibilité puisque l'odeur de la colonie change un peu en fonction de l'alimentation, des matériaux du nid ou de la composition en individus. La fourmi doit en permanence ajuster l'odeur qu'elle porte avec celle des autres (labels), mais aussi l'image mentale qu'elle mémorise dans son cerveau que l'on appelle Template. En fait, l'évolution a imaginé un processus très économique pour permettre la reconnaissance des ongénères : il suffit d'apprendre et mémoriser en toute confiance l'odeur des congénères du nid [16] ! C'est ainsi que les fourmis amazones Polyergus esclavagistes vont faire des razzias de cocons dans

les colonies de l'espèce hôte. Quand les jeunes ouvrières éclosent dans la colonie esclavagiste, elles s'imprègnent de l'odeur de leur colonie d'adoption et vont considérer les amazones comme leurs sœurs et travailler à leur service.

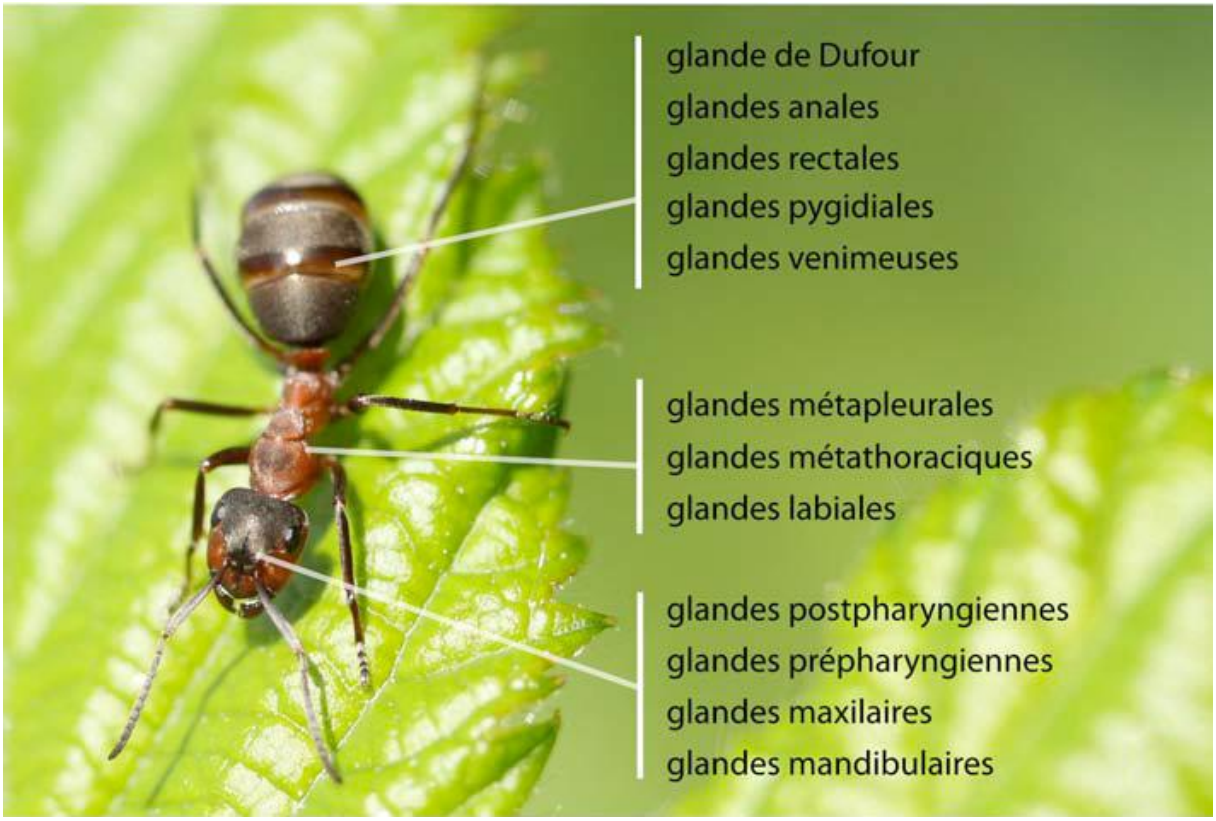

Figure 2-3 -L'importance des glandes chez les fourmis - photo N. Monmarché.

2.3 Intelligence et comportements collective des fourmis

Lorsque l'on regarde un nid de fourmis, comme on l'a dit plus haut, cela grouille dans tous les sens ! A première vue, les comportements individuels sont fantaisistes et semblent indépendants. C'est une observation courante que de voir une fourmi tirer la charge dans le mauvais sens et il est impossible de prédire exactement ce qu'un individu va faire à un moment donné. Pourtant, ce fouillis apparent est remarquablement efficace : même si certains individus échouent, la tâche est accomplie, la grosse proie rentrée au nid, la brindille insérée correctement dans le dôme et la colonie fonctionne de manière prévisible. Comment expliquer cela alors qu'il n'existe pas d'organe central donnant des ordres ? En effet, la reine

ne joue pas ce rôle, il n'y a ni commandant ni despote. On peut marquer individuellement des ouvrières avec une pastille numérotée (ou des tâches de couleur), suivre leur activité dans un nid artificiel et les soumettre à diverses tâches. On s'aperçoit que certaines sont hyperactives, d'autres peu actives ou même totalement inactives, ces dernières représentent même la majorité. Tout se passe comme s'il y avait un volant important d'ouvrières inactives en attente de travail. On constate que la fourmi adapte son comportement aux besoins de la colonie, qui peuvent varier, et aussi en fonction de l'environnement. En dotant ainsi la fourmi d'un simple comportement d'adaptation de ses seuils de réponse à différents stimuli de son quotidien, celle-ci se spécialise, c'est-à-dire répond plus rapidement à un stimulus donné et accomplit l'action correspondante plus efficacement, et ceci d'autant plus que la colonie s'agrandit. Plus la colonie est importante, plus les besoins sont importants mais plus les fourmis se spécialisent et accomplissent certaines tâches très efficacement : ceci peut expliquer que, la taille de la colonie augmentant, la proportion de fourmis inactives augmente également. L'ensemble devient ainsi de plus en plus flexible et robuste, s'il faut beaucoup d'individus pour exploiter une source de nourriture importante, les réserves sont disponibles.

Donc a partir des données d'observation, on peut construire des modèles pour essayer de simuler le fonctionnement des sociétés. Ce travail permet d'approfondir nos connaissances fondamentales sur les sociétés de fourmis. Un pas décisif a été franchi avec les modèles récents inspirés de la théorie de l'auto-organisation. L'auto-organisation correspond à un ensemble de processus décrits par I. rigogine (Prix Nobel de chimie en 1977), au cours desquels des structures émergent au niveau collectif, à partir d'une multitude d'interactions entre individus, sans être codées explicitement au niveau individuel ([17] et voir par exemple [18,19]).

Le concept d'auto-organisation permet d'expliquer, du moins de prédire, de nombreux phénomènes macroscopiques en définissant des règles comportementales au niveau microscopique. Dans la perspective scientifique habituelle, ces règles sont sélectionnées en respectant le principe de minimalité (le système le plus réduit/simple qui peut induire ou expliquer le phénomène est plébiscité). Un certain nombre de propriétés générales ont été dégagées de ces systèmes auto-organisés pour qu'il y ait des chances d'observer l'émergence d'une structure stable :

- La présence de deux types de renforcements : positif (positive feedback), qui induit l'amplification d'un phénomène, et négatif (négative feedback), qui contrebalance l'amplification et assure donc une stabilisation du phénomène.

- L'amplification des fluctuations : les fluctuations sont la composante aléatoire du phénomène et l'amplification de certaines manifestations repose sur les mécanismes de renforcement.

- La multitude des interactions : plus les interactions sont nombreuses, plus la stabilité des structures construites est forte.

Le phénomène observé macroscopiquement prend la forme d'une structure présentant une certaine stabilité spatio-temporelle, c'est-à-dire observable dans l'espace et/ou le temps, dont les manifestations chez les fourmis sont, par exemple, la construction de pistes, la construction du nid (creusement de galeries). Ces structures apparaissent dans un milieu plus ou moins homogène (la surface de la terre ou le sous-sol) et plusieurs états stables peuvent coexister (plusieurs chemins vers différentes sources de nourriture). Ces phénomènes auto-organisés présentent aussi parfois des bifurcations dans le comportement observé, notamment lorsque certains paramètres environnementaux sont modifiés : par exemple, l'évaporation des phéromones devenant plus importante à cause des conditions climatiques, les chemins qui ont été construits et qui représentent des structures stables peuvent subitement se disloquer.

La simplicité des entités considérées et participant à l'auto-organisation ne doit cependant pas dissimuler que la fourmi reste à son échelle une merveille de miniaturisation et d'autonomie. En effet, les fourmis ne se comportent pas comme de simples molécules [20] et leurs capacités individuelles, notamment cognitives, ne sont pas à négliger pour expliquer des comportements évolués, qu'ils soient collectifs ou individuels. La suite de cette section donne quelques illustrations de modèles de comportement.

2.3.1 Les pistes

Comme on l'a vu, une fourmi qui découvre une source de nourriture importante dépose une trace de phéromone en rentrant au nid. La règle dans ce cas est très simple : les fourmis suivent la piste chimique et l'on observe une croissance exponentielle du nombre de fourmis sur cette source. D'un comportement simple individuel (le dépôt d'une gouttelette de phéromone et l'attirance pour un chemin marqué par de la phéromone), émerge un comportement collectif (le suivi de piste et le recrutement de masse). S'il y a plusieurs chemins possibles pour accéder à la nourriture, le chemin le plus court sera alors choisi collectivement sans aucune décision individuelle et perception individuelle de la distance, tout simplement parce que la piste la plus courte sera renforcée plus vite, comme cela a été observé chez la fourmi d'Argentine [21,22]. C'est un processus auto-catalytique, les fourmis rentrant au nid choisissent et renforcent toujours davantage la piste la plus marquée (renforcement positif) jusqu'à ce que la source s'épuise ou que l'accès à la nourriture devienne trop embouteillé (renforcement négatif). Ces expériences sont maintenant classiques et sont décrites dans des manuels comme celui de M. Dorigo et T. Stützle [23].

Une recherche récente montre aussi que les fourmis sont capables d'estimer le trafic sur une piste : quand la densité devient trop importante, elles choisissent le chemin le moins embouteillé. Cela a été montré chez la petite fourmi noire des jardins, Lasius niger, par A. Dussutour [24]. Cette fourmi est capable de recrutement de masse et forme des pistes plus ou moins permanentes. Dès que la densité sur la piste atteint un certain seuil, une deuxième piste est formée avant que le trafic ne soit affecté, ce qui garantit une efficacité optimale.

Elles sont aussi capables d'ajuster le trafic à la largeur de la piste. Ce phénomène a été étudié, toujours chez Lasius niger, où l'on peut observer sur les pistes un trafic très important dans les deux sens. On a examiné les effets d'un goulot d'étranglement sur la piste. A. Dussutour [25] a réuni un nid de Lasius à une aire de fourragement à l'aide d'un pont. Deux types de pont ont été utilisés : avec ou sans goulot. Les fourmis arrivent à réguler leur trafic : avec les deux types de pont, le volume de trafic et la quantité totale de nourriture rapportée au nid restent identiques. Cela est possible, quand il y a un goulot d'étranglement, par une réorganisation temporelle du flux : quand la largeur de la voie décroît, les fourmis se répartissent en groupes alternant dans les deux sens (aller ou retour) comme s'il y avait un

feu rouge virtuel. Cela limite le nombre de rencontres en face à face entre deux fourmis allant en sens contraire et permet la même durée de trajet. On peut imaginer que ce type de problème se pose sans arrêt dans les galeries des nids de fourmis ou de termites.

Ces capacités des fourmis réelles à résoudre des problèmes, comme trouver le meilleur chemin pour arriver à une source de nourriture, à l'aide de phéromones, a inspiré des algorithmes nombreux, par exemple à des chimistes pour élaborer des polymères [26]. Cependant, les pistes ne sont pas utiles qu'à la recherche de nourriture, par exemple chez les Magnans on observe une organisation particulière quand le nid est déplacé : les soldats surveillent et protègent la piste empruntée par les ouvrières (voir figure 2-4).

Figure 2-4 -Colonne de Magnans - photo A. Lenoir.

Ainsi, la recherche d'un nouveau nid et le déplacement des colonies est un autre modèle pour comprendre les interactions entre décisions individuelles et collectives. N. Franks et

son équipe à Bristol étudient depuis plusieurs années une toute petite fourmi qui vit dans les fissures de rochers (Temnothorax albipennis). Si l'on donne le choix entre deux sites possibles, l'un proche mais de mauvaise qualité, l'autre plus éloigné et sur la route du premier mais meilleur, le plus souvent les colonies commencent à émigrer simultanément vers les deux nids. Mais, un peu plus tard, elles redirigent tout le trafic exclusivement vers le meilleur nid. Un modèle a montré que ce choix minimise l'exposition aux risques liés au déménagement à l'air libre [27].

On peut enfin noter que certains travaux [28] ont montré qu'en situation de panique, les fourmis pouvaient perdre leur aptitude à distribuer équitablement la charge de trafic et ont tendance à s'engouffrer vers la sortie la plus engorgée (à l'image d'une foule entraînée par une seule sortie).

2.3.2 Trie d'objets

Une application particulière du tri d'objets est le transport et l'agrégation des cadavres, qui aboutissent à l'élaboration des fameux « cimetières » de fourmis observés par les anciens naturalistes qui n'hésitaient pas à dessiner les cadavres bien alignés. Les fourmis rejettent les cadavres hors du nid afin de réduire les risques d'infection dans la colonie et les regroupent dans un endroit éloigné. Lorsque l'on disperse au hasard des cadavres dans une arène expérimentale, en quelques heures, les fourmis vont les regrouper en petits tas. Il est possible de modéliser ce comportement en attribuant à chaque fourmi une probabilité d'exécuter une action, tout d'abord de saisir le cadavre rencontré, puis de le déposer dès qu'elle rencontre un autre cadavre. Cependant, il ne faut pas prendre un cadavre déjà déposé sur un tas. Pour résoudre ce problème, on introduit une nouvelle règle : plus le tas est gros, moins la fourmi aura tendance à prendre un cadavre et plus elle le déposera facilement. Le modèle permet de prédire qu'il existe une densité critique de cadavres en deçà de laquelle l'agrégation n'aura pas lieu : les tas n'ont pas le temps de se former [29].

Figure 2-5 -Le couvain chez Lasius niger : les cocons de sexués et d'ouvrières sont séparés (ici sur la photo) et se trouvent dans des chambres différentes des larves et oeufs - photo N. Monmarché.

Un autre exemple de tri est le tri du couvain (figure 2.5). En effet, dans le nid, les divers éléments de couvain ne sont pas mélangés en vrac mais répartis dans des chambres différentes selon leur âge. C'est ainsi que les œufs et petites larves sont entassées au centre du nid, les larves plus grosses dans des chambres périphériques. Les cocons sont régulièrement montés à la surface du nid pour être chauffés au soleil, ce qui accélère leur développement. Dans le modèle correspondant, les fourmis prennent et déposent tout élément de couvain en fonction du nombre d'items présents et de leur catégorie [30]. Le chapitre 1 du volume 2 [31] donne des détails sur les modèles construits à partir de ces notions de tri d'objets par les fourmis, et notamment leurs utilisations en classification de données.

2.3.3 Activités de creusement du nid

Dans le cas du creusement du nid, les ouvrières dégagent les chambres du nid et rejettent les grains de sable en dehors de l'entrée du nid. C'est ce que l'on peut observer après une pluie quand la terre est meuble et plus facile à creuser. Le dépôt de centaines de charges conduit à la formation d'un cratère régulier. Comment cela est-il possible ? E.J.H. Robinson et al. ont étudié ce problème [32] : des règles simples permettent de le résoudre.

Figure 2-6 -Cratères creusés après une pluie (Camponotus, Maroc) - photo A.Lenoir.

La fourmi rejette la particule préférentiellement au sommet du cratère vers la pente la plus douce depuis l'entrée du nid. D'autres règles simples, comme la mémoire de la direction des sorties précédentes, permettent d'affiner le modèle et un comportement régulier et prévisible émerge (voir figure 2.6).

2.3.4 Agrégation d'individus

L'agrégation est une caractéristique de tous les animaux sociaux, mais elle est fondamentale pour les insectes sociaux qui ne peuvent survivre longtemps hors de leur colonie. Toutes les fourmis présentent, à divers degrés, cette attraction mutuelle qui permet l'agrégation [33].

La distribution spatiale des individus dans la colonie est déterminante pour l'efficacité de celle-ci, il faut être au bon endroit au bon moment. Les individus vont se regrouper en fonction des tâches comme le couvain ou la reine à soigner (voir la revue de [34]). L'agrégation peut s'effectuer en fonction de repères physiques ou physico-chimiques dans le nid (par exemple, taille des chambres, humidité, luminosité, etc.). Des critères chimiques peuvent aussi intervenir. C'est ainsi que l'on a montré chez la blatte (qui n'est pas sociale comme les fourmis, mais grégaire, ce qui peut être considéré comme une forme basique de la socialité) que les individus reconnaissent l'odeur de leurs congénères et s'en servent pour se regrouper dans des abris [35]. En l'occurrence, les odeurs sont aussi des hydrocarbures, comme chez les fourmis. Par exemple, chez Lasius niger, des traces de marquage colonial sur le sol favorisent l'agrégation [36,37]. Ce système de reconnaissance est donc apparu tôt dans l'évolution. On a même fabriqué des robots imprégnés de l'odeur des blattes qui arrivent à induire un regroupement des individus, les blattes sont leurrées et considèrent les robots comme des congénères [38].

2.4 Les fourmis artificielles

2.5 Principe

En informatique, fourmis artificielles représentent des modes multi-agents inspirés par le comportement des fourmis réelles. La communication à base de phéromones des fourmis biologiques est souvent le paradigme prédominant utilisé. Combinaisons de fourmis artificielles et des algorithmes de recherche locale sont devenus une méthode de choix pour de nombreuses tâches d'optimisation impliquant une sorte de graphique par exemple

véhicule acheminement et le routage internet. L'activité en plein essor dans ce domaine a conduit à des conférences dédiées uniquement aux fourmis artificielles, et pour de nombreuses applications commerciales par des entreprises spécialisées telles que AntOptima. À titre d'exemple, l'optimisation de colonie de fourmi est une classe d'algorithmes d'optimisation sur le modèle des actions d'une colonie de fourmis. Les fourmis artificielles (par exemple des agents de simulation) localiser des solutions optimales en se déplaçant dans un espace de paramètres représentant toutes les solutions possibles. Fourmis réelles fixent phéromones diriger l'autre aux ressources tout en explorant leur environnement. Les Fourmis simulées même enregistrer leurs positions et la qualité de leurs solutions, ainsi que dans les itérations de simulation plus tard plus de fourmis localiser meilleures solutions. Une variante de cette approche est l'algorithme des abeilles, qui s'apparente davantage aux habitudes d'alimentation de l'abeille, un autre insecte social.

Les inventeurs sont Frans Moyson et Bernard Manderick. Pionniers du champ comprennent Marco Dorigo, Luca Maria Gambardella.

2.6 Fourmis réelles Vs fourmis artificielles

Les fourmis virtuelles ont une double nature. D'une part, elles modélisent les comportements abstraits de fourmis réelles, et d'autre part, elles peuvent être enrichies par des capacités que ne possèdent pas les fourmis réelles, afin de les rendre plus efficaces que ces dernières. Nous allons maintenant synthétiser ces ressemblances et différences.

2.6.1 Points commun

- Colonie d'individus coopérants. Comme pour les fourmis réelles, une colonie virtuelle est un ensemble d'entités non-synchronisés, qui se rassemblent ensemble pour trouver une "bonne" solution au problème considéré. Chaque groupe d'individus doit pouvoir trouver une solution même si elle est mauvaise.

- Pistes de phéromones. Ces entités communiquent par le mécanisme des pistes de phéromone. Cette forme de communication joue un grand rôle dans le comportement

des fourmis : son rôle principal est de changer la manière dont l'environnement est perçu par les fourmis, en fonction de l'historique laissé par ces phéromones.

- Évaporation des phéromones. La méta-heuristique ACO comprend aussi la possibilité d'évaporation des phéromones. Ce mécanisme permet d'oublier lentement ce qui s'est passé avant. C'est ainsi qu'elle peut diriger sa recherche vers de nouvelles directions, sans être trop contrainte par ces anciennes décisions.

- Recherche du plus petit chemin. Les fourmis réelles et virtuelles partagent un but commun : recherche du plus court chemin reliant un point de départ (le nid) à des sites de destination (la nourriture).

- Déplacement locaux Les vraies fourmis ne sautent pas des cases, tout comme les fourmis virtuelles. Elles se contentent de se déplacer entre sites adjacents du terrain.

- Choix aléatoire lors des transitions. Lorsqu'elles sont sur un site, les fourmis réelles et virtuelles doivent décider sur quel site adjacent se déplacer. Cette prise de décision se fait au hasard et dépend de l'information locale déposée sur le site courant. Elle doit tenir compte des pistes de phéromones, mais aussi du contexte de départ (ce qui revient à prendre en considération les données du problème d'optimisation combinatoire pour une fourmi virtuelle).

2.6.2 Points de différence

Les fourmis virtuelles possèdent certaines caractéristiques que ne possèdent pas les fourmis réelles :

- Elles vivent dans un monde non-continu. Leurs déplacements consistent en des transitions d'état.

- Mémoire (état interne) de la fourmi. Les fourmis réelles ont une mémoire très limitée. Tandis que nos fourmis virtuelles mémorisent l'historique de leurs actions. Elles peuvent aussi retenir des données supplémentaires sur leurs performances.

- Nature des phéromones déposées. Les fourmis réelles déposent une information physique sur la piste qu'elles parcourent, là où les fourmis virtuelles modifient des informations dans les variables d'états associées au problème. Ainsi, l'évaporation des phéromones est une simple décrémentation de la valeur des variables d'états à chaque itération.

- Qualité de la solution. Les fourmis virtuelles déposent une quantité de phéromone proportionnelle à la qualité de la solution qu'elles ont découverte.

- Retard dans le dépôt de phéromone. Les fourmis virtuelles peuvent mettre à jour les pistes de phéromones de façon non immédiate : souvent elles attendent d'avoir terminé la construction de leur solution. Ce choix dépend du problème considéré bien évidemment.

- Capacités supplémentaires. Les fourmis virtuelles peuvent être pourvues de capacités artificielles afin d'améliorer les performances du système. Ces possibilités sont liées au problème et peuvent être :

- L'anticipation : la fourmi étudie les états suivants pour faire son choix et non seulement l'état local.

- le retour en arrière : une fourmi peut revenir à un état déjà parcouru car la décision qu'elle avait prise à cet état a été mauvaise.

2.7 Algorithme général (ACO)

Les algorithmes de colonies de fourmis sont des algorithmes inspirés du comportement des fourmis et qui constituent une famille de métaheuristiques d'optimisation.

Initialement proposé par Marco Dorigo et al. dans les années 1990, pour la recherche de chemins optimaux dans un graphe, le premier algorithme s'inspire du comportement des fourmis recherchant un chemin entre leur colonie et une source de nourriture. L'idée originale s'est depuis diversifiée pour résoudre une classe plus large de problèmes et

plusieurs algorithmes ont vu le jour, s'inspirant de divers aspects du comportement des fourmis.

En anglais, le terme consacré à la principale classe d'algorithme est « Ant Colony Optimization » (abrégé ACO). Les spécialistes réservent ce terme à un type particulier d'algorithme. Il existe cependant plusieurs familles de méthodes s'inspirant du comportement des fourmis. En français, ces différentes approches sont regroupées sous les termes : algorithmes de colonies de fourmis, optimisation par colonies de fourmis, fourmis artificielles, ou diverses combinaisons de ces variantes.

2.7.1 Principes

L'idée originale provient de l'observation de l'exploitation des ressources alimentaires chez les fourmis. En effet, celles-ci, bien qu'ayant individuellement des capacités cognitives limitées, sont capables collectivement de trouver le chemin le plus court entre une source de nourriture et leur nid.

Des biologistes ont ainsi observé, dans une série d'expériences menées à partir de 1989, qu'une colonie de fourmis ayant le choix entre deux chemins d'inégale longueur menant à une source de nourriture avait tendance à utiliser le chemin le plus court.

Un modèle expliquant ce comportement est le suivant :

- une fourmi (appelée « éclaireuse ») parcourt plus ou moins au hasard l'environnement autour de la colonie.
- si celle-ci découvre une source de nourriture, elle rentre plus ou moins directement au nid, en laissant sur son chemin une piste de phéromones.
- ces phéromones étant attractives, les fourmis passant à proximité vont avoir tendance à suivre, de façon plus ou moins directe, cette piste.
- en revenant au nid, ces mêmes fourmis vont renforcer la piste.

- si deux pistes sont possibles pour atteindre la même source de nourriture, celle étant la plus courte sera, dans le même temps, parcourue par plus de fourmis que la longue piste.
- la piste courte sera donc de plus en plus renforcée, et donc de plus en plus attractive.
- la longue piste, elle, finira par disparaître, les phéromones étant volatiles.
- à terme, l'ensemble des fourmis a donc déterminé et « choisi » la piste la plus courte.

La figure suivante (figure1.7) peut expliquer facilement les procédures précédentes.

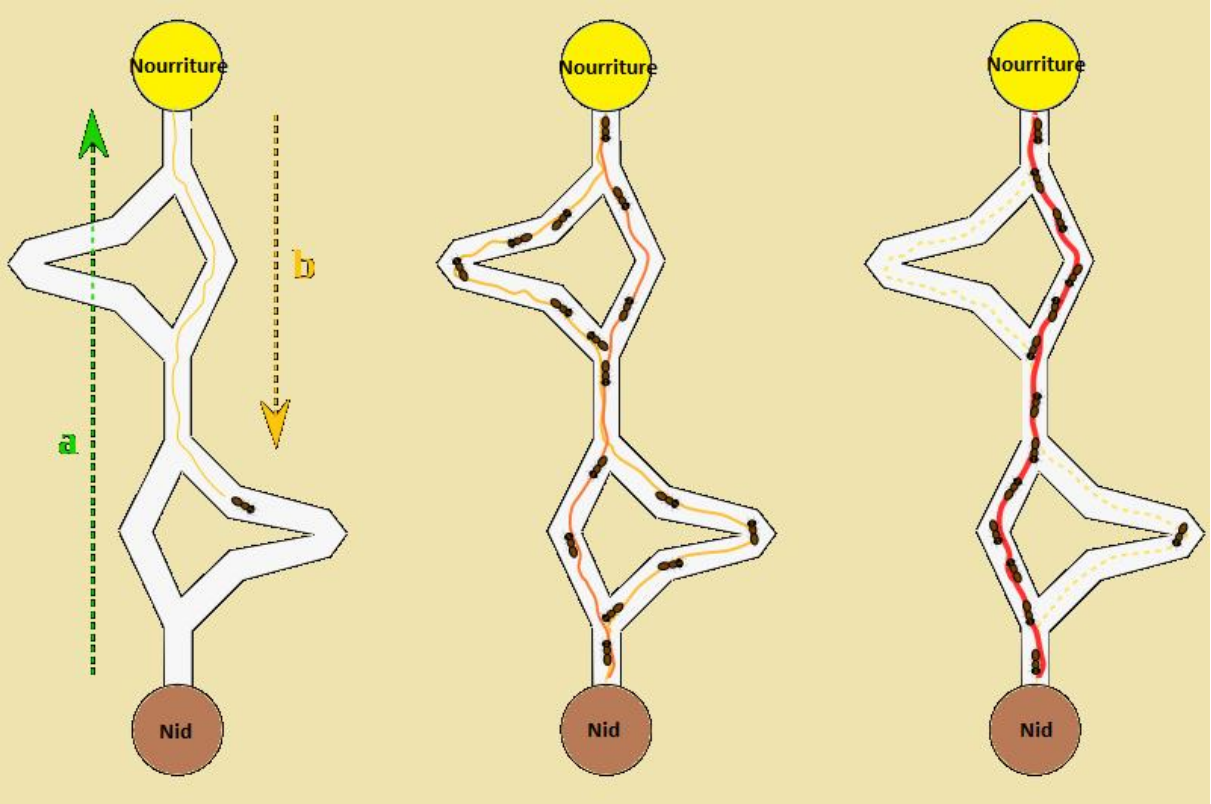

Figure 2-7 -Démarche des fourmis pour assebler de nourriture.

Les fourmis utilisent l'environnement comme support de communication : elles échangent indirectement de l'information en déposant des phéromones, le tout décrivant l'état de leur « travail ». L'information échangée a une portée locale, seule une fourmi située à l'endroit où les phéromones ont été déposées y a accès. Ce système porte le nom de «

stigmergie », et se retrouve chez plusieurs animaux sociaux (il a notamment été étudié dans le cas de la construction de piliers dans les nids de termites).

Le mécanisme permettant de résoudre un problème trop complexe pour être abordé par des fourmis seules est un bon exemple de système auto-organisé. Ce système repose sur des rétroactions positives (le dépôt de phéromone attire d'autres fourmis qui vont la renforcer à leur tour) et négatives (la dissipation de la piste par évaporation empêche le système de s'emballer). Théoriquement, si la quantité de phéromone restait identique au cours du temps sur toutes les branches, aucune piste ne serait choisie. Or, du fait des rétroactions, une faible variation sur une branche va être amplifiée et permettre alors le choix d'une branche. L'algorithme va permettre de passer d'un état instable où aucune branche n'est plus marquée qu'une autre, vers un état stable où l'itinéraire est formé des « meilleures » branches.

2.7.2 Extensions

Voici quelques-unes des variantes les plus populaires des algorithmes ACO :

- Elitist ant system: La meilleure globale solution depose phéromone sur chaque itération avec toutes les autres fourmis.
- Max-Min ant system (MMAS): Ajouté quantités Maximum et Minimum de phéromones [T_{max}, T_{min}], seulement meilleure globale ou meilleure itération tour dépose phéromone.
- Ant Colony System: précédement présenté.
- Rank-based ant system (ASrank): Toutes les solutions sont classées à l'aide de leur longueur. La quantité de phéromone déposée est ensuite pondérée pour chaque solution. les solutions avec des chemins plus courts déposent des phéromones plus que les solutions des chemins plus longs.
- Continuous orthogonal ant colony (COAC): le mécanisme de dépôt de phéromones de COAC est de permettre aux fourmis de rechercher des solutions collaborative et efficace. En utilisant une méthode de conception orthogonale, des fourmis dans le

domaine de faisabilité peuvent explorer leurs régions choisies rapidement et efficacement, avec une meilleure capacité de recherche mondiale et de précision.

- Ant Colony Optimization with Fuzzy Logic: Cette méthode présente l'intelligence floue en fourmis pour accélérer la recherche capacité.

2.7.3 Applications

ACO algorithmes ont été appliquées à de nombreux problèmes d'optimisation combinatoire, allant de l'affectation quadratique pour le repliement des protéines ou routage des véhicules et de beaucoup de méthodes dérivées ont été adaptés aux problèmes dynamiques variables réelles, problèmes stochastiques, multi-cibles et des implémentations parallèles. Il a également été utilisé pour produire des solutions quasi-optimales pour le problème du voyageur de commerce. Ils ont un avantage sur le recuit simulé et algorithme génétique approches des problèmes similaires lorsque le graphe peut changer dynamiquement, l'algorithme de colonie de fourmis peut être exécuté en permanence et de s'adapter aux changements en temps réel. Il s'agit de l'intérêt pour le routage du réseau et des systèmes de transport urbain. Brièvement on peut compter :

- Problème d'ordonnancement (Scheduling problem)
  - Job-shop scheduling problem (JSP)
  - Open-shop scheduling problem (OSP)
  - Permutation flow shop problem (PFSP)
  - Single machine total tardiness problem (SMTTP)
  - Single machine total weighted tardiness problem (SMTWTP)
  - Resource-constrained project scheduling problem (RCPSP)
  - Group-shop scheduling problem (GSP)

- Single-machine total tardiness problem with sequence dependent setup times (SMTTPDST)
- Multistage Flowshop Scheduling Problem (MFSP) with sequence dependent setup/changeover times

➢ Problème routage de véhicule (Vehicle routing problem)

- Capacitated vehicle routing problem (CVRP)
- Multi-depot vehicle routing problem (MDVRP)
- Period vehicle routing problem (PVRP)
- Split delivery vehicle routing problem (SDVRP)
- Stochastic vehicle routing problem (SVRP)
- Vehicle routing problem with pick-up and delivery (VRPPD)
- Vehicle routing problem with time windows (VRPTW)
- Time Dependent Vehicle Routing Problem with Time Windows (TDVRPTW)
- Vehicle Routing Problem with Time Windows and Multiple Service Workers (VRPTWMS)

➢ Problème d'affectation (Assignment problem)

- Quadratic assignment problem (QAP)
- Generalized assignment problem (GAP)
- Frequency assignment problem (FAP)
- Redundancy allocation problem (RAP)
- Problème de set (Set problem)

- Set covering problem(SCP)
- Set partition problem (SPP)
- Weight constrained graph tree partition problem (WCGTPP)
- Arc-weighted l-cardinality tree problem (AWlCTP)
- Multiple knapsack problem (MKP)
- Maximum independent set problem (MIS)

➢ Autres

- Classification
- Connection-oriented network routing
- Connectionless network routing
- Data mining
- Discounted cash flows in project scheduling
- Distributed Information Retrieval
- Grid Workflow Scheduling Problem
- Image processing
- Intelligent testing system
- System identification
- Protein Folding
- Power Electronic Circuit Design

2.8 Historique

- 1959, Pierre-Paul Grassé invente la théorie de la stigmergie pour expliquer le comportement de construction du nid chez des termites.
- 1983, Deneubourg et ses collègues étudient le comportement collectif des fourmis.
- 1988, Moyson et Manderick présentent un article sur l'auto-organisation chez les fourmis.
- 1989, travaux de Goss, Aron, Deneubourg et Pasteels, sur le comportement collectifs des fourmis Argentines, qui donneront l'idée des algorithmes de colonies de fourmis.
- 1989, implémentation d'un modèle de comportement de recherche de nourriture par Ebling et ses collègues14 ;
- 1991, M. Dorigo propose le Ant System dans sa thèse de doctorat (qui ne sera publiée qu'en 19922). Il fait paraître, avec V. Maniezzo et A. Colorni, un rapport technique15, qui sera publié cinq ans plus tard.
- 1995, Bilchev et Parmee publient la première tentative d'adaptation aux problèmes continus.
- 1996, publication de l'article sur le Ant System.
- 1996, Stützle et Hoos inventent le MAX-MIN Ant System.
- 1997, Dorigo et Gambardella publient le Ant Colony System.
- 1997, Schoonderwoerd et ses collègues conçoivent la première application aux réseaux de télécommunications.
- 1997, Martinoli et ses collègues s'inspirent des algorithmes de colonies de fourmis pour le contrôle de robots.

- 1998, Dorigo lance la première conférence dédiée aux algorithmes de colonies de fourmis.
- 1998, Stützle propose les premières implémentations parallèles.
- 1999, Bonabeau et ses collègues font paraître un livre traitant principalement des fourmis artificielles.
- 1999, premières applications pour le routage de véhicule, l'assignement quadratique, le sac à dos multi-dimensionnel.
- 2000, numéro spécial d'une revue scientifique sur les algorithmes de colonies de fourmis.
- 2000, premières applications à l'ordonnancement, l'ordonnancement séquentiel, la satisfaction de contraintes.
- 2000, Gutjahr donne la première preuve de convergence pour un algorithme de colonies de fourmis.
- 2001, première utilisation des algorithmes de colonies de fourmis par des entreprises (Eurobios et AntOptima).
- 2001, Iredi et ses collègues publient le premier algorithme multi-objectif.
- 2002, premières applications à la conception d'emploi du temps, les réseaux bayésiens.
- 2002, Bianchi et ses collègues proposent le premier algorithme pour problème stochastique.
- 2004, Zlochin et Dorigo montrent que certains algorithmes sont équivalents à la descente stochastique de gradient, l'entropie croisée et les algorithmes à estimation de distribution8.
- 2005, premières applications au repliement de protéines.

2.9 Conclusion

D’après ce chapitre qui m’a bien introduire et expliquer la fourmi et leurs colonie ainsi qu’après ce chapitre j’étais capable de construire une conception de notre fourmi de simulation (artificielle), ces comportements, leur propriétés générales et les besoins qui donne la simulation les bons conditions pour qu’elle marche bien, maintenant je veux la modélise dans le chapitre suivant.

Chapitre 3

Modélisation & Implémentation

Introduction

Comme nous avons expliqués dans le chapitre précédent que les phénomènes collectifs en biologie sont une source d'inspiration pour proposer des méthodes multi-agents de résolution de problèmes. Ils permettent l'élaboration de comportements individuels simples produisant collectivement des phénomènes complexes, dans ce chapitre nous allons modéliser le phénomène collectif de colonie de fourmis en biologie (recherche de nourriture).

Mais d'abord pour faire une bonne réalisation, il faut faire une bonne modélisation, Dans notre modélisation, nous avons choisi le langage de modélisation AUML (AgentUML), qui est nécessaire de la démarche de conception.

3.1 Langage de modélisation (AUML = Agent + UML)

La modélisation consiste à créer une représentation simplifiée d'un problème:

Le Modèle

Grâce au modèle il est possible de représenter simplement un problème, un concept et le simuler. Le modèle constitue ainsi une représentation possible du système pour un point de Vue donné.

AUML est un mécanisme de modélisation des interactions entre agents.

- Pourquoi AUML?

« Pour parvenir à vendre la technologie agent, il faut arriver à réduire le risque inhérent à toute nouvelle technologie en la présentant comme une extension de méthodes déjà éprouvées et en fournissant des outils pour son utilisation»

- L'adoption large dont UML a fait l'objet
- Le paradigme agent pourrait succéder valablement à celui objet
- Les limitations d'UML face au paradigme agent
- Les agents sont proactifs et autonomes
- Les agents interagissent les uns avec les autres

3.2 Architecture générale du système

Dans notre système nous avons distingués trois catégories d'agents :

- Un agent de gestion du système (agent moniteur), qui contient les informations nécessaires au fonctionnement du système multi-agents.
- Agent fourmi qui représente l'agent réactif et autonome pour chercher et rassembler de nourriture pour leur nid.

Des agents nommés agents d'environnement (agent terre, agent nid, agents nourritures, agents obstacles), chargés de simuler l'environnement pour l'agent fourmi.

La figure ci-dessous représente l'architecture générale du système proposé :

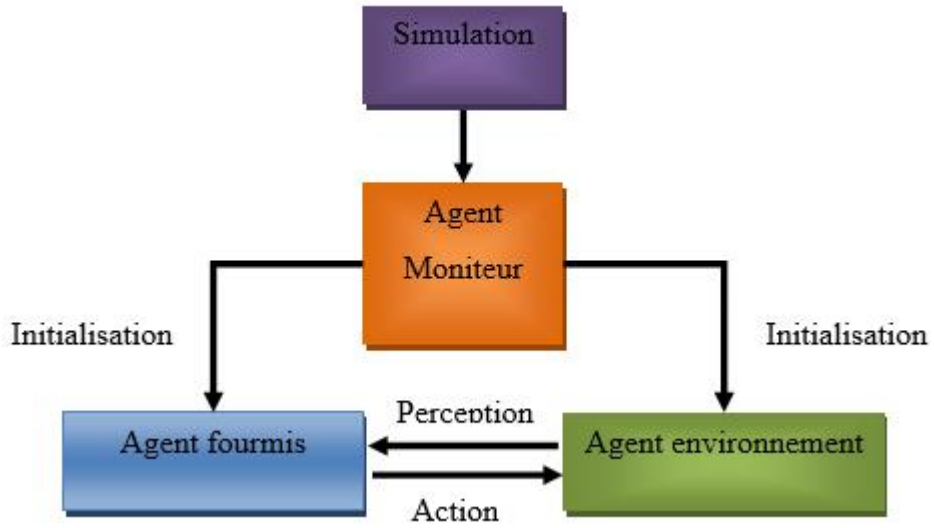

Figure 3-1 -architecture générale du système proposé.

Ci-dessous une figure représentant le diagramme de classe du système multi-agent modélisé :

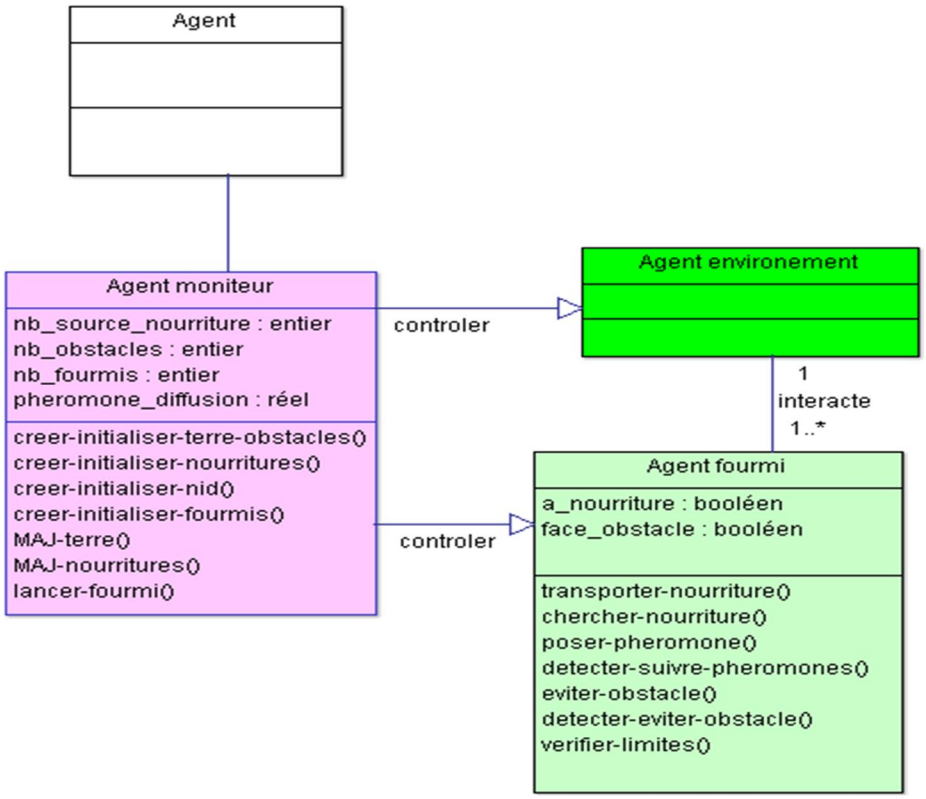

Figure 3-2 -Architecture générale du système proposé (diagramme de classe AUML).

3.3 Architecture de chaque agent

Nous allons maintenant détailler l'architecture de chaque agent du système :

3.3.1 Agent moniteur

L'agent moniteur a les informations nécessaires pour la simulation qui sont : nombre des obstacles, nombre des sources de nourriture, nombre des fourmis, ration de diffusion de phéromone. Son rôle peut être décrit dans les notes suivantes :

3.3.1.1 Création et initialisation de l'agent

Dans cette étape l'agent moniteur va :

- Créer l'agent terre et initialiser ces attributs (pheromone, couleur)
- Créer les agents obstacles et initialiser l'attribut (taille) de chacun puis les placer dans des lieux aléatoires
- Créer les agents nourriture dans des lieux aléatoires mais différent des agents obstacles
- Créer et placer l'agent nid dans un lieu aléatoire à condition que ce lieu ne porte pas l'un des types des deux agents précédents
- Créer le nombre des fourmis spécifier par (nb_fourmis) et initialiser ces emplacements dans une seul place celle de l'emplacement de nid.

3.3.1.2 Lancement

Dans le lancement l'agent moniteur va boucler dans deux choses en principe :

- la mise à jour de l'agent environnement, dans lequel il va colorer les pièces de terre à base de l'attribut (pheromone) et supprimer les agents nourriture qui a l'attribut (quantite_nourriture < n telle que n nombre entier positif).
- lancement de l'agent fourmis.

L'agent moniteur est représenté par la classe AUML dans la figure ci-dessous.

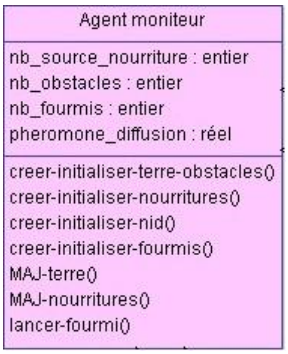

Figure 3-3 -La classe de l'agent moniteur.

3.3.2 Agent environnement

L'agent environnement représente les quatre types d'agents (terre, nid, obstacle, nourriture). Ce dernier est joue le rôle de faire simule l'environnement réel d'une fourmi dans le quelle :

- Agents terre garde la quantité de phéromone poser par l'agent fourmis
- Agent nourriture est le cible des agents fourmis au temps de cherche et qui à l'attribut (quantite_nourriture) qui signifie le reste de quantité de nourriture dans cette place
- Agent nid est le point de dépôt de nourriture
- Agent obstacle est l'obstacle qui encombre les agents fourmis au temps de leur mouvement.

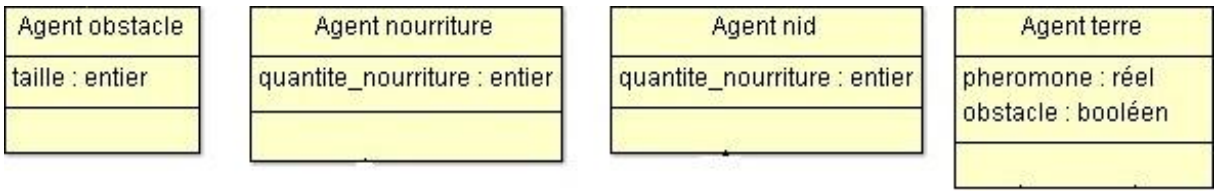

Figure 3-4 -les 4 les classes qui représente l'agent environnement.

3.3.3 Agent fourmi

Les agents fourmis simulent le comportement de la colonie de fourmis à la recherche de nourriture dans le quelle le fonctionnement de chacun de ces dernier basé sur :

3.3.3.1 Perceptions et actions :

Les perceptions offrent les informations locales disponibles dans l'environnement et sur lesquelles se base le choix d'action de l'agent :

- Il détecte la nourriture dans le point où il se localise puis la prend
- S'il porte de nourriture, il l'amène au nid, en posant la phéromone
- S'il y a d'obstacle face à lui, éviter le (changer la direction)
- S'il est au nid, déposer la nourriture
- Il détecte de phéromone, il suive la phéromone
- Il n'a pas trouvé de nourriture, avancer et chercher de nourriture

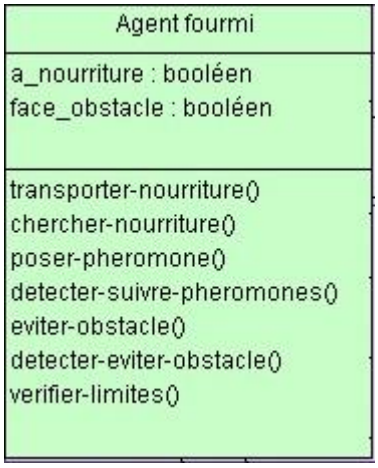

Figure 3-5 -classe de l'agent fourmi.

3.3.3.2 Les interactions des agents fourmis

Les interactions des agents sont basées sur le principe de coordination par stigmergie : les actions des agents modifient l'environnement, de son côté l'environnement aussi modifie les actions futurs des agents. Ce principe est modélisé implicitement dans le comportement de déplacement influencé par la quantité de phéromone.

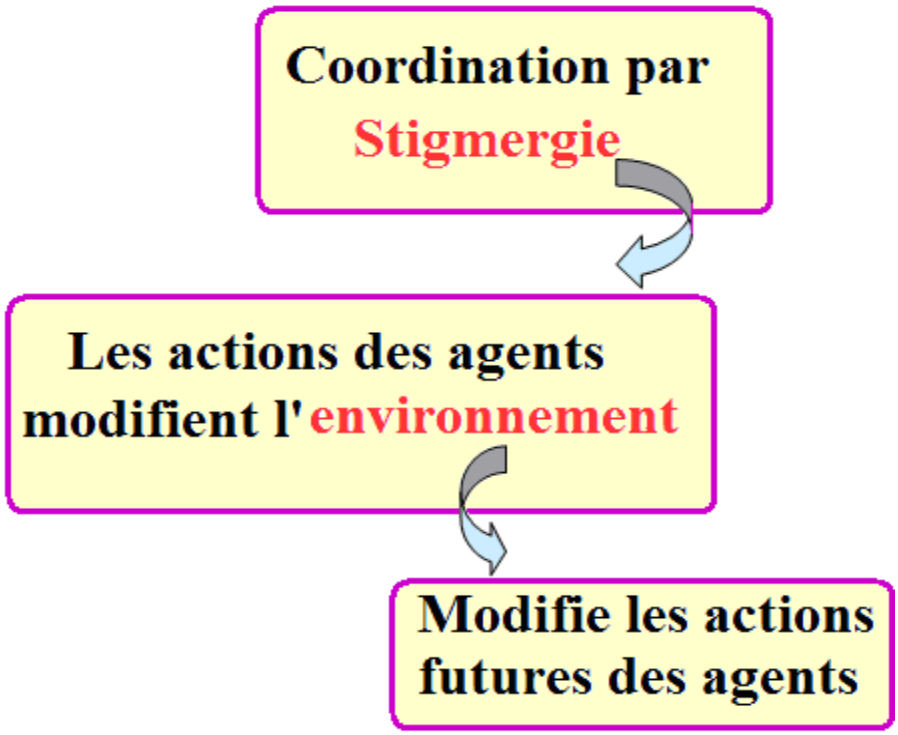

Figure 3-6 -principe de coordination par stigmergie.

3.4 L'interaction entre les agents

3.4.1 Diagramme de séquence

Les diagrammes de séquences sont la représentation graphique des interactions entre les agents et le système selon un ordre chronologique dans la formulation

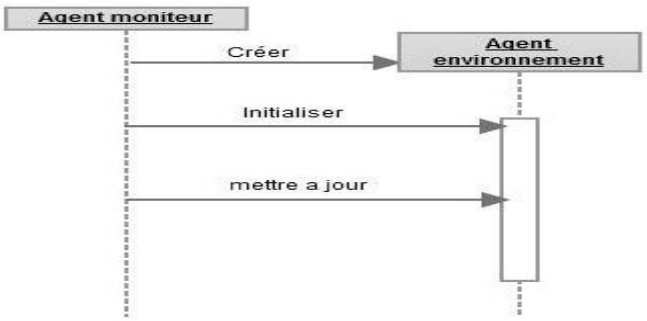

Figure 3-7 -diagramme de séquence entre les agents moniteur et agent environnement.

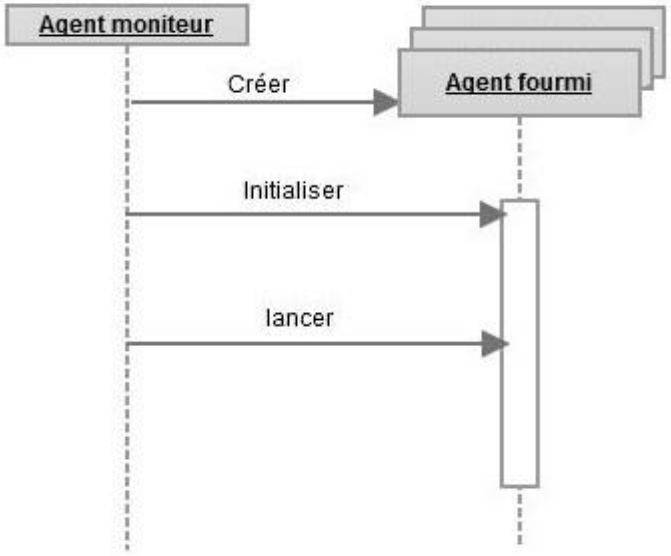

Figure 3-8 -diagramme de séquence entre les agents moniteur et agent fourmi.

3.4.2 Diagramme d'activités

Un diagramme d'activité permet de modéliser un processus interactif, global ou partiel pour un système donné. Il est recommandable pour exprimer une dimension temporelle sur une partie du modèle, à partir de diagrammes de classes ou de cas d'utilisation.

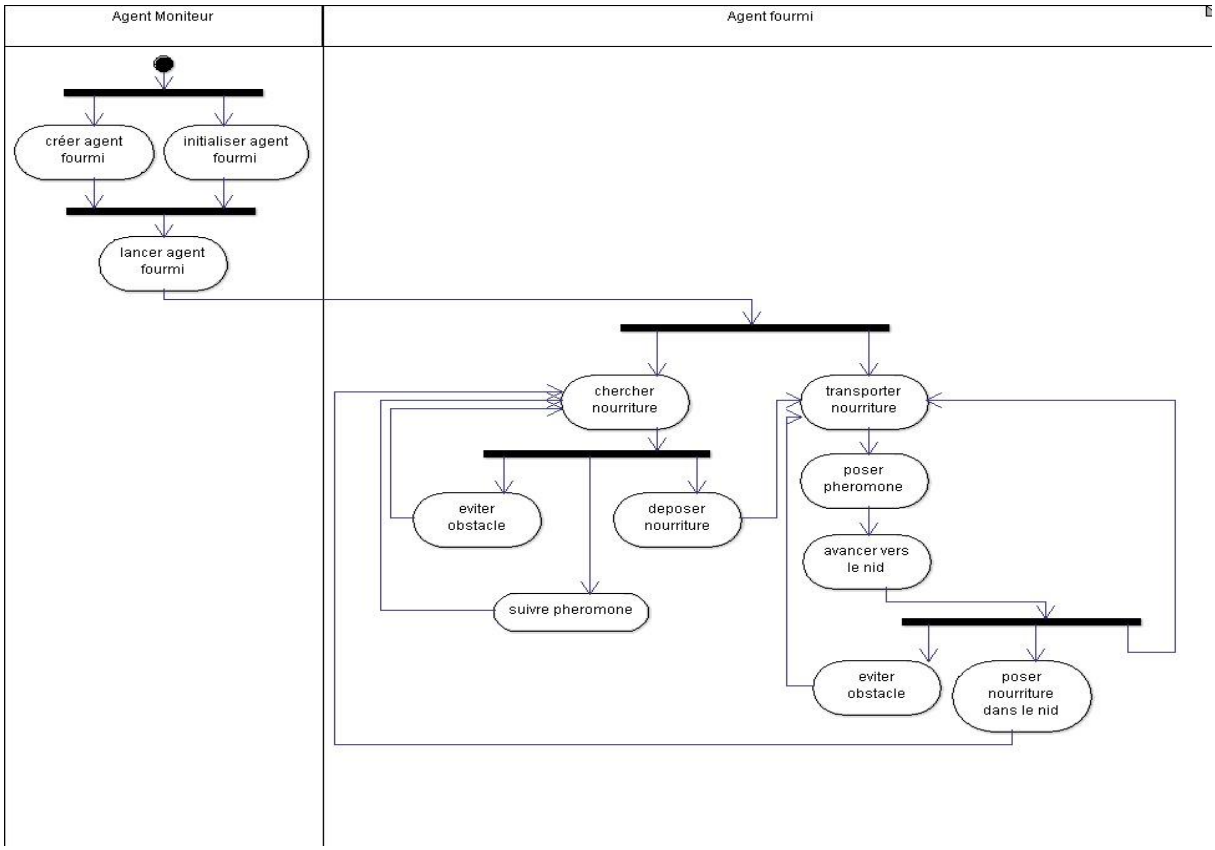

Figure 3-9 -diagramme d'activité entre agent moniteur et agent fourmis.

3.5 L'implémentation

Nous allons implémenter chaque composant de système proposé en suivant la modélisation détaillée dans le chapitre précédent. C'est dans la phase d'implémentation que nous allons découvrir les capacités et les avantages offertes par le système multi-agent du point de vue du parallélisme d'exécution des différents composants du système, l'autonomie des agents, et leur interaction. C'est avec la mise en œuvre des agents, puis des systèmes multi-agents, que nous voyons apparaître des systèmes dans lesquels, en plus de résoudre les problèmes pour lesquels elles ont été conçues, les entités computationnelles vont simultanément être capables de résoudre le problème de leur propre organisation, soit pour éviter les conflits, soit pour améliorer leur fonctionnement collectif.

3.6 Implémentation de l'architecture du système

Après l'étude et la modélisation du phénomène biologique « recherche de nourriture dans la colonie des fourmis », nous commençons la simulation de ce phénomène collectif.

3.6.1 La simulation du phénomène

3.6.1.1 Description du langage NetLogo

NetLogo est un environnement de modélisation programmable permettant de simuler des phénomènes naturels et sociaux. Il a été créé par Uri Wilenski en 1999 et son développement est poursuivi de manière continue par le Center for Connected Learning and Computer-Based Modeling.

Il convient tout particulièrement à la modélisation de systèmes complexes évoluant au cours du temps. Les « modélisateurs » peuvent donner des instructions à des centaines ou des milliers d'« agents » opérant indépendamment les uns des autres. Ce qui permet d'explorer les liens entre les comportements des individus à leur niveau et les schémas généraux (comportements de groupe ou de masse) qui émergent des interactions entre de nombreux individus.

Aussi il permet aux étudiants d'ouvrir des modèles et de « jouer » avec leurs simulations pour explorer leurs comportements en faisant varier diverses conditions (paramètres). C'est aussi un environnement de programmation permettant aux étudiants, aux enseignants et aux développeurs de cours de formation de créer leurs propres modèles. NetLogo est suffisamment simple pour permettre aux étudiants et aux ensei-gnants d'utiliser les simulations sans trop de difficultés, voire d'en concevoir eux-mêmes. Mais il est aussi assez performant pour servir d'outil puissant pour des chercheurs dans de nombreux domaines.

Fonctionnalités :

Consultez la liste ci-dessous pour prendre connaissance des fonctionnalités offertes par NetLogo.

- ✓ Système :

Multi-plateforme : tourne sur Mac, Windows, Linux, et autres

- ✓ Langage :

Entièrement programmable

Structure du langage simple

Dialecte Logo étendu pour supporter les agents

Agents mobiles (les tortues) se déplaçant sur une grille formée d'agents stationnaires (les patches)

Création de liens entre les tortues pour former des agrégats, des réseaux et des graphes

Important vocabulaire de primitives intégrées au langage

Calculs en virgule flottante double précision (IEEE 754)

Fonctionnement absolument identique sur toutes les plateformes

- ✓ Environnement :

Observation des modèles en 2D ou en 3D

Formes vectorielles redimensionnables et pivotantes

Étiquetage des tortues et des patches

Centre de commande pour un pilotage interactif (en direct) de la simulation

Interface pouvant recevoir des boutons, des curseurs, des commutateurs, des traceurs de courbes, des sélecteurs, des moniteurs, des boîtes de texte, des notes, des zones de sortie

Variateur de vitesse pour accélérer ou ralentir la simulation

Système de sortie graphique puissant et souple

Panneau d'informations pour annoter les modèles

- ✓ HubNet: simulation participative utilisant des machines en réseau

Moniteurs d'agent pour inspecter et contrôler les agents

Fonctions d'exportation et d'importation (exportation des données, sauvegarde et restauration de l'état d'un modèle, création et enregistrement d'une animation)

Outil BehaviourSpace (espace de comportements) utilisé pour collecter des données provenant de plusieurs sessions de simulations

Modélisation de systèmes dynamiques

- ✓ Web:

Modèles enregistrables sous forme d'applets pouvant ensuite être intégrés dans des pages web (note: certaines fonctionnalités ne sont pas disponibles dans les applets, telles que certaines ex-tensions et la visualisation 3D)

*Pour plus d'information vous pouvez consultez le manuel.

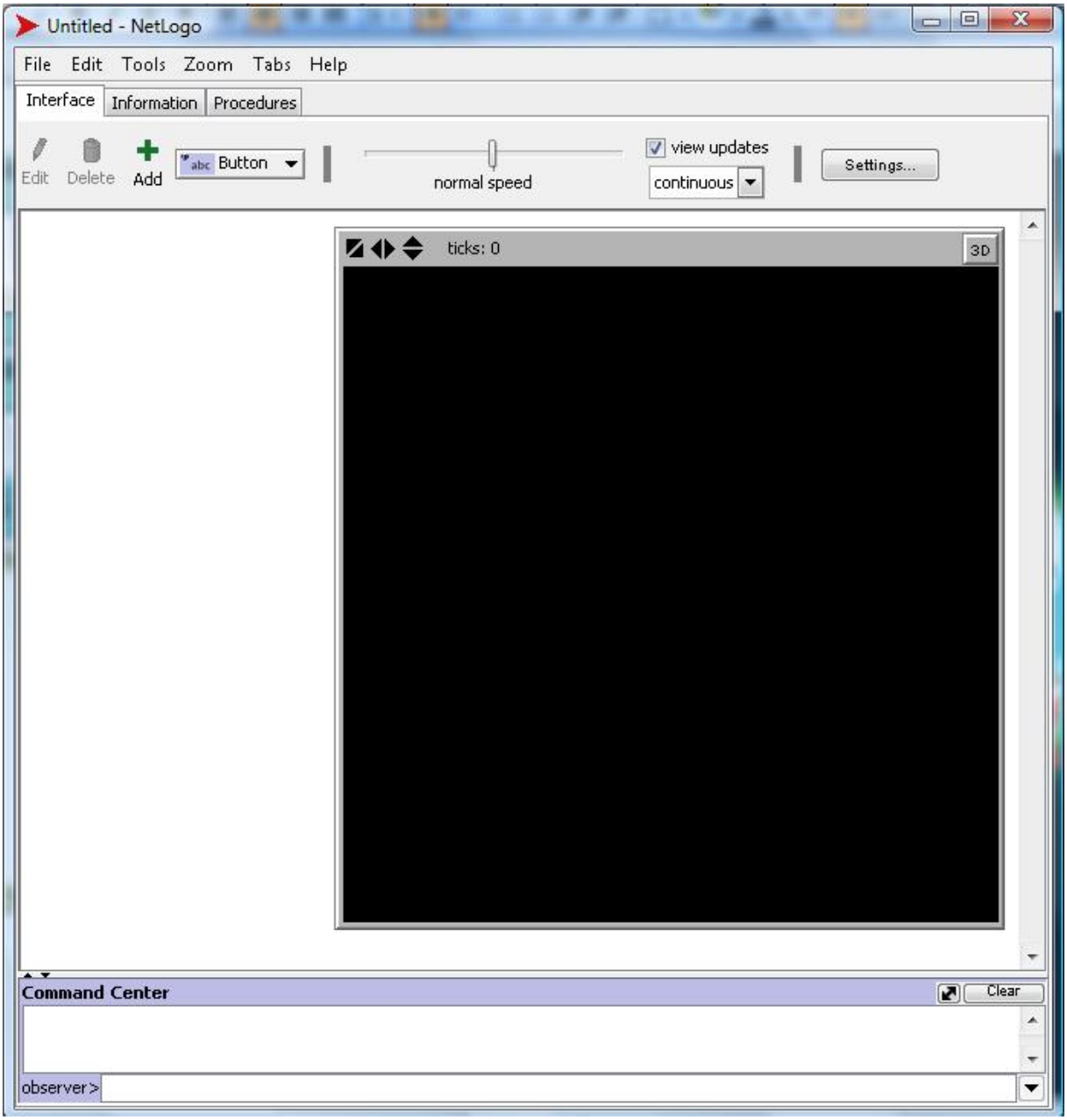

Figure 3.11 La plate-forme NetLogo

La figure ci-dessous est un exemple d'exécution mon projet.

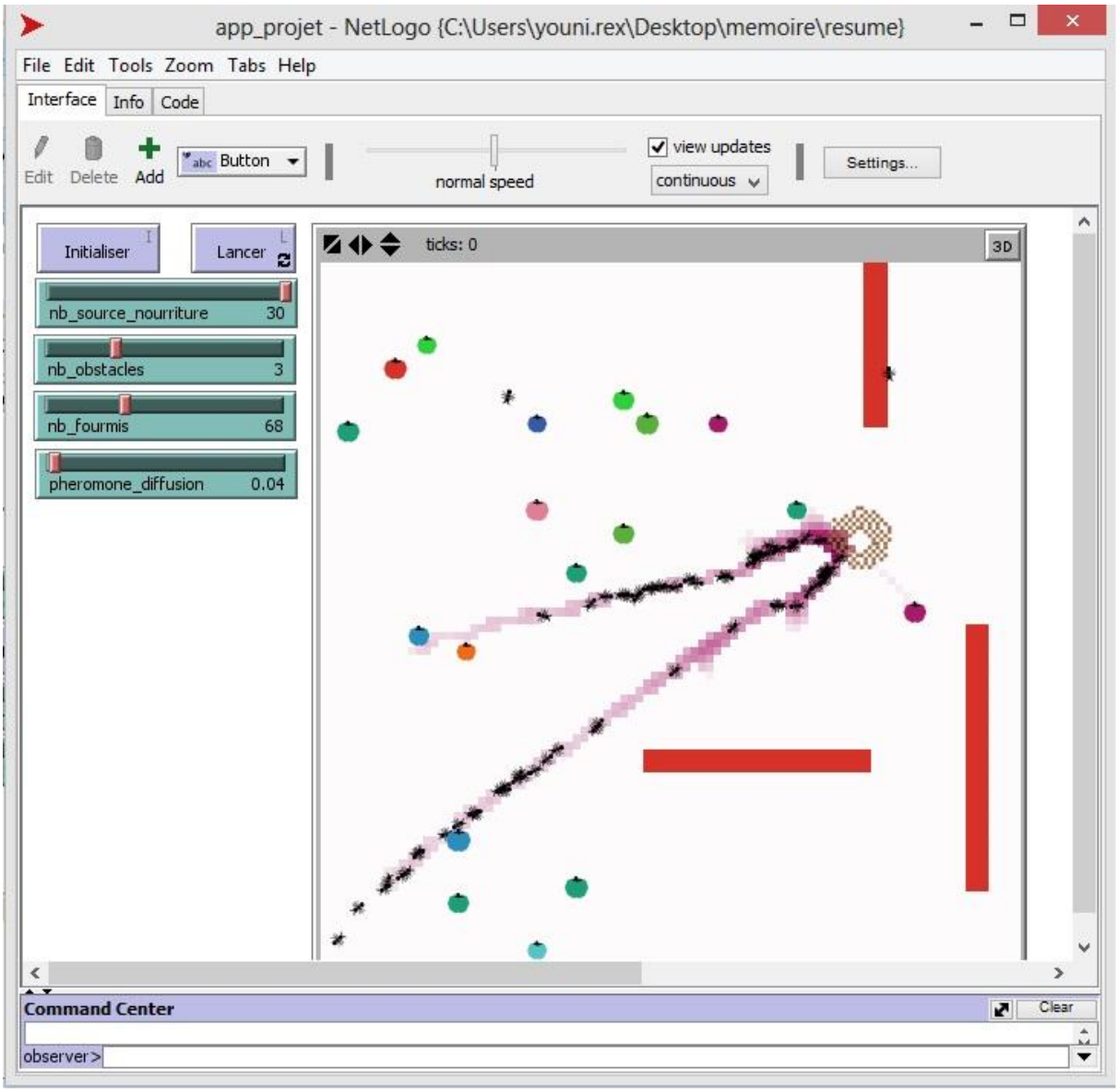

Figure 3.12 Exemple du mon projet simulation

3.6.1.2 Implémentation de la simulation

L'environnement est modélisé sous la forme d'une grille carrée, chaque case étant associée à un piquet, dans le NetLogo le piquet est représenté par le « *patch* » qui va représenter l'agent terre. L'agent représenté dans le NetLogo par le « *turtle* » qui va nous présente les agents (fourmis, nid, nourriture, obstacle), et n'oublie pas l'agent moniteur qui on va le représenter par le « observer » donc :

Agents : fourmis, nid, nourriture, obstacle → Turtle

Agent moniteur → Observer

Agent terre → patch

Comme j'ai expliqué précédemment dans la partie de la modélisation que les agent terre, nourriture, nid n'ont pas que des attributs (pas de fonctions ou procédures) alors j'ai précis les codes des deux essentiels agent (moniteur et fourmis).

- **Agent moniteur**

A deux procédures principales :

- Procédure d'initialisation de simulation (monde virtuel) :

```
Procédure initialiser()
    créer-initialiser-environnement()
    créer-initialiser-fourmis(cordX cordY)
Fin procédure
```

- Procédure de mise à jour et le lancement des agents fourmis

```
Procédure lancer()
        MAJ-environnement()
        lancer-fourmi()
Fin procédure
```

- **Agent fourmis**

```
Procédure lancer-fourmi()
    Si(nourriture)
        transporter-nourriture()
    Sinon
        chercher-nourriture()
    Fin Si
Fin procédure
```

3.6.2 Teste de projet

Apres la lancement de simulations on peut clairement voire que l'espace de simulation dans le programme comme est-il un vrais pièce de terre sur elle des sources de nourriture (pomme), des obstacles (rectangles rouges) et des fourmis qui bougent aléatoirement cherchant de nourriture.

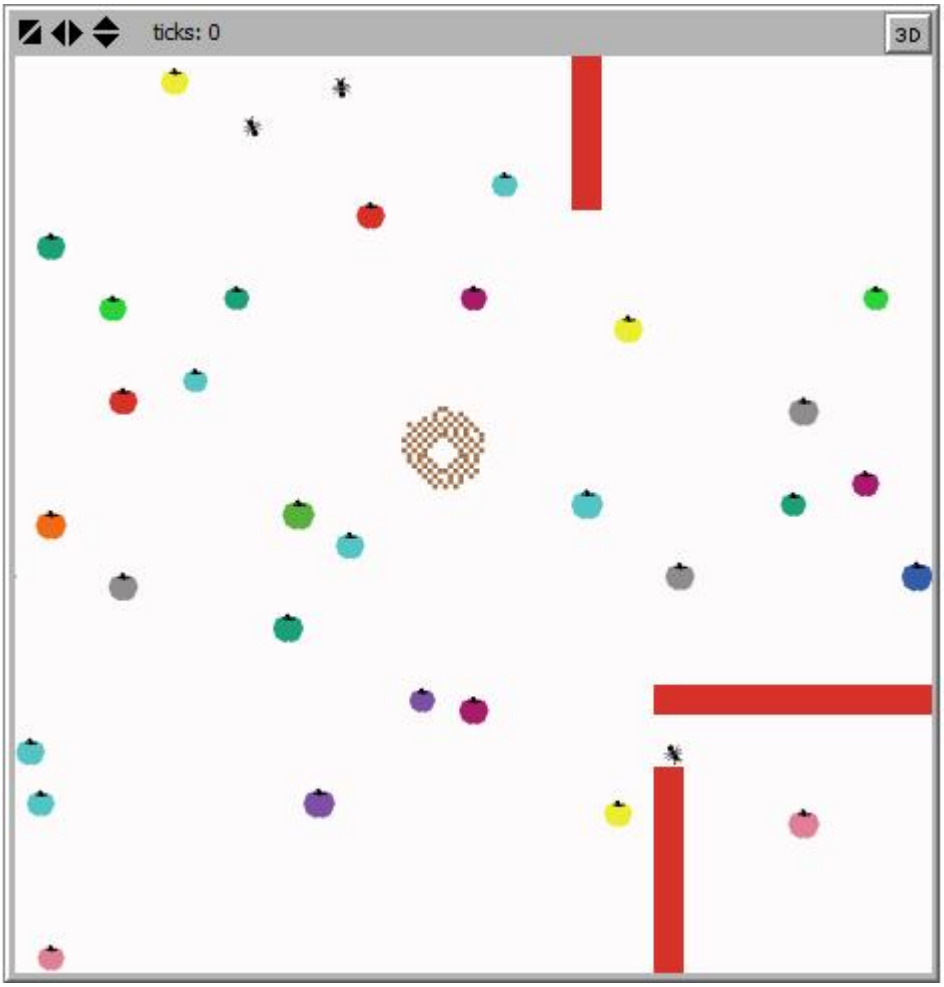

Figure 3-10 -Fourmis tant de recherche de nourriture.

Quand une fourmi touche une pomme elle redirige directement vers le nid et déplace pas par pas vers mais elle colore chaque bloque elle passe sur lui par le couleur violet, cet couleur représente le phéromone, qui on peut le voir change de couleur vers le blanc (ça c'est la phénomène d'évaporation).

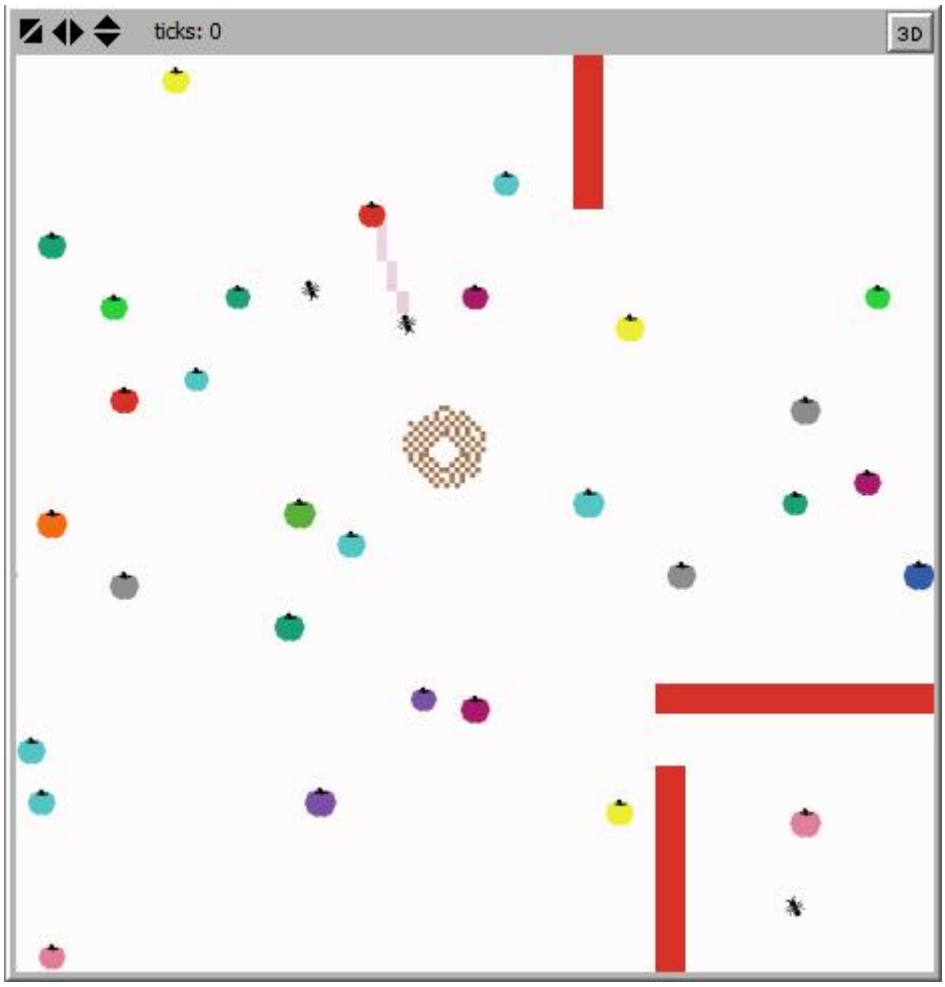

Figure 3-11-Fourmis a de nourriture et dirige vers le nid.

Apres que la fourmi arrive au nid elle retourne sur le même chemin qui a la phéromone qu'elle a déposée jusqu'à les sources de nourriture précèdent.

Si une autre fourmi croise la trace de la phéromone elle suive la trace jusqu'à l'emplacement de nourriture.

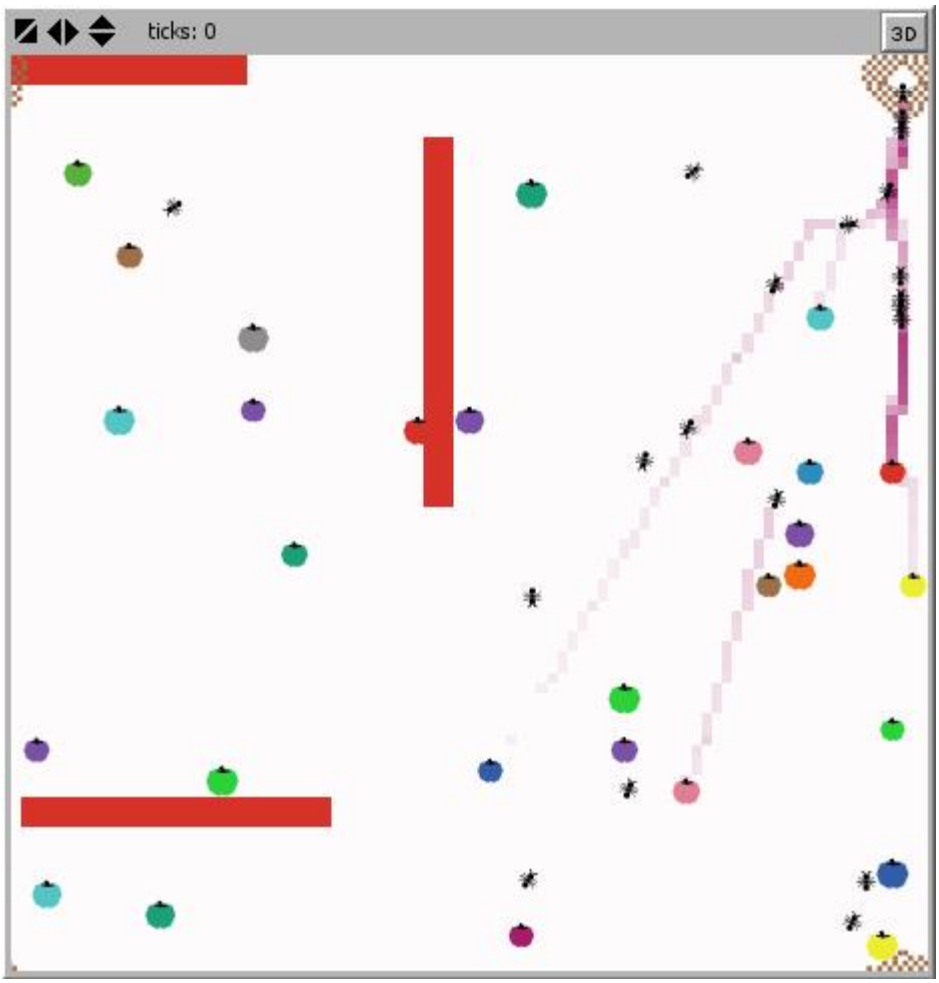

Figure 3-12 -Fourmis suivent la trace de phéromone.

Quand une fourmi face un obstacle elle immédiatement l'évite par changer de direction.

3.6.3 Remarques

Apres avoir faire la simulation plusieurs fois j'ai facilement compris que :

- les nombres des fourmis dans une colonie est vraiment influence sur la performance du nid (plus de fourmis = plus de nourriture dans un temps plus bas).
- Ici seulement la phéromone joue le rôle de communication entre une fourmi et les autres.

3.7 Conclusion

Dans ce chapitre nous avons vus que l'utilité de la modélisation de notre système est faciliter l'implémentation, , dans ce contexte nous avons proposé une modélisation en s'inspirant le phénomène collectif de colonie de fourmis Dans ce chapitre nous avons donné les détails de cette modélisation, en détaillant l'architecture de chaque composant, nous utilisons le langage de modélisation AUML, car il est adapté pour modelé le système multi agents, les diagramme de AUML (processus de modélisation) utilisé dans cette modélisation est : la diagramme de classe , la diagramme de séquence et diagramme d'activité. Encore la simulation des agents et du phénomène en générale a nous donné une expérience et une voire de l'importance de simulation et ces outils surtout dans l'univers des agents.

Conclusion générale et Perspectives

Dans ce mémoire nous avons fait la réalisation de la simulation de comportement collectif de colonie de fourmis en se basant sur le système multi-agents.

L'une des difficultés au sein des systèmes multi-agents destinés à effectuer de la résolution collective de problème est de concevoir des formes d'interaction entre les agents et leur environnement qui permettent l'observation de propriétés collectives complexes. L'étude des modèles sociaux biologiques est alors une manière d'aborder la résolution de tels problèmes par une approche décentralisée privilégiant un comportement individuel et fournissant des mécanismes robustes, adaptatifs et simples.

Deux objectifs à la modélisation du comportement des fourmis : permettre aux biologistes de mieux comprendre et observer les mécanismes qui régissent ces sociétés (vérification de la capacité de prédiction du modèle) et imaginer des nouvelles méthodes de résolution de problèmes complexes (allocation des taches en robotique, optimisation combinatoire, routage dans les réseaux, ...)

Après la réalisation de l'application, nous avons remarqué l'émergence de nouvelles structures, qu'on n'a pas modélisé ou programmé comme la recherche de plus court chemin entre le nid et la nourriture.

Dans le futur on peut utiliser la simulation de comportement collectif de fourmis pour la résolution de plusieurs type de problèmes telle que l'optimisation par colonie de fourmis..

Bibliographie

[1] Stuart J. Russell & Peter Norvig: Artificial Intelligence: A Modern Approach. Pearson Education, 2003.

[2] Michael Wooldridge: Intelligent agents. In Gerhard Weiss, éditeur: Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, chapitre 1, pages 27–77. The MIT Press, Cambridge, MA, 1999.

[3] M.E. Bratman, D.J. Israel et M.E. Pollack: Plans and resource-bounded practical reasoning. Computational intelligence, 4(3):349–355, 1988.

[4] Nicholas Jennings, Katia Sycara & Michael Wooldridge: A roadmap of agent research and development. Autonomous Agents and Multi-Agent Systems, 1(1):7 – 38, July 1998.

[5] Brahim Chaib-draa, Imed Jarras et Bernard Moulin : Systèmes multiagents : Principes généraux et applications. In J. P. Briot et Y. Demazeau, éditeurs: Agent et systèmes multiagents. Hermès, 2001.

[6] Y. Shoham, K. Leyton-Brown et K. Leyton-Brown : Multiagent Systems : Algorithmic, Game-Theoretic, and Logical Foundations. Cambridge University Press, 2008.

[7] N. Vlassis : A Concise introduction to multiagent systems and distributed artificial intelligence. Synthesis Lectures on Artificial Intelligence and Machine Learning, 1(1):1–71, 2007.

[8] Clint Heinze, Bradley Smith et Martin Cross: Thinking quickly: Agents for modeling air warfare. In Proceedings of the 9th Australian Joint Conference on AI (Ai'98), Australie, 1998.

[9] Fipa acl message structure specification, 2000. Foundation for Intelligent Physical Agents, http://www.fipa.org/specs/fipa00061/XC00061D.html/.

[10] Sandip Sen et Gerhard Weiss: Learning in multiagent systems. In Gerhard Weiss, éditeur: Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, chapitre 6, pages 259–298. The MIT Press, Cambridge, MA, 1999.

[11] PASSERA L. La véritable histoire des fourmis, Fayard, Paris, 2006.

[12] PASSERA L. ARON S. Les fourmis : comportement, organisation sociale et évolution, Les Presses scientifiques du CNRC, Ottawa, 2005.

[13] HAMILTON W. « The genetical evolution of social behaviour I », J. Theor. Biol. vol.7 , n°1-16, 1964.

[14] HAMILTON W. « The genetical evolution of social behaviour II », J. Theor. Biol.vol. 7, p. 17–52, 1964.

[15] DAHBI A. JAISSON P. LENOIR A. HEFETZ A. « Comment les fourmis partagent leur odeur », La Recherche, p. 32–34, 1998.

[16] LENOIR A. FRESNEAU D. ERRARD C. HEFETZ A. « Individuality and colonial identity in ants : the emergence of the social reprensentation concept », DETRAIN C. DENEUBOURG J.PASTEELS J. Eds. Information Processing in Social Insects, p. 219–237, Birkhäuser Verlag Basel, Suisse, 1999.

[17] NICOLIS S. C. P. I. Self-organization in non-equilibrium systems, Wiley and Sons,

New York, 1977.

[18] BONABEAU E. THERAULAZ G. DENEUBOURG J. ARON S. CAMAZINE S. «Selforganization in social insects», Trends Ecol. Evol. vol. 12, p. 188–193, 1997.

[19] CAMAZINE S. DENEUBOURG J.-L. FRANKS N. R. SNEYD J. THERAULAZ G. BONABEAU E. Self-Organization in Biological Systems, Princeton University Press, Princeton, New Jersey, USA, 2001.

[20] DETRAIN C. DENEUBOURG J. « Self-organized structures in a superorganism : do ants "behave" like molecules ? », Physics of Life Reviews, vol. 3, p. 162–187, 2006.

[21] DENEUBOURG J. ARON S. GOSS S. PASTEELS J. « The self-organizing exploratory pattern of the Argentine ant », J. Insect Behav. vol. 3, p. 159–168, 1990.

[22] GOSS S. ARON S. DENEUBOURG J. PASTEELS J. « Self-organized shortcuts in the Argentine ant », Naturwissenschaften, vol. 76, p. 579–581, 1989.

[23] DORIGO M. ST¨UTZLE T. Ant Colony Optimization, MIT Press, Cambridge, MA,

2004.

[24] DUSSUTOUR A. FOURCASSI´E V. HELBING D. DENEUBOURG J. « Optimal traffic organization in ants under crowded conditions », Nature, vol. 428, p. 70–73, 2004.

[25] DUSSUTOUR A. DENEUBOURG J. FOURCASSI´E V. « Temporal organization of bidirectional traffic in the ant Lasius niger (L.) », The Journal of Experimental Biology, vol. 208, p. 2903–2912, 2005.

[26] MARTINS B. V. C. BRUNETTO G. SATO F. COLUCI V. R. GALV˜AO D. « Designing conducting polymers using bioinspired ant algorithms », Chemical Physics Letters, vol. 453, n°4-6, p. 290–295, 2008.

[27] FRANKS N. HARDCASTLE K. A. COLLINS S. SMITH F. D. SULLIVAN K. M. E. ROBINSONE. J. H. SENDOVA FRANKS A. « Can ants choose a far-and-away better nets over an in the way poor one ? », Animal Behaviour, vol. 75, n°2, p. 323–334, 2008.

[28] ALTSHULER E. RAMOS O. N´U˜NEZ Y. FERN´ANDEZ J. BATISTA-LEYVA A. NODA C. « Symmetry Breaking in Escaping Ants », The American Naturalist, vol. 166, n°6, p. 643–649, 2005.

[29] THERAULAZ G. BONABEAU E. NICOLIS S. C. SOL´E R. V. FOURCASSI´E V. BLANCO S. FOURNIER R. JOLY J. FERNANDEZ P. GRIMAL A. DALLE P.

DENEUBOURG J. « Spatial patterns in ant colonies », Proc. Natl. Acad. Sci. USA 99, p. 9645–9649, 2002.

[30] DENEUBOURG J. GOSS S. FRANKS N. SENDOVA FRANKS A. DETRAIN C. CHR´ETIEN L. « The dynamics of collective sorting : robot-like ants and ant-like robots », MEYER J. A.WILSON E. O. Eds. Simulations of animal behavior : from animals to animals, Cambridge

University Press. Cambridge, Mass. p. 356–365, 1991.

[31] HAMDI A. ANTOINE V.MONMARCH´E N. ALIMI A. SLIMANEM. « Fourmis artificielles

et classification automatique », MONMARCH´E N. GUINAND F. SIARRY P. Eds. Fourmis Artificielles,

nouvelles directions pour une intelligence collective, vol. 2 de Traité IC2, Chapitre

1, Hermès, Paris, 2009.

[32] ROBINSON E. J. H. HOLCOMBE M. RATNIEKS L. W. F. « The organization of soil disposal by ants », Animal Behaviour, vol. 75, p. 1389–1399, 2007.

[33] DEPICK`ERE S. RAMIREZ AVILA G.-M. FRESNEAU D. DENEUBOURG J. « Polymorphism : a weak influence on worker aggregation level in ants », Ecological Entomology, vol. 33, p. 225–231, 2008.

[34] ANDERSON C. THERAULAZ G. DENEUBOURG J. « Self-assemblages in insect societies», Insectes Sociaux, vol. 49, p. 99–110, 2002.

[35] RIVAULT C. CLOAREC A. SRENG L. « Cuticular extracts inducing aggregation in the German cockroach, Blatella germanica », Animal Behaviour, vol. 55, p. 177–184, 1998.

[36] DEPICK`ERE S. FRESNEAU D. DENEUBOURG J. « A basis for spatial and social patterns in ant species : dynamics and mechanisms of aggregation », Journal of Insect Behavior, vol. 17, p. 81–97, 2004.

[37] DEPICK`ERE S. FRESNEAU D. DETRAIN C. DENEUBOURG J. « Marking as a decision factor in the choice of new nesting site in Lasius niger », Insectes Sociaux, vol. 51, p. 243–246, 2004.

[38] HALLOY J. SEMPO G. CAPRARI G. RIVAULT C. ASADPOURM. TˆACHE F. SA¨ID I. DURIER V. CANONGE S. AM´E J. M. DETRAIN C. CORRELL N. MARTINOLLI A. MONDADA F. SIEGWART R. DENEUBOURG J. « Social integration of robots into groups of cockroaches to control selforganized choices », Science, vol. 318, p. 1155–1158, 2007.

Annex

Procédure de création et initialisation d'environnement :

```
to creer-initialiser-environnement

  __clear-all-and-reset-ticks
  set-default-shape nid "nid"
  set-default-shape fourmi "ant"
  set-default-shape nourriture "apple"

  creer-initialiser-terre-obstacles
  creer-initialiser-nourritures
  creer-initialiser-nid
end
```

Procédure de création et initialisation de terre et obstacles :

```
to creer-initialiser-terre-obstacles
  ask patches
  [
    set pcolor white
    set obstacle? false
  ]
  let var1 0
  while [var1 < nb_obstacles]
  [
    create-obstacle 1
    [
      setxy random-x random-y
      set heading (random 4 * 90)
      set taille random 15 + 20
```

```
    let var2 0
    while [var2 < taille]
    [
      ask patch-here
      [
        set obstacle? true
        ask neighbors
        [
          set obstacle? true
        ]
      ]
      verifier-limites
      fd 1
      set var2 var2 + 1
    ]
    die
  ]
  set var1 var1 + 1
 ]
 ask patches
 [
  if (obstacle?)
  [set pcolor red]
 ]
end
```

Procédure de création et initialisation de nourriture :

```
to creer-initialiser-nourritures

 create-nourriture nb_source_nourriture
 [
  setxy random-x random-y
  while[[obstacle?] of patch-here]
  [
    setxy random-x random-y
  ]
  set quantite_nourriture random 40 + 60
```

```
    set size sqrt (quantite_nourriture / 10)
  ]
end
```

Procédure de création et initialisation de nid :

```
to creer-initialiser-nid
  create-nid 1
  [
    set color brown
    set cordX random-x
    set cordY random-y
    setxy cordX cordY
    while[[obstacle?] of patch-here]
    [
      set cordX random-x
      set cordY random-y
      setxy cordX cordY
    ]
    set quantite_nourriture 0
    set size 10
  ]
end
```

Procédure de création et initialisation des fourmis :

```
to creer-initialiser-fourmis [x y]

  create-fourmi nb_fourmis
    [
      setxy x y
      set color black
      set size 2
      set heading random 360
      set a_nourriture? false
      set face-obstacle? false
    ]
end
```

Procédure qui mettre à jour l'état de phéromone sur la terre :

```
to MAJ-terre

  diffuse pheromone pheromone_diffusion
  ask patches
  [
    if pheromone > 9
    [
      set pheromone 9
    ]
    if pheromone < .2
    [
      set pheromone 0
    ]
    set pcolor 129.9 - pheromone
    if obstacle?
    [
      set pcolor red
      set pheromone 0
    ]
  ]
end
```

```
to MAJ-nourritures

  ask nourriture
  [
    if(quantite_nourriture < 5)
    [
      die
    ]
    set size sqrt (quantite_nourriture / 10)
  ]
end
```

Procédure de pose de nourriture dans le nid de fourmi :

```
to transporter-nourriture

 ifelse(any? nid-here)
    [
      ask one-of nid-here
     [
       set quantite_nourriture quantite_nourriture + 5
     ]
     set a_nourriture? false
     rt 180
    ]
    [
     ifelse face-obstacle?
     [
       eviter-obstacle
     ]
     [
       set heading towards-nowrap one-of nid
     ]
     ifelse(any? nid-on neighbors)
     [
       fd 1
     ]
     [
       if(not face-obstacle?)
       [
         detecter-suivre-pheromones
       ]
       if ([obstacle?] of patch-ahead 1)
       [
         detecter-eviter-obstacle
       ]

       verifier-limites
       poser-pheromone
```

```
        fd 1
      ]
    ]
end
```

Procédure de pose de nourriture dans le nid de fourmi :

```
to transporter-nourriture

 ifelse(any? nid-here)
    [
      ask one-of nid-here
      [
        set quantite_nourriture quantite_nourriture + 5
      ]
      set a_nourriture? false
      rt 180
    ]
    [
      ifelse face-obstacle?
      [
        eviter-obstacle
      ]
      [
        set heading towards-nowrap one-of nid
      ]
      ifelse(any? nid-on neighbors)
      [
        fd 1
      ]
      [
        if(not face-obstacle?)
        [
          detecter-suivre-pheromones
        ]

        if ([obstacle?] of patch-ahead 1)
        [
```

```
          detecter-eviter-obstacle
        ]
        verifier-limites
        poser-pheromone

        fd 1
      ]
    ]
end
```

Procédure de détection et suivis de phéromone de fourmi :

```
to detecter-suivre-pheromones

  let chem [pheromone] of patch-ahead 1
  let ret 0
  let turn -90
  while [turn <= 90]
  [
    if [pheromone] of patch-right-and-ahead turn 2 > chem
    [
      set chem [pheromone] of patch-right-and-ahead turn 2
      set ret turn
    ]
    set turn turn + 30
  ]
  set heading heading + ret
end
```

Procédure de détection et évite des obstacles pour les fourmis :

```
to detecter-eviter-obstacle

  let turn 0
  while[[obstacle?] of patch-ahead 1]
  [
    ifelse(not [obstacle?] of patch-right-and-ahead turn 1)
    [
```

```
    set heading heading + turn
    set face-obstacle? true
   ]

   [
    if(not [obstacle?] of patch-left-and-ahead turn 1)
    [
     set heading heading - turn
     set face-obstacle? true
    ]
   ]
   set turn turn + 15
  ]
end
```

```
to eviter-obstacle

 ifelse([obstacle?] of patch-right-and-ahead 90 2)
       [
        rt 45
        set face-obstacle? false
       ]
       [
        lt 45
        set face-obstacle? false
       ]
end
```

Procédure de dépôt de nourriture :

```
to deposer-nourriture

 ask one-of nourriture-here [set quantite_nourriture quantite_nourriture - 5]
      set a_nourriture? true
end
```