وزارة التعليم العالي و البحث العلمي

BADJI MOKHTAR-ANNABA UNIVERSITY

UNIVERSITE BADJI MOKHTAR-ANNABA

جامعة باجي مختار– عنابـــــة

Année : 2019

Faculté des Sciences de L'Ingéniorat

Département d'Informatique

THESE

Présentée en vue de l'obtention
du diplôme de Doctorat en Sciences

Démarche de développement des systèmes logiciels biomorphiques

Option

Intelligence Artificielle

Par

Mr ZEGHIDA Djamel

Jury:

Président :	Mr	GHANEMI	Salim	**Pr**	Université Badji Mokhtar-Annaba
Directeur :	Mr	MESLATI	Djamel	**Pr**	Université Badji Mokhtar-Annaba
Examinateurs :	Mr	AMIRAT	Abdelkrim	**Pr**	Université Med Cherif Mesaadia-Souk Ahras
	Mr	MOKHATI	Farid	**Pr**	Université Larbi Ben M'Hidi-Oum El Bouaghi

REMERCIEMENTS

Je remercie vivement mon directeur de thèse ***Pr Djamel Meslati*** d'avoir accepté de m'encadrer et m'avoir donné l'occasion de travailler sur un thème qui me passionnait tant. De m'avoir donné l'opportunité de découvrir ses vertus humaines et scientifiques, avec son savoir, savoir-faire et savoir être. Son suivi, ses conseils constructifs, et sa confiance, sans réserve, m'ont poussé à toujours positiver et à achever ce travail.

Je tiens à exprimer ma profonde gratitude au Dr *Fabien Michel* d'avoir accepté de m'accueillir au sein du Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier (LIRMM), ses critiques constructives ont été très bénéfiques.

Je souhaite exprimer toute ma reconnaissance aux membres de jury:
Je remercie Pr *Salim Ghanemi*, d'avoir accepté de présider ce jury.
Je remercie Pr *Abdelkrim Amirat* et Pr *Farid Mokhati* pour l'honneur qu'ils m'ont fait en acceptant d'évaluer mon travail.

Je remercie tous les membres du Laboratoire d'Ingénierie des Systèmes Complexes d'Annaba (LISCO) et particulièrement Pr *Labiba Souici-Meslati*, Pr *Fadila Atil* et Dr Nora Bounour pour leur soutien.

Enfin, je tiens à remercier les membres de ma famille qui m'ont couvert par leur soutien et leur encouragements .

DEDICACES

A ma famille.

RESUME

De nos jours, les approches bio-inspirées sont largement utilisées. Certaines d'entre elles sont devenues des paradigmes dans plusieurs domaines, tels que les colonies de fourmis (Ant Colony Optimization : ACO) et les Algorithmes Génétiques (Genetic Algorithm : GA). Malgré les défis inhérents à la survie, dans le monde naturel, les organismes biologiques évoluent, s'auto-organisent et s'auto-réparent avec seulement des connaissances locales et sans aucun contrôle centralisé. L'analogie entre les systèmes biologiques et les systèmes multi-agents "SMA" (Multi-Agent System) est plus qu'évidente. En fait, chaque entité dans les systèmes réels et naturels est facilement identifiée en tant qu'agent. Par conséquent, il sera plus efficace de les modéliser avec des agents. Dans un contexte de simulation, les SMAs ont été utilisés pour imiter les caractéristiques comportementales, fonctionnelles ou structurelles des systèmes biologiques. Dans un contexte général, les systèmes bio-inspirés sont réalisés avec des modèles de conception ad hoc ou avec un modèle multi-agents à objectif unique. Par conséquent, ces travaux souffrent de deux faiblesses : Le premier est l'utilisation de modèles dédiés à des fins restrictives (comme les projets académiques) ; Le second est l'absence d'un modèle de conception.

Dans cette thèse, nous proposons un modèle générique multi-paradigme pour les systèmes biomorphiques. Ce modèle est basé agent et intégrera différents paradigmes bio-inspirés avec le respect de leur concepts. Nous étudions dans quelle mesure il est possible de préserver les principales caractéristiques des systèmes naturels et artificiels. Par conséquent, nous introduisons le principe influence / réaction pour gérer les interactions au sein de ces systèmes multi-agents biomorphiques. Notre modèle a été validé par une étude expérimentale.

Mots Clés : Approche bio-inspiré, Système biomorphique, Système multi-agent, Ingénierie logicielle orientée agent, Principe Influence/Réaction.

ABSTRACT

Nowadays bio-inspired approaches are widely used. Some of them became paradigms in many domains, such as Ant Colony Optimization (ACO) and Genetic Algorithms (GA). Despite the inherent challenges of surviving, in the natural world, biological organisms evolve, self-organize and self-repair with only local knowledge and without any centralized control. The analogy between biological systems and Multi-Agent Systems (MAS) is more than evident. In fact, every entity in real and natural systems is easily identified as an agent. Therefore, it will be more efficient to model them with agents. In a simulation context, MAS has been used to mimic behavioral, functional or structural features of biological systems. In a general context, bio-inspired systems are carried out with ad hoc design models or with a one target feature MAS model. Consequently, these works suffer from two weaknesses: The first is the use of dedicated models for restrictive purposes (such as academic projects); the second one is the lack of a design model.

In this thesis, we propose a generic multi-paradigms model for biomorphic systems. This model is agent-based and will integrate different bio-inspired paradigms with respect of their concepts. We investigate to which extent is it possible to preserve the main characteristics of both natural and artificial systems. Therefore, we introduce the influence/reaction principle to deal with interactions in these biomorphic multi-agent systems. Our model has been validated by an experimental study.

Keywords: Bio-inspired approach, Biomorphic system, Multi-agent system, Agent-oriented software engineering, Influence/Reaction principle.

ملخص

في الوقت الحاضر يتم استخدام المقاربات (المناهج) المستوحاة حيويا (بيولوجيا)على نطاق واسع. أصبح بعضها نماذج في مجالات عدة ، مثل التحسين بمستعمرات النمل (Ant Colony Optimization : ACO) و الخوارزميات الجينية (Genetic Algorithm : GA).

على الرغم من التحديات المتأصلة للبقاء على قيد الحياة ، في العالم الطبيعي ، تتطور الكائنات الحية ، وتنتظم و تصطلح ذاتيا باستخدام معرفة محلية فقط ودون أي سيطرة (تسيير) مركزية. التشابه بين الأنظمة البيولوجية والنظم متعددة الوكلاء (Multi-Agent System : MAS) أكثر من واضح. في الواقع، يتم التعرف بسهولة على كل كيان في النظم الحقيقية والطبيعية كوكيل. لذلك ، سيكون من الأكثر فاعلية تصميمها بوكلاء. في سياق المحاكاة ، تم استخدام النظم متعددة الوكلاء (MAS) لتقليد السمات السلوكية أو الوظيفية أو الهيكلية للأنظمة البيولوجية. في سياق عام، يتم التكفل بالأنظمة المستوحاة بيولوجيا بواسطة نماذج تصميم مخصصة أو بنموذج نظام متعددة الوكلاء بخاصية مستهدفة وحيدة. وبالتالي، فإن هذه الأعمال تعاني من ضعفين: الأول هو استخدام نماذج مخصصة مقيدة الأهداف (مثل المشاريع الأكاديمية); والثاني هو عدم وجود أنموذج تصميم.

في هذه الرسالة، نهدف إلى اقتراح أنموذج تصميم متعدد النماذج للأنظمة المستوحاة بيولوجيًا. هذا الأنموذج قائم على مفهوم الوكيل وسيقوم بدمج عدة نماذج مستوحاة بيولوجيًا مع احترام مفاهيمها. نحن نحقق إلى أي مدى يمكن الحفاظ على الخصائص الرئيسية لكل من النظم الطبيعية والاصطناعية. لذلك، نقدم مبدأ تأثير/ تفاعل للتعامل مع تفاعلات مكونات هذه الأنظمة متعددة الوكلاء المستوحاة بيولوجيًا .تم التحقق من فعالية نموذجنا من خلال دراسة تجريبية.

الكلمات الرئيسية:

مقاربة مستوحاة بيولوجيًا، نظام بيولوجي، نظام متعدد الوكلاء، هندسة البرمجيات الموجهة للوكيل، مبدأ تأثير/ تفاعل.

TABLE DES MATIERES

LISTE DES FIGURES

LISTE DES TABLEAUX

CHAPITRE 1

INTRODUCTION GENERALE

CHAPITRE 1

INTRODUCTION GENERALE

1. Contexte de recherche

La tendance à imiter ce qui est naturel était, par le passé, limitée par diverses considérations d'ordre technologique et scientifique, on assiste aujourd'hui à une prolifération des systèmes dits bio-inspirés. Ces derniers puisent dans la nature et dans les organismes vivants des métaphores capables de résoudre des problèmes complexes et apportent de nouvelles dimensions aux systèmes que nous concevons. Cette prolifération s'explique aussi bien par le développement technologique et méthodologique que par les avancées en termes de compréhension des mécanismes naturels et de l'organisation des systèmes biologiques associés.

En informatique, les approches biologiques suscitent un intérêt particulier. Leurs mécanismes et leurs caractéristiques comportementales, fonctionnelles ou structurelles demeurent des domaines d'étude et d'inspiration favorables aux recherches multidisciplinaires.

La plupart des chercheurs conviennent que les systèmes biologiques sont complexes. Dans ces systèmes, la distribution et la décentralisation sont des caractéristiques innées. Malgré les défis inhérents à la survie, dans le monde naturel, les organismes biologiques évoluent, s'auto-organisent et s'auto-réparent avec seulement des connaissances locales et sans aucun contrôle centralisé.

L'analogie entre ces systèmes biologiques et les *Systèmes Multi-Agents* (*SMA*) est plus qu'évidente. En effet, chaque entité dans les systèmes biologiques est facilement identifiable à un agent.

Formellement, ces deux types de systèmes se distinguent clairement par deux niveaux de granularité qui sont:

- Le niveau micro: tenu par le concept *Agent* dans les *SMA* ou par un individu dans le système biologique. Un agent ou un individu sont tous deux: *autonomes*, *réactifs*, *proactifs* et *sociaux*.

- Le niveau macro: se référant au concept *SMA* (un agrégat d'agents en interaction) ou à un sous-système ou à l'ensemble du système biologique tout entier. Ces deux types de systèmes sont par définition des systèmes *ouverts* : *ils peuvent changer, durant l'exécution, en nombre de composants et en structure organisationnelle*. D'autres propriétés peuvent être observées, telles que: diversité et distribution des connaissances, contrôle distribué, calculs et traitements asynchrones, tolérance aux fautes et fiabilité, communication locale asynchrone et fonctionnalités émergentes.

Cette analogie suggère qu'il serait plus efficace, dans le monde artificiel, d'opter pour une modélisation agent pour ces systèmes naturels.

2. Problématique

En informatique comme dans d'autres disciplines, nous continuons à chercher de nouveaux cas d'application des *métaphores* biologiques existantes ou dégager de nouvelles pour produire des systèmes bio-inspirés.

De nos jours, les systèmes bio-inspirés ont connu un grand succès. Certaines de leurs approches correspondantes sont devenues des paradigmes dans le domaine, tels que les algorithmes à fourmis (Ant Colony Optimization : *ACO*) et les algorithmes génétiques (Genetic Algorithms : *GA*).

Par ailleurs, à côté des réussites, parfois spectaculaires, des systèmes biomorphiques développés, de nouveaux défis surgissent sur différents plans. Premièrement, on peut constater que la prolifération des approches bio-inspirées se fait en absence d'un cadre général qui permettrait de canaliser les efforts vers une direction ou une autre, de comparer les approches, de les évaluer, de mettre en évidence leurs points faibles, de les étudier, et finalement de guider les développeurs pour le choix d'une approche parmi les différentes solutions existantes lors du développement d'un système complexe.

Deuxièmement, l'absence d'un cadre général implique l'absence d'une démarche de développement qui s'accommode aux diverses approches bio-inspirées afin de tirer profit de ce qui fait la force de chacune.

Dans un contexte général, nous soulignons que les systèmes bio-inspirés existants sont réalisés soit avec des modèles de conception Ad hoc ou en utilisant des modèles, agent ou non, dédiés à des fins restrictives (les projets académiques basés agent comme le cas d'*Adelfe*).

Dans un contexte de simulation, les *SMAs* ont été utilisés pour imiter les caractéristiques comportementales, fonctionnelles ou structurelles des systèmes biologiques.

Par conséquent, la nécessité d'une modélisation bio-inspirée associée et spécifique devient de plus en plus urgente. Une telle représentation abstraite unifiée aidera au moins à surmonter le manque de réutilisation dans ce domaine.

3. Motivations

Les systèmes multi-agents (*SMA*) profitent de l'effort d'une large communauté scientifique misant sur le fait que leur approche s'accommode à différents niveaux d'abstraction. En effet, d'agents cognitifs complexes aux agents réactifs très simples, il est possible de modéliser des réalités très variées.

Beaucoup d'arguments ont été donnés, en faveur de l'utilisation des approches orientées-agent pour le développement de systèmes complexes (Jennings 01). Dans l'optique de prendre en charge cette complexité, l'ingénierie orientée-agent fournit les structures, les techniques et les outils fondamentaux, se référant à la décomposition, l'abstraction et l'organisation (Jennings 01).

Il s'ajoute à la complexité de ces systèmes le problème de la modélisation de l'action simultanée de leurs composants. Le principe Influence/Réaction pour la modélisation des actions simultanées est une solution qui a fait ses preuves dans le domaine de la simulation agent (Michel 07) et s'apprête bien à la modélisation de l'action dans ces systèmes bio-inspirés.

En effet, en exploitant l'analogie évidente entre les systèmes biologiques et multi-agents et en soulignant le fait que les concepts agent/multi-agent sont un dénominateur commun pour les paradigmes bio-inspirés; il est tout à fait naturel de modéliser ces systèmes en utilisant des agents autonomes.

En plus de cette analogie, une dualité *Micro*/*Macro* est observée dans tous les concepts relatifs à ces deux types de systèmes :

- ✓ Dans la définition des SMAs, dualité *Micro*/*Macro* portée par l'*agent*/*SMA* (ou un sous-système de ce dernier) ;
- ✓ Dans la définition des systèmes biologiques, dualité *Micro*/*Macro* portée par l'*individu*/*système biologique* (ou un sous-système de ce dernier) ;
- ✓ Dans les approches biologiques, dualité *Micro*/*Macro* portée par l'*individu*/*système biologique* (ou un sous-système de ce dernier) dans la nature et qui peuvent être portées par l'*agent*/*Agrégation d'agent* (*SMA* ou un sous-système de ce dernier) dans le monde artificiel ;
- ✓ Dans les caractérisations d'une approche biomorphique proposées par *Looding*, qui peuvent être identifiées de nature *Micro* (donc portées par un individu) ou *Macro* (et donc portées par un agrégat d'individus);
- ✓ Dans les caractérisations d'une approche multi-agent biomorphique proposées par *Parunak*. Certaines de ces caractéristiques se réfèrent au niveau *Micro*, qui est la composante individuelle ou atomique (l'*agent*) et d'autres à leur *agrégat* (*multi-agent*) donc à un niveau *Macro* ;
- ✓ Dans le Principe Influence/Réaction, la dualité *Micro*/*Macro* se manifeste à un niveau *Micro* dans la phase *Influence* (par le "cumul" des interactions individuelles) et à niveau *Macro* dans la phase Réaction (par la prise en charge des interactions individuelles et leur rendu niveau *agrégat* : système ou sous-système).

4. Objectifs

Les enjeux d'une vision multi-paradigmes d'un système, résident dans la démarche de développement de ce système. La démarche inclut principalement des processus qui peuvent être manuels ou automatisés et des artefacts de modélisation (ou simplement modèles). La démarche de développement des systèmes biomorphiques a comme objectif le développement d'un système moyennant divers paradigmes bio-inspirés.

Dans un contexte de développement multi-paradigmes, il se pose naturellement l'idée de faire appel à un formalisme unificateur pour faire face à la diversité des concepts spécifiques aux approches bio-inspirées considérées. Plutôt que de proposer une approche qui soit la somme de divers concepts, ou chercher à fusionner des concepts

proches, notre vision d'un formalisme unificateur est d'envelopper ces concepts divers par un concept de base et d'opérer par la suite, à des raffinements successifs qui peuvent être menés dans les contextes spécifiques à chaque approche bio-inspirée.

Notre contribution principale se résume en la proposition d'un modèle multi-paradigmes pour les systèmes bio-inspirés appelé : *BIO-IR-M* (Zeghida 18a). Ce modèle générique basé agent est capable d'intégrer différents paradigmes et de préserver les principales caractéristiques des systèmes biologiques et artificiels. Nous introduisons le principe Influence/Réaction pour la prise en charge des interactions dans ces systèmes multi-agents biomorphiques.

L'utilisation combinée des systèmes biologiques et des SMA aura des retombés dans le domaine du génie logiciel (*GL*) comme dans l'intelligence artificielle (*IA*). En *GL*, cette utilisation se manifeste par des méthodologies et des modèles de conception. En *IA*, elle se manifeste par des outils (algorithme ou plateforme) et des modèles de calcul (solveur ou optimiseur). Il est à noter qu'une bonne contribution dans un domaine aura, fort possible, des conséquences positives sur l'autre.

A travers l'étude expérimentale que nous avons menée (Zeghida 18b), notre modèle utilisé avec les systèmes à fourmis pour le problème du voyageur de commerce a montré une amélioration des mécanismes de diffusion de l'information ce qui a augmenté la réactivité des agents et du système. Ceci constitue une nouvelle utilisation du principe Influence/Réaction, en plus des objectifs initiaux pour lesquels il a été défini pour.

5. Contenu de la thèse

Outre l'introduction générale, cette thèse est structurée en six autres Chapitres:

Chapitre 2 : Il est consacré à la présentation des systèmes biologiques : leur spécificité et caractéristiques, leur analogie avec les systèmes artificiels et les chalenges de leur conception.

Chapitre 3 : Nous nous intéressons dans ce Chapitre aux systèmes multi-agents ainsi qu'aux méthodologies et plateformes orientées agents. Avec une présentation du Principe Influence/Réaction et ses utilisations dans le domaine agent.

Chapitre 4 : Il sera consacré à une synthèse et une réflexion de l'état de l'art de notre domaine de recherche mettant en évidence les prérequis et les conséquences de la confrontation des concepts biologiques et agent.

Chapitre 5 : Il est consacré à la description de notre Modèle Agent Influence/Réaction Multi-paradigmes Biologiques. Ensuite, il présente quelques cas d'application. Enfin, il positionne notre contribution avec des travaux similaires dans le domaine ; tout en soulignant l'apport de notre proposition dans le domaine de l'ingénierie de logiciels complexes, en général, et de l'ingénierie logicielle orientée agents particulièrement.

Chapitre 6 : Nous nous concentrons dans ce Chapitre sur l'étude expérimentale de notre modèle. L'accent sera mis, évidemment dans cette étude, sur les retombés de notre proposition, dans le domaine de l'intelligence artificielle distribuée (*IAD*).

Chapitre 7 : Il constitue la conclusion générale qui donnera une récapitulation des contributions et de leurs perspectives relatives couvrant les aspects méthodologique ou algorithmique se référant aux domaines *GL* et *IAD*.

CHAPITRE 2

LES SYSTEMES BIOMORPHIQUES

CHAPITRE 2

LES SYSTEMES BIOMORPHIQUES

1. Introduction

De nos jours, les approches bio-inspirées sont largement utilisées dans divers domaines. On assiste aujourd'hui à une prolifération des systèmes dits bio-inspirés. Ces derniers puisent dans la nature et les organismes vivants des métaphores capables de résoudre des problèmes complexes et apportent de nouvelles dimensions aux systèmes que nous concevons. Cette prolifération s'explique aussi bien par le développement technologique et méthodologique que par les avancées en termes de compréhension des mécanismes naturels et de l'organisation des systèmes biologiques associés. Ainsi, de multiples métaphores ont vu le jour.

En informatique, les approches biologiques suscitent un intérêt particulier. Leurs mécanismes et leurs caractéristiques comportementales, fonctionnelles ou structurelles restent des domaines d'étude et une source d'inspiration favorables pour des domaines de recherche multidisciplinaires.

Afin de décrire l'approche logicielle bio-inspirée, le terme *biomorphique* a été inventé par le zoologiste britannique *Desmond Morris* et popularisé par le darwinien *Richard Dawkins* dans son livre "*L'horloger aveugle*" ("*The Blind Watchmaker*" 1996) (Lodding 04).

Un système biomorphique est simplement conçu à base de concepts algorithmiques inspirés des systèmes et des processus biologiques. Cependant, il n'y a pas de caractéristiques structurelles, faciles à identifier, pour certifier qu'une architecture donnée est bio-inspirée ou pas (Lodding 04).

2. Historique des paradigmes bio-inspirés

On entend par *métaphore* biologique une analogie qu'on cherche à déterminer entre le monde biologique et le monde artificiel, de façon à pouvoir proposer des outils qui imitent certains aspects du premier, tout en ignorant d'autres.

Fondamentalement, les métaphores ne cherchent pas à reproduire ce qui est biologique, mais plutôt à l'interpréter en fonction de ce qu'il est *possible* et *raisonnable* de *réaliser*. De ce fait, on peut déduire que les métaphores biologiques sont *évolutives* et dépendent de notre compréhension de la réalité et de notre aptitude à en extraire des éléments pratiques et bénéfiques. Il s'agit d'enrichir le domaine informatique en général, et les *SMAs*, en particulier, à l'aide de principes et de mécanismes provenant de ces systèmes biologiques (Ballet 00).

L'inspiration de l'évolution naturelle des espèces en générale et des mécanismes génétiques particulièrement, se concrétise par la naissance de la *programmation évolutionnaire* par *David B. Fogel* et des *algorithmes génétiques* par *J. Holland* (1962). En 1965 et avec les mêmes convictions, on parle de *stratégies évolutionnaires* dues à *I. Rechenberg* et *H.P. Schwefel*, et le carré se boucle avec la *programmation génétique* de *J. Koza* en 1989 (Talbi 00, Dréo 06).

Norman H. Packard, *Alan S. Perelson* (1986) et *H. Bersini*, *FJ. Varela* (1990) exploitent les mécanismes opératoires des cellules du système immunitaire humain pour aboutir aux *systèmes immunitaires artificiels*. Le système nerveux et précisément la structure et le fonctionnement du cerveau donnent naissance aux réseaux de neurones suite aux travaux de *F. Rosenblatt* (1958, 1962) et de *John Hopfield* (1982). Avant cela, en 1965, *Lotfi A. Zadeh* définissait les *systèmes flous* (ensembles et logique flous) en mimant le raisonnement humain qui ne se limite pas aux deux états, vrai/faux, de la *logique de Boole* (Talbi 00, Dréo 06).

L'étude du comportement social de colonies et d'essaims vivants a fait émerger une nouvelle forme d'intelligence dite d'essaims (*Swarm Intelligence*), marqué par les méta-heuristiques d'optimisation par les colonies de fourmis (Ant Colony Optimization) de *Marco Dorigo* (1992) et d'optimisation par l'essaim de particules (Particle Swarm Optimization) de *J. Kennedy* et *RC. Everhart* (1995) (Dréo 06, Talbi 00). Un récapitulatif est donné dans le Tableau 2.1.

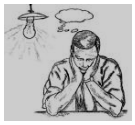
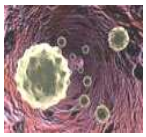
paradigmes	***Origine***	***Métaphore***	***Caractéristiques***
Réseaux de neurones Artificiels Artificial Neural Network "ANN" *F. Rosenblatt,1958,1962* *John Hopfield*, 1982	 Système Nerveux	Structure et fonctionnement du cerveau	Imiter la structure et le fonctionnement du cerveau humain Couches d'entrée, couches de sortie et couches intermédiaires
Algorithme évolutionnaire Evolutionary Computing "EC" EX : Algorithme génétique Genetic Algorithm "GA" *J. Holland*, 1962	 Evolution Naturelle (génétique)	Mécanismes génétiques	Evolution d'une solution : Diversifier par Combinaison et Mutation Intensifier par Sélection
Systèmes flous Fuzzy Systems "FS" *Lotfi A.Zadeh*, 1965	 Cerveau	Raisonnement humain	Imiter le raisonnement humain ne se limitant pas aux deux états de l'algèbre de Boole : vrai /faux.
Système Immunitaire Artificiel Artificial Immune System "AIS" *Farmer, Packard, Perelson 1986.* *Bersini, Varela 1990.*	 Systèmes Immunitaires Humain	Mécanismes opératoires des cellules du système immunitaire	Exploiter les caractéristiques de mémorisation et d'apprentissage du système immunitaire et ses mécanismes d'émergence et d'auto-organisation
Swarm Intelligence "SI" : Ant Colony Optimization "ACO" *M. Dorigo, 1992*	 Etude des colonies sociaux	Comportement de la colonie de fourmis	intelligence émergente du comportement social d'un grand nombre d'individus morphologiquement différents performant une communication stigmergique locale par le dépôt de phéromone accumulé et évaporé.
Swarm Intelligence "SI" : Particle Swarm Optimization "PSO" *J. Kennedy , RC. Everhart, 1995*	 Etude des essaims sociaux	Comportement d'essaim d'oiseau en vol	intelligence émergente du comportement social d'un petit nombre d'individus morphologiquement identique.

Tableau 2.1 Taxonomie de quelques paradigmes bio-inspirés

La plupart des chercheurs conviennent que les systèmes biologiques sont complexes où la distribution et la décentralisation sont des caractéristiques innées. Malgré les défis inhérents à la survie, dans le monde naturel, les organismes biologiques évoluent, s'auto-organisent et s'auto-réparent avec seulement des connaissances locales et sans aucun contrôle centralisé.

Concernant l'aspect applicatif de ces approches biologiques, de nombreuses expériences réussies maintiennent une motivation continue pour le développement de systèmes bio-inspirés (Nagpal 08, Dorigo 06, Chang 06, Blum 05, Dréo 06, Beurier 07). Nous citons par exemple, les approches immunitaires artificielles qui sont devenues incontournables face au défi que pose la protection des systèmes informatiques (Tarakanov 01, Skormin 01). Ou encore avec le mécanisme d'évolution des espèces qui a pu apporter des solutions très satisfaisantes aux problèmes d'optimisation complexes (Fogel 88). Certaines de ces approches bio-inspirées sont devenues des paradigmes dans de nombreux domaines comme heuristique pour l'optimisation difficile (Dréo 06), mettant en évidence les algorithmes génétiques, le recuit simulé (Simulated Annealing : *SA*) et la méta-heuristique *ACO* de *Dorigo* (Dorigo 99).

Nous pouvons trouver des cas d'utilisation, par exemple en optimisation avec une approche évolutionnaire (Fogel 88) ou avec une intelligence d'essaim (Swarm Intelligence : *SI*) en utilisant les colonies de fourmis (Ant Colony : *AC*) (Colorni 91). D'autres exemples sont présentés pour l'utilisation de réseaux de neurones artificielles (Artificial Neural Network : *ANN*) dans les systèmes de contrôle et de décision (Lin 91) ou pour la détection de classe d'objet (spécifiquement la détection de visage) (Rowley 98). Alors que (Tarakanov 01, Tsang 05) présentent respectivement l'utilisation des systèmes immunitaires artificiels (Artificial Immune System : *AIS*) et AC dans le domaine de la sécurité. Alors que (Xiang 08) présente l'utilisation de l'intelligence colonies de fourmis (*AC*) avec une modélisation agent pour la planification et (Bari 09) illustre le routage avec les algorithmes génétiques (Genetic Algorithm : *GA*).

Certaines applications récentes peuvent être trouvées, comme l'extension parallèle de l'algorithme à fourmis dans (Hong 14) et l'algorithme d'optimisation par essaim de particule (Particle Swarm Optimization : *PSO*) dans (Ma 14). On peut citer deux autres applications de la méta-heuristique *ACO* pour la découverte de ressources dans une grille utilisant la technologie agent (Olaifa 15) et pour les réseaux domotiques (Wang 15). Un algorithme à fourmi (Ant Algorithm : *AA*) comme agent mobile a été utilisé

dans un système de cyberdéfense basé fourmis (Fink 14), alors qu'un algorithme hybride fourmis-abeilles (Ant-Bee Algorithm : *A-B A*) a été utilisé pour la couverture multi-robots dans (Broecker 15). Pour l'optimisation multi-objectif, on trouve l'utilisation d'un algorithme de chauve-souris (Bat Algorithm : *BA*) dans (Amine 15) et un algorithme évolutionnaire dans (Perez-Cara 16). En vision, les réseaux de neurones artificiels (Artificial Neural Networks : *ANN*) sont utilisés pour la reconnaissance de place (Cuperlier 16).

3. Caractéristiques d'un système naturel

À l'image d'un écosystème ou un biotope, un système naturel est un *système complexe* et *distribué*, cela revient à sa composition d'une très grande diversité d'entités autonomes, dotées d'une multiplicité de mécanismes (dominants/dominés) qui font apparaitre des *comportements variés* et où règne une *multiplicité d'interactions* et *d'interdépendances* complexes (Figure 2.1).

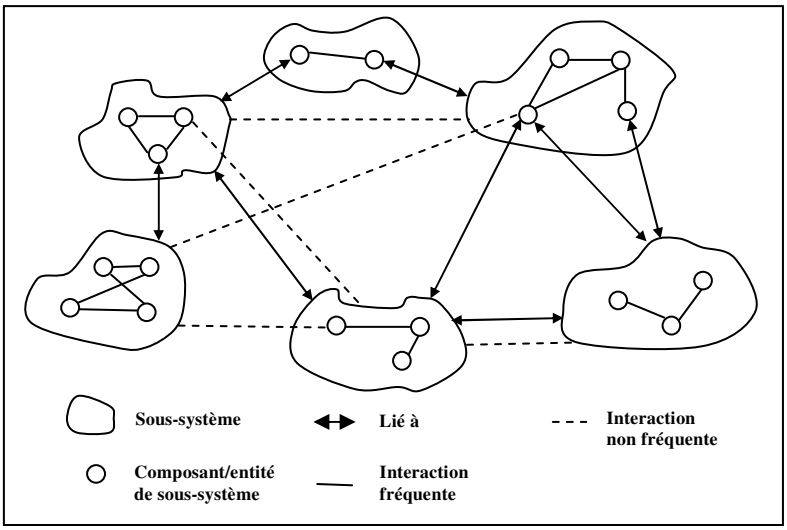


Figure 2.1 Vue canonique d'un système naturel complexe (Jennings 01)

Les biologistes définissent un biotope comme une boite dotée d'un ensemble distinct de conditions environnementales (climatiques et géologiques) qui supporte une communauté écologique composée de plantes et d'animaux. Dans un biotope, l'interdépendance est complexe et la survie des espèces en dépend. Il est important de noter que tout le biotope forme un système cohérent et que diverses espèces y

cohabitent alors qu'ils diffèrent énormément en terme de mécanismes et de comportements.

Par exemple, les fourmis utilisent un mécanisme à base de phéromones qui leur permet de trouver le chemin le plus court entre leur nid et la source de nourriture. La métaphore de ce mécanisme, nommée algorithmes de colonies de fourmis, est utilisée avec succès dans la résolution de certains problèmes d'optimisation (Colorni 91, Dorigo 91a, 91b, 96, 99, 04, 06). Dans le même biotope, les espèces évoluent d'une génération à l'autre selon la théorie Darwinienne de l'évolution. Cette théorie, qui explique la diversification des espèces et leur survie dans des environnements changeants et contraignants (Chang 06), est à l'origine d'une métaphore nommée algorithmes génétiques ou algorithmes évolutionnaires. Cette métaphore est couramment utilisée dans les problèmes d'optimisation.

D'un autre côté, en considérant l'individu d'une espèce de manière isolée, on remarque qu'il se compose de nombreux organes qui coopèrent étroitement pour préserver sa survie. Le système immunitaire concerne tous les organes de l'individu, et les mécanismes qui régissent ce système lui permettent de faire face à des attaques virales ou bactériennes qui le menacent.

Ainsi, en analysant le comportement d'un biotope, de ses espèces et leurs individus on se rend compte de la multitude de mécanismes que la ''nature'' a su développer sur des millions d'années. Il est important de noter que ce développement a préservé divers mécanismes et a rendu d'autres universels. A titre d'exemple, les mécanismes de la génétique sont universels dans le sens où ils régissent le processus de reproduction et de croissance de tous les organismes vivants (espèces végétales et animales). La survie de tous ces mécanismes naturels dénote de l'optimalité qu'a chaque entité pour résoudre un type particulier de problème.

Tous ces composants baignent dans des *environnements changeants et contraignants* imposant des *changements continus d'organisation* (décomposition/regroupement). Il faut noter que la *distribution* et la *complexité* sont des caractéristiques innées de ces systèmes plutôt que des fonctions occasionnelles. Ces systèmes sont des groupes *d'individus auto-organisés*. Ces derniers sont *autonomes*, *simples* et *coopératifs*, mis ensemble via une *communication locale* pour effectuer des opérations complexes de *manière distribuée* et *parallèle*. Où le *comportement* montré par le groupe n'est pas explicitement programmé dans les membres mais *émerge* de leurs interactions. Ces

membres *rejoignent* et *quittent librement* le groupe en perpétuel changement. Tout cela est effectué *sans aucun contrôle central* (Lodding 04). Avec tout ce *chaos* et ces *interactions anarchiques*, l'organisation *continue* à *grandir*, à *vivre*, à *s'auto-adapter* et à *s'auto-réparer*.

Dans le monde artificiel, les concepteurs constatent que cette complexité présente plusieurs régularités (Jennings 01):

1. La complexité prend fréquemment la forme d'hiérarchie. Le système est composé de sous-systèmes interconnectés, qui ont également, une structure hiérarchique jusqu'à des sous-systèmes élémentaires (de bas niveau).

Les relations organisationnelles sont de nature variées et ne sont pas statiques et peuvent souvent varier dans le temps.

2. Le choix des composants primitifs dans le système est arbitraire et il est défini par les buts et les objectifs d'un observateur.

3. Les systèmes hiérarchiques évoluent plus vite qu'un système non hiérarchique de taille semblable.

4. Il est possible de distinguer les interactions (prédictibles lors de la conception, pour la plus part) inter et intra sous-systèmes, les dernières sont plus fréquentes et plus prévisibles. Ces sous-systèmes peuvent être traités comme étant indépendants (ou presque).

4. Analogie avec les systèmes artificiels

D'un point de vu organisationnel et à l'image du biotope, un système informatique, peut être composé de sous-systèmes interdépendants où chacun est régi par une métaphore biologique donnée donc par un paradigme donné.

L'intérêt sous-jacent à cela est de tirer profit des meilleurs paradigmes pour chaque problème posé. C'est donc une synergie des divers paradigmes que nous souhaitons atteindre.

A leur tour, les sous-systèmes peuvent être décomposés et chacun obéira à un paradigme donné. La relation elle-même entre les divers sous-systèmes peut être régie par un paradigme différent de ceux qui régissent les sous-systèmes (Figure 2.2).

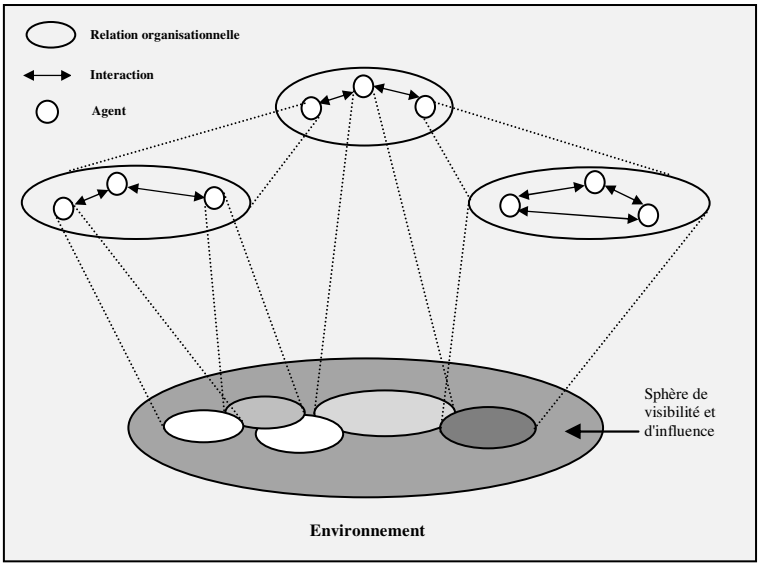


Figure 2.2 Vue canonique d'un Système Artificiel (Jennings 01)

Par analogie avec le biotope où l'objectif est le maintien d'un état d'équilibre entre les individus, les espèces et l'environnement, l'objectif que nous assignons à une approche multi-paradigmes est de fournir un système avec des performances relativement meilleures et une bonne qualité (fiabilité, facilité de développement, maintenabilité, portabilité, etc.).

5. Challenge de la conception des systèmes bio-inspirés

Historiquement, l'évolution de toute approche ou paradigme doit s'accompagner d'une évolution méthodologique pour prendre en charge l'aspect modélisation. Ce qui n'est, malheureusement, pas le cas pour ces différentes approches bio- inspirées.

Dans un contexte de simulation, les *SMAs* ont été utilisés, avec succès, pour imiter des caractéristiques comportementales, fonctionnelles ou structurelles de systèmes biologiques.

Dans un contexte plus général, nous soulignons que les systèmes bio-inspirés existants sont réalisés soit avec des modèles de conception *Ad hoc* ou en utilisant des modèles, agent ou non, dédiés à des fins très restrictives (cas des projets académiques, par exemple).

En plus, comme nous avons déjà noté, au début de ce Chapitre, ils n'existent pas de caractéristiques structurelles, faciles à identifier, pour certifier qu'une architecture donnée est bio-inspirée ou pas (Lodding 04).

Plusieurs critères ont été identifiés dans (Lodding 04) pour caractériser le comportement des systèmes biomorphiques. Ces critères font ressortir qu'un système biomorphique est matérialisé par une multitude d'entités qui collaborent. Les critères sont les suivants :

1- Interaction collective : Le comportement du système résulte de l'interaction collective de plusieurs entités similaires et indépendantes.

2- Emergence : Le comportement du système émerge de l'interaction des entités sans être explicitement décrit dans ces dernières.

3- Action autonome : Les entités agissent d'une manière autonome.

4- Information et interaction locale : Les entités opèrent en fonction des informations et des interactions locales et leur portée spatiale est plutôt locale.

5- Naissance et mort : Les entités naissent et disparaissent librement au fur et à mesure de l'évolution du système (évolution libre du groupe).

6- Adaptation : Les entités ont la capacité de s'adapter aux changements d'objectifs, des connaissances et de conditions.

7- Evolution : L'espèce a la capacité d'évoluer dans le temps.

Par conséquent, une réflexion poussée sur une modélisation bio-inspirée associée et spécifique aux paradigmes devient extrêmement urgente et fortement conseillée.

6. Conclusion

Nous avons décrit dans ce Chapitre une présentation incluant la définition d'approches et paradigmes biologiques, les caractéristiques des systèmes dits bio-inspirés ou biomorphiques. Nous avons, par la suite, mis l'accent sur l'analogie qui existe entre ces systèmes biologiques et les systèmes artificiels. Concluant par la spécificité de la conception des systèmes bio-inspirés et le besoin d'une représentation abstraite unifiée des différents paradigmes biologiques associés, qui aidera, au moins, à surmonter le manque de réutilisation dans ce domaine. Ce qui justifie la présentation, dans le Chapitre suivant, des *SMA* et des méthodologies orientées agent comme sous thème de l'ingénierie logicielle orientée agent (Agent Oriented Software Engineering : *AOSE*).

CHAPITRE 3

LES SYSTEMES MULTI-AGENTS

CHAPITRE 3

LES SYSTEMES MULTI-AGENTS

1. Introduction

Pour faire face à la complexité croissante des problèmes traités en informatique, l'Intelligence Artificielle Distribuée (*IAD*) a été introduite pour surmonter les "limites" de l'Intelligence Artificielle (*IA*) "classique" et depuis le milieu des années 70, l'*IAD* évolua et se diversifia rapidement.

L'*IAD* est un domaine qui rassemble plusieurs disciplines incluant l'*IA*, la sociologie, l'économie, l'organisation et le management des entreprises…. etc (Bourdon 01, Wooldridge 99). L'*IAD* s'accentue sur le problème de fourniture de support de coordination et d'intégration de plusieurs solveurs (humains et automatiques) travaillant sur des problèmes multiples simultanément. Elle considère les concepts tels que l'interaction, l'organisation sociale et la société comme des métaphores et des générateurs de problèmes pour l'*IA*.

La recherche et les problèmes *IAD* relèvent naturellement de deux domaines inter-reliés:

- La résolution coopérative de problème (*RCP*) (Cooperative Problem Solving : *CPS*): Où un problème complexe, particulier, peut être divisé entre plusieurs éléments de traitement ou nœuds, qui coopèrent et interagissent en terme de division et de partage d'expertise pour développer une solution globale. Chaque nœud possède une expertise suffisante de résolution du problème pour formuler une solution partielle au problème global en employant ses propres connaissances. Une coopération entre nœuds sera obligatoire pour développer une solution globale.
- Les Systèmes Multi-Agents (*SMAs*) : Concerné par la coordination du comportement intelligent d'une agrégation d'agents intelligents, caractérisés par leur domaine d'expertise et qui sont situés à des locations distinctes et géographiquement distribuées. L'objectif est la coordination des connaissances, d'expertise et de plans

d'agents. L'interaction est guidée par des stratégies de coopération pour atteindre un objectif commun de haut niveau.

Comme dans les *CPS*, les agents doivent partager des connaissances, mais en plus dans les *SMAs*, ils doivent aussi raisonner sur la façon de coordonner leurs activités vu qu'ils ont différentes perspectives et croyances et il n'y a pas de contrôle global ni de connaissances globalement consistantes. On est ainsi naturellement conduit à chercher à donner plus d'autonomie et d'initiative aux différents modules logiciels. Le concept de système multi-agent propose un cadre de réponse à ces deux enjeux complémentaires : l'autonomie et l'organisation (Demazeau 01).

Jennings (Jennings 98a, 98b), explique que la recherche sur les systèmes composés de plusieurs agents autonomes était classiquement appelée intelligence artificielle distribuée. En tant qu'évolution de l'*IA*, un *SMA* s'adapte pour permettre l'interaction d'un ensemble d'individus appelés "agents".

2. Le concept d'agent

Le concept d'agent est le résultat de quarante années de recherche en *IA* et en robotique. L'idée d'une entité conceptuelle capable de réaliser des tâches au profit d'un utilisateur était déjà bien établie au milieu des années 70. De cette origine, découlent des bases théoriques telles que les concepts de raisonnement, de représentation de connaissances et d'apprentissage.

La technologie agent est originaire de plusieurs domaines qui sont (Caglayan 98) :

• L'intelligence artificielle : avec les systèmes intentionnels, les systèmes de production, la théorie de raisonnement et les réseaux de neurones.

• Le génie logiciel : avec les objets distribués, la commande à distance et le contrôle temps réel.

• Les interfaces homme-machine : avec l'ingénierie cognitive, les expérimentations homme-machine, la modélisation de l'utilisateur, les systèmes intelligents d'enseignement assisté et la vision assistée par ordinateur.

Par ailleurs, sur le plan pratique, l'approche agent est plus général et apporte des outils qui s'intègrent à de multiples applications et à des bases de données avec des extensions

réseaux. Leur développement est motivé par un très grand nombre de champs d'application, on peut, entre autres, citer (Bourdon 01):

• Le commerce électronique ;

• La gestion et le suivi temps-réel des réseaux de télécommunication ;

• La modélisation et l'optimisation de flux de marchandises ou de données ;

• Le traitement de l'information dans des systèmes de type Internet (recherche, filtrage, présentation...) ;

• La gestion du trafic routier et aérien ;

• La planification automatique de réunions ;

• L'optimisation des processus industriels de fabrication ;

• L'analyse des stratégies d'entreprise ;

• Les jeux électroniques ;

• La conception et la réingénierie des composants informationnels dans les organisations ouvertes ;

• L'étude et la simulation de phénomènes complexes dans des organisations humaines ou naturelles, comme la réaction à une situation de crise (tremblement de terre), et l'évolution des rôles ou des normes dans une société.

Toutes ces applications ont en commun les deux propriétés suivantes :

- Une répartition/hétérogénéité intrinsèque : dans la mesure où les données à traiter :
  - existent dans des endroits différents (répartition dans l'espace),
  - apparaissent à des moments différents (répartition dans le temps),
  - se structurent dans des communautés dont les accès et les usages nécessitent de partager au moins une ontologie et au plus un langage et des protocoles d'échange et de communication associés (répartition sémantique),
  - et / ou se structurent dans des communautés dont les accès et les usages nécessitent des capacités de perception, de restitution et de raisonnement particulières (répartition fonctionnelle).
- Une complexité intrinsèque : dans la mesure où la dimension de ces systèmes n'est pas abordable par une seule machine et un seul logiciel. Cette dimension comprend

aussi bien le nombre d'entités en jeu, que les interactions produites ou encore les distances parcourues entre ces acteurs, le volume des informations échangées, la fréquence de leur évolution ou encore leur diversité.

2.1. Définition d'un agent

Il n'existe pas actuellement une définition du concept agent acceptée universellement par les chercheurs concernés (Caglayan 98).

l'une des premières définitions de l'agent a été fournie par *Ferber* (Ferber 99):

Un agent est une entité autonome, réelle ou abstraite, qui est capable d'agir sur elle-même et sur son environnement, qui, dans un univers multi-agent, peut communiquer avec d'autres agents, et dont le comportement est la conséquence de ses observations, de ses connaissances et des interactions avec les autres agents.

M. Wooldridge propose la définition suivante (Wooldridge 95) :

Un agent est un programme informatique qui est situé dans un environnement et qui est doté de comportements autonomes (actions) lui permettant d'atteindre, dans cet environnement, les objectifs qui lui ont été fixé à sa conception.

En outre, *Jennings*, *Sycara* et *Wooldridge* (Jennings 98b) ont proposé la définition suivante pour un agent:

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

Les notions "*situé*", "*autonomie*" et "*flexible*" sont définies comme suit:

• *situé*: l'agent est capable d'agir sur son environnement à partir des entrées sensorielles qu'il reçoit de ce même environnement. Exemples: systèmes de contrôle de processus, systèmes embarqués, etc ;

• *autonome*: l'agent est capable d'agir sans l'intervention d'un tiers (humain ou agent) et contrôle ses propres actions ainsi que son état interne;

• *flexible* : l'agent se caractérise par :

- la *réactivité*, Capable de répondre à temps : l'agent doit être capable de percevoir son environnement et d'élaborer une réponse dans les temps requis ;

- la *pro-activité*, l'agent doit exhiber un comportement proactif et opportuniste tout en étant capable de prendre l'initiative au bon moment ;
- l'aspect *social*, l'agent doit être capable d'interagir avec les autres agents « logiciels et humains » quand la situation l'exige afin d'accomplir ses tâches.

Il ressort de ces définitions les propriétés clés de l'agent comme *l'autonomie*, *l'action*, la *perception* et la *communication*. D'autres propriétés peuvent être attribuées aux agents. On cite en particulier, la *rationalité*, *l'engagement* et *l'intention* (Jarras 02).

2.2. Caractéristiques d'un agent

Commençons par les attributs les plus importants, nous présentons dans ce qui suit, une liste d'attributs que peut posséder un agent. Tous ne sont pas indispensables pour correspondre au profil d'agent, mais ils permettent de caractériser un agent particulier.

1. *Autonomie*

L'agent peut, spontanément, effectuer certaines tâches et prendre des initiatives.

2. *Interactivité*

Les interactions sont à des niveaux différents en raison de la complexité différente des composants avec les quels l'agent interagit (Interaction avec le hardware ou avec des capteurs, actionneurs, ou logiciel).

3. *Intelligence*

L'intelligence est la faculté de raisonnement et d'apprentissage. C'est la capacité de l'agent à accepter les demandes de l'utilisateur et de mener à bien la tâche qui lui est déléguée.

4. *Mobilité*

Certains agents peuvent être fixes, qu'ils résident sur la machine de l'utilisateur, ou sur le serveur. D'autres agents peuvent être mobiles, c'est-à-dire qu'ils se baladent sur le réseau. Ils peuvent se déplacer de machine à machine pendant leur exécution en transportant avec eux les données accumulées. Ces agents peuvent se présenter à des agences qui peuvent leur fournir certains services ou servir de point de rencontre entre différents agents. La mobilité implique de résoudre des problèmes de sécurité, de protection de la vie privée et des données confidentielles et de gestion. Tant qu'une

infrastructure adaptée ne sera pas mise en place, les agents seront surtout fixes. Il faudra faire appel à des agents chargés de la sécurité pour surveiller l'activité des agents qui viennent s'installer sur une machine.

Au part avant, aucun agent n'était vraiment mobile (Smets 98). Mais, il s'agissait d'une "pseudo-mobilité", c'est-à-dire que l'agent utilise le réseau pour rechercher et consulter des données, des informations, mais il ne se déplace pas, le code s'exécute toujours sur la machine de l'utilisateur. Actuellement, les agents sont mobiles et peuvent *immigrer* d'une machine à une autre au sens propre du terme.

5. *Réactivité*

L'agent doit pouvoir faire face aux modifications de l'environnement, que ce soit la modification des objectifs de l'utilisateur ou des ressources disponibles.

6. *Délégation*

L'agent est mandaté par l'utilisateur pour effectuer une tâche. Dans le cas du commerce électronique, un degré de délégation faible impliquerait que l'agent va chercher les opportunités intéressantes et les propose à l'utilisateur. A un degré plus élevé, l'agent effectue lui-même l'achat. Dans ce cas, il y a un risque que l'agent effectue une mauvaise opération, il faut donc que l'utilisateur ait une certaine confiance.

7. *Domaine d'application*

Le domaine d'application est important par rapport au risque couru. S'il s'agit de trier des e-mails, le risque est relativement faible, mais s'il s'agit de contrôler une centrale nucléaire, il faudra réfléchir deux fois avant de confier cette tâche à un agent, car son comportement n'est pas parfaitement prévisible.

8. *Personnalisation*

L'utilisateur détermine la façon dont l'agent interagit. Dans beaucoup de cas, l'agent s'adapte à l'utilisateur.

9. *Prévisibilité*

Il est nécessaire que l'utilisateur sache quels résultats il peut attendre de l'agent.

10. *Rentabilité*

Il faut que l'avantage pour l'utilisateur (gain en temps, information fournie, filtrage des informations,...) soit supérieur au coût (en argent, en temps,...).

11. *Communication*

Une communication entre l'utilisateur et l'agent est établie pour déterminer la tâche de l'agent.

12. Dégradation progressive

Si un problème se pose, par exemple un problème de communication ou une tâche qui ne peut plus être effectuée, l'agent ne doit pas être complètement bloqué et doit pouvoir continuer à exécuter ce qui peut l'être, éventuellement par d'autres moyens.

2.3. Types d'agent

La granularité des agents impliqués dans une application varie selon deux écoles co-existantes : L'école cognitive et l'école réactive. Suivant le type d'agent utilisé on parlera de système cognitif ou de système réactif (Labidi 93).

a. Agents réactifs

L'école réactive est basée sur le principe suivant : dans un système multi-agents, il n'est pas nécessaire que chaque agent soit individuellement intelligent pour parvenir à un comportement global intelligent.

En effet, les agents intervenants dans ces systèmes n'ont pas de connaissances explicites ni de l'environnement, ni des autres agents, ni de leur passé, pas plus que de leurs buts (pas de planification de leurs actions, pas de moyens de mémorisation). Ils possèdent une fonction de perception / action (la loi stimulus / réponse) sur l'environnement qui constitue l'unique protocole de communication avec les autres agents. Il s'agit d'une communication via l'environnement (Figure 3.1).

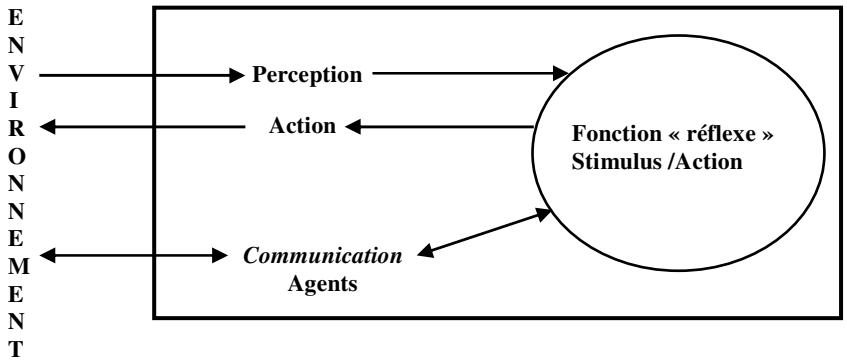


Figure 3.1 Modèle d'agent réactif (Bergia 99)

b. Agents cognitifs

A l'opposé des agents réactifs, les agents cognitifs (ou Intelligents) sont capables, à eux seuls, de réaliser des opérations relativement complexes et généralement coopèrent les uns avec les autres pour atteindre un but commun (Labidi 93, Bergia 99).

Ils ont une représentation explicite de l'environnement, sur les autres agents et sur eux-mêmes décrite dans une base de connaissances sur laquelle ils peuvent raisonner. Ils réagissent en fonction de leurs connaissances, leurs buts et en fonction des échanges avec les autres agents et de la perception de l'environnement en organisant leurs actions suivant une planification. Ils sont dotés de moyens et mécanismes de communication pour gérer les interactions avec les autres agents (coopération, négociation). Ce modèle d'agent est une métaphore du modèle humain et s'appuie plutôt sur les sciences humaines et sociales alors que le modèle de l'agent réactif s'appuie sur les sciences de la vie (Figure 3.2).

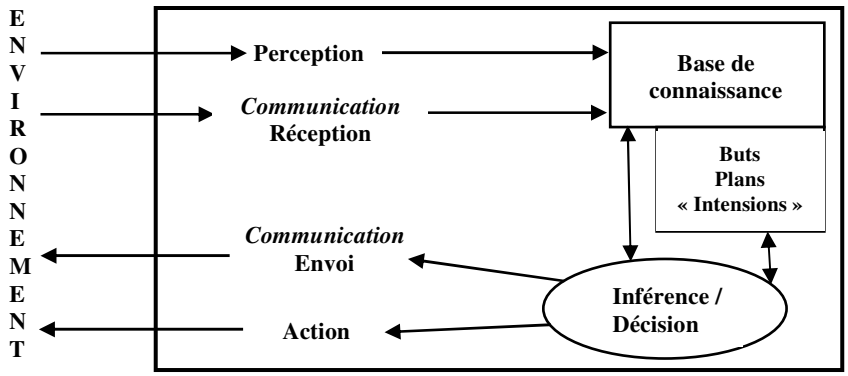


Figure 3.2 Modèle d'agent cognitif (Bergia 99)

Un agent est "intelligent" s'il est capable de réaliser des actions flexibles et autonomes pour atteindre les objectifs qui lui ont été fixé.

c. Agents hybrides

En pratique, un compromis entre complexité et efficacité est souvent recherché, ce qui se traduit par des agents qui sont à la fois réactifs et cognitifs. On parle alors d'agents hybrides (Bergia 01).

2.4. Architecture fonctionnelle d'un agent intelligent

Sur le plan fonctionnel (Caglayan 98), un agent intelligent est vue comme un ensemble de compétences (Figure 3.3) :

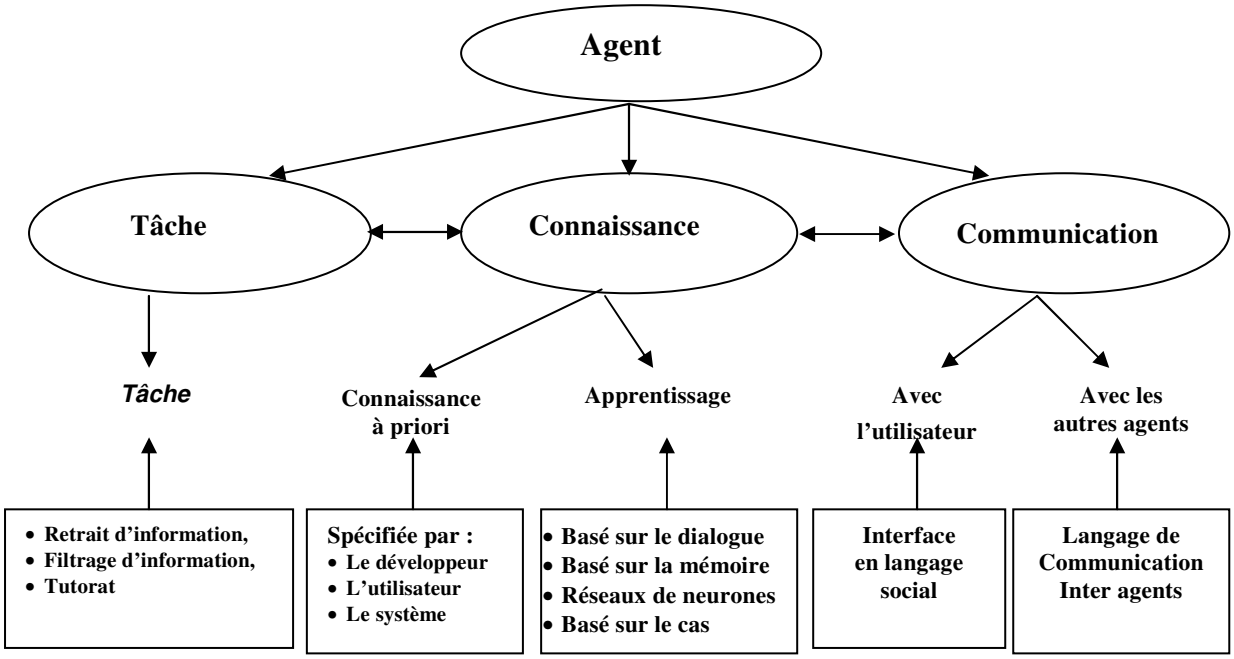


Figure 3.3 Modèle d'agent (Caglayan 98)

- Compétences de niveau tâche :

Correspond à la capacité des agents à accomplir leur propre but. Ces compétences spécifient la fonction d'un agent, pour cela ce dernier doit pouvoir percevoir son environnement par le biais de capteurs et agir sur son environnement pour la réalisation de sa tâche.

- Compétences de niveau connaissance :

Correspond aux règles à suivre par l'agent pour réaliser sa tâche. Effectivement pour entreprendre une action, l'agent doit connaître son environnement, cette connaissance introduite par le concepteur ou acquise par des techniques d'apprentissage, et peut être :

- Spécifiée par le développeur : modèles du domaine d'application ou d'un utilisateur ; une connaissance formelle organisée en règles ou en frames.

Inconvénient : manque d'adaptabilité après mise en place de ces agents.

- Spécifiée par l'utilisateur : définition d'une tâche utilisateur par les règles de programmation dans les systèmes à base de règles.

Inconvénient : l'utilisateur n'a pas le choix de son environnement de programmation.

- Dérivée d'autres sources de connaissances : acquisition d'expertise de la communauté agent par les langages agent.
- Apprise par le système : acquise par l'agent de son environnement et de l'utilisateur.

➢ Compétences de niveau communication :

Les capacités de communication influencent les interactions avec l'utilisateur, via l'interface qui est considérée comme canal de communication, il faut noter que dans un environnement distribué le module de communication doit également contrôler les interactions avec les autres agents, pour cela les agents doivent disposer d'un protocole de communication inter-agents, ce qui implique un langage de communication (et même des applications intermédiaires) : *Telescript General Magic*, *SmallTalkAgents Quaser Knowledge Systems*, *AgentTcl* (Tool Command Language) *Gray* 1995, *KIF* (Knowledge Interface Format) et *ACL* (Agent Communication Language) *Genesereth* 1995.

2.5. Les avantages de la technologie agent

Ces avantages (Tableau 3.1) reposent sur les niveaux de compétence de l'agent (ou des agents) pour réaliser la tâche désirée, et peuvent être constatés selon plusieurs catégories de fonctionnalités ou caractéristiques visées.

Caractéristiques	**Avantages**	**Bénéfices**
Automatisation	Réaliser des tâches répétitives : Comportement répétitif d'un utilisateur ou similaire de plusieurs dans un groupe.	Augmenter la productivité individuelle ou de groupe.
Personnalisation	Interagir avec des informations personnalisées (selon le profil et les préférences de l'utilisateur).	Réduire la quantité d'information présentée.
Notification	Prévenir l'utilisateur des événements importants.	Réduire la charge de travail.
Apprentissage	Apprendre à reconnaître le comportement des utilisateurs: cibler les tâches d'automatisation, pour une meilleure personnalisation.	Assistance proactive.
Tutorat	Suivre et diriger l'utilisateur dans un contexte particulier.	Réduire le temps de travail.
Communication	Réaliser les tâches à distance.	Travail local en mode déconnecté.

Tableau 3.1 Caractéristiques, avantages et bénéfices de la technologie agent

3. Système Multi-Agents : SMA

Un système multi-agents est un système distribué composé d'un ensemble d'agents. Contrairement aux systèmes d'*IA*, qui simulent dans une certaine mesure les capacités du raisonnement humain, les *SMAs* sont conçus et implantés idéalement comme un ensemble d'agents interagissant, le plus souvent, selon des modes de coopération, de concurrence ou de coexistence. Ils sont une métaphore de l'organisation sociale (Capera 03), basés sur la distribution des connaissances et du contrôle.

Les *SMAs* sont des systèmes idéaux pour représenter des problèmes possédant de multiples méthodes de résolution, de multiples perspectives et/ou de multiples solveurs. Ces systèmes possèdent les avantages traditionnels de la résolution distribuée et concurrente de problèmes comme la modularité, la vitesse (avec le parallélisme), et la fiabilité (due à la redondance).

Ils sont à l'intersection de plusieurs domaines scientifiques: informatique répartie, génie logiciel, intelligence artificielle et interfaces homme-machine. Ils s'inspirent également d'études issues d'autres disciplines connexes notamment la sociologie, la psychologie sociale, les sciences cognitives et bien d'autres. C'est ainsi qu'on les trouve parfois à la base des:

- Systèmes distribués;
- Interface hommes-machines;
- Bases de données et bases de connaissances distribuées coopératives;
- Systèmes pour la compréhension du langage naturel;
- Protocoles de communication et réseaux de télécommunications;
- Programmation orientée agents et génie logiciel;
- Robotique cognitive et coopération entre robots;
- Applications distribuées comme le web, l'Internet, le contrôle de trafic routier, le contrôle aérien, les réseaux d'énergie, etc.

Les *SMAs* à base d'agents réactifs se caractérisent par un grand nombre d'agents simples, par l'émergence et l'éco-résolution. Les *SMAs* basés sur des agents cognitifs

sont caractérisés par un petit nombre d'agents intelligents et par la coordination (Bergia 99).

3.1. Propriétés des SMAs

Un *SMA* est généralement caractérisé (Bourdon 01, Jarras 02) par :

– Diversité et diffusion des connaissances: Chaque agent a des informations ou des capacités de résolution de problèmes limitées (informations incomplètes et champ d'action limité), ainsi chaque agent a une vision partielle du système,

– Décentralisation des données,

– Calculs et traitement asynchrones,

– Contrôle distribué: il n'y a pas de contrôle global du système multi-agents (contrôle réparti).

De par leur nature répartie, les *SMAs* offrent en outre des propriétés intéressantes comme :

• Efficacité des traitements : les agents travaillent en parallèle et communiquent de façon asynchrone ;

• Robustesse, tolérance aux fautes et fiabilité (sûreté de fonctionnement): la mise hors fonctionnement de quelques agents ne modifie pas sensiblement le comportement global du système ;

• Flexibilité et traitement de systèmes à grande échelle : on peut toujours augmenter le nombre d'agents pour traiter des systèmes de plus en plus gros, sans pour autant perturber le travail des agents existants qui peuvent s'adapter;

• Coût de fonctionnement faible : en principe la répartition des traitements, entre de nombreuses unités simples (agents), conduit à des coûts faibles ; en fait cette propriété est une contrainte de l'approche. En effet, il est extrêmement difficile de maîtriser/prévoir des comportements à l'échelle macroscopique issus de l'interaction de comportements individuels (à l'échelle microscopique) complexes ;

• Coût de développement et de réutilisation intéressant : là encore en théorie il devrait être plus simple de faire développer par des spécialistes des agents indépendamment les uns des autres (approche objets répartis dans l'architecture *CORBA* : ***C****ommon* ***O****bject*

***R**equest **B**roker **A**rchitecture*, utilisant le protocole *IIOP* : ***I**nternet **I**nter**O**bject **P**rotocol*), pour les réutiliser dans divers scénarios applicatifs ; sauf que l'une des différences essentielle entre les agents et les objets est justement dans les notions d'autonomie et de flexibilité liées aux comportements des agents ; dans la mesure où le passage entre les niveaux microscopique et macroscopique pose en soit un réel problème, il paraît difficile aujourd'hui d'affirmer quoi que soit de concret sur ces notions de coût de développement et de réutilisation des agents !

• Plans d'interaction sophistiqués : ils incluent la coopération, la coordination et la négociation.

3.2. Interaction et communication

Un système multi-agents se distingue d'une collection d'agents indépendants par le fait que les agents interagissent en vue de réaliser conjointement une tâche ou d'atteindre conjointement un but particulier. Les agents peuvent interagir en communiquant directement entre eux ou par l'intermédiaire d'un autre agent ou même en agissant sur leur environnement.

Les protocoles d'interaction s'intéressent à des séries de messages échangés entre des agents; ce que l'on appelle des conversations. Il existe différentes formes de protocoles d'interaction selon les systèmes visés. Dans le cas d'agents compétitifs (buts conflictuels ou simplement des intérêts propres), l'objectif des protocoles est de maximiser l'utilité de chaque agent. Dans le cas où des agents ont des buts identiques ou des problèmes communs, comme c'est le cas en résolution distribuée de problèmes, les protocoles d'interaction cherchent à maintenir des invariants globaux sans compromettre l'autonomie de chaque agent, ni avoir recours à un contrôle centralisé (Bourdon 01).

3.2.1. Coordination entre agents

De nombreux exemples de coordination existent dans la vie quotidienne: deux déménageurs déplaçant un meuble lourd, deux jongleurs échangeant des balles avec lesquelles ils jonglent, des personnes qui parlent à tour de rôle en se passant un micro, etc. Les composantes fondamentales de la coordination entre agents sont l'allocation de ressources rares et la communication de résultats intermédiaires (Malone 94). Dans ce contexte, les agents doivent être capables de communiquer entre eux de façon à pouvoir

échanger les résultats intermédiaires. Pour l'allocation des ressources partagées, les agents doivent être capables de faire des transferts de ressources.

Bien entendu, la coordination est une question centrale pour les *SMAs* et la résolution de systèmes distribués. En effet, sans coordination un groupe d'agents peut dégénérer rapidement en une collection chaotique d'individus (Jarras 02).

Il existe plusieurs types de coordination, suivant la nature des agents en jeu. On parlera de coopération entre des agents non-antagonistes et de négociation entre des agents compétitifs ou simplement indépendants (Figure 3.4). Une bonne coopération nécessite que chaque agent maintienne une représentation de ses collègues et des futures interactions entre eux. Cela présuppose des comportements qui intègrent les autres (sociabilité).

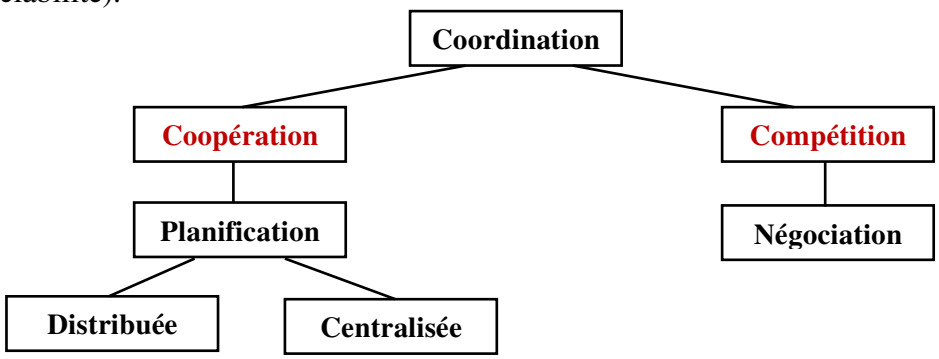


Figure 3.4 Types de coordination (Jarras 02)

3.2.1.1. Négociation entre agents compétitifs

La négociation joue un rôle fondamental dans les activités de compétition en permettant aux personnes de résoudre des conflits qui pourraient mettre en péril des comportements coopératifs. La négociation est définie comme étant le processus d'améliorer les accords (en réduisant les inconsistances et l'incertitude) sur des points de vue communs ou des plans d'action grâce à l'échange structuré d'informations pertinentes. En général, les chercheurs en *IA* distribuée utilisent la négociation comme un mécanisme pour coordonner un groupe d'agents.

Un des protocoles les plus étudiés pour la négociation s'appuie sur une métaphore organisationnelle. Le protocole du réseau contractuel ("Contract-Net") a été une des approches les plus utilisées pour les *SMAs* (Boudon 01, Jarras 02). Les agents coordonnent leurs activités grâce à l'établissement de contrats pour atteindre des buts spécifiques. Un agent, agissant comme un gestionnaire ("manager"), décompose son

contrat (une tâche ou un problème) en sous-contrats qui pourront être traités par des agents contractants potentiels. Le gestionnaire annonce chaque sous-contrat sur un réseau d'agents. Les agents reçoivent et évaluent l'annonce.

Les agents, qui ont les ressources appropriées, l'expertise ou l'information requise, envoient au gestionnaire des soumissions qui indiquent leurs capacités à réaliser la tâche annoncée. Le gestionnaire évalue les soumissions et accorde les tâches aux agents les mieux appropriés. Ces agents sont appelés des contractants ("contractors"). Enfin, gestionnaires et contractants échangent les informations nécessaires durant l'accomplissement des tâches. Par exemple, ce protocole a été utilisé pour développer un système de contrôle de production.

- Négociation centrée environnement

  Dans ce cas, l'idée est de voir comment on peut agir sur l'environnement, en décrivant les règles qui régissent son fonctionnement, pour faciliter le bon fonctionnement des agents en particulier dans les résolutions de conflits via les négociations.

- Négociation centrée agent

  Ici le problème n'est plus d'adapter le contexte à la négociation, mais le comportement de l'agent compte-tenu des propriétés du contexte donné. La plupart des stratégies de négociation de ce type ont été faites pour des problèmes spécifiques, et par conséquent peu de principes généraux ont été proposés.

3.2.1.2. Planification entre agents coopératifs

La coopération peut être considérée comme une attitude intentionnelle adoptée par les agents qui décident de travailler ensemble (Ferber 99). Dans ce cas les agents s'engagent dans une action après avoir identifié et adopté un but commun considéré comme un élément essentiel de l'activité sociale.

Les techniques de planification centralisée pour des groupes d'agents, de conciliation de plans, de planification distribuée, d'analyse organisationnelle, sont toutes des façons d'aider les agents à aligner leurs activités en assignant les tâches après avoir raisonné sur les conséquences de réaliser ces tâches dans des ordres particuliers. Dans une approche de planification multi-agents, un plan multi-agents est un plan qui est créé afin que plusieurs agents puissent l'exécuter. La création d'un tel plan peut être faite par un seul ou par plusieurs agents.

En planification multi-agents centralisée, un agent est responsable de la création du plan qui spécifie les actions planifiées pour tous les agents concernés. Dans une autre façon d'implémenter la planification multi-agent centralisée, les plans des agents sont d'abord créés de façon individuelle; ensuite un agent centralisateur rassemble ces plans et les analyse pour identifier les conflits. L'agent centralisateur en question essaye de résoudre les conflits en modifiant les plans locaux des autres agents et en introduisant des commandes de communication afin que les agents se synchronisent de façon appropriée.

Par ailleurs, dans une approche de planification distribuée, les activités de planification sont réparties au sein d'un groupe d'agents. Cette approche est utilisée quand un seul agent ne peut pas avoir une vue globale des activités du groupe.

En général, la planification multi-agents nécessite une forme ou une autre de synchronisation de plans qui peut être réalisée à divers moments: pendant la décomposition de plan, pendant la construction de plan ou après celle-ci. Les plans des agents peuvent être en conflits en raison d'incompatibilités d'états des systèmes, de l'ordre des activités ou de l'usage des ressources. De tels conflits peuvent être résolus par un agent en particulier (coordonnateur ou médiateur) ou une solution peut être obtenue par négociation.

Dans un *SMA*, les agents doivent gérer des ressources distribuées qui peuvent être physiques (capacités de communication, matières premières, argent) ou informationnelles (telles que les informations au sujet de la décomposition du problème). Les agents doivent adapter leurs plans pour tenir compte de la disponibilité des ressources. Bien entendu, la planification contribue à la coordination, dans la mesure où lorsque les agents adoptent un plan " bien fait ", ils agissent généralement de manière coordonnée.

3.2.2. Types de communication

Les agents peuvent interagir soit en accomplissant des actions linguistiques (en communiquant entre eux), soit en accomplissant des actions non-linguistiques qui modifient leur environnement. En communiquant, les agents peuvent échanger des informations et coordonner leurs activités. Dans les *SMAs*, deux stratégies principales ont été utilisées pour supporter la communication entre agents:

• Envoi de message : Echange direct de messages ou de plans ;

• Partage d'information: Accès à une base de données partagée, appelée tableau noir, dans laquelle les informations sont postées.

Tout protocole de communication (point-a-point, multicast ou broadcast) est représenté par une structure de données qui contient les champs suivants : Emetteur, Récepteur(s), Langage utilisé dans le protocole, Fonctions d'encodage et de décodage et Actions à faire par le(s) récepteur(s).

a. Transfert de plans ou de messages

Dans l'approche de transfert de plans, un agent *X* communique son plan en totalité à un agent *Y* et *Y* communique son plan en totalité à *X*. Cette approche présente plusieurs inconvénients: le transfert de plans est coûteux en ressources de communication et c'est une approche difficile à mettre en œuvre dans des applications réelles car il y a en général un fort degré d'incertitude au sujet des états actuel et futurs du monde.

Dans des situations réelles en plus, il n'est pas possible de formuler à l'avance des plans en totalité. Par conséquent, on a besoin de communiquer des stratégies générales aux agents plutôt que des plans, et donc il est nécessaire que les agents puissent échanger des messages.

b. Echange d'informations par tableau noir

En intelligence artificielle, la technique du tableau noir "blackboard" est très utilisée pour spécifier une mémoire partagée par divers systèmes. Dans un *SMA* utilisant un tableau noir, les agents peuvent écrire des messages, insérer des résultats partiels de leurs calculs et obtenir de l'information. Le tableau noir est en général partitionné en plusieurs niveaux qui sont spécifiques à l'application. Les agents qui travaillent sur un niveau particulier peuvent accéder aux informations contenues dans le niveau correspondant du tableau noir ainsi que dans des niveaux adjacents. Les données peuvent, donc, être synthétisées à n'importe quel niveau et transférées aux niveaux supérieurs alors que les buts de haut niveau peuvent être filtrés et passés aux niveaux inférieurs pour diriger les agents qui œuvrent à ces niveaux.

c. Actes de langages et conversations

On considère que les agents sont dotés de structures de données appelées "états mentaux" à partir desquelles ils peuvent raisonner. Diverses catégories d'états mentaux ont été retenues par les chercheurs dont notamment les croyances, les désirs, les intentions ou buts, les capacités, etc. Ce sont ces mêmes états mentaux qui caractérisent

les modèles des agents cognitifs en intelligence artificielle distribuée. Ainsi, un agent cognitif raisonne sur ses croyances qui représentent sa compréhension du monde dans lequel il évolue. Il raisonne aussi sur ses désirs et intentions en relation avec ses croyances et capacités afin de prendre des décisions auxquelles sont associés les plans qu'il va accomplir pour agir dans le monde.

L'idée est d'utiliser ce que l'on comprend du dialogue entre humains comme modèle pour la communication entre agents. Dans cette approche la production d'un énoncé, est un acte qui sert avant tout à produire des effets sur son destinataire (Bourdon 01). Les principaux actes de langages décrits sont les suivants :

1. Les assertifs servent à donner une information sur le monde en affirmant quelque chose (ex. il fait beau).

2. Les directifs sont utilisés pour donner des directives au destinataire (ex. donne-moi ta montre).

3. Les promissifs engagent le locuteur à accomplir certains actes dans l'avenir (ex. je viendrai à la réunion).

4. Les expressifs servent à donner au destinataire des indications concernant l'état mental du locuteur (ex. je suis heureux).

5. Les déclaratifs accomplissent un acte par le fait même de prononcer l'énoncé (ex. je déclare la séance ouverte).

Les actes de langage, considérés comme des structures complexes, sont définis par trois composantes :

1. La locutoire : c'est la composante physique de la communication (envoi d'ondes sonores ou production de caractères) ; production de phrases à l'aide d'une grammaire et d'un lexique donné.

2. L'illocutoire : elle se rapporte à la réalisation de l'acte effectué par le locuteur sur le destinataire de l'énoncé. En pragmatique du langage, un acte illocutoire est caractérisé par une force illocutoire (ex. affirmer, questionner, demander de faire, promettre, prévenir ...) et par un contenu propositionnel, objet de la force illocutoire. Par exemple "affirmer qu'il pleut" prend la forme "Affirmer (il pleut)".

3. La perlocutoire : cette composante porte sur les effets que les actes illocutoires peuvent avoir sur l'état du destinataire, ses actions, ses croyances et ses jugements. Par exemple convaincre, inspirer, effrayer, persuader...

Lorsqu'un agent *X* communique avec un autre agent *Y*, c'est pour influencer les états mentaux de *Y*. Ainsi, *X* peut transférer à *Y* une information qui va changer les croyances de *Y* ou bien lui proposer d'adopter un but que *Y* va transformer en une de ses intentions. Il est donc pertinent de s'intéresser à la façon dont les messages échangés peuvent être structurés pour refléter ces mécanismes d'influence sur les états mentaux des agents.

***Exemple* : Communication utilisant KQML, ACL et les conversations**

Le langage *KQML* (Knowledge Query and Manipulation Language) a été proposé pour supporter la communication inter-agents (Jarras 02). Ce langage définit un ensemble de types de messages et des règles qui définissent les comportements suggérés pour les agents qui reçoivent ces messages. Les types de messages *KQML* sont de natures diverses:

• Simples requêtes et assertions telle que "ask", "tell";

• Instructions de routage de l'information telle que "forward" et "broadcast";

• Commandes persistantes telle que "subscribe", "monitor";

• Commandes qui permettent aux agents consommateurs de demander à des agents intermédiaires de trouver les agents fournisseurs pertinents ("advertise", "recommend", "recruit" and "broker").

Ce langage a été développé pour les besoins des développeurs d'agents logiciels: le terme "performatif" a été utilisé pour nommer diverses commandes qui ont une certaine ressemblance avec des verbes utilisés de façon performative dans le langage naturel; l'interprétation sémantique actuelle de ces commandes n'est pas satisfaisante.

On appelle performatif un verbe qui permet d'identifier dans une communication, la force illocutoire correspondante. Les performatifs serviront précisément à définir les différents types d'actes de langage que pourront émettre et interpréter les agents sans ambiguïté. Cela va simplifier énormément la conception des agents.

Un message contenu dans le protocole de communication peut être ambigu, il peut ne pas avoir de réponse correspondante simple ou nécessite une décomposition et

l'assistance d'autres agents. Dans tous les cas, le protocole d'échange doit clairement identifier le type du message échangé.

Le principe de *KQML* est de séparer la sémantique liée au protocole de communication (indépendante du domaine d'application), de celle liée au contenu des messages (dépendante du domaine d'application).

Le protocole de communication doit donc être universel (pour tous les types d'agents), concis et constitué d'un nombre restreint de primitives de communication.

Dans *KQML* un message contient toutes les informations nécessaires à sa compréhension :

(KQML-performatif

:émetteur <texte>

:récepteur <texte>

:langage <texte>

:ontologie <texte>

:contenu <expression>

...)

Ces dernières années, *KQML* semble perdre du terrain au profit d'un autre langage plus riche sémantiquement : *ACL*. C'est un langage mis en avant par la *FIPA* (Foundation for Intelligent Physical Agents) qui s'occupe de standardiser les communications entre agents. *ACL* est basé également sur la théorie du langage et a bénéficié grandement des résultats de recherche de *KQML*. Si toutefois, les deux langages se rapprochent au niveau des actes du langage, il n'en est rien au niveau de la sémantique et il semble qu'un grand soin a été apporté au niveau de *ACL* tant au niveau de certains protocoles qui sont plus explicites qu'au niveau de la sémantique des actes eux-mêmes.

4. Le principe Influence/Réaction

Dans l'*IA*, le traitement des actions dans le système est immédiatement validé en tant que modification directe des variables d'état du système. Dans le cas de l'*IAD*, ces validations mettront le système dans un état *incohérent* et devront être évitées. En *IAD*,

on s'attend, effectivement à des actions simultanées résultant du comportement autonome des individus dans de tels systèmes.

En effet, la validation des actions directes adoptée en *IA* peut être source de problèmes. Ces derniers sont considérés comme des faiblesses et des limites de l'*IA*, ils peuvent être illustrés par l'exemple générique suivant:

Dans un système donné, une ressource critique *RC* "*utilisée*" (la signification dépend du contexte) simultanément par deux personnes (agents) effectuant respectivement *Action1* et *Action2* sur *RC* (par exemple: une imprimante pour deux utilisateurs, un produit pour deux consommateurs ou une balle pour deux joueurs).

En *IA*, le nouvel état du système sera le résultat de l'une des situations suivantes:

1. Une seule action ; l'*Action1* ou l'*Action2* sera exécutée et l'effet de l'autre action sera perdu (quelle que soit la signification selon le contexte);

2. Les deux actions ; l'*Action1* et l'*Action2* seront exécutées ensemble (quelle que soit la signification selon le contexte).

Dans les deux cas, *l'autonomie* des agents sera *compromise* et / ou le nouvel état du système sera *incohérent* au contexte. Pour résoudre ce problème, on peut proposer l'utilisation d'un gestionnaire de conflits introduisant des moniteurs et des sémaphores (plus de détails et d'exemples dans (Michel 04)).

Avec une vision plus générale, *M. Wooldridge*, *N R. Jennings* dans (Wooldridge 98) ont prévenu les intéressés des pièges du développement orienté-agent, car utiliser des agents dans un système implique avoir des entités ayant des caractéristiques fondamentales, sinon ça devient une pure prétention. C'est ce qu'a poussé les auteurs de (Drogoul 02) à se demander où sont les agents ? C'est effectivement très rare de trouver dans l'implémentation d'une simulation multi-agents des entités qui aient les caractéristiques qu'on leur prête dans le modèle conceptuel, conséquence du manque de spécifications dont la notion d'agent est victime, notamment en ce qui concerne son implémentation (Michel 04).

Il est aujourd'hui fondamental de définir et de proposer des interprétations purement informatiques à ces différentes propriétés, pour faire du paradigme agent un véritable outil de modélisation, et non seulement un concept métaphorique.

D'autre part, les systèmes multi-agents avec leurs agents autonomes se confondaient à l'*IAD*. Ces systèmes fournissent une approche basée population et définissent un type particulier d'*IAD* appelé Intelligence d'essaims (*Swarm Intelligence* : *SI*). Cette intelligence distribuée *n'est pas explicitement codée* mais *émerge* dans le système *à travers les interactions* des individus.

En *SMA*, pour combler les limites de l'*IA*, il a été proposé (Ferber 96), d'utiliser le *Principe Influence / Réaction* pour traiter les actions simultanées des agents. En ce qui concerne le paradigme agent, ce principe fournit, fondamentalement, les moyens de gérer les interactions et de modéliser les comportements concurrents dans de tel système distribué.

4.1. Le principe Influence/réaction pour la modélisation des actions simultanées

Ayant affaire à des entités autonomes, la simultanéité des actions est une caractéristique inhérente au paradigme agent et elle est difficile à implémenter de manière satisfaisante.

Contraints, les agents ne doivent pas avoir de contrôle sur les conséquences de leurs actes, seul l'environnement est habilité à les calculer et pour lequel la structure interne d'un agent est inaccessible, comme solution, on propose un modèle agent selon le principe Influence/Réaction de (Ferber 96) pour la modélisation des actions simultanées, avec les adaptations faites par (Michel 04) pour la simulation multi-agents (Ferber 96, Michel 04, 07, 09).

En deux points, ce principe, (Figure 3.5) se résume dans le fait que :

1. Les agents n'ont pas un contrôle direct sur le résultat de leurs actions.
2. L'ensemble des influences produites à un instant doit être connu pour pouvoir calculer le nouvel état du monde.

Chaque application de ce principe donnera un modèle pour sa mise en œuvre.

En plus d'être une solution pour la simultanéité, le principe Influence / Réaction fournit des bases de bonne modélisation / programmation d'agents (Michel 04, 09) pour accomplir plus formellement certains aspects du paradigme agent.

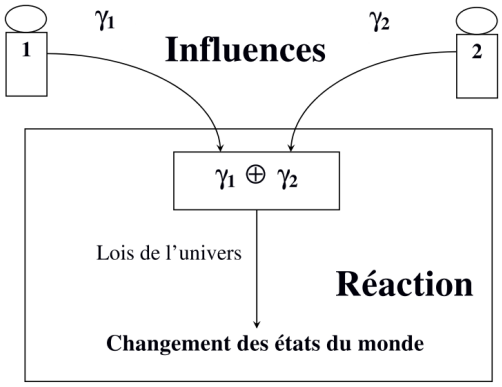


Figure 3.5 Illustration du principe Influence/Réaction (Michel 04)

En tant que principe de modélisation, le principe Influence / Réaction a été défini pour sa capacité à modéliser des comportements concurrent, mais son intérêt va au-delà de cet objectif (Michel 04, 09) :

1. La plus fondamentale conséquence : L'abandon de la représentation de l'action comme une modification de l'état global. (Argument pour le point 3)
2. Permet de donner une véritable sémantique à la gestion des interactions lors de la phase de réaction (grâce à la notion d'influence).
3. Favorise l'isolation d'un module d'interactions (la modélisation / programmation de l'interaction est difficile à identifier vue qu'elle est diluée dans les autres parties du modèle, ce qui prête confusion dans la modélisation du système et facilite le phénomène de divergence d'implémentation).
4. Implique de distinguer, clairement, les variables d'état du système décisionnel de l'agent, son esprit (accédées/modifiées uniquement durant la phase Influence) de celles qui concernent son aspect physique qui font partie intégrante de l'environnement, son corps (modifiées uniquement durant la phase Réaction), pour avoir des agents véritablement autonomes.

La distinction esprit/corps, couplée avec le principe Influence/ Réaction est le seul moyen de faire interagir les agents sans violer leur intégrité interne (leur architecture interne ne doit être accédée ni en lecture ni en écriture).

4.2. Le principe Influence/réaction pour la modélisation des interactions

Dans le contexte de modélisation d'interactions, une démarche correctrice a été proposée en tant qu'aspect de validation des *SMA* (Michel 04): l'étude de la cohérence

paradigmatique. Selon laquelle, le non-respect de l'autonomie des agents par une modélisation des interactions la rend incohérente avec le cadre expérimental (le paradigme agent), et la relation de modélisation ne sera, donc, pas vérifiée. Ce qui va être à la charge du principe Influence / Réaction.

Suite à des expériences selon différents modèles d'interaction par rapport au respect de l'autonomie des agents (Michel 04), il a été constaté que certaines interactions nécessitent de considérer l'ensemble des comportements simultanément et d'autres non (Reproduction : situation d'interaction forte, Consommation de ressource : situation d'interaction faible).

D'une part, plusieurs actions constituent une situation d'interaction forte lorsque la réalisation du but de chaque agent dépend de l'action des autres agents (Michel 04). Le résultat d'une telle interaction ne peut être calculé sans prendre en compte la délibération de l'ensemble des agents concernés, sans quoi leur autonomie décisionnelle ne peut être garantie. Le résultat (en cas de succès) ne peut être observé si le nombre d'agent présents dans le système n'est pas suffisant.

D'autre part, plusieurs actions constituent une situation d'interaction faible lorsque le but de chaque agent peut être réalisé indépendamment des autres agents (Michel 04). Dans cette situation, bien que ces actions puissent interférer entre elles, et quelle que soit la gestion des interactions, l'autonomie des agents n'est pas remise en cause par la technique utilisée.

Les agents interagissent au travers de l'environnement et produisent des influences qui ne sont pas directement corrélées. La faisabilité du but recherché par cette action n'est pas conditionnée par les actions des autres agents. Un seul agent peut mener au même résultat, il suffit d'être un peu plus patient. Les influences peuvent être traitées simultanément ou indépendamment et séquentiellement.

Exemple:

Dans la Figure 3.5, notons δ (t) l'état dynamique d'un système à l'instant t et γ_1, γ_2 deux influences produites à ce moment. Le nouvel état δ (t + dt) est donné par la fonction de réaction (équation1):

$$\delta (t + dt) = \text{Réaction } (\delta (t), \gamma_1, \gamma_2) \qquad (1)$$

Le caractère parallèle des actions peut être obligatoire ou facultatif selon les situations (voir plus de détails dans (Michel 04, 09)).

- ***Interaction forte***

C'est un cas de traitements parallèles obligatoire, nous avons des réactions parallèles (réactions non linéaires), nécessitant une composition de comportement explicite. Pour préserver la cohérence du système et assurer l'autonomie décisionnelle de tous les agents impliqués, nous calculons la réaction de l'environnement en traitant toutes ses influences simultanément comme une unité (équation 2):

$$\delta (t + dt) = \text{Réaction } (\delta (t), \bigcup_{i=1,2} \gamma_i) \qquad (2)$$

- ***Interaction faible***

Dans ce cas, le caractère parallèle n'est plus une obligation (c'est juste un choix de modélisation). Maintenant, nous avons des réactions en série (non parallèles). La cohérence du système et l'autonomie de l'agent ne seront pas compromises par le processus utilisé; on peut utiliser l'équation 2 ou décomposer le calcul global en réactions élémentaires et indépendantes. Nous les exécutons les unes après les autres (équation 3 puis équation 4).

Nous calculons donc d'abord: $\delta'(t') = \text{Réaction } (\delta (t), \gamma_1) \quad t < t' < t+dt \qquad (3)$

Et puis: $\delta (t + dt) = \text{Réaction } (\delta'(t'), \gamma_2) \qquad (4)$

(ou γ_2 puis γ_1)

L'application du principe Influence/Réaction pour le traitement des interactions exige un module d'interaction séparé.

5. L'ingénierie logicielle orientée agent

L'ingénierie logicielle orientée agent (Agent-Oriented Software Engineering : *AOSE*) a évoluée pour inclure des thèmes de haut niveau tels que: *méthodologies*, architectures, implémentations de framework, langages de programmation et communication (Figure 3.6) (Sturm 14a, 14b, Bernon 05, Tveit 01).

Figure 3.6 Carte thématique du génie logiciel orienté agent (Sturm 14a)

5.1. Méthodologies orientées agent

Plusieurs méthodologies pour développer les systèmes basés-agent ont été proposées (Iglesias 99, O'Malley 01), cependant leur application reste limitée vu leur manque de maturité ou leur caractère académique. Citons, à titre d'exemple, les méthodologies : *Adelfe* (Bernon 02, 04), *Gaia* (Wooldridge 00, Moraitis 06), *Passi* (Cossentino 01) et *Ingenias* (Pavón 03). Nous donnons, ci-dessous, des fiches techniques de quelques méthodes et modèles d'ingénierie logicielle orientée-agent.

5.1.1. Modèle organisationnel AGR (Agent/Groupe/Rôle)

AGR est considéré comme un méta-modèle plutôt qu'une véritable méthodologie. Il se concentre essentiellement sur les aspects de conception.

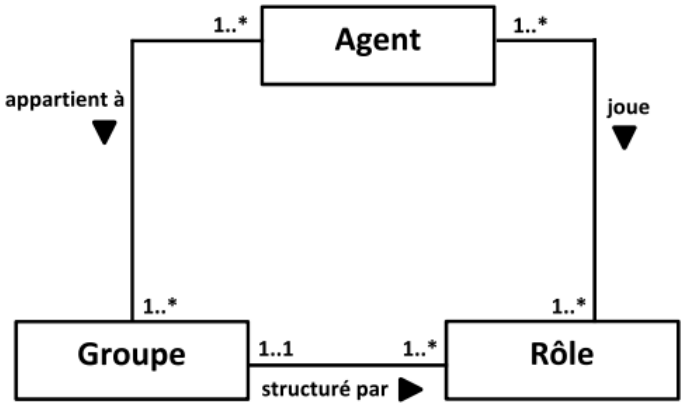


Figure 3.7 Modèle UML du modèle AGR (Ferber 98)

Auteurs: *Olivier Gutknecht; Jacques Ferber, LIRMM, Montpellier, France,* 1999.

Domaine d'origine: Intelligence Artificielle-Ingénierie de Connaissance (*IA-IC*).

Concepts: *agent* (aucune contrainte sur le type et la structure interne de l'agent); *groupe*; *rôle* (Figure 3.7).

Notations: notations dédiées et peut réutiliser *UML* et *AUML*.

Outils: plateforme *MadKit* (Multi-agent development Kit).

Processus: Aucun processus de développement ne lui est véritablement associé.

Dans le cadre d'un projet de grille de calcul, une extension d'*AGR* a été proposée par F*erber* pour inclure une modélisation de l'environnement appelée *AGRE* (Figure 3.8).

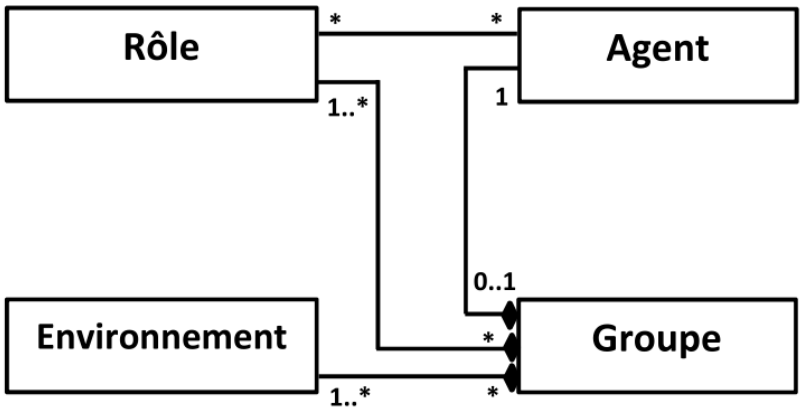


Figure 3.8 Modèle UML du modèle AGRE (Ferber 04)

5.1.2. MaSE (Multi-agent Systems Engineering)

Auteurs: *Scott A. DeLoach*, 1999; *Scott A. DeLoach; Mark F. Wood; Clint H. Sparkman*, 2001 (DeLoach 99, 01, 04).

Domaine d'origine: Génie Logiciel (*GL*).

Concepts: *agent* (les architectures d'agents ne sont pas imposées ; *BDI* ou autres); *tâches*; *rôles*; *buts*.

Notations: inspirées de la conception objet + *UML*.

Outils: grand support d'outils sous forme d'*Agenttool* (permet une génération de code partielle), la version 2.0 implémente les 7 étapes du processus de développement et fournit un support automatique pour transformer les modèles en produit de conception;

Deux langage de description: Agent Modeling language (*AgML*) et Agent Definition Language (*AgDL*);

Elle n'est pas dédiée à un langage ou une plateforme spécifique.

Processus: *MaSE* a un processus de sept (07) étapes divisées en deux (02) phases :

A. **Phase d'Analyse** : inclut trois (03) étapes :

1. *Capture de buts (computation)*: cette phase consiste en deux sous- parties:
   - Identification de buts; identifier les rôles;
   - Structuration de buts (les représenter sous forme d'hiérarchie de buts).
2. *Application des scenarios de cas d'utilisation :*
   - Création des cas d'utilisation – définit le comportement du système.
   - Création des diagrammes de séquence– identifie les messages passés entre rôles.
3. *Raffinage de rôles; transformer les buts en ensembles de rôles en créant les modèles suivants:*
   - Modèle de Rôle – définit les rôles du système et leurs tâches.
   - Modèle de tâche concurrent – définit le comportement des tâches.

B. **Phase de conception :** inclue quatre (04) étapes:

4. *Création des classes d'agent:*

   Assigner les rôles à des classes d'agents spécifiques, le résultat est un diagramme de classe d'agent qui décrit le *SMA* en entier, montrant les agents et leurs rôles et les liens libellés entre eux qui représentent la conversation et son nom;

5. *Construction de conversations:*

   Détails de conversation en utilisant des diagrammes de classes de communication (sous forme de machine à états finis);

6. *Assemblage des classes d'agent (architecture d'agent)*

   On a besoin de définir l'architecture de l'agent et les composants de cette architecture (ne vise et n'impose pas une plateforme particulière d'implémentation);

7. *Conception du système:*

   Construire un diagramme de déploiement qui spécifie l'emplacement des agents dans le système.

5.1.3. Prometheus

Auteurs: *Padgham L.*; *Winikoff M.*, RMIT University, Melbourne, *Australia*, 2002 (Padgham 02, 03, Winikoff 04).

Domaine d'origine: Génie Logiciel (*GL*) & Intelligence Artificielle-Ingénierie de Connaissance (*IA-IC*).

Concepts: *agent* intelligent (*BDI*: Belief, Desire, Intention): fortement proactifs et sont à gros grains; *buts*; *plans*; *groupe* (fixé).

Notations: notations dédiées et réutilise *UML* et *AUML*.

Outils: supportée par deux outils (les deux ne supportent pas la phase de spécification).

1. Plateforme *JACK* Development Environment (*JDE*).
2. Prometheus Design Tool (*PDT*).

Le langage *Java* est préférable pour l'implémentation, conséquence du couplage *Prometheus* et *JACK*.

Processus: Son processus comprend trois (03) phases :

A. ***Spécification du système*** : implique 02 activités :

1. Déterminer l'environnement du système → identifier les interfaces avec l'environnement (actions et perceptions): à travers le développement des scénarios de cas d'utilisation ce qui permet l'identification des buts et sous-buts du système (sous forme de listes) ;
2. Déterminer les buts et les fonctionnalités qui seront regroupées dans un descripteur préliminaire de capacités, ce qui nécessite, en parallèle, la définition des données de travail associées grâce à des diagrammes de couplage de données.

B. ***Conception architecturale*** : implique 03 activités :

1. Définition du type d'agent résultant du regroupement par agent des fonctionnalités déterminées précédemment;
2. Conception de la structure générale du système: Les interactions entre agents, données et perceptions sont spécifiées dans un diagramme de vue du système.

Un diagramme de couplage est aussi utilisé pour spécifier les relations entre données et fonctionnalités regroupées ;

3. Définition des interactions entre agents; qui sont décrites grâce à des diagrammes de séquences.

C. ***Conception détaillée*** : commence par la vue interne des agents et implique 02 activités:

1. Définition de toutes les composantes internes aux agents (selon l'architecture choisie ''BDI vaut mieux pour profiter des outils offerts'', il convient alors de préciser ou de raffiner – capacités, plans, croyances et données – dans des diagrammes de vue d'agent et de capacités;
2. Structures de données détaillées.

5.1.4. Tropos

Auteurs: *Paolo Giorgini*; *Paolo Bresciani*; *Fausto Giunchiglia*; *John Mylopoulos*; *Anna Perini*, University of TRENTO, Department of Information and Communication Technology, Povo–Trento (*Italy*), (projet 2001), 2004 (Giunchiglia 01, Giorgini 04, Bertolini 05).

Domaine d'origine: Génie Logiciel (*GL*) & Intelligence Artificielle-Ingénierie de Connaissance (*IA-IC*).

Concepts: *l'acteur* (*BDI* de préférence), le *rôle*, la *position*, le *but* (soft ou hard), le *plan*, la *ressource* et la *dépendance* pour modéliser les intervenants (stake-holders) dans le domaine cible et leurs intentions.

Notations: notation dédiée et *AUML*.

Outils: une plateforme *BDI*, spécifiquement *JACK* Intelligent Agents.

JACK fournit 5 principales constructions de langage : Agents, capacités, relations de bases de données, événements et plans. *Tropos* fournit des directives et des heuristiques pour correspondre les concepts *Tropos* aux concepts *BDI* et les concepts *BDI* aux 5 constructions de *JACK*.

Processus: *Tropos* comprend cinq (05) phases :

A. ***Exigences premières "early requirements"*** : elle est influencée par le framework de modélisation d'Eric Yu'si : i* modelling framework.

Tropos divise les buts en deux types :

1) Les buts durs (hardgoals) : qui mènent à des exigences fonctionnelles.
2) Les buts doux (softgoals) : qui mènent à des exigences non fonctionnelles.

Il y a deux modèles pour représenter les buts et les acteurs:

1) Diagramme d'acteur : met l'accent sur les intervenants et leurs relations dans le domaine, appelées "dépendance sociales" qui reflètent comment les acteurs dépendent l'un de l'autre pour l'accomplissement des buts, l'exécution des plans et la disponibilité des ressources.
2) Diagramme des buts : montre l'analyse des buts et des plans par rapport à un acteur spécifique ayant la responsabilité de les réaliser.

B. ***Exigences tardives "late requirements" :*** Implique l'extension des modèles crées dans la phase précédente.

C. ***Conception architecturale*** : définit 3 étapes:

1. Introduction et description de nouveaux acteurs par un diagramme d'acteurs étendu, ces acteurs doivent existés pour accomplir des exigences non fonctionnelles ou pour supporter les sous-buts décomposés à la phase précédente.
2. Identification des capacités.
3. Grouper ces capacités pour former des types d'agents ou chaque type est formé par jonction (joining) d'un nombre de capacités.

D. ***Conception détaillée*** : Implique la définition de la spécification d'agents à un niveau micro. Le concepteur doit produire trois types de diagramme décrivant les capacités et les plans.

1. Diagramme de capacités : utilise les diagrammes d'activité d'UML.
2. Diagramme de plans : utilise les diagrammes d'activité d'UML (représentations de fine granularité de chaque nœud dans le diagramme de capacité).
3. Diagramme d'interaction d'agents : c'est des diagrammes d'interactions d'*AUML* (interactions entre agents).

E. Implémentation : *Tropos* choisit une plateforme *BDI*. A ce stade le développeur a besoin de faire une correspondance entre chaque concept de la phase de conception et les 5 constructions de *JACK*.

5.2. Plateformes agent

Les plateformes contribuent dans la composante Outils d'une méthodologie. Elles peuvent être à objectif général (Tableau 3.2) ou dédiées simulation et modélisation (Tableau 3.3) ou à d'autres objectifs spécialisés (Tableau 3.4).

Outils	Langage	Caractéristiques
AGLOBE*, *AglobeX Simulation	***Java***	Conçu pour le test (et aussi pour la modélisation et le développement) de systèmes multi-agent décentralisés.
Cybele	***Similaire à Java***	Commercialisé et utilisé par le gouvernement, l'industrie et le secteur académique en robotique, planification, data mining (extraction de données), modélisation et simulation et contrôle de systèmes de transport aériens et urbains, communication réseaux.
JaCaMo	***AgentSpeak (Jason)***	Un framework de programmation Multi-agent combinant trois technologies séparées: *Jason* pour la programmation d'agent autonomes; *CartAgO* (Common ARTifact infrastructure for AGents Open environments) pour la programmation des artéfacts d'environnement et *Moise* pour la programmation des organisations multi-agent.
JACK	***JACK Agent Language, a super-set of Java***	Un framework de développement de systèmes multi-agent, utilisant (parmi peu d'autres) le modèle *BDI* et fournissant son propre Langage de plan basé-*Java* et des outils de planification graphiques.
JADE (Java Agent DEvelopment)	***Java, C#/.NET (JADE LEAP)***	Framework implémenté en *Java* simplifiant l'implémentation de systèmes multi-agent à travers un middleware compilé avec les spécifications *FIPA* et des outils graphiques supportant les phases de débogage et de déploiement.
JADEX (JADE extension)	***Java***	Une couche rationnelle au-dessus de *JADE* permettant un développement facile d'agents rationnels selon un paradigme de modélisation *BDI* (Belief, Desire, Intention). Les agents sont écrits en *XML* et les corps de plan en *Java*. *ActiveComponents* nouvelle version se focalisant sur les web services.

BDI4JADE	***Java***	Fourni une implémentation d'architecture *BDI* comme une couche au-dessus de *JADE*. Le but est de fournir un environnement pour l'implémentation d'applications d'entreprise, ce qui implique une intégration totale des technologies standards.
Jason	***AgentSpeak***	Plateforme de développement de systèmes multi-agent. *AgentSpeak* est un des plus influant langages abstrait basé sur l'architecture *BDI*. Ses agents sont parfois mentionnés comme des systèmes de planification réactifs. *Jason* inclut aussi une communication inter-agent basée speech-act.
MADKIT	***Java***	Une bibliothèque *Java* légère pour la conception et la simulation de systèmes multi-agent. *MADKIT* est conçu pour construire facilement des applications et des simulations distribuées et suit une approche centrée organisation (*OCMAS*) au lieu d'une centrée agent (*ACMAS*) et donc n'impose pas un modèle prédéfini d'agent. *Madkit* est construite sur le modèle organisationnel *AGR*: *l'Agent* joue des *Rôles* dans des *Groupes* et crée donc des sociétés artificielles.
Soar	***Soar***	*Soar* est une architecture cognitive générale pour développer les systèmes qui exhibent un comportement intelligent. *Soar Markup language* permit aux agents la communication avec des environnements externes et même avec des agents écrits en d'autres langages.

Tableau 3.2 Plateformes à objectif général (encore utilisées) (Florin 15)

Outils	Langage	Caractéristiques
Gama	***Java***	Plateforme de simulation fournissant aux experts de domaines, aux informaticiens et aux modeleurs un environnement de développement complet de modélisation et de simulation pour construire des simulations multi-agents spatiales explicites.
NetLogo	***Logo dialect***	Environnement programmable de modélisation pour la simulation de phénomènes naturels et sociaux. Privilégié pour la modélisation de systèmes complexes se développant dans le temps.
PRESAGE2	***Java***	Plateforme de simulation de prototypage rapide de sociétés d'agents.
Repast	***Java, Python, C#/.NET, C++, ReLogo, Groovy***	Conçu pour des applications de science sociale, supporte le développement de modèles très flexibles d'agents en interaction pour utilisation en stations de travail.

Tableau 3.3 Plateformes de simulation et de modélisation (encore utilisées) (Florin 15)

Outils	Langage	Caractéristiques
AgentSheets	***Programmation conversationnelle***	Création de jeux et simulation, démonstration interactive, applications informatiques pour les multimédias.
Framsticks	***FramScript (similar à JavaScript, Java), C++***	Simulation 3D de vie (modélisation de structures mécanique "corps" et systèmes de contrôle "cerveaux").
MATSim	***Java pour développeurs. Interface graphique, fichiers de configuration pour les utilisateurs.***	Implémente des simulations de transport basées-agent à grande échelle.
Mobile-C	***JACK Agent Language, a super-set of Java***	Un framework de développement de systèmes multi-agent supportant des agents mobiles, utilisant (parmi peu d'autres) le modèle *BDI* et fournissant son propre Langage de plan basé-*Java* et des outils de planification graphiques.
Orleans	***C#***	Construire des applications de calcul distribué à grande échelle, sans avoir besoin d'apprendre des techniques de programmation complexes pour gérer la concurrence, tolérance aux pannes et gestion de ressources. Déployable en cloud.
StarLogo	***Graphical user interface, modèles construits avec "blocks"***	Environnement de programmation permettant aux étudiants et aux enseignants de créer des jeux et simulations 3D pour comprendre les systèmes complexes, jeux vidéo éducationnels.
TuCSoN (Tuple Centres Spread over the Network)	***ReSpecT, a logic-based coordination language***	Bibliothèque *Java*, c'est un modèle de coordination de processus distribués d'agents autonomes, intelligents et mobiles.

Tableau 3.4 Plateformes à objectif spécial (encore utilisées) (Florin 15)

6. Conclusion

A travers ce Chapitre nous avons introduit les concepts d'agent et de système multi-agent via la description de leurs propriétés et caractéristiques et la présentation du domaine des méthodologies et plateformes orientées agent. La gestion des interactions dans ces *SMA* était confiée au principe Influence/Réaction.

Ce principe fait du paradigme agent un véritable outil de modélisation, et non seulement un concept métaphorique. En plus d'être une solution pour la simultanéité, le principe Influence / Réaction fournit des bases de bonne modélisation / programmation agents pour accomplir plus formellement certains aspects du paradigme agent. En tant que principe de modélisation, le principe Influence / Réaction a été défini pour sa capacité à modéliser des comportements concurrent, mais son intérêt va au-delà de cet objectif comme cela sera démontré dans les Chapitres suivants.

CHAPITRE 4

SYSTEMES MULTI-AGENTS BIOMORPHIQUES : ANALYSES ET REFLEXIONS

CHAPITRE 4

SYSTEMES MULTI-AGENTS BIOMORPHIQUES : ANALYSES ET REFLEXIONS

1. Introduction

Les deux Chapitres précédents représentaient l'état de l'art de notre travail de thèse. Ce Chapitre sera consacré à des réflexions et des constats sur cet état de l'art pouvant servir d'arguments de soutiens et de motivations pour notre contribution.

Il sera, essentiellement, question de revoir les caractéristiques des systèmes naturels et artificiels et établir une analogie entre les deux. De discuter le défi de conception des premiers et les potentialités de développement offertes par le paradigme agent couvrant les aspects modélisation et outils. Nous soulevons l'existence d'une *dualité Micro/Macro* observée entre tous les concepts relatifs à ces deux types de systèmes à différents niveaux : définition, caractérisation séparée/commune, conception.

2. Conception bio-inspirée

Historiquement, l'évolution de toute approche ou paradigme doit s'accompagner d'une évolution méthodologique pour faire face à l'aspect conception. Cependant, à côté de l'utilisation étendue et la prolifération des systèmes biomorphiques, de nouveaux défis surgissent sur différents plans. Cela se manifeste par l'absence de cadre général de canalisation des efforts menant à une orientation vers une direction ou une autre, par l'étude et la comparaison des approches disponibles, bénéfiques et pratiques lors du développement d'un système complexe. Ce qui dévoile l'absence d'une démarche de développement qui s'accommode aux diverses approches bio-inspirées afin de tirer profit de ce qui fait la force de chacune.

Malgré les difficultés inhérentes à la survie, dans le monde naturel, les organismes biologiques évoluent, s'auto-organisent et s'auto-réparent usant seulement d'un savoir local et sans contrôle centralisé. L'analogie entre ces systèmes biologiques et les systèmes multi-agents est plus qu'évidente. En fait, chaque entité dans les systèmes réels et naturels est facilement identifiable à un *agent* ; ce qui fait de ce dernier un *candidat de choix pour leur modélisation*. Dans un contexte de simulation, les *SMA* ont été utilisés pour imiter des comportements, fonctionnalités ou structures de systèmes biologiques. Dans un contexte général, les systèmes bio-inspirés sont réalisés avec des modèles de conception ad hoc ou avec un modèle multi-agents à fonctionnalité ciblée. Par conséquent, ces travaux souffrent de deux faiblesses : l'utilisation de modèles dédiés à des fins restrictives (tels que les projets académiques) ou l'absence de modèle de conception.

Par conséquent, la nécessité d'une modélisation bio-inspirée associée et spécifique devient de plus en plus urgente. L'idée est d'opter pour une approche unificatrice (à base d'agent, pourquoi pas ?) pour la conception de systèmes bio-inspirés. Une telle représentation abstraite unifiée aidera au moins à surmonter le manque de réutilisation dans ce domaine.

2.1. Prémisses d'une conception bio-inspirée

Les prémisses de toute démarche de développement de systèmes biomorphiques se répartissent en deux points :

1- ***Caractérisation des approches biomorphiques*** : Il s'agit d'arriver à identifier les processus de base et à décrire formellement leur modèle de calcul. Comme les approches sont nombreuses, il convient de distinguer les approches de base des approches composées ou hybrides.

2- ***Caractérisation du contexte d'applicabilité de chaque approche biomorphique de base*** : Il s'agit de parvenir à un état où connaissant certains critères sur un problème donné, il sera possible de choisir l'approche bio-inspirée à appliquer ou indiquer des éventuelles combinaisons.

2.2. Conséquences d'une conception bio-inspirée

Sur la base de ces deux prémisses, nous nous intéressons ici à quelques considérations relatives à une démarche de développement *multi-paradigmes* biomorphique.

Il convient de constater que l'aspect biomorphique concerne tout le cycle de vie d'un système logiciel. Concernant la phase des exigences qui est censée délivrer les exigences fonctionnelles et non fonctionnelles du système, une détermination préliminaire du type d'approche biomorphique à utiliser pour chaque exigence ou groupe d'exigences est nécessaire. C'est à ce niveau qu'on peut, par exemple, déterminer que telle ou telle exigence présente des caractéristiques qui suggèrent l'utilisation d'une optimisation par colonies de fourmis ou l'utilisation d'une classification par réseau de neurones. La détermination de l'approche biomorphique adéquate pour une exigence donnée, est étroitement liée aux prémisses précédemment introduites.

La phase de conception est aussi une phase clé. Dans la conception architecturale, cette phase permet de décomposer le système en sous-systèmes et de déterminer le rôle joué par chacun et les interactions qui doivent exister entre les sous-systèmes. Pour cela, il faut d'abord déterminer l'approche biomorphique principale à utiliser à partir des exigences principales du système. En fonction de ces exigences, il est possible qu'aucune des approches biomorphiques de base ne corresponde et, à ce moment, il serait convenable de considérer des combinaisons (i.e. hybridations). La deuxième étape dans la conception est la conception détaillée. Lorsqu'un sous-système doit être conforme à une approche biomorphique donnée, sa conception détaillée devrait spécifier les entrées et les sorties et les adaptations nécessaires de ces derniers pour appliquer l'approche.

2.3. Règles d'application d'une approche multi-paradigme

Cette vision des systèmes complexes et la multitude et la variété de paradigmes bio-inspirés disponibles aujourd'hui soulèvent des remarques qu'il convient de citer :

1. L'approche multi-paradigmes n'est autre qu'une bio-inspiration plus poussée qui fait l'analogie entre un système informatique et un biotope. Elle n'est donc pas une approche circonscrite par une seule métaphore mais par plusieurs.

2. L'approche multi-paradigmes est une approche systémique qui cherche à intégrer, voire à hybrider, les paradigmes pour tirer profit de leur synergie. Par exemple, un système peut être modélisé comme une colonie de fourmis qui utilisent les algorithmes génétiques comme modèle de calcul.
3. Aucun paradigme ne prédomine les autres dans l'absolu mais, un paradigme peut-être au premier plan dans un contexte et en second plan dans un autre. Par exemple, un système peut être modélisé comme une espèce en évolution (application d'une approche évolutionnaire) où les individus sont des réseaux de neurones pour lesquels on cherche à améliorer la configuration ou les poids synaptiques. L'inverse est aussi possible, il suffit de considérer un réseau de neurones où chaque nœud calcule sa fonction de combinaison par un algorithme génétique.
4. Les paradigmes peuvent être utilisés de manière réentrante. Par exemple, un réseau de neurones dont les sorties servent à sélectionner un autre réseau de neurones parmi plusieurs.

Notons pour finir cette section, que la persistance de divers langages de programmation et leur coexistence parfois étroite est une réalité qui illustre l'intérêt pratique d'une approche multi-paradigme (Cas de la plateforme .NET "Dotnet" de Microsoft qui est indépendante de tout langage de programmation et en supporte nativement un grand nombre). Les sections suivantes mettront l'accent sur le choix d'une approche intégrante, unificatrice, générique et multi-paradigmes bio-inspirés pour faire face aux systèmes biomorphiques.

3. Le paradigme Agent

Notre utilisation du terme *approche* fait référence à une vision ou un processus pour faire face à un problème ou une situation donnée. Cette vision sera désignée comme *paradigme* sous réserve d'être bien définie et largement utilisée, au point d'être reconnue comme *référence*. Par exemple, l'agent et l'objet sont paradigmes dans de nombreux domaines (programmation, conception, modélisation, etc.), quand on les qualifie d'approches, on sous-entend la vision globale et la façon dont ils procèdent.

3.1. Agent versus objet et acteur

En tant que concept de modélisation, pour surmonter la nature passive de l'*Objet*, le concept moins connu d'*Acteur* a été lancé.

L'*Acteur* est un modèle mathématique de calcul concurrent utilisé avec plusieurs implémentations pratiques de systèmes distribués. Il a été construit avec une valeur ajoutée principale; son comportement *asynchrone* (Figure 4.1). Le concept d'*Acteur* a été initié en 1997 et laissé de côté et ignoré pendant une décennie. Il a été relancé d'abord par *Gul* (Agha 85) et ensuite par *Karmani* et *Gul* (Karmani 09, 11). Le concept d'*Agent* devance l'*Acteur* par ses compétences d'*interactivité* (Figure 4.1). Les trois concepts sont devenus des *paradigmes* dans le domaine Informatique.

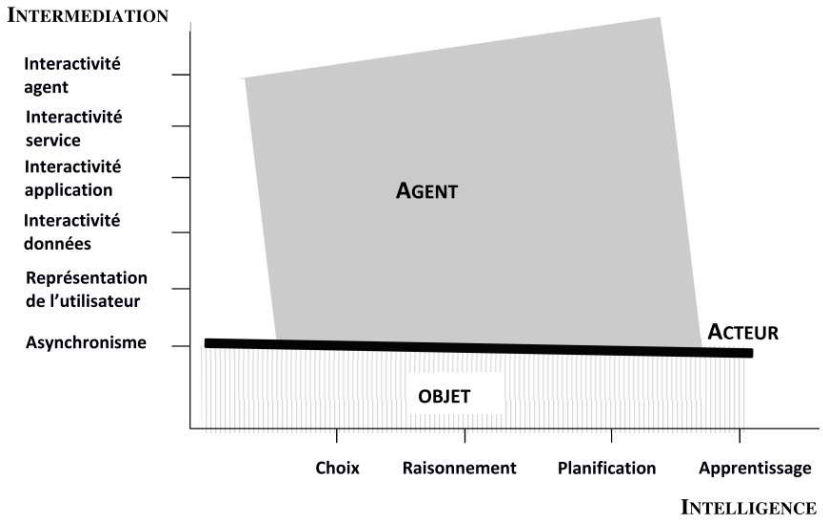


Figure 4.1 Positionnement des concepts Agent, Acteur et Objet selon les caractéristiques Intelligence et Intermédiation

Si on prend les caractéristiques les plus importantes et illustratives; *Intelligence* et *intermédiation*, la Figure 4.1 décrit les emplacements respectives des trois paradigmes *Agent*, *Acteur* et *Objet*. Cette figure a été inspirée d'une description graphique de type et de fonctionnalités *Agent* (Caglayan 98). Elle a été affinée après dans (Zeghida 03, 04) et étendue aux concepts d'*Acteur* et d'*Objet* (Zeghida 18a).

Sur l'axe *Intelligence*, les trois paradigmes peuvent assurer cette caractéristique plus ou moins facilement. Sur l'autre axe (*Intermédiation*), on distingue une relation d'inclusion (Figure 4.2). En effet, l'*Objet* ne peut même pas assurer la première étape: *l'asynchronisme*, qui est bien géré par l'*acteur*. L'*Agent* atteint des étapes plus avancées, avec ses moyens sophistiqués de *communication* (appelée *interaction* agent).

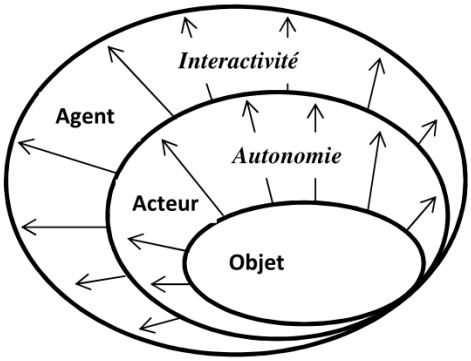


Figure 4.2 L'inclusion Objet, Acteur et Agent

Dans l'interaction agent, on distingue un mode indirect, utilisé seulement pour une coordination limitée (phéromones dans les colonies de fourmis) et un mode direct. Ce dernier type est largement utilisé: langage *Agent* (*KQML*, *ACL-FIPA*), ontologies et support de communication (présent dans les plateformes d'agents telles que *Jade* (Bellifemine 99) ou *MadKit* (Gutknecht 00). Le mode direct est structuré en utilisant; des protocoles, des jeux de dialogue ou des systèmes d'argumentation (Huget 14).

Dans le Tableau 4.1, nous situons l'*Agent* par rapport au concept le plus connu et largement utilisé d'*Objet*.

Critères de comparaison	***Approche Objet***	***Approche Agent***
Nature	Passive	Active et Autonome
Réalisation d'état et de comportement	Encapsulée	Encapsulée
Activation de comportement	Non encapsulée	Encapsulée
Système de fonctions générique	Important	Négligé
Description de types d'interaction	Mécanismes primitifs	Mécanismes avancés
Modèles d'interaction	Rigides et prédéfinis	Flexibles et sophistiqués
Moyens d'abstraction	Insuffisants	Suffisants
Spécification et gestion des relations organisationnelles	Support Minimal (hiérarchies d'héritage statique)	Support Avancé
Modélisation de systèmes complexes	Non supportée	Supportée par des concepts / mécanismes

Tableau 4.1 Comparaison des approches Objet & Agent

L'utilisation abusive du terme *Agent* intelligent, a fait que chacun possède une vision personnelle de cette entité, et plusieurs définitions lui ont étaient données selon les

détenteurs de cet "atout magique" qui n'est qu'un programme qui exécute ce que pourquoi il a était conçu avec de nouvelles manières d'utilisation des technologies actuelles.

Les agents représentent, essentiellement, une nouvelle application des technologies existantes et ne sont, donc ni une nouvelle technologie ni une technologie à part entière. De plus les agents ne sont, nécessairement, pas une nouvelle forme d'application isolée, mais de nouvelles potentialités ajoutées aux applications existantes ou une implication distincte des technologies existantes pour en concevoir de nouvelles.

La technologie agent ne constitue, généralement pas, l'aspect structurel d'une application, c'est plutôt un ensemble de caractéristiques qui permet l'extension des fonctionnalités ou la qualité d'une application : on ne crée pas d'application à base d'agents, mais on donne une valeur ajoutée, à une application nouvelle ou existante par l'approche *Agent* (Caglayan 98).

3.2. Agrégation d'agents

Pour l'agent, les caractéristiques les plus souhaitées sont donc l'*autonomie* et l'*interactivité*. La première leur donne l'aspect asynchrone et la dernière fait référence à leurs techniques et moyens sophistiqués d'interaction. Pour une agrégation d'agents ou un *SMA* la caractéristique la plus significative au contexte sera l'*ouverture* du système.

Cela découle du fait qu'on peut toujours *augmenter/diminuer* (éventuellement) *le nombre d'agents* pour traiter des systèmes de plus en plus gros, sans pour autant perturber le travail des agents existants qui peuvent s'adapter. Cette propriété reste encore ouverte dans la mesure où aujourd'hui encore, elle est à elle seule un sujet de recherche. Lorsqu'elle est jumelée avec l'aptitude du système à *changer sa structure organisationnelle durant l'exécution* elles définissent un *système ouvert*.

Notons que les systèmes naturels sont, par définition, des systèmes ouverts, ainsi les systèmes artificiels bio-inspirés doivent l'être aussi, par conséquence.

Outre leurs caractéristiques innées, un *système ouvert* doit avoir les trois caractéristiques suivantes (Zeghida 18a):

1. Le nombre de composants du système peut changer; le système accepte de nouveaux composants et permet le départ de membres existants.

2. La structure organisationnelle du système peut changer ; il n'y a pas d'organisation prédéfinie et fixe à respecter, les composants peuvent former et dissoudre librement des agrégations et des groupes.

3. Les deux caractéristiques précédentes doivent être effectuées avec des systèmes en cours d'exécution (en action).

Les deux premières caractéristiques sont suffisantes dans la nature pour définir un *système ouvert*. La troisième caractéristique peut être ignorée avec les organisations et les organismes vivants, car elle est vérifiée naturellement: l'écosystème ne sera pas contraint de s'arrêter ni même d'attendre les changements de sa structure et le nombre de ses composants. Dans le monde artificiel (tel en informatique), la troisième caractéristique est très importante. Nous pouvons modifier la structure et le nombre de composants d'un système en modifiant son code lorsqu'il est arrêté; dans ce cas, le système n'est pas ouvert. Pour être ouvert, les deux modifications précédentes doivent être observées dans un système en cours d'exécution (Zeghida 18a).

Les *SMAs* basés sur des agents cognitifs misent, essentiellement, sur l'intelligence de ces agents. Lorsqu'ils sont basés sur des agents réactifs (non intelligents), ils misent, alors, sur les interactions des agents pour obtenir un comportement collectif intelligent. Ils définissent, dans ce dernier cas, un type particulier d'intelligence artificielle distribuée appelée intelligence d'essaims (*Swarm Intelligence* : SI). C'est le fait que des *fonctionnalités intelligentes*, qui n'ont pas été explicitement codées dans le système, peuvent *émerger* à travers les *interactions des agents* (Zeghida 18a).

3.3. L'aspect méthodologique

En général, une méthode peut être vue comme l`ensemble de représentations abstraites (modèles) de tout ou partie du réel, avec les formalismes et les moyens d'exploitation et de manipulation de ces modèles (outils) et une démarche de mise en œuvre (Zeghida 18a):

Méthode = {***Modèles***; ***Outils***} + ***Démarche***.

Donc, une méthodologie de conception comprendra (Zeghida 18a):

1. ***Modèles***: représentations abstraites du monde réel ou d'une partie de ce dernier, on parle aussi de notations;

2. ***Outils***: moyens pour représenter, manipuler et mettre en œuvre les modèles;
3. ***Démarche***: ensemble coordonné d'étapes, de phases et de tâches indiquant le processus à suivre pour la conception du système.

L'aspect méthodologique est un des thèmes de haut niveau de l'ingénierie logicielle orientée agent (Agent-Oriented Software Engineering : AOSE). Toute contribution en rapport à cet aspect touchera, plus au moins, les domaines de *modèles*, *outils* et *démarche*.

4. Analogie entre systèmes biologiques et multi-agents

Les sections précédentes peuvent être considérées comme des motivations et/ou des arguments qui plaident pour une approche unificatrice basée *Agent* pour la conception de systèmes bio-inspirés. Ces arguments et motivations peuvent être consolidés par l'exploitation des constats suivants.

4.1. Intérêt mutuel des systèmes biologiques et multi-agents

En intelligence artificielle, penser *bio* est parfois comme penser système multi-agent, et penser *SMA* c'est penser modélisation et simulation. Cette transitivité des *SMAs* est un pont naturel entre le monde réel et la simulation et la modélisation en informatique. C'est une généralisation de ce qui a été attesté pour l'immunologie par *Bakhouya* (Bakhouya 03). Donc, pour la biologie et les *SMAs*, l'intérêt est mutuel.

La biologie soutient les *SMAs* en particulier et le domaine informatique en général, en fournissant aux systèmes artificiels des principes, des processus et des mécanismes disponibles dans les systèmes biologiques. Ceci est réalisé à travers des *métaphores* biologiques qui sont des analogies établies entre le monde biologique et le monde artificiel, afin de proposer des approches imitant certains aspects du monde naturel tout en ignorant d'autres (Tableau 4.2). Un aperçu historique des approches bio-inspirées peut être trouvé dans (Lodding 04). Fondamentalement, les métaphores n'essaient pas de reproduire ce qui est biologique, mais plutôt de l'interpréter en termes de ce qu'il est possible et raisonnable de faire. Ainsi, nous pouvons conclure que les métaphores biologiques évoluent et dépendent de notre *compréhension* de la réalité et de notre capacité à extraire des éléments *bénéfiques* et *pratiques*.

Paradigmes	***Métaphore***	***Nature d'inspiration***
Réseaux de Neurones Artificiels (Artificial Neural Network: ANN)	Fonctionnement et structure du cerveau	Structurelle & Fonctionnelle.
Algorithme Génétique (Genetic Algorithm : GA)	Mécanismes génétiques	Fonctionnelle.
Systèmes Flous (Fuzzy System: FS)	Raisonnement humain	Fonctionnelle.
Système Immunitaire Artificiel (Artificial Immune System: AIS)	Mécanismes opératoires & organisationnels de cellules immunitaires	Structurelle & Fonctionnelle.
Optimisation par Colonie de Fourmis (Ant Colony Optimization : ACO)	Comportement social de colonie de fourmis	Comportementale.
Optimisation par Essaim de Particule (Particle Swarm Optimization : PSO)	Comportement social d'essaim d'oiseaux en vol	Comportementale.

Tableau 4.2 Nature d'inspiration de quelques paradigmes bio-inspirés

De l'autre côté, les *SMAs* permettent la construction et la conception de systèmes complexes hautement distribués et adaptables aux changements environnementaux. Les *SMAs* offrent aux biologistes la possibilité de modéliser et simuler, aussi simplement que possible, des systèmes naturels complexes (cellules/molécules en interaction, insectes, oiseaux, poissons ou autres organismes vivants) procurant une reproduction artificielle d'un phénomène naturel pour:

a. Comprendre leurs processus / mécanismes.

b. Identifier de nouvelles métaphores: modèles de calcul/mémorisation ou outils de résolution/optimisation.

4.2. La dualité Micro/Macro entre systèmes naturels et artificiels

Mise à part leur soutien mutuel, une analogie explicite peut être observée entre les systèmes biologiques et *SMAs*. Par exemple, certaines approches bio-inspirées sont facilement identifiables à un agrégat d'agents et manifestent, par conséquent, une analogie directe avec le concept *SMA* (le niveau macro). D'autres peuvent être utilisés dans le niveau micro comme un modèle de calcul détenu par le concept d'agent, comme le montre la Tableau 4.3.

Niveau Micro (Agent)	*Niveau Macro (SMA)*
- Réseaux de Neurones Artificiels (ANN) - Algorithme Génétique (GA) - Systèmes flous (FS)	- Systèmes Immunitaire Artificiels (AIS) - Optimisation par Colonie de Fourmis (ACO) - Optimisation par Essaim de Particule (PSO)

Tableau 4.3 Classification de quelques paradigmes bio-inspirés

Notons que pour un usage particulier et un besoin d'abstraction spécifique, nous pouvons utiliser un paradigme de niveau micro à un niveau macro selon le Tableau 4.4. Par exemple, avec une métaphore fonctionnelle (Tableau 4.4), *GA* a été classé au niveau micro (Tableau 4.3), mais avec un niveau d'abstraction plus profond, il peut être utilisé dans un niveau macro, où chaque génotype, par exemple, sera détenu par un agent.

Nature de la métaphore	*Niveau Micro (Agent)*	*Niveau Macro (SMA)*
Fonctionnelle	Ok	
Structurelle		Ok
Comportementale		Ok

Tableau 4.4 Classification de paradigmes bio-inspirés selon la nature de la métaphore

D'autre part, nous pouvons faire une confrontation avec ce qui a été présentés dans le Chapitre 2. En effet, en plus du fait que les *SMAs* comme les approches biomorphiques considèrent que les systèmes sont formés d'entités en interaction pouvant former, occasionnellement, des regroupements ; sous-systèmes, on constate une grande ressemblance dans les critères de caractérisation des approches *SMA* et biomorphiques.

Il est possible de partager ces caractéristiques en deux catégories : la catégorie intra-entité et la catégorie inter-entités. En d'autres termes, nous caractérisons aussi bien les entités prises isolément ainsi que leurs interactions. Pour ces raisons, nous pensons que l'approche agent se place naturellement comme un candidat de choix pour jouer le rôle d'un formalisme unificateur des systèmes biomorphiques.

En plus de l'analogie et l'intérêt mutuel qui règnent entre les systèmes biologiques et les systèmes artificiels, représentés par les *SMAs*, cette dualité *Micro*/*Macro* observée dans

les métaphores biologiques et leurs natures (Tableau 4.3 et Tableau 4.4) est aisément généralisable.

En effet, tous les concepts relatifs à ces deux types de systèmes obéissent à cette dualité (Tableau 4.7) : les définitions, les caractéristiques des approches de conception bio-inspirées, les caractéristiques des approches de conception agent bio-inspirées ou dans le principe influence/réaction, soient, donc dans:

- ✓ Les caractérisations d'une approche biomorphique proposées par *Lodding* comme illustré par le Tableau 4.5 : Certaines de ces caractéristiques se réfèrent au niveau *micro*, qui est la composante individuelle ou atomique (*individu*) et d'autres à leur agrégat (*système* ou *sous-système*) donc à un niveau *macro* ;

Critères	***Nature***
Interaction collective : Le comportement du système résulte de l'interaction collective de plusieurs entités similaires et indépendantes.	Niveau Macro: SMA
Emergence : Le comportement du système émerge de l'interaction des entités sans être explicitement décrit dans ces dernières.	Niveau Micro: Agent
Action autonome : Les entités agissent d'une manière autonome.	Niveau Macro: SMA
Information et interaction locale : Les entités opèrent en fonction des informations et des interactions locales et leur portée spatiale est plutôt locale.	Niveau Micro: Agent
Naissance et mort : Les entités naissent et disparaissent librement au fur et à mesure de l'évolution du système (évolution libre du groupe).	Niveau Micro: Agent
Adaptation : Les entités ont la capacité de s'adapter aux changements d'objectifs, des connaissances et de conditions.	Niveau Micro: Agent
Evolution : L'espèce a la capacité d'évoluer dans le temps.	Niveau Micro: Agent

Tableau 4.5 Caractéristiques d'une approche biomorphique

- ✓ Les caractérisations d'une approche multi-agent biomorphique proposées par *Parunak* (Tableau 4.6) : Certaines de ces caractéristiques se réfèrent au niveau

micro, qui est la composante individuelle ou atomique (*agent*) et d'autres à leur agrégat (*niveau multi-agen*) donc à un niveau *macro* ;

Critères	***Nature***
Les agents doivent correspondre à des entités et non à des fonctions abstraites.	Niveau Micro: Agent
Les agents doivent être petits en taille (petite partie du système total), en temps (capable d'oublier) et en porté (éviter les connaissances et les actions globales).	Niveau Micro: Agent
La communauté des agents doit être décentralisée, sans aucun point singulier de contrôle ou d'échec.	Niveau Macro: MAS
Les agents doivent être divers, ni homogènes ni incompatibles. L'aléatoire et la répulsion sont d'importants outils pour le maintien et la stabilisation de cette diversité.	Niveau Micro: Agent
Les communautés d'agents doivent inclure un mécanisme de dissémination de l'information afin d'augmenter leur réactivité.	Niveau Macro: MAS
Les agents doivent avoir les moyens pour capter et partager ce qu'ils savent/apprennent.	Niveau Micro: Agent
Les agents planifient et tournent d'une façon concurrente et non séquentielle.	Niveau Micro: Agent

Tableau 4.6 Caractéristiques d'une approche multi-agent biomorphique

✓ Le Principe Influence/Réaction, le calcul de la *Réaction* du système (*Macro*) se fait par la prise en compte de toutes les *Influences* : interactions individuelles des agents (*Micro*).

	Niveau Micro	***Niveau Macro***
Système multi-agent	Agent	Groupe (un SMA)
Principe Influence/Réaction	Phase Influence	Phase Réaction
Système biologique	Individus	Sous-Système biologique
Paradigme bio-inspiré	Basé Individu	Basé Population
Conception bio-inspirée	Caractéristiques Individuelles	Caractéristiques Collectives
Système multi-agent biomorphique	Caractéristiques de l'Agent	Caractéristiques de Groupe (SMA)

Tableau 4.7 La dualité Micro/Macro dans les systèmes biologiques et artificiels

4.3. Conséquences de l'utilisation conjointe des paradigmes biologiques et concepts agent

Nous résumons dans la Figure 4.3 et la Figure 4.4, les travaux effectués dans des domaines particuliers de l'informatique et nous positionnons notre contribution.

La Figure 4.3 décrit l'utilisation combinée / séparée d'approches bio-inspirées et de concepts agents / multi-agents dans le domaine de l'intelligence artificielle distribuée (Distributed Artificial Intelligence : DAI) ou de l'intelligence artificielle traditionnelle (Artificial Intelligence : AI).

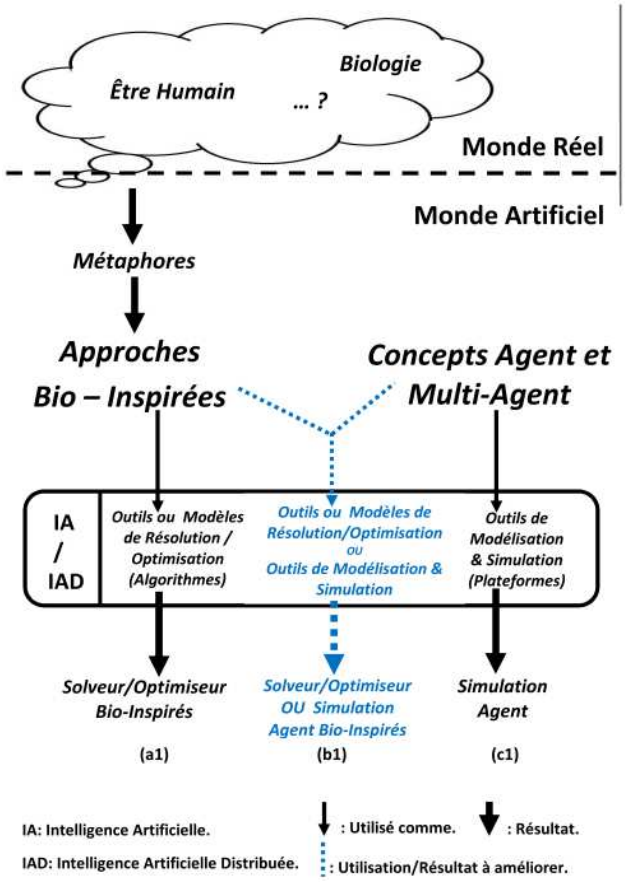


Figure 4.3 Utilisation des approches bio-inspirées dans le domaine IA / IAD combinées ou non avec les SMAs

Le cas (a1), illustre l'utilisation d'outils de résolution / optimisation bio-inspirés (Algorithmes: modèles de calcul / mémorisation ou outils de résolution / optimisation) pour résoudre des problèmes. Tous les exemples et applications cités dans le Chapitre 2 appartiennent à ce cas (sauf s'il a été mentionné l'utilisation d'agent).

Le cas (b1) illustre l'utilisation d'outils de modélisation / simulation multi-agents (plateformes) multi-agents bio-inspirés pour modéliser / simuler des systèmes multi-agents bio-inspirés. Par exemple, l'utilisation de l'outil *Turtlekit* dans la plateforme *Madkit* (Gutknecht 00) pour simuler des systèmes de vie artificielle / réactifs et l'utilisation de la plateforme *Repast* pour simuler des applications en sciences sociales (Florin 15). Il peut également illustrer l'utilisation d'outils et de modèles agents / multi-agents bio-inspirés (Algorithmes) tels que (Broecker 15, Fink 14, Olaifa 15, Tsang 05, Xiang 08).

Le cas (c1) illustre l'utilisation d'outils de modélisation / simulation multi-agent (plates-formes) pour modéliser et simuler des systèmes multi-agents. *Gama*, *NetLogo* et *PRESAGE2* sont des exemples de plateformes de simulation d'agents encore utilisées (Florin 15).

Par ailleurs, la Figure 4.4 décrit l'utilisation combinée / séparée d'approches bio-inspirées et de concepts agents / multi-agents dans le domaine de l'ingénierie logicielle (Software Engineering : *SE*).

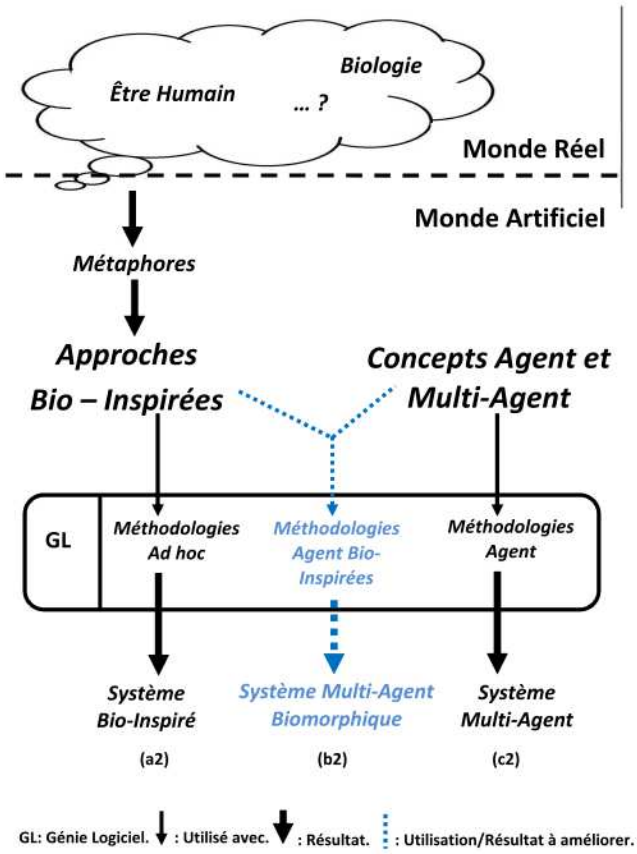


Figure 4.4 Utilisation des approches bio-inspirées dans le domaine GL combinées ou non avec les SMAs

Le cas (a2) illustre l'utilisation de méthodologies ad hoc bio-inspirées (modèles / processus / outils) pour développer des systèmes bio-inspirés. Cela concerne la plupart des systèmes bio-inspirés existants.

Le cas (b2) illustre l'utilisation de méthodologies agent / multi-agent bio-inspirées (modèles / processus / outils) pour développer des systèmes multi-agents bio-inspirés. Il n'y a pas de modèle ni de méthodologie pour traiter ce cas (Gengan 14, Gonçalves 13, Lee 09, Mochalov 15, Perez-Diaz 16, Qian 16). Mise à part des méthodologies supportant une caractéristique cible unique (par exemple, la méthodologie agent *Adelfe* (Bernon 02, 04) prend en charge les fonctionnalités émergentes dans les *SMAs* adaptatifs).

C'est *le cas qui a besoin d'amélioration, et où nous visons à contribuer dans cette thèse. Une telle amélioration aura des conséquences positives sur le cas (b1) et vice versa.*

Le cas (c2) illustre l'utilisation de méthodologies Agent / multi-agents (modèles / processus / outils) pour développer des systèmes multi-agents. Par exemple, nous pouvons citer les modèles organisationnels *AGR* et *AGRE* (Ferber 99, Ferber 04).

Pour les méthodologies nous avons, par exemple: *Gaia*, *MaSE*, *O-MaSE*, *Passi*, *Prometheus*, *INGENIAS*, *Tropos* (Florin 15, Padmanaban 16, Sturm 14a). Quelques exemples de leur application peuvent être trouvés dans (Manate 14, Massawe 09, Silva 10), et des exemples qui ne mentionnent aucune méthodologie (Kosakaya 16, Rehberger 16, Xiang 08).

5. Le formalisme unificateur

L'idée d'utiliser un formalisme unificateur pour faire face à la diversité des concepts spécifiques pour les paradigmes bio-inspirés considérés est devenue évidente. Il s'agit d'un contexte de développement multi-paradigmes. Plutôt que de proposer une approche qui soit la somme des différents concepts bio-inspirés, ou d'essayer de fusionner des concepts similaires, notre vision d'un formalisme unificateur est d'envelopper les différents concepts par des concepts de base et d'opérer, par la suite, des raffinements successifs selon les contextes spécifiques à chaque paradigme bio-inspiré.

5.1. Adéquation de l'approche agent au développement de systèmes naturels

Les *SMAs* profitent de l'effort d'une large communauté scientifique misant sur le fait que leur approche s'accommode à différents niveaux d'abstraction. En effet, des agents cognitifs complexes aux agents réactifs très simples, il est possible de modéliser des réalités très variées.

Beaucoup d'arguments ont été donnés, en faveur de l'utilisation des approches orientées-agent pour le développement des systèmes complexes (Jennings 01). Dans l'optique de prendre en charge cette complexité, l'ingénierie orientée-agent fournit les structures, les techniques et les outils fondamentaux, se référant principalement à la décomposition, l'abstraction et l'organisation (Jennings 01).

a. Mérites des décompositions orientées-agents

Limitant le champ et la portée du concepteur, la décomposition est la technique de base qui aide à contrer les gros problèmes et leur complexité, par la division de ces derniers en parties plus petites, maniables et traitables de manière relativement séparée.

Il est apparent que la façon naturelle de modéliser un système complexe c'est en fonction de plusieurs composant autonomes qui peuvent agir et interagir d'une manière flexible pour réaliser leur ensemble d'objectifs ; L'approche orientée agent est, tout simplement, la mieux adaptée.

b. La convenance des abstractions orientées-agents

Limitant à un moment donné, l'intérêt et le champ de vision du concepteur, le processus de définir un modèle simplifié du système, aide à contrer sa complexité, en mettant l'accent sur quelques détails et ignorer d'autres.

Dans le cas des systèmes complexes composés de sous-systèmes, de composants de sous-systèmes et de relations organisationnelles, il sera naturel de faire correspondre les sous-systèmes à des organisations d'agents, les composants des sous-systèmes à des agents et l'interaction entre les sous-systèmes et entre leurs composants sera visualisée en termes d'interactions sociales de haut niveau.

Cette vue s'accorde précisément avec le traitement "au niveau connaissance" des interactions offert par le paradigme orienté-agent. Les systèmes agents sont inévitablement décrits en termes de "coopération pour atteindre des objectifs communs", "coordination de leurs actions" ou "négociation pour résoudre des conflits".

c. La nécessité d'une gestion flexible de l'évolution des structures organisationnelles

Offrant la capacité de préciser et d'adopter des relations organisationnelles, le processus de définition et de gestion des interactions entre les divers composants de résolution de problème (les sous-systèmes et les liens d'interaction), aide les concepteurs à faire face à la complexité en permettant de regrouper des composants, de les traiter comme unité d'analyse de haut niveau et de fournir les moyens de description des relations de haut niveau entre divers unités.

Les systèmes orientés-agents ont des mécanismes de calcul simultanés pour former, maintenir et dissoudre des organisations de manière flexible. Cette force de représentation permet aux systèmes d'agents d'exploiter deux aspects de la nature des systèmes complexes.

Les systèmes multi-agents constituent aujourd'hui une technologie pour la conception et le contrôle de systèmes complexes, flexibles et évolutifs (Azaiez 07).

5.2. L'environnement dans les systèmes multi-agents biomorphiques

Dans le modèle voyelles *AEIO* (Da Silva 02); *Da Silva* distingue quatre dimensions pour un *SMA*: *Agent*, *Environnement*, *Interaction* et *Organisation*. Nous remarquons que le composant *environnement* a été identifié comme un élément clé pour un *SMA* (Weyns 06). Pour les systèmes bio-inspirés, ce composant est d'une importance vitale. C'est l'endroit où les agents doivent coexister et interagir avec la possibilité de former, maintenir et dissoudre des organisations. Tous ces changements ne peuvent avoir lieu qu'à travers l'environnement (Weyns 06).

Parunak (Parunak 97) insiste sur une réelle prise en compte de l'environnement pour les *SMAs* "*biomorphiques*". Dans ce contexte, il établit qu'un tel système peut être défini comme trois composantes:

SMA = {Agents, Environnement, Couplage}

Où un $Agent_i$ est un ensemble de quatre éléments comme suit:

$$\boldsymbol{Agent_i = \{A.état_i, A.entrée_i, A.sortie_i, A.processus_i\}}$$

L'$Environnement_i$ (en tant que portée d'un $Agent_i$) est composé de deux éléments:

$$\boldsymbol{Environnement_i =< E.état_i, E.processus_i >}$$

La nature exacte du *couplage* dépend de la façon dont nous modélisons les agents et les états et processus de l'environnement. Ce *couplage* peut être très complexe. Lorsque les agents et l'environnement sont des événements discrets, le couplage de $A.entrée_i$ et $A.sortie_i$ avec $E.état_i$ est simplement une correspondance des états des agents et de l'environnement. Ce type de représentations, dominant dans le domaine de l'intelligence artificielle, est critiqué car il génère des situations incohérentes.

Évidemment, l'autonomie des entités et la simultanéité de leurs actions est cruciale pour les *SMAs* biomorphiques. Une validation directe des actions doit, donc, être évitée dans de telles approches. En respect de ces exigences, nous proposons l'utilisation du principe *Influence/Réaction* lorsqu'on a affaire aux *systèmes multi-agents biomorphiques*.

6. Conclusion

Dans ce Chapitre, nous avons présenté une synthèse et une réflexion sur l'état de l'art concernant les systèmes biologiques et multi-agents. Nous avons mis en évidence les prérequis et les conséquences de la confrontation des concepts biologiques et agents.

Il est à noter que les solutions de développement offertes par l'approche agent sont prometteuses et permettent d'obtenir des systèmes flexibles et évolutifs. Cependant, leur mise en œuvre reste problématique. Ceci est dû au manque de standardisation des techniques d'ingénierie adaptées à ce genre de systèmes.

Notre objectif est, donc, de donner les prémisses d'une méthodologie basée-agent pour l'ingénierie des systèmes bio-inspirés, via un modèle et une démarche orientée-agent, dans un contexte multi-paradigme. Ce qui fera l'objet du Chapitre suivant.

CHAPITRE 5

MODELISATION MULTI-PARADIGMES DE SYSTEMES MULTI-AGENTS BIOMORPHIQUES

CHAPITRE 5

MODELISATION MULTI-PARADIGMES DE SYSTEMES MULTI-AGENTS BIOMORPHIQUES

1. Introduction

Les enjeux d'une vision multi-paradigmes d'un système, résident dans la démarche de développement de ce système. La démarche inclut principalement des processus qui peuvent être manuels ou automatisés et des artefacts de modélisation (ou modèles). La démarche de développement des systèmes biomorphiques a comme objectif le développement d'un système moyennant divers paradigmes bio-inspirés. Dans le contexte d'une conception bio-inspiré, notre objectif est d'utiliser un modèle générique pour unifier la diversité des concepts spécifiques aux paradigmes biologiques considérés.

2. Modélisation Bio-IR

Il est question dans cette section d'utiliser les réflexions établies dans le Chapitre précédent pour proposer notre modélisation de paradigmes biologiques basée agent et utilisant le principe Influence/Réaction pour la gestion des interactions et la prise en charge des actions simultanées des agents.

En effet, et comme récapitulative, outre le fait que les *SMAs*, à l'instar des systèmes naturels, considèrent que les systèmes sont composés d'entités en interaction, il existe une grande similitude dans les critères de caractérisation des approches bio-inspirées, *SMA* et *SMA* bio-inspirées. Nous optons à classer ces caractéristiques en deux catégories: les caractéristiques intra-entité et inter-entités. En d'autres termes, nous caractérisons les entités prises séparément (atomique, se référant à des individus) et nous caractérisons leurs interactions (composées, se référant à un agrégat d'individus).

Nous remarquons que le même fait a été établi pour la classification des paradigmes bio-inspirés.

2.1. Modélisation structurelle de système multi-agents biomorphique

Nous proposons une modélisation supportant plusieurs paradigmes bio-inspirés simultanément (Figure 5.1). Le méta-modèle est considéré comme un système multi-agent multi-paradigme bio-inspiré avec un agent biomorphique et un groupe biomorphique. Nous remarquons qu'il inclut les six paradigmes bio-inspirés cités, à titre d'exemple, dans le Chapitre 2.

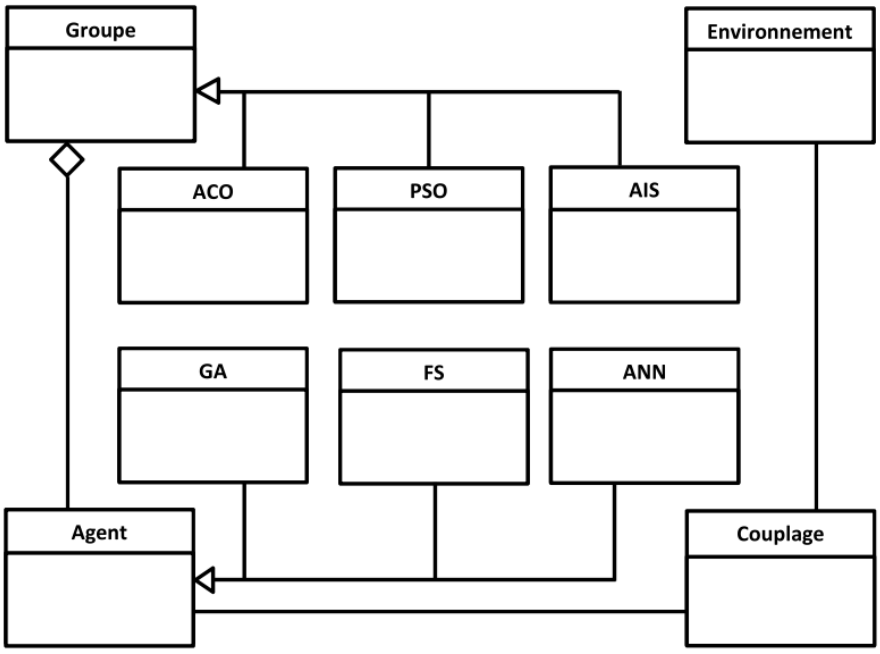


Figure 5.1 Méta-modèle multi-paradigmes de système multi-agents bio-inspiré (Zeghida 18a)

Pour un nouveau paradigme bio-inspiré, nous devons le classer au niveau micro / macro. Pour cela nous suivrons les recommandations du Tableau 4.4 (Chapitre précédant), selon la nature de la métaphore bio-inspirée, son utilisation particulière et le niveau d'abstraction nécessaire. S'il appartient à un niveau macro, nous l'ajoutons comme une spécialisation du groupe (héritage). Sinon, il sera ajouté en tant que spécialisation de l'agent (étant au niveau micro).

La nature complexe des systèmes biomorphiques se manifeste par différents aspects, allant du simple calcul ou optimisation, à la coordination complexe et aux résolutions symboliques. En utilisant un *SMA* pour prendre en charges tous ces aspects dans un contexte multi-paradigme, nous identifions trois scénarios possibles:

1. Approche intra-agent: où l'agent encapsule un traitement selon un paradigme bio-inspiré donné (comme un modèle de calcul par exemple). Le système est vu comme une agrégation d'agents biomorphiques. Ce scénario a l'avantage d'encapsuler la diversité des paradigmes dans les agents, ce qui est intéressant en termes de développement: division du travail entre équipes (c'est le cas d'une modélisation avec des agents biomorphiques et sans groupes biomorphiques, Figure 5.2);

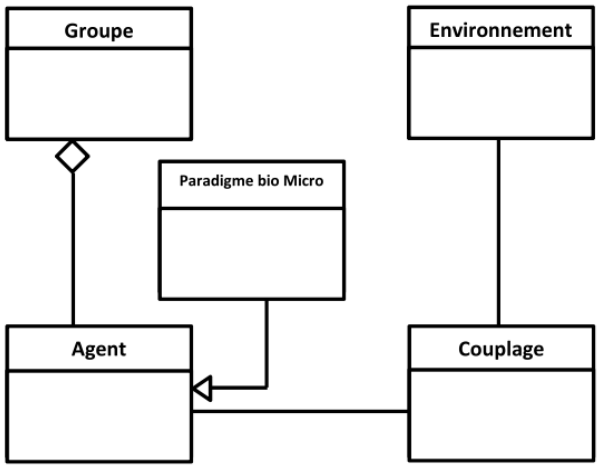


Figure 5.2 Méta-modèle de système multi-agents avec agent bio-inspiré (Zeghida 18a)

2. Approche inter-agents: Lorsque l'aspect bio-inspiré apparaît à travers les interactions des agents (c'est-à-dire le *SMA*), nous convergeons vers un comportement de groupe biomorphique avec des agents non biomorphiques (Figure 5.3);

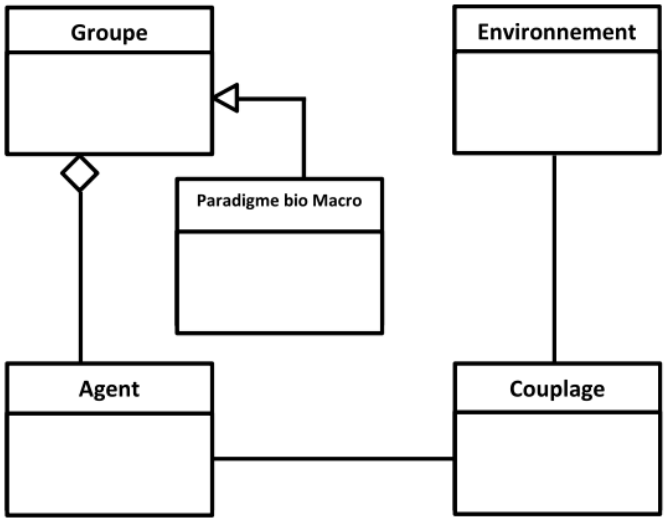


Figure 5.3 Méta-modèle de système multi-agents avec groupe bio-inspiré (Zeghida 18a)

3. Approche hybride: où les deux scénarios précédents sont combinés. Le système est alors vu comme un groupe biomorphique d'agents biomorphiques (le cas d'une

modélisation avec des groupes biomorphiques et des agents biomorphiques, Figure 5.4).

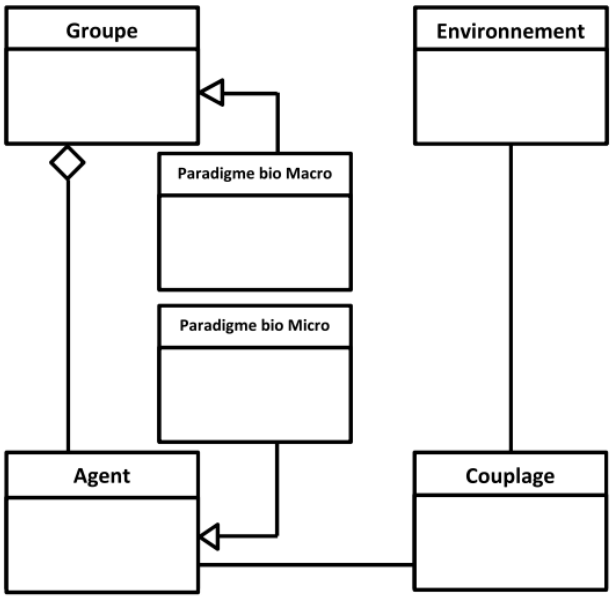


Figure 5.4 Méta-modèle de système multi-agents avec agent et groupe bio-inspirés (Zeghida 18a)

Nous soulignons que, dans notre modèle, il n'y a pas de contraintes sur le type / l'architecture de l'agent. Si la bio-inspiration est prise en charge à un niveau micro, l'agent sera cognitif selon l'approche bio-inspirée qu'il détient et son module de calcul doit être, par conséquent, sophistiqué. Si la bio-inspiration est prise en charge à un niveau macro, l'agent est généralement réactif.

2.2. Modélisation des interactions dans un système multi-agents biomorphique

Le couplage (Coupling) dans les modèles des systèmes biomorphiques présentés précédemment est pris en charge par le principe Influence / Réaction. Formellement et à un niveau d'abstraction plus élevé, en *SMA* biomorphique, les trois scénarios précédents seront reflétés dans deux niveaux comme suit:

- *Niveau agent*

Nous utilisons un modèle d'agent qui doit supporter la dimension biologique; il sera conçu en assurant une *autonomie* réelle avec la séparation entre les variables d'état du système décisionnel (son esprit) et la composante physique (son corps). Ces agents interagissant peuvent être structurés en groupes.

- Niveau groupe

Le système résultant est un système *ouvert* d'agrégat d'agents en *interaction*. Ces interactions seront gérées par un module d'interaction distinct. Nous soulignons le caractère actif de l'environnement à modéliser explicitement. Cette caractéristique est due au fait que l'environnement possède son propre processus et qu'il peut changer son état, indépendamment des actions de ses agents. Les états de divers agents sont couplés à l'état de l'environnement, ce couplage sera effectué en utilisant le principe Influence/Réaction.

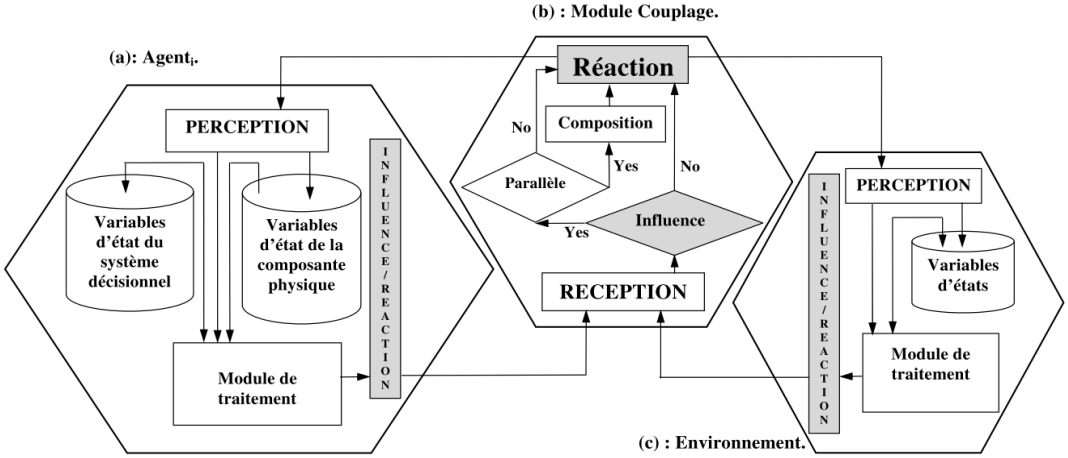


Figure 5.5 Bio-IR-M: Modélisation Influence/Réaction bio-inspirée; (a) Bio-IR-Agent, (b) Bio-IR-Couplage, (c) Bio-IR-Environnement (Zeghida 18a)

Les flèches sortantes d'une base de données sont des lectures, les entrantes sont des mises à jour.

Nous modélisons un système Influence/Réaction multi-agent biomorphique comme suit (Figure 5.5):

Bio.IR.M = {{Bio.IR.A}, Bio.IR.E, Bio.IR.C},

Où:

1. Bio-IR-A : c'est le composant Agent ; un agent n'a pas de contrôle direct sur le résultat de ses influences sur l'environnement, y compris sur ses variables d'état de composant physique. L'agent doit émettre des influences vers le module d'interaction. Mais au contraire, l'agent peut utiliser et modifier ses variables d'état du système décisionnel, ses variables d'état de composant physique peuvent être modifiées par un composant externe (comme une réaction au composant environnement par exemple) (Figure 5.5.a).

2. Bio-IR-C: c'est le composant Couplage ; le module de couplage gère les interactions par la composition des influences simultanées agent/environnement et ensuite il transmet le résultat au composant environnement/agent (Figure 5.5.b).

3. Bio-IR-E : c'est le composant Environnement ; en tant que composant actif, l'environnement réagit (par sa propre influence) aux influences des agents en fonction de ses propres processus et état. L'environnement peut non seulement utiliser et modifier ses variables d'état, mais également modifier les variables d'état physique du composant agent via le module de couplage (Figure 5.5.c). Cependant, l'environnement ne peut pas atteindre les variables du système décisionnel de l'agent (pour ne pas enfreindre son autonomie).

Ce modèle peut préserver l'intégrité des agents en séparant leurs variables d'état. Les variables du système décisionnel sont accédées/modifiées seulement par l'agent pendant la phase influence. Les variables des composants physiques font partie de l'environnement et ne sont modifiées que par l'environnement pendant la phase réaction.

La réaction de l'agent/environnement est dans notre cas une influence désirée sur l'environnement/agent et n'est plus une action traditionnelle, au sens de l'intelligence artificielle.

Même si le principe Influence/Réaction n'affecte pas l'action simultanée et la modélisation de l'interaction, ce principe améliore les mécanismes de diffusion de l'information pour augmenter la réactivité du système (Zeghida 18b).

À cette fin, nous résumons les principales caractéristiques de notre proposition dans:

1. L'application du principe Influence/Réaction.

- Capable de modéliser les comportements simultanés et joints.
- Abandonner la représentation de l'action comme une modification de l'état global du système.
- Améliorer les mécanismes de diffusion de l'information pour augmenter la réactivité des agents.

2. Isoler un module d'interaction (le module de couplage). Utilisez toutes les influences produites à un moment donné pour calculer le nouvel état du monde.

3. La garantie de l'intégrité de l'agent (autonomie) par la distinction entre les variables d'état du système décisionnel d'un agent et les variables concernant son aspect physique.

4. La modélisation explicite de l'environnement.

3. Études de cas

Nous prenons, dans une première étude de cas, l'utilisation d'un algorithme à fourmis appliqué au célèbre problème du voyageur de commerce (Travelling Salesman Problem : *TSP*), avec deux illustrations différentes.

La Figure 5.6 illustre la modélisation d'un système à fourmis TSP, selon notre modèle et en utilisant un modèle organisationnel *AGR* étendu et adapté ou un modèle organisationnel *AGRE* adapté (Ferber 04), où nous optons pour:

- Une considération particulière pour l'environnement;
- Un double cercle pour l'aspect bio-inspiré.

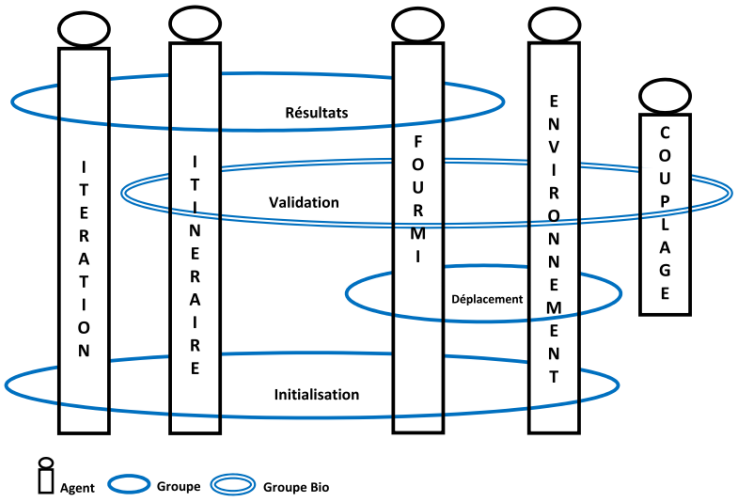


Figure 5.6 Modélisation Influence/Réaction bio-inspirée du TSP avec un groupe biomorphique ACO (Zeghida 18a)

Dans ce cas, nous avons une bio-inspiration de niveau macro représentée par le groupe biomorphique "*Validation*", mettant en œuvre l'approche d'optimisation par colonies de fourmis (*ACO*) pour trouver le circuit le plus court de villes. Les agents fourmis dans cette implémentation utilisent la probabilité en fonction de la distance et de la densité de phéromone sur chaque chemin entre les villes pour choisir la prochaine ville de déplacement (le méta-modèle correspondant est donné par la Figure 5.7).

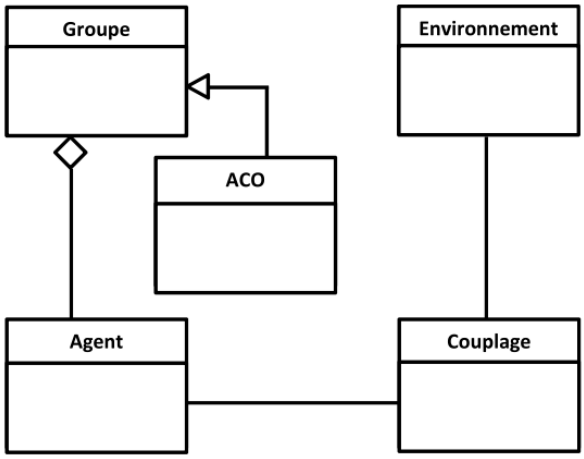


Figure 5.7 Méta-modèle d'un TSP avec un groupe biomorphique ACO (Zeghida 18a)

La Figure 5.8 montre la modélisation d'un système à fourmis *TSP* avec une bio-inspiration macro: un groupe biomorphique "Validation", mettant en œuvre l'approche *ACO* et une bio-inspiration micro: un agent biomorphique "Fourmi", utilisant par exemple, comme modèle de calcul, un algorithme génétique (*GA*) pour choisir la prochaine ville de déplacement (son méta-modèle est présenté par la Figure 5.9).

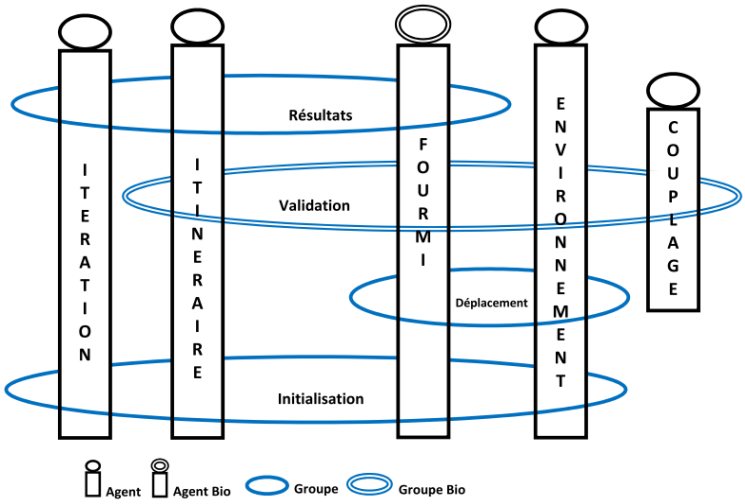


Figure 5.8 Modélisation Influence /Réaction bio-inspirée du TSP avec un agent biomorphique GA et un groupe biomorphique ACO (Zeghida 18a)

Dans les deux cas, le couplage est réalisé par le principe influence/réaction. L'environnement peut être vu comme un graphe, où les nœuds sont des villes et les arcs/poids sont des chemins/distances entre les villes. Une implémentation sur la plate-forme *Jade* pour le premier cas peut être trouvée dans (zeghida 18b) où nous comparons les trois variantes de base du système à fourmis (Ant System : *AS*): Ant-Cycle, Ant-Density et Ant-Quantity (Dorigo 91, 97).

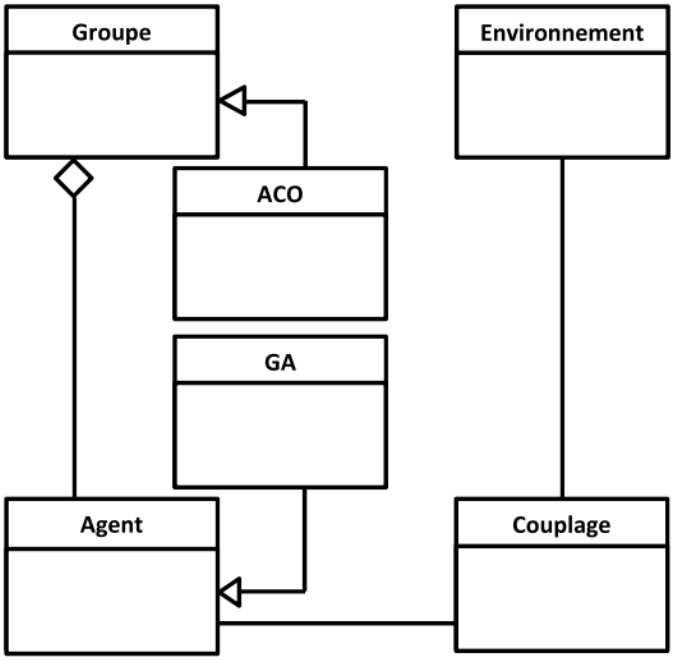


Figure 5.9 Méta-modèle d'un TSP avec un agent biomorphique GA et un groupe biomorphique ACO (Zeghida 18a)

Les résultats obtenus sont prometteuses dans les domaines du génie logiciel et de l'intelligence artificielle distribuée. Cela nous encourage à investiguer les variantes améliorées de *AS*, comme le max-min ant system (Stützle 98) et à explorer d'autres aspects en utilisant les plates-formes *Jade* et *Madkit* dans une perspective de proposer notre algorithme à fourmis amélioré.

Une troisième étude de cas concerne le problème d'un emploi du temps (Time Tabling Problem : *TTP*) résolu avec un algorithme à fourmis aussi. La Figure 5.6 et la Figure 5.7 peuvent illustrer respectivement la modélisation fourmis *TTP* et son méta-modèle. Dans ce cas, l'environnement est un graphe, où les nœuds sont les extrémités des sessions (début/fin); les arcs et leurs poids sont la durée et les classes/salles de classe. Par conséquent, les fourmis (enseignants) exécutent un processus adapté au contexte.

Un autre cas d'étude traite le *TTP*, en utilisant une optimisation par loup gris (Gray Wolf Optimization : *GWO*) (Mirjalili 14). Dans ce cas, il suffit de remplacer *Fourmi* par *Loup* dans la Figure 5.6 et *ACO* par *GWO* dans la Figure 5.7 pour illustrer respectivement la modélisation du système d'optimisation par loup gris *TTP* et son méta-modèle.

4. Travaux similaires

Divers exemples de systèmes multi-agents bio-inspirés ont été développés et sont pour la plupart des travaux spécifiques. Cette spécificité consiste dans le fait de cibler une seule fonctionnalité bio-inspirée ou d'être conçus à l'aide d'un processus ou d'une "méthodologie" ad hoc.

(Bernon 02, 04) présente un premier exemple de méthodologie agent dédiée ; la méthodologie *ADELFE*. Cette dernière se consacre à la conception de systèmes multi-agents adaptatifs et coopératifs et s'appuie sur la théorie des systèmes multi-agents adaptatifs ; les *AMAS* : "Adaptive Multi-Agent Systems" (Georgé 03). Cette methodologie semble être un bon exemple de méthodologie manipulant une classe de systèmes biomorphiques caractérisés par une intelligence d'essaims.

Un deuxième exemple est tiré de l'ingénierie de l'auto-organisation dans les systèmes multi-agents. Inspiré d'organismes multicellulaires, *Nagpal* dans (Nagpal 03) donne un ensemble de primitives d'ingénierie bio-inspirées en robotique.

Dans (Brun 08), l'auteur donne un autre exemple pour la construction de systèmes auto-adaptatifs bio-inspirés; il vise des systèmes logiciels particuliers et présente l'utilisation de styles architecturaux dans une perspective d'architecture logicielle appliquée à des problèmes de caractéristiques communes. Cela consiste principalement à créer un modèle pour un système biologique donné. Ce modèle doit être étudié jusqu'à ce qu'il soit complètement compris. Après cela, dans un cycle itératif, les concepteurs construisent sur ce modèle initial le système biologique cible. Un cas concret a été donné pour un problème de distribution discrète: distribuer un calcul sur un grand réseau, où chaque petit groupe de nœuds ignore le problème qu'ils aident à résoudre.

Ces travaux, parmi d'autres, restent spécifiques à des domaines et à des catégories de problèmes particuliers et ne soutiennent ni encouragent guère la réutilisation.

À l'opposé et avec une vision plus générale, des directives utiles pour une meilleure définition et caractérisation d'un *SMA* biomorphique ont été données dans (Parunak 97, Weyns 06), encourageant une bio-inspiration avancée pouvant conduire à un processus générique respectant les considérations de notre sujet de thèse.

Un autre travail suggère l'extension du modèle organisationnel *AGR* (Agent, Rôle et Groupe) (Ferber 98), ce qui donne lieu au modèle *AGRE* (Ferber 04). *AGRE* comprend

la dimension environnementale et rejoint notre vision au sujet du développement de système multi-agents biomorphique.

Dans (Zambonelli 15a, 15b), les auteurs présentent un framework multi-agents général appelé *SAPERE* (Self-aware Pervasive Service Ecosystems) (écosystème de services auto-conscients). *SAPERE* traite des systèmes "pervasives" considérés comme un écosystème où les services informatiques "pervasives" sont supposés être des systèmes multi-agents. Leur contribution vise à réaliser les interactions entre ces services (*SMA*) selon des lois bio-inspirées.

Dans notre cas, nous traitons des systèmes naturels, vues comme un biotope ou un écosystème (Système de systèmes en interaction), avec une approche de modélisation multi-paradigmes. Ces systèmes en interaction implémentent un paradigme bio-inspiré donné et l'interaction entre eux peut être, elle-même, régie par un paradigme bio-inspiré aussi. Notre contribution vise à modéliser ces systèmes en interaction et leur interaction avec des systèmes multi-agents utilisant le principe Influence/Réaction.

Nous utilisons donc tous deux, mais à différents niveaux, des concepts et des terminologies communs:

- ✓ Ils abordent, dans un écosystème et avec une approche bio-inspirée, le problème de l'interaction entre ses systèmes supposés multi-agents ;
- ✓ Nous abordons, dans un écosystème et avec une approche multi-agents, le sujet de la modélisation de ses systèmes et leur interaction qui sont, tous deux bio-inspirées.

Leur travail se rapproche du notre dans le sens où il peut être considéré comme un exemple d'application idéal de notre modèle, si leurs services informatiques "pervasives" étaient tous bio-inspirés avec un modèle d'interaction Influence/Réaction.

Dans (Mariani 16), les auteurs permettent aux agents, dans les technologies *SMA*, d'adopter de manière dynamique un type d'interaction entre plusieurs différents types possibles. Concrètement, ils ont utilisé la plate-forme agents dédiée *TuCSoN* (Tuple Center Spread over the Network) avec les plates-formes *Jada* et *Jason*. *TuCSoN* utilise un langage de coordination logique (*ReSpecT*), qui est une bibliothèque *Java* permettant de modéliser la coordination dans des processus distribués (comme des agents autonomes, intelligents et mobiles).

L'idée est intéressante et peut être utilisée avec notre vision multi-paradigmes intégrant différents paradigmes bio-inspirés. Lorsque le paradigme bio-inspiré est pris en charge au niveau *micro* par un agent (qui doit être intelligent) ou au niveau macro par un *SMA* d'un petit nombre d'agents intelligents, l'idée en vaut la peine d'être explorée. Toutefois, lorsque le paradigme bio-inspiré est pris en charge au niveau macro par un *SMA* d'un grand nombre d'agents simples (non intelligents), l'idée sera moins utile.

5. Conclusion

Dans ce Chapitre, nous avons proposé un modèle basé agent Influence/Réaction générique qui intègre divers paradigmes bio-inspirés comme démarche de développement de systèmes biomorphiques.

Le Chapitre commence par la présentation de notre modèle incluant la structure et l'architecture générale d'un système multi-agents biomorphique et mettant en évidence le module Interaction qui utilise le principe Influence/Réaction. Il était, en suite, question d'une mise en œuvre conceptuelle à travers quelque cas d'application. Et cela se termine par une étude comparative pour se situer parmi des travaux similaires.

Nous considérons cette contribution comme un apport méthodologique pour le développement de *SMA* biomorphiques. L'aspect expérimental sera abordé dans le Chapitre suivant.

CHAPITRE 6

ETUDE EXPERIMENTALE

CHAPITRE 6

ETUDE EXPERIMENTALE

1. Introduction

Un grand nombre de solveurs/optimiseurs distribués peuvent être observés dans la nature usant de capacités individuelles très limitées et montrant un comportement collectif très complexe. Conséquence d'interactions coordonnées, le comportement d'une colonie de fourmis est très structuré. Cependant, les fourmis de cette colonie ont des capacités limitées de communication et d'action.

Le but de ce Chapitre est de donner une expérimentation de notre modèle Bio-IR M appliqué au problème du voyageur de commerce (TSP) avec une approche à fourmis. C'est le premier cas d'étude du Chapitre précédent. Sa modélisation est illustrée par la Figure 5.6 et son méta-modèle correspondant est donné par la Figure 5.7.

Cette étude expérimentale sera suivie d'une évaluation de la mise en œuvre d'une implémentation Agent des trois variantes de base de l'heuristique *AS* (Ant System) avec et sans l'application du Principe Influence/Réaction. Les valeurs ajoutées de ce principe et ses utilités supplémentaires seront démontrées à travers divers benchmarks de *TSP* symétrique.

2. L'heuristique Ant System

Pour faire face aux problèmes NP-difficiles d'optimisation combinatoire discrète (Combinatorial Optimization Problem : *COP*), il y eu recours à la méta-heuristique d'optimisation par colonie de fourmis (Ant Colony Optimizaton : *ACO*). Cela commença par l'utilisation de l'heuristique Ant System (*AS*) ou algorithme à fourmis (Ant Algorithm ; *AA*) appliquée au problème symétrique du voyageur de commerce (Travelling Salesman Problem : *TSP*).

L'heuristique *AS* est une approche basée population utilisant une Intelligence d'essaims (Swarm Intelligence : *SI*). Tout d'abord, trois implémentations de base ont été appliquées: Ant-Density, Ant-Quantity et Ant-Cycle. Selon leurs auteurs, la dernière variante réalise de meilleures performances que les deux premières.

2.1. Le problème du voyageur de commerce

Ce problème consiste en un vendeur ambulant qui cherche le tour le plus court pour visiter des villes sans se rendre deux fois dans la même ville et revenir à la ville de départ. La longueur du tour est le cumul des distances entre chaque paire de villes visitées. Les distances peuvent être réelles (géographiques) ou euclidiennes. Lorsque les chemins entre deux villes données (dans le cas de plusieurs chemins) sont égaux, le *TSP* est symétrique sinon (au moins deux chemins différents) le *TSP* est asymétrique (*ATSP* : Asymmetric Travelling Salesman Problem).

En théorie des graphes, ce problème équivaut à trouver un cycle Hamiltonien dans le graphe des villes; Les nœuds sont des villes, les arcs et leurs poids sont respectivement les chemins et les distances entre ces villes (Figure 6.1).

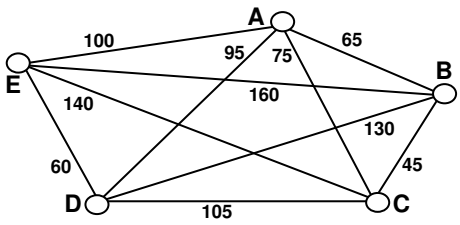


Figure 6.1 Exemple de graphe pour TSP

La Figure 6.1 montre un graphe *TSP* de cinq (05) villes: A, E, D, B et C. ABCEDA est un tour dans ce graphe, sa longueur est égale à: ABCEDA = AB + BC + CE + ED + DA = 65 + 45 + 140 + 60 + 95 = 405. AEDBCA est un autre tour, sa longueur est: DBCAED = DB + BC + CA + AE + ED = 130 + 45 + 75 + 100 + 60 = 410. Pour un voyageur de commerce, la question est de trouver le tour le plus court.

Dans un *TSP* symétrique, le nombre de solutions possibles pour un graphe avec n villes est: $(n-1)!/2$. Un algorithme exact ou déterministe ne peut être utilisé que pour un petit nombre de villes. Par conséquent, seules les heuristiques sont utilisées pour résoudre les grands *TSP*.

2.2. Motivations et expérimentations

Les fourmis peuvent trouver le chemin le plus court entre une source de nourriture et leur nid (Figure 6.2). Dans des conditions spécifiques (Colorni 91), si nous avons affaire à des systèmes hautement distribués avec des individus de calcul très limités, nous devons simplement imiter les fourmis.

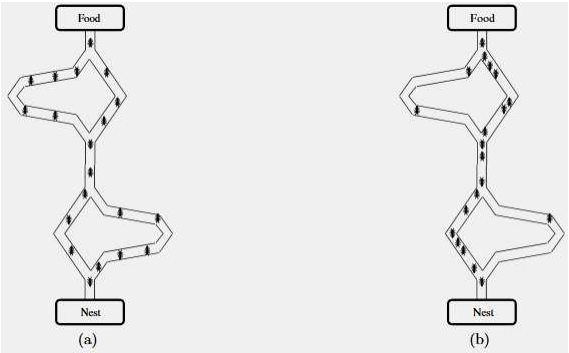


Figure 6.2 Le processus auto-catalytique des fourmis (Dréo 06)

Les fourmis interagissent dans une communication de base indirecte et locale appelée S*tigmergie*. Ils agissent avec des capacités de calcul limitées: lors du déplacement, chaque fourmi laisse derrière elle de la *phéromone* pour aider les autres fourmis à choisir la voie la plus "phéromonée" (Figure 6.2). Ce processus conduit à une boucle de renforcement positive et crée un comportement collectif auto-catalytique émergeant (Dorigo 91, 96, 97, 99). Ceci définit la base d'un algorithme distribué pour résoudre le *TSP* en tant que *COP* discret NP-dur. Dans ce cas, la fonctionnalité émergente est de trouver le tour de villes visitées le plus court, où les fourmis n'ont qu'à se déplacer et à laisser de la phéromone; ils ignorent faire de l'optimisation puisqu'ils n'étaient pas explicitement programmées pour.

2.3. Les Algorithmes à fourmis de base

Les auteurs de cette heuristique traitent un *TSP* symétrique avec une distance euclidienne et utilisent les notations suivantes :

- n : le nombre de villes ;
- m : le nombre de fourmis ;
- $Tabulist_k$ mémorise pour la fourmi Ant_k les villes déjà visitées jusqu'à compléter un cycle (tour) et évite le passage deux fois par la même ville ;

- d_{ij}, distance euclidienne, entre deux villes T_i et T_j, calculée par la formule: $d_{ij} = \sqrt{\left(x_1^i - x_1^j\right)^2 + \left(x_2^i - x_2^j\right)^2}$;

- L'intensité de phéromone sur le chemin $Path_{ij}$ à l'instant $t+1$ est calculée par l'équation 1 :

$$\tau_{ij}(t+1) = \rho \,.\, \tau_{ij}(t) + \Delta\tau_{ij}(t, t+1) \qquad (1)$$

Où:

- ρ est le coefficient d'évaporation de phéromone,
- La quantité de phéromone laissée sur le chemin $Path_{ij}$ par la fourmi Ant_k entre t et $t+1$ est donnée par : $\Delta\tau_{ij}(t, t+1) = \sum_{k=1}^{m} \Delta\tau_{ij}(t, t+1)$,

A l'instant t=0, on peut déposer une quantité de phéromone initiale égale à: $\tau_{ij}(0) = Q_c$;

- $\eta_{ij} = 1 / d_{ij}$, représente la visibilité, d_{ij} la distance entre les villes T_i et T_j ;

- Pour choisir la ville de déplacement, la fourmi Ant_k utilise la probabilité de transition entre les villes T_i et T_j calculée par l'équation 2:

$$P_{ij}^k(t) = \frac{\tau_{ij}^{\alpha}(t).\eta_{ij}^{\beta}}{\sum_{k \in Allowed_k} \tau_{ik}^{\alpha}(t).\eta_{ik}^{\beta}} \qquad (2)$$

$Allowed_k = n - Tabulist_k$ (les villes pas encore visitées par la fourmi Ant_k), α et β deux paramètres pour contrôler l'importance relative des traces de phéromone versus visibilité.

Sans aucun modèle mathématique de référence pour tester et évaluer l'influence de divers paramètres pour mettre en œuvre *AS*, de nombreuses questions ouvertes peuvent seulement être résolues par une étude expérimentale (Dorigo 91, 96, 97, 99). Les plus importantes sont:

1. Quand mettre à jour la phéromone? Après un simple déplacement entre deux villes ou après un tour complet (L_k: longueur de tour de la fourmi Ant_k)?

2. Quelle est l'importance de la distance (d_{ij} / L_k) entre les villes? Utilisez cette distance avec la densité de phéromone Q pour calculer sa mise à jour ou non?

3. Comment pouvons-nous choisir les paramètres **m**, ρ, α, β, Q_c et $Q_{variant}$? Quel ensemble combiné de leurs valeurs aura un effet positif sur le résultat?

Si les auteurs d'*AS* (*Dorigo et al.*) choisissent de mettre à jour après chaque déplacement simple entre deux villes, deux modèles ont été utilisés dans la pratique, l'un considère la distance entre les villes et l'autre non.

Dans le premier modèle, une fourmi Ant_k allant de T_i à T_j dépose une quantité Q_1 de phéromone sur le chemin $Path_{ij}$, c'est la variante *Ant-Densité* (Algorithme 1), entre t et $t+1$, la quantité de de phéromone à déposer est donnée par l'équation 3:

$$\Delta\tau_{ij}^{k}(t,t+1)=Q_1 \qquad (3)$$

Algorithm 1. Ant-Density / Quantity Algorithm

```
Initialization                                          \t=0
for k=1 to m do                                         \m Ant
 Get Random i                                           \i=1, .., n
 Tabulist_k[1]=T_i                                      \Ant_k Start Town T_i
end for
for t=1 to t_max do                                     \t_max iterations
 for k=1 to m do                                        \each Ant_k
  for i=2 to n do                                       \Perform a tour
   Select T_j, using equation (2)                       \Best P_ij
   Tabulist_k[i]=T_j                                    \Ant_k next Town
   Update τ_ij, using equations (1) and (3)/(4)         \for a single move
  end for
 end for
 Get shortest tour among the m tours
end for
Get shortest tour among the t_max tours
```

Dans le deuxième modèle, une quantité Q_2 de phéromone est laissée sur le chemin $Path_{ij}$ chaque fois qu'une fourmi Ant_k passe de T_i à T_j pour chaque unité de longueur, c'est la variante *Ant-Quantité* (Algorithme 1). Entre t et $t+1$, la quantité de de phéromone à déposer est calculée par l'équation 4:

$$\Delta\tau_{ij}^{k}(t,t+1)=Q_2 / d_{ij} \qquad (4)$$

Algorithm 2. Ant-Cycle Algorithm

```
Initialization                                          \t=0
for k=1 to m do                                         \m Ant
 Get Random i                                           \i=1, .., n
 Tabulist_k[1]=T_i                                      \Ant_k Start Town T_i
end for
for t=1 to t_max do                                     \t_max iterations
 for k=1 to m do                                        \each Ant_k
  for i=2 to n do                                       \Perform a tour
   Select T_j, using equation (2)                       \Best P_ij
   Tabulist_k[i]=T_j                                    \Ant_k next Town
  end for
   Update τ_ij, using equations (5) and (6)             \entire Ant_k tour
 end for
 Get shortest tour among the m tours
end for
Get shortest tour among the t_max tours
```

Si *Dorigo et al.* choisissent de mettre à jour après une tournée entière, ils introduisent une différence majeure par rapport aux deux modèles précédents et définissent le troisième modèle.

Ce modèle définit la variante *Ant-Cycle* (Algorithme 2), si Ant_k passe de T_i à T_j entre t et $t+n$ (L_k est la longueur du tour de la fourmi Ant_k), en utilisant une quantité Q_3, le dépôt de phéromones n'est pas effectué à chaque déplacement ou étape, mais après un tour complet (n étapes) en utilisant l'équation 5:

$$\Delta\tau_{ij}^{k}(t, t+1) = Q_3 / L_k \qquad (5)$$

L'équation 1 devient l'équation 6:

$$\tau_{ij}(t+n) = \rho \,.\, \tau_{ij}(t) + \Delta\tau_{ij}(t, t+n) \qquad (6)$$

2.4. Les résultats de *Dorigo* et al.

L'étude expérimentale permet aux auteurs de cette heuristique de prendre les choix et les décisions suivantes :

- Ne pas utiliser un nombre élevé de fourmis, le prendre exactement égal au nombre de villes : $n = m$;
- Le dispatching aléatoire est sans effet, mettre une fourmi par ville ;
- La quantité de phéromone initiale ou à déposer est peut influente, prendre par exemple : $\boldsymbol{Q_c = 5}$, $\boldsymbol{Q_{variant} = Q_1 = Q_2 = Q_3 = 100}$;
- Pour les variantes *Ant-Density* et *Ant-Quantity* les meilleurs valeurs des paramètres sont : $\boldsymbol{\alpha = 1}$, $\boldsymbol{\beta = 5}$, $\boldsymbol{\rho = 0.99}$; l'évaporation est égale à : $\boldsymbol{1-\rho = 1-0.99 \approx 0}$, (pas d'évaporation) ;
- Pour la variante *Ant-Cycle* les meilleurs valeurs des paramètres sont : $\boldsymbol{\alpha = 1}$, $\boldsymbol{\beta = 5}$, $\boldsymbol{\rho = 0.5}$; l'évaporation est égale à : $\boldsymbol{1-\rho = 1-0.5 \approx 0.5}$, (évaporation de la moitié de quantité de phéromone) ;

L'étude a aussi permet aux auteurs de conclure que la variante *Ant-Cycle* donnait de meilleurs résultats que les deux autres variantes *Ant-Density* et *Ant-Quantity*. Cela est dû, d'après les auteurs à l'utilisation d'***information globale***. Ils ne donnaient cependant pas trop de détails concernant leurs implémentations, mise à part un package développé

en *ANSI C* sous *Linux* disponible sur le web quelque années après (Stützle 04), et un autre en *Java* ajouté récemment.

3. Modélisation agent Influence/Réaction des variantes de base de Ant system (Zeghida 18a)

Avant de présenter notre modélisation Agent des algorithmes à fourmis de base proposés par *Dorigo* et ses collègues, faisons quelques remarques sur leurs travaux.

3.1. Commentaires sur les travaux de *Dorigo* et al.

La première réflexion évidente et flagrante sur les travaux de *Dorigo* et ses collègues s'illustre par la question suivante : Pourquoi ce qui fonctionne parfaitement dans la nature ne l'est pas dans le monde artificiel?

Les algorithmes *Ant-Density*/*Quantity* reflètent ce que les fourmis font dans la nature; se déplacer et déposer de la phéromone; ils n'ont pas à attendre d'effectuer une tournée entière pour le faire. De même l'évaporation de la phéromone n'attend pas la tournée entière de toutes les fourmis ni d'une fourmi pour se faire. "Malheureusement", la variante "anormale" avec des fourmis "mutantes" donne de meilleurs résultats. La mise à jour de la phéromone (dépôt/évaporation) "attend" que ces fourmis "mutantes" effectuent leurs tournées pour être réalisée.

Bien que ce que nous sommes sur le point de proposer n'est pas "normal" non plus. Nous utilisons des fourmis "mutantes" et avec des comportements "anormaux" plus poussés; Si avec *Dorigo et al.,* nous devons attendre une tournée de fourmis pour procéder à une mise à jour de la phéromone / évaporation, dans notre cas, nous devons attendre chaque déplacement unitaire de toutes les fourmis; ***l'anomalie est plus profonde***.

3.2. Les algorithmes Agent fourmis

L'algorithme 3 donne une version *Agent* des algorithmes *Ant-Density* / *Quantity* (Algorithme 1, Section 2.3). Il présentera nos deux premières variantes:

• A A-D A: Agent Ant-Density Algorithm.

• A A-Q A: Agent Ant-Quantity Algorithm.

Algorithm 3. Agent Ant-Density / Quantity Algorithm

```
Initialization                                          \t=0
for k=1 to m do                                         \m Agent Ant
 Tabulist_k[1]=T_k                                      \Ant_k Start Town T_k
end for
for t=1 to t_max do                                     \t_max iterations
 Parallel                                               \Parallel treatment
  Agent AgentAnt_1                                      \First Ant
    …
  end Agent
  ..
  Agent AgentAnt_k                                      \k^th Ant
   for i=2 to n do                                      \Perform a tour
   Select T_j, using equation (2)                       \Best P_ij
   Tabulist_k[i]=T_j                                    \Ant_k next Town
   Update τ_ij, using equations (1) and (3)/(4)         \For a single move
   end for
  end Agent
  ..
  Agent AgentAnt_m                                      \Last Ant
    …
  end Agent
 end Parallel
 Get shortest tour among the m tours
end for
Get shortest tour among the t_max tours
```

Cependant, l'algorithme 4 donne une version *Agent* de l'algorithme *Ant-Cycle* (Algorithme 2, Section 2.3). Il sera utilisé pour créer notre troisième variante:

• A A-C A: Agent Ant-Cycle Algorithm.

Algorithm 4. Agent Ant-Cycle Algorithm

```
Initialization                                          \t=0
for k=1 to m do                                         \m Agent Ant
 Tabulist_k[1]=T_k                                      \Ant_k Start Town T_k
end for
for t=1 to t_max do                                     \t_max iterations
 Parallel                                               \Parallel treatment
  Agent AgentAnt_1                                      \First Ant
    …
  end Agent
  ..
  Agent AgentAnt_k                                      \k^th Ant
   for i=2 to n do                                      \Perform a tour
   Select T_j, using equation (2)                       \Best P_ij
   Tabulist_k[i]=T_j                                    \Ant_k next Town
   end for
   Update τ_ij, using equations (5) and (6)             \Entire Ant_k tour
  end Agent
  ..
  Agent AgentAnt_m                                      \Last Ant
    …
  end Agent
 end Parallel
 Get shortest tour among the m tours
end for
Get shortest tour among the t_max tours
```

3.3. Les algorithmes Agent Influence/Réaction ant-density / quantity

L'algorithme 5 donne une version Influence / Réaction des algorithmes Agent Ant-Density / Quantity (Algorithme 3, Section 3.2).

Algorithm 5. Influence/Reaction Agent Ant-Density/Quantity Algorithm

```
Initialization                                          \t=0
for k=1 to m do                                         \m Agent Ant
 Tabulist_k[1]=T_k                                      \Ant_k Start Town T_k
end for
for t=1 to t_max do                                     \t_max iterations
 Parallel                                               \Parallel treatment
  Agent AgentAnt_1                                      \First Ant
    …
  end Agent
  ..
  Agent AgentAnt_k                                      \k^th Ant
   for i=2 to n do                                      \Perform a tour
   Select T_j, using equation (2)                       \Best P_ij
   Tabulist_k[i]=T_j                                    \Ant_k next Town
   Request τ_ij Update                                  \For a single move
   end for
  end Agent
  ..
  Agent AgentAnt_m                                      \Last Ant
    …
  end Agent
  Agent Agent_Interaction                               \An Agent
   if m request received then
    Update τ_ij, using equations (1) and (3)/(4)        \For m single moves
   end if
  end Agent
 end Parallel
 Get shortest tour among the m tours
end for
Get shortest tour among the t_max tours
```

Cet algorithme sera utilisé pour produire nos deux dernières variantes:

- I/R A A-D A: Influence / Reaction Agent Ant-Density Algorithm.
- I/R A A-Q A: Influence / Reaction Agent Ant-Quantity Algorithm.

4. Expérimentation: Implémentations et résultats

Cette section fournit le contexte expérimental de notre étude: les implémentations Agent des algorithmes à fourmis de base sans et avec l'utilisation du principe Influence / Réaction qui sera suivi d'une discussion portant sur l'analyse des résultats. Elle sera conclue par une étude de l'effet du paramétrage sur la convergence et le temps d'exécution de ces algorithmes.

4.1. Les implémentations Agent

Pour réaliser ces implémentations Agent, nous avons utilisé la plate-forme orientée agent : *Jade* (http://jade.tilab.com). *Jade* (Java Agent Development) est un framework de développement open source basé *Java* pour les systèmes multi-agents. *Jade* est conforme aux normes de la fondation pour les agents physiques intelligents, *FIPA* (Foundation for Intelligent Physical Agents) qui implémente l'*ACL FIPA* (Agent Communication Language) pour l'interaction des agents. Nous implémentons d'abord l'Algorithme 3 (Sections 3.2) avec ses deux variantes correspondant à l'algorithme A A-D A: Agent Ant-Density Algorithm et l'algorithme. A A-Q A: Agent Ant-Quantity Algorithm. Ensuite, nous implémentons l'Algorithme 4 (Section 3.2) pour produire l'algorithme A A-C A: Agent Ant-Cycle Algorithm. La première comparaison sera effectuée sur ces trois variantes précédentes: A A-D A, A A-Q A et A A-C A.

Après cela, nous procédons à l'implémentation de l'Algorithme 5 (Section 3.3) avec ses deux variantes correspondant à l'algorithme I/R A A-D A: Influence/Reaction Agent Ant-Density Algorithm et l'algorithme I/R A A-Q A: Influence/Reaction Agent Ant-Quantity Algorithm. Pour effectuer la seconde comparaison impliquant ces deux dernières variantes: I/R A A-D A, I/R A A-Q A et la variante A A-C A.

4.2. Résultats : Analyses et réflexions

Le Tableau 6.1 fournit une synthèse des résultats de la première comparaison entre les algorithmes : A A-D A: Agent Ant-Density Algorithm, A A-Q A: Agent Ant-Quantity Algorithm et A A-C A: Agent Ant-Cycle Algorithm.

	Algorithmes		
Benchmarks	**A A-D A**	**A A-Q A**	**A A-C A**
10.tsp	**585.0282395**	**585.0282395**	**585.0282395**
Ulysses16.tsp	**84.42764187**	**84.42764187**	**84.26972339**
Ulysses22.tsp	**85.29376531**	**85.29376531**	*86.81363806*
bayg29.tsp	**10810.48103**	**10810.48103**	**10810.48103**
30.tsp	**28836.82797**	**28518.70836**	**28836.82797**
att48.tsp	**38689.79869**	*40931.69606*	**38462.74449**
Eil51.tsp	*510.7839293*	**489.055727**	**487.7401788**
Berlin52.tsp	*8422.979948*	**8093.352348**	**8093.352348**

St70.tsp	**732.0497763**	*806.1246417*	**783.4198359**
Eil76.tsp	**619.0593045**	*646.0931385*	**583.293779**
Pr76.tsp	**122837.2728**	*131037.8591*	**130660.2977**
rat99.tsp	**1367.746084**	**1367.746084**	*1436.632020*
rd100.tsp	**9449.854420**	*9531.009032*	**9274.337650**
KroA100.tsp	**25897.47515**	*26480.88356*	**24673.23949**
Eil101.tsp	*760.5774858*	**751.8951685**	**749.5914276**
lin105.tsp	**15946.13436**	*16235.61910*	**15441.64204**

Valeurs grisées: *Meilleurs résultats;* ***Valeurs en gras***: *Second meilleurs résultats.*

Tableau 6.1 Comparaison des longueurs de tours des variantes de base du système à fourmis basé Agent sur les benchmarks TSP symétrique

Le Tableau 6.2 fournit une synthèse des résultats de la seconde comparaison entre les algorithmes : I/R A A-D A: Influence/Reaction Agent Ant-Density Algorithm, I/R A A-Q A: Influence/Reaction Agent Ant-Quantity Algorithm et A A-C A: Agent Ant-Cycle Algorithm.

	Algorithmes		
Benchmarks	**I/R A A-D A**	**I/R A A-Q A**	**A A-C A**
10.tsp	**585.0282395**	**585.0282395**	**585.0282395**
Ulysses16.tsp	**84.26972339 ***	**84.26972339 ***	**84.26972339**
Ulysses22.tsp	**86.46881220 ***	**86.46881220 ***	*86.81363806*
bayg29.tsp	**9621.144925 ***	**9371.469827 ***	*10810.48103*
30.tsp	**28193.92627**	**28535.00499**	*28836.82797*
att48.tsp	*39528.06846*	**38800.77598**	**38462.74449**
Eil51.tsp	**472.5737199**	**472.5737199**	**487.7401788**
Berlin52.tsp	*8172.421497*	**8093.352348**	**8093.352348**
St70.tsp	**736.5853503 ***	**747.4435722**	*783.4198359*
Eil76.tsp	*609.4187010*	**598.9653970**	**583.293779**
Pr76.tsp	**118223.0926 ***	**126270.5087 ***	*130660.2977*
rat99.tsp	**1271.891991 ***	**1346.519586**	*1436.632020*
rd100.tsp	**8517.113165 ***	**8985.403224**	*9274.337650*
KroA100.tsp	*25918.97162*	**25588.39430**	**24673.23949**
Eil101.tsp	*758.7904023*	**731.5922711**	**749.5914276**
lin105.tsp	**15197.88123 ***	**15240.10616**	*15441.64204*

Valeurs grisées: *Meilleurs résultats;* ***Valeurs en gras***: *Second meilleurs résultats ; * : Nouvelle valeur de* ρ

Tableau 6.2 Comparaison des longueurs de tours des algorithmes Agent Ant-Cycle et Agent influence / réaction Ant-Density et Ant-Quantity sur les benchmarks TSP symétrique

Une lecture et une analyse des résultats récapitulés dans les deux tableaux précédents révèlent les premiers constats suivants :

1. Pour l'algorithme A A-C A: Agent Ant-Cycle Algorithm qui utilise de l'information "globale" (argument donné par les auteurs pour le succès de cette variante), l'ensemble des valeurs des paramètres reste valable : $\alpha = 1$, $\beta = 5$, $\rho = 0.5$ (Tableau 6.1);
2. L'ensemble des valeurs des paramètres pour les deux algorithmes A A-D A: Agent Ant-Density Algorithm et A A-Q A: Agent Ant-Quantity Algorithm reste aussi valable : $\alpha = 1$, $\beta = 5$, $\rho = 0.99$ (Tableau 6.1);
3. Dans notre implémentation Agent, pris ensemble, les deux algorithmes A A-D A: Agent Ant-Density Algorithm et A A-Q A: Agent Ant-Quantity Algorithm sont aussi bons que l'algorithme A A-C A: Agent Ant-Cycle Algorithm qui n'est plus meilleur que dans environ 8/16 des cas (Tableau 6.1);
4. Pour un nombre limité de villes (10 villes; benchmark *10.tsp*, Tableau 6.2), le principe Influence/Réaction est sans effet: les interactions ne suffisent pas pour surmonter le temps perdu à retarder la mise à jour des phéromones : dépôt et évaporation (l'effet commence à partir des benchmarks *Ulysses16.tsp* et *Ulysses22*.tsp, Tableau 6.2);
5. Pour les algorithmes I/R A A-D A: Influence/Reaction Agent Ant-Density Algorithm, I/R A A-Q A: Influence/Reaction Agent Ant-Quantity Algorithm, l'ensemble des valeurs de paramètres: $\alpha = 1$ et $\beta = 5$ reste utile. Par contre au lieu de $\rho = 0.99$ de meilleures valeurs ont été trouvées: $\rho = 0.3$, $\rho = 0.5$ ou au moins, $\rho = 0.7$ (au lieu de ne rien évaporer et de garder toute l'expérience des étapes précédentes avec $\rho = 0.99$, le système a besoin d'oublier une partie de son passé et doit évaporer de la phéromone : 0,7, 0,5 ou 0,3, Tableau 6.2);
6. Lorsque nous retardons l'évaporation et le dépôt de la phéromone, le système dispose d'une façon très efficiente d'une quantité d'information "globale" et dans ce cas il doit oublier une partie de son passé (les exploits des fourmis aux étapes précédentes) on doit donc travailler avec un $\rho < 0,99$ pour que l'évaporation soit > 0 ; points 1 et 5 précédents);

7. L'algorithme I/R A A-D A: Influence/Reaction Agent Ant-Density Algorithm performe, légèrement, mieux que l'algorithme I/R A A-Q A: Influence/Reaction Agent Ant-Quantity Algorithm (Tableau 6.2).
8. Les algorithmes I/R A A-D A: Influence/Reaction Agent Ant-Density Algorithm, I/R A A-Q A: Influence/Reaction Agent Ant-Quantity Algorithm performent mieux que l'algorithme A A-C A: Agent Ant-Cycle Algorithm qui reste meilleur dans seulement 3/16 des cas (Tableau 6.2).

Ces résultats sont la conséquence d'une distribution et d'une disponibilité efficientes de l'information dans le système, résultant de la modélisation Influence/Réaction et portée par le paradigme agent. Cela répond aux sept critères (en particulier les cinquième et sixième) proposés par *Parunak* dans (Parunak 97) et résumés dans (Zeghida 12), lorsqu'il avait demandé à la communauté informatique de "se tourner et allez vers les fourmis" ("Go to the Ants") dans le cas d'ingénierie de *SMA* biomorphiques. Ces critères permettent aux agents de couvrir à la fois les caractéristiques artificielles et naturelles de tels systèmes.

En effet, même si le principe Influence/Réaction n'affecte pas l'action simultanée et la modélisation des interactions dans les agents, ce principe améliore les mécanismes de diffusion de l'information pour augmenter la réactivité du système. C'est la troisième utilité remarquable de ce principe.

4.3. Paramétrage et analyse de convergence et de la durée d'exécution d'algorithmes à fourmis

Dans cette étude expérimentale, nous ne proposons pas de variante améliorée d'algorithme à fourmis de base; nous comparons simplement leurs implémentations agents avec ou sans le principe Influence/Réaction. Puisque, la convergence des Algorithmes à fourmis est évidente (les trois points suivants, en particulier a et b) et a été "expérimentalement" prouvés par notre expérience; Nous nous concentrons, comme valeur ajoutée, sur l'effet du paramétrage sur les performances des variantes Agent fourmis implémentées. Pour les paramètres les plus importants: α, β et ρ, nous avons utilisé également d'autres valeurs. Nos choix et premiers commentaires et constats sont présentés dans le Tableau 6.3.

Lorsque nous explorons le comportement des chercheurs face à l'effet du paramétrage sur la convergence de leur algorithme proposé (Zhou 09), nous pouvons distinguer trois étapes:

a) Analyse expérimentale de convergence: lors de la proposition d'une nouvelle heuristique ou l'amélioration d'une existante; l'objectif est d'obtenir des résultats améliorés. Les auteurs procèdent à des réglages de paramètres expérimentaux, la convergence est "vérifiée expérimentalement" puisqu'ils atteignent leur objectif et obtiennent des algorithmes ayant de meilleur performance. Des études expérimentales récentes sur l'effet de paramètre dans *MMAS* et *ACO* sont disponibles dans (Pellegrini 06) et (Stützle 11). Sachant que l'heuristique *AS* était proposé pour résoudre le *TSP* vers les années 90, il a fallu, donc, plus de dix (10) ans à cette étape pour effectuer la première vérification théorique de la convergence (années 2000).

La convergence des AA est " expérimentalement vérifiée ".

b) Analyse théorique de la convergence: cette étape commence dans les années 2000, avec la première vérification de la convergence pour une variante particulière d'*ACO* appelée *GBAS* (Graph-Based Ant System) (Gutjahr 00) et pour une autre variante de *GBAS* dans (Gutjahr 02). Dans (Stützle 02), *Stützle* et *Dorigo* ont présenté une brève preuve de convergence du *MMAS* et d'une classe d'algorithmes *ACO*. *Meyer* (Meyer 04) a étudié le contrôle de l'exploration de l'espace de recherche (diversification) et l'influence du paramètre α dans un *ACO* (α détermine, qualitativement, le comportement en diversité et convergence et améliore la robustesse des algorithmes à fourmis). (Kötzing 12) présente une autre tentative d'analyse des performances de l'algorithme ACO.

La convergence des AA est " théoriquement vérifiée ".

c) Analyse de la durée d'exécution (runtime): *Neumann* et *Witt* (Neumann 06) ont proposé la première analyse de la durée d'exécution d'un algorithme *ACO* simple appelé le *1-ANT* appliqué à l'optimisation de fonctions pseudo-booléennes. Ils ont démontré que le facteur d'évaporation (ρ) avait un impact crucial sur la durée d'exécution. Il en a été de même pour *Gutjahr* pour deux algorithmes *ACO*: *GBAS* et *AS* (Gutjahr 08a). *Doerr et al.* ont étudié l'effet du mécanisme de mise à jour des phéromones pour l'algorithme *1-ANT* (Doerr 07a) et (Doerr 07b). À l'instar des

exemples précédents (Gutjahr 08b) et (Neumann 09) ont analysé la complexité d'exécution d'un algorithme *ACO* appliqué à l'optimisation de fonctions pseudo-booléennes.

Pour l'application des *ACO* pour les *COP*, *Neumann* et *Witt* (Neumann 10) ont présenté l'analyse de la durée d'exécution d'un algorithme *ACO* simple *1-Ant*, alors que *Attiratanasunthron* et *Fakcharoenphol* (Attiratanasunthron 08) ont prouvé que la durée d'exécution d'un algorithme à fourmis multiple est polynomiale.

Zhou (Zhou 09) a prouvé la faisabilité d'appliquer les algorithmes *ACO* pour *TSP* et a fourni une analyse théorique de la convergence et de la durée d'exécution de cet algorithme. Pour un cas d'application similaire, *Nallaperuma et al.* ont utilisé l'analyse statistique pour comprendre l'effet des paramètres et des caractéristiques d'instance sur les performances de l'algorithme (Nallaperuma 13).

	Algorithmes					
Paramètres	**A A-D A**	**A A-Q A**	**A A-C A**	**I/R A A-D A**	**I/R A A-Q A**	**Commentaires**
n	n = m	n = m	n = m	n = m	n = m	n : Nombre de ville, taille du benchmark.
m	m = n	m = n	m = n	m = n	m = n	m : Nombre de fourmis égal au nombre de ville n donc une fourmi par ville.
Q_c	5	5	5	5	5	Une valeur initiale.
$Q_{variant}$	100	100	100	100	100	Une valeur quelconque.
T_{max}	100 ~ 500	100 ~ 500	100 ~ 500	100 ~ 500	100 ~ 500	**Critère d'arrêt.**
α	1	1	1	1	1	**Affecte la convergence.**
β	5	5	5	5	5	**Affecte la convergence.**
ρ	0.999	0.999	0.5	0.5 (0.3 & 0.7)	0.5 (0.3 & 0.7)	**Affecte la performance.**

Tableau 6.3 Paramétrage des variantes Agents fourmis

5. Conclusion

Pour gérer l'aspect parallèle des algorithmes à fourmis, nous avons préconisé, dans ce Chapitre, l'utilisation de notre modèle Agent Influence/Réaction pour l'implémentation d'algorithmes à fourmis pour le *TSP*.

Le premier résultat remarquable de notre étude expérimentale est l'amélioration des algorithmes Ant-Density/Quantity par rapport à l'algorithme Ant-Cycle, lorsqu'ils sont implémentés comme Agent utilisant le principe Influence/Réaction. En conséquence, nous pouvons affirmer l'émergence d'une nouvelle utilisation du principe Influence/Réaction: son rôle de processus de diffusion de l'information dans le système, en fournissant une disponibilité/distribution efficiente de l'information pour augmenter la réactivité des agents du système.

Il faut noter qu'à ce stade nous n'essayons pas du tout de positionner notre variante parmi les autres heuristiques appliquées au *TSP* (Recuit Simulé, Recherche Tabou, Algorithme Génétique, Optimisation par essaim de particules ou Optimisation par colonie de fourmis etc.). Malgré nos résultats prometteurs, nous ne prétendons pas (encore) que notre variante est meilleure que les autres. Tout fois, une analyse poussée sur l'effet des paramètres sur la convergence et les performances d'un algorithme serait vivement souhaitable.

CHAPITRE 7

CONCLUSION GENERALE

CHAPITRE 7

CONCLUSION GENERALE

1. Résumé des contributions

Les systèmes artificiels que nous développons aujourd'hui sont à forte dominance logicielle et sont de plus en plus complexes. Cette complexité ne réside pas seulement dans la multitude d'entités qui forment un système opérationnel, mais dans la nature diversifiée de ces entités et les interactions multiples et variées qu'ils peuvent avoir. Les systèmes bio-inspirés illustrent parfaitement ce constat, d'où la nécessité d'une vision multi-paradigmes d'un système artificiel. Les enjeux de cette vision résident dans la démarche de développement de ce système. La démarche inclut principalement des processus qui peuvent être manuels ou automatisés et des artefacts de modélisation (ou de modèles). La démarche de développement des systèmes biomorphiques a comme objectif le développement d'un système moyennant divers paradigmes bio-inspirés.

Pour faire face à la prolifération des systèmes biomorphiques, il est devenu nécessaire de focaliser l'attention et les recherches sur leurs modèles. Une telle modélisation doit englober tous les différents concepts bio-inspirés.

Sous cette perspective nous avons proposé un modèle multi-paradigmes générique pour les systèmes multi-agents biomorphiques. Notre modèle prend en charge la diversité structurelle et comportementale des entités d'un système artificiel. Nous nous basons sur le principe Influence/Réaction pour gérer les interactions dans le système biomorphique. Notre modèle a été validé à travers une étude expérimentale (Zeghida 18b).

Avec une vision panoramique, quand on confronte l'agent et la biologie l'impact peut se manifester dans deux domaines de l'informatique comme suit (Zeghida 18a):

1. En Génie Logiciel (*GL*): où il y sera question de méthodologies et de techniques agent, nous parlons ici de génie logiciel orienté agent (Agent Oriented Software

Engineering : *AOSE*). Notre contribution se concrétise par notre *modèle générique BIO-IR-M* pour la modélisation multi-paradigmes basée agent Influence/Réaction de systèmes biomorphiques (Zeghida 18a).

2. En Intelligence Artificielle (*IA*): ou plus précisément l'Intelligence Artificielle Distribuée (*IAD*), il y sera question d'outils et de modèles agent (algorithmes/plates-formes), nous parlons ici de solveurs/*optimiseurs* ou de modélisation/simulation agent. Notre contribution se concrétise par la proposition d'une "nouvelle famille" *d'algorithmes d'optimisation* à travers les deux variantes:

   - I/R A A-D A: Influence/Reaction Agent Ant-Density Algorithm.
   - I/R A A-Q A: Influence/Reaction Agent Ant-Quantity Algorithm.

   Les résultats obtenus ont inversé les rapports de force détenus par la variante : Ant-Cycle par rapport aux deux variantes : Ant-Density et Ant-Quantity en faveurs des variantes Agent Influence/Réaction de ces deux dernières (Zeghida 18b).

Avec une vision détaillée, nous résumons les principales caractéristiques de nos contributions dans:

1. L'application du principe Influence/Réaction ayant les conséquences suivantes :
   a. La garantie de l'intégrité de l'agent ; son autonomie, par la distinction entre les variables d'état du système décisionnel d'un agent et les variables concernant son aspect physique.
   b. La garantie de la cohérence du système par la modélisation de comportements simultanés et joints.
   c. *L'amélioration des mécanismes de diffusion de l'information pour augmenter la réactivité des agents.*
   d. L'isolation d'un module d'interaction (le module de couplage) qui utilise toutes les influences produites à un moment donné pour calculer le nouvel état du monde.
2. La garantie d'un aspect actif de l'environnement par sa modélisation explicite.
3. Le développement de système ouvert.
4. Le respect conjoint des caractéristiques biologiques et agent.

2. Perspectives

Avec la même vision panoramique utilisée pour présenter nos contributions, leurs perspectives peuvent se voir de la façon suivante:

1. En Génie Logiciel (*GL*): où il était question de méthodologies orientées agent ; partant du fait que les *SMAs* représentent un paradigme potentiellement unificateur, une ***première perspective*** consiste à établir une synthèse des méthodologies basées agents et à identifier un noyau à adapter, afin d'incorporer notre *modèle* bio-inspiré générique "*BIO-IR-M*". Le degré d'adaptation d'une approche de développement, à cet objectif, dépend non seulement de la diversité des approches bio-inspirées considérées mais aussi de leurs combinaisons possibles, enrichissant leur champ d'application existant.

   Dans un tel contexte multi-paradigme, une ***seconde perspective*** serait de reconsidérer ce noyau pour exploiter la force des approches bio-inspirées ; où pour un problème donné et connaissant tous ses critères spécifiques, nous serons en mesure d'atteindre l'état pour une véritable orientation de l'utilisateur à choisir le paradigme bio-inspiré à appliquer ou indiquer des combinaisons possibles.

2. En Intelligence Artificielle (*IA*): où il était question d'outils et de modèles agent, nous parlons ici d'algorithmes *optimiseurs* basés agent. La ***troisième perspective*** propose de reconsidérer toutes les variantes améliorées d'algorithmes à fourmis développés dans le domaine et basés sur la supériorité, compromise avec notre modèle, de l'algorithme Ant-Cycle parmi les trois variantes de base ; redéfinir et redévelopper les algorithmes à fourmis améliorés basés Ant-Cycle sur la base de nos nouvelles meilleures variantes : les algorithmes Agent Influence/Reaction Ant-Density et Agent Influence/Reaction Ant-Quantity.

   Une ***quatrième perspective*** sera d'explorer l'aspect organisationnel dans ces systèmes à fourmis sur une plateforme agent prenant en charge l'aspect organisation telle que *MadKit*.

   Enfin une ***cinquième perspective*** envisage l'utilisation de nos variantes agent Influence/Réaction de base et améliorées pour d'autres domaines d'application.

REFERENCES

REFERENCES

(Agha 85) Agha, G. A., 1985, Actors: A model of concurrent computation in distributed systems, technical report, DTIC Document,

(Amine 15) Amine, L. M. and Nadjet, K., 2015, A Multi-objective Binary Bat Algorithm, Proceedings of the International Conference on Intelligent Information Processing, Security and Advanced Communication, ACM, 75,

(Attiratanasunthron 08) Attiratanasunthron, N., Fakcharoenphol, J., 2008, A running time analysis of an ant colony optimization algorithm for shortest paths in directed acyclic graphs. Information Processing Letters, Elsevier, 105(3), 88–92.

(Azaiez 07) Azaiez Selma, 2007, Approche dirigée par les modèles pour le développement de systèmes multi-agents. Thèse de doctorat, Université de Savoie.

(Bakhouya 03) Bakhouya, M., Gaber, J. and Koukam, A., 2003, Bio-inspired model for behavior emergence: Modelling and case study, Procs. Of Knowledge Grid and Grid Intelligence workshop (KGGI'03) at IEEE WI/IAT'03.

(Ballet 00) Ballet P. 2000, Intérêts Mutuels des Systèmes Multi-Agents et de l'Immunologie. Thèse, Université de Bretagne Occidentale.

(Bari 09) Bari, A., Wazed, S., Jaekel, A. and Bandyopad- hyay, S., 2009, A genetic algorithm based approach for energy efficient routing in two- tiered sensor networks, Ad Hoc Networks Journal, Elsevier, (7), (4), 665–676,

(Bellifemine 99) Bellifemine, F., Poggi, A. and Rimassa, G., 1999, JADE–A FIPA- compliant agent framework, Proceedings of PAAM, London, (99), 97–108.

(Bergia 99) Loris Bergia, 1999, Approches Multi agents pour les environnements interactifs d'apprentissage avec ordinateur, Laboratoire Leibniz- Institut IMAG.Grenoble.

(Bergia 01) Loris Bergia, 2001, Conception et réalisation d'une plate-forme pour l'apprentissage et l'enseignement à Distance, Mémoire présenté en vue d'obtenir le diplôme d'Ingénieur CNAM Laboratoire Leibniz-IMAG, GRENOBLE, France-n° 20.

(Bernon 02) Bernon, C., Gleizes, M-P., Peyruqueou, S. and Picard, G., 2002, ADELFE: a methodology for adaptive multi-agent systems engineering, International Workshop on Engineering Societies in the Agents World, Springer, 156–169.

(Bernon 04) Bernon C., Cossentino M., M.P., Gleizes Turci P., and Zambonelli F., 2004, A study of some Multi-Agent Meta-Models. In Proceedings of the fourth international workshop on software engineering for large-scale multi-agent systems.

(Bernon 05) Bernon C., Cossentino M. and Pavón J., 2005, Agent Oriented Software Engineering. The Knowledge Engineering, ACM Publication.

(Bertolini 05) Bertolini D., Perini A., Susi A., and Mouratidis H., 2005, The Tropos visual modeling language. A MOF 1.4 compliant meta-model. Contribution for the AOSE TFG meeting.

(Beurier 07) Beurier, G. 2007, Codage indirect de la forme dans les systèmes multi-agents; émergence multi-niveaux, morphogenèse et évolution. Thèse, Université de Montpellier II.

(Blum 05) Blum, C. 2005, Ant colony optimization: Introduction and recent trends. Elsevier, Physics of Life Reviews 2, 353–373.

(Bourdon 01) Bourdon, F. 2001, Systèmes Multi-Agents. Cours exposé dans le cadre du DEA-IAA-(tronc commun). Université de Caen.

(Bresciani 02) Bresciani Paolo, Giorgini Paolo, Giunchiglia Fausto, Mylopoulos John, and Perini Anna. , 2002, TROPOS: An agent-oriented software development methodology, Technical Report # DIT-02-0015, University Of Trento.

(Broecker 15) Broecker, B., Caliskanelli, I., Tuyls, K., Sklar, E. and Hennes, D., 2015, Social insect-inspired multi-robot coverage, Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, International Foundation for Autonomous Agents and Multiagent Systems, 1775–1776.

(Brun 08) Brun, Y., 2008, Building Biologically- Inspired Self-Adapting Systems, Dagstuhl Seminar Proceedings, Schloss Dagstuhl- Leibniz-Zentrum fr Informatik.

(Caglayan 98) Caglayan, A. et C. Harrison, 1998, Les agents Applications bureautiques, Internet et intranet. Intereditions, Paris.

(Capera 03) Capera, D., George, J.P., Gleizes, M.P. and Glize, P., 2003, The AMAS theory for complex problem solving based on self-organizing cooperative agents,

Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003. WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on, IEEE, 383–388.

(Chang 06) Chang. H. S. 2006, Computational Intelligence Optimization: Particle Swarm Optimisation and Ant Colony Optimization, A Gental Introduction. cours, Sogang University.

(Colorni 91) Colorni, A., Dorigo, M., Maniezzo, V. and others, 1991, Distributed optimization by ant colonies, Proceedings of the first European conference on artificial life, (142), 134–142.

(Cossentino 01) Cossentino M. and Potts C., 2001, PASSI : a process for specifying and implementing Multi-Agent Systems Using UML.

(Cuperlier 16) Cuperlier, N., Guedjou, H., de Melo, F., and Miramond, B., 2016, Attention-based smart-camera for spatial cognition, Proceedings of the 10th International Conference on Distributed Smart Camera, ACM, 121–127.

(Da Silva 02) Da Silva, J.L.T. and Demazeau, Y., 2002, Vowels co-ordination model, Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 3, ACM, 1129-1136.

(DeLoach 99) DeLoach Scott A., 1999, Multiagent Systems Engineering: A Methodology And Language for Designing Agent Systems, Presented at Agent-Oriented Information Systems (AOIS) ’99.

(DeLoach 01) DeLoach Scott A. , 2001, Wood Mark F., and Sparkman Clint H., MULTIAGENT SYSTEMS ENGINEERING, International Journal of Software Engineering and Knowledge Engineering, Vol. 11, No. 3, 231-258.

(DeLoach 04) DeLoach Scott A., 2004, Chapter 6: THE MASE METHODOLOGY, Methodologies and Software Engineering for Agent Systems, Springer.

(Demazeau 01) Yves Demazeau, et al, 2001, Principes et architecture des systèmes multi-agents, Hermes Science Publication.

(Doerr 07a) Doerr, B., Johannsen, D., 2007, Refined runtime analysis of a basic ant colony optimization algorithm. in Proc. IEEE Congress on Evolutionary Computation, CEC 2007, NJ: IEEE Press, 501–507.

(Doerr 07b) Doerr, B., Neumann, F., Sudholt, D., Witt, C., 2007, On the runtime analysis of the 1-ANT ACO algorithm. in Proceedings of the 9th annual conference on

Genetic and evolutionary computation (GECCO), London, U.K.: ACM, 33–40.

(Dorigo 91a) Dorigo Marco, Maniezzo Vittorio et Colorni Alberto, 1991, Positive Feedback as a search strategy, Technical Report n° 91-016.

(Dorigo 91b) Dorigo, M., Maniezzo, V. and Colorni, A., 1991, The ant system: An autocatalytic optimizing process. Tech. rep., Italy: Dipartimento di Elettronica, Politecnico di Milano.

(Dorigo 96) Dorigo, M., Maniezzo, V., Colorni, C., 1996, Ant system: optimization by a colony of cooperating agents. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions, 26(1), 29–41.

(Dorigo 97) Dorigo, M. and Gambardella, L., 1997, Ant colonies for the travelling salesman problem. BioSystems, Elsevier, 43(2), 73–81.

(Dorigo 99) Dorigo, M. and Di Caro, G., 1999, Ant colony optimization: a new meta-heuristic, Evolutionary Computation, CEC 99. Proceedings of the 1999 Congress on, IEEE, (2), 1470–1477.

(Dorigo 04) Dorigo, M., Stützle, T., 2004, Ant Colony Optimization. MIT Press, Cambridge, MA, USA.

(Dorigo 06) Dorigo, M. et K. Socha, 2006. An Introduction to Ant Colony Optimization, IRIDIA, Technical Report No.TR/IRIDIA/2006-010.

(Dréo 06) Dréo, J., Pétrowski, A., Siarry, P. and Taillard, E., 2006, Metaheuristics for hard optimization: methods and case studies, Springer Science & Business Media.

(Drogoul 02) Drogoul, Alexis, Vanbergue, Diane, & Meurisse, Thomas. , 2002, Multi-Agent Based Simulation: Where are the Agents? Multi-Agent-Based Simulation II, Proceedings of MABS 2002, Third International Worshop. LNAI, vol. 258, Pages 89–104. Springer-Verlag.

(Ferber 96) Ferber, J., and Müller, J-P., 1996, Influences and reaction: a model of situated multiagent systems, Proceedings of Second International Conference on Multi-Agent Systems (ICMAS-96), 72–79.

(Ferber 98) Ferber, J. and Gutknecht, O., 1998, A meta-model for the analysis and design of organizations in multi-agent systems, MultiAgent Systems, Proceedings. International Conference on, IEEE, 128–135.

(Ferber 99) Ferber, Jacques, 1999, Multi-agent systems: an introduction to distributed

artificial intelligence, Addison-Wesley Reading.

(Ferber 04) Ferber, J., Michel, F. and Báez, J., 2004, AGRE: Integrating environments with organizations, International Workshop on Environments for Multi-Agent Systems, Springer, 48–56.

(Fink 14) Fink, G.A., Haack, J.N., McKinnon, A.D. and Fulp, E.W., 2014, Defense on the move: ant-based cyber defense, IEEE Security & Privacy, IEEE, (12), (2), 36–43.

(Florin 15) Florin Leon, Marcin Paprzycki, and Maria Ganzha. 2015, A Review of Agent Platforms, Multi-Paradigm Modelling for Cyber-Physical Systems (MPM4CPS), ICT COST Action IC1404, 1–15.

(Fogel 88) Fogel, D.B., 1988, An evolutionary approach to the traveling salesman problem, Biological Cybernetics Journal, Springer, (60), (2), 139–144.

(Gengan 14) Gengan, D., Schoeman, M.A. and Van Der Poll, J.A., 2014, An Ant-based Mobile Agent Approach to Resource Discovery in Grid Computing, Proceedings of the Southern African Institute for Computer Scientist and Information Technologists Annual Conference 2014 on SAICSIT 2014 Empowered by Technology, ACM, 1.

(Georgé 03) Georgé Jean-Pierre, Gleizes Marie-Pierre, Glize Pierre, 2003, Conception de systèmes adaptatifs à fonctionnalité émergente: la théorie des AMAS, Revue d'intelligence artificielle.

(Giorgini 04) Giorgini, Paolo and Kolp, Manuel and Mylopoulos, John and Pistore, Marco, The tropos methodology, Methodologies and software engineering for agent systems, 89--106, Springer.

(Giunchiglia 01) Giunchiglia Fausto, Mylopoulos John, and Perini Anna, 2001, THE TROPOS SOFTWARE DEVELOPMENT METHODOLOGY: PROCESSES, MODELS AND DIAGRAMS, Technical Report # DIT-02-0008, UNIVERSITY OF TRENTO.

(Gonçalves 13) Gonçalves, F. A.C.A., Guimarães, F.G. and Souza, Marcone J.F., 2013, An evolutionary multi-agent system for database query optimization, Proceedings of the 15th annual conference on Genetic and evolutionary computation, ACM, 535–542.

(Gutjahr 00) Gutjahr, W.J., 2000, A graph-based Ant System and its convergence. Future Generation Computer Systems, Elsevier, 16(8), 873–888.

(Gutjahr 02) Gutjahr, W.J., 2002, ACO Algorithms with Guaranteed Convergence to the

Optimal Solution. Information Processing Letters, Elsevier Science, 82(3), 145–153.

(Gutjahr 08a) Gutjahr, W.J., 2008, First steps to the runtime complexity analysis of ant colony optimization. Computers & Operations Research, Elsevier, 35(9), 2711–2727.

(Gutjahr 08b) Gutjahr, W.J., Sebastiani, G., 2008, Runtime analysis of ant colony optimization with best-so-far reinforcement. Methodology and Computing in Applied Probability, 10(3), 409–433.

(Gutknecht 00) Gutknecht, O. and Ferber, J., 2000, The madkit agent platform architecture, Workshop on Infrastructure for Scalable Multi-Agent Systems at the International Conference on Autonomous Agents, Springer, 48–55.

(Hong 14) Hong, T-P., Huang, L-I. and Lin, W-Y., 2014, A Different Perspective on Parallel Sub-Ant-Colonies, Proceedings of the 12th International Conference on Advances in Mobile Computing and Multimedia, ACM, 322–325.

(Huget 14) Huget, M-P., 2014, Agent Communication, Agent-Oriented Software Engineering, Springer, 101–133.

(Iglesias 99) Iglesias C.A., Garijo M., and Gonzalez J.C., 1999, A survey of agent-oriented methodologies. In Proceeding of the Fifth International Workshop on Agent Theories Architectures, and Languages, Budapest, Hungary.

(Jarras 02) Jarras, I. et B.Chaib-draa, 2002, Aperçu sur les systèmes multiagents. Série scientifique. 2002s-67. Montréal.

(Jennings 98a) Jennings N. R., M. Wooldridge, 1998, Applications of Intelligent Agents; Queen Mary & Westfield College. University of London.

(Jennings 98b) Jennings, N.R., Sycara, K. and Wooldridge, M., 1998, A roadmap of agent research and development, Autonomous agents and multi-agent systems, Kluwer Academic Publishers, (1), (1), 7–38.

(Jennings 01) Jennings, N.R., 2001, An agent-based approach for building complex software systems, Communications ACM Journal, ACM, (44), (4), 35–41.

(Karmani 09) Karmani, R.K., and Shali, A. and Agha, G., 2009, Actor frameworks for the JVM platform: a comparative analysis, Proceedings of the 7th International Conference on Principles and Practice of Programming in Java, ACM, 11–20.

(Karmani 11) Karmani, R.K. and Agha, G., 2011, Actors, Encyclopedia of Parallel Computing, Springer, 1–11.

(Kosakaya 16) Kosakaya, J., 2016, Multi-agent-based SCADA system, Event-based Control, Communication, and Signal Processing (EBCCSP), 2016 Second International Conference on, IEEE, 1–5.

(Kötzing 12) Kötzing, T., Neumann, F., Röglin, H., Witt, C., 2012, Theoretical analysis of two ACO approaches for the traveling salesman problem. Swarm Intelligence, 6, 1–21.

(Labidi 93) S.Labidi, W.Lejouad, 1993, De l'Intelligence Artificielle Distribuée au Systèmes Multi Agents, Rapport de recherche INRIA n° 2004.

(Lee 09) Lee, U., Magistretti, E., Gerla, M., Bellavista, P., Li´o, P. and Lee, K.W., 2009, Bio-inspired multi-agent data harvesting in a proactive urban monitoring environment, Ad Hoc Networks Journal, Elsevier, (7), (4), 725–741.

(Lin 91) Lin, C-T. and Lee, C.S.G., 1991, Neural-network-based fuzzy logic control and decision system, IEEE Transactions on computers, IEEE, (40), (12), 1320–1336.

(Lodding 04) Lodding, K.N., 2004, The Hitchhiker's Guide to Biomorphic Software, ACM Queue Journal, ACM, (2), (4), 66–75.

(Ma 14) Ma, J., Man, K.L., Ting, T. Zhang, N., Guan, S-U. and Wong, P.WH., 2014, Accelerating Parameter Estimation for Photovoltaic Models via Parallel Particle Swarm Optimization, Computer, Consumer and Control (IS3C), International Symposium on, IEEE, 175–178.

(Malone 94) Thomas W.Malone, Kevin Crowston, 1994, The interdisciplinary study of coordination; ACM computer surveys, Vol 26, n° 1, 87-117.

(Manate 14) Manate, B., Fortis, F. and Moore, P., 2014, Applying the Prometheus methodology for an Internet of Things architecture, Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing, IEEE Computer Society, 435–442.

(Mariani 16) Mariani, S. and Omicini, A., 2016, Multi-paradigm Coordination for MAS: Integrating Heterogeneous Coordination Approaches in MAS Technologies, WOA, 91–99.

(Massawe 09) Massawe, L.V., Aghdasi, F. and Kinyua, J., 2009, The development of a multi-agent based middleware for RFID asset management system using the PASSI

methodology, Information Technology: New Generations, ITNG'09. Sixth International Conference on, IEEE, 1042–1048.

(Meyer 04) Meyer, B., 2004, Convergence control in ACO. Genetic and Evolutionary Computation Conference (GECCO), Seattle, WA.

(Michel 04) Michel, F., 2004, Formalism, tools and methodological elements for the modeling and simulation of multi-agents systems, PhD thesis, LIRMM, Montpellier, France.

(Michel 07) Michel, F., 2007, The IRM4S Model: The Influence/Reaction Principle for Multiagent Based Simulation (short paper), Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2007), Honolulu, Hawaii, USA, ACM, 903–905.

(Michel 09) Michel, F., Ferber, J., Drogoul, A. and others, 2009, Multi-Agent Systems and Simulation: a Survey From the Agents Community's Perspective, Multi-Agent Systems: Simulation and Applications Journal, 3–52.

(Mirjalili 14) Mirjalili, S., Mirjalili, S.M. and Lewis, A., 2014, Grey wolf optimizer, Advances in engineering software, Elsevier, (69), 46–61.

(Mochalov 15) Mochalov, V., 2015, Multi-agent bio-inspired algorithms for wireless sensor network design, Advanced Communication Technology (ICACT), 17th International Conference on, IEEE, 33–42.

(Moraitis 06) Moraitis Pavlos, Spanoudakis Nikolaos, 2006, THE GAIA2JADE PROCESS FOR MULTI-AGENT SYSTEMS DEVELOPMENT, Applied Artificial Intelligence, 20:251–273, Taylor & Francis Group, LLC.

(Nagpal 03) Nagpal, R., 2003, A catalog of biologically-inspired primitives for engineering self-organization, International Workshop on Engineering Self-Organising Applications, Springer, 53–62.

(Nagpal 08) Nagpal Radhika, Yu Chih-han, Yamins Daniel, 2008, Demo: Engineering Self-Organizing Multi-Agent Systems , Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008), Estoril, Portugal, pp 1717.

(Nallaperuma 13) Nallaperuma, S., Wagner, M., Neumann, F., 2013, Ant colony optimisation and the traveling salesperson problem: Hardness, features and parameter settings. Proceedings of the 15th annual conference companion on Genetic and evolutionary computation, ACM, 13–14.

(Neumann 06) Neumann, F., Witt, C., 2006, Runtime analysis of a simple ant colony optimization algorithm. in Proc. 17th Int. Symp. Algorithms Comput. (ISAAC), LNCS vol. 4288. Kolkata, India, Berlin, Germany: Springer-Verlag, 618–627.

(Neumann 09) Neumann, F., Sudholt, D., Witt, C., 2009, Analysis of different MMAS ACO algorithms on unimodal functions and plateaus. Swarm Intelligence, Springer, 3(1), 35–68.

(Neumann 10) Neumann, F., Witt, C., 2010, Ant colony optimization and the minimum spanning tree problem. Theoretical Computer Science, Elsevier, 411(25), 2406–2413.

(Olaifa 15) Olaifa, M., Mapayi, T. and Van Der Merwe, R., 2015, Multi Ant LA: An adaptive multi agent resource discovery for peer to peer grid systems, Science and Information Conference (SAI), IEEE, 447–451.

(O'Malley 01) O'Malley S. A. and DeLoach S. A., 2001, Determining when to use an agent-oriented software engineering. In Proceedings of the Second International Workshop On Agent-Oriented Software Engineering (AOSE-2001), pages 188–205, Montreal.

(Padgham 02) Padgham Lin Winikoff Michael, 2002, Prometheus: A Pragmatic Methodology for Engineering Intelligent Agents, Rapport, RMIT University.

(Padgham 03) Padgham Lin and Winikoff Michael, 2003, The Prometheus Methodology, Agent-oriented software engineering III, 2003 – Springer.

(Padmanaban 16) Padmanaban, R., Thirumaran, M., Suganya, K. and Priya, R.V., 2016, AOSE Methodologies and Comparison of Object Oriented and Agent Oriented Software Testing, Proceedings of the International Conference on Informatics and Analytics, ACM, 119.

(Parunak 97) Parunak, H.V.D., 1997, "Go to the ant": Engineering principles from natural multi-agent systems, Annals of Operations Research Journal, JC BALTZER AG, (75), 69–102.

(Pavón 03) Pavón Juan and Gómez-Sanz Jorge, V. Mařík et al., 2003, Agent Oriented Software Engineering with INGENIAS, (Eds): CEEMAS 2003, LNAI 2691, pp. 394-403, 2003. Springer-Verlag Berlin Heidelberg.

(Pellegrini 06) Pellegrini, P., Favaretto, D., Moretti, E., 2006, On MAX-MIN ant system's parameters. In Proceedings of the 5th International Conference on Ant Colony

Optimization and Swarm Intelligence (ANTS), Springer, 203–214.

(Perez-Cara 16) Perez-Carabaza, S., Besada-Portas, E., Lopez-Orozco, J.A. and de la Cruz, J.M., 2016, A Real World Multi-UAV Evolutionary Planner for Minimum Time Target Detection, Proceedings of the 2016 on Genetic and Evolutionary Computation Conference, ACM, 981–988.

(Perez-Diaz 15) Perez-Diaz, F., Zillmer, R. and Groß, R., 2015, Firefly-Inspired Synchronization in Swarms of Mobile Agents, Proceedings of the International Conference on Autonomous Agents and Multiagent Systems, International Foundation for Autonomous Agents and Multiagent Systems, 279–286.

(Picard 04) Picard Gauthier, Gleizes Marie-Pierre, 2004, Chapter 8: THE ADELFE METHODOLOGY Designing Adaptive Cooperative Multi-Agent Systems, Published in F. Bergenti, M.-P. Gleizes, and F. Zambonelli, editors, Methodologies and Software Engineering for Agent Systems, pages 157-176. Kluwer Publishing.

(Qian 16) Qian, B. and Cheng, H.H., 2016, A mobile agent-based coalition formation system for multi-robot systems, Mechatronic and Embedded Systems and Applications (MESA), 2016 12th IEEE/ASME International Conference on, IEEE, 1–6.

(Rehberger 16) Rehberger, S., Spreiter, L. and Vogel-Heuser, B., 2016, An agent approach to flexible automated production systems based on discrete and continuous reasoning, Automation Science and Engineering (CASE), IEEE International Conference on, IEEE, 1249–1256.

(Rowley 98) Rowley, H.A., Baluja, S. and Kanade, T., 1998, Neural network-based face detection, Pattern Analysis and Machine Intelligence Journal, IEEE Transactions on, IEEE, (20), (1), 23–38.

(Silva 10) Silva, D.C., Braga, R.A-M. and Reis, L.P., and Oliveira, E., 2010, A generic model for a robotic agent system using GAIA methodology: Two distinct implementations, Robotics Automation and Mechatronics (RAM), IEEE Conference on, IEEE, 280–285.

(Skormin 01) Skormin Victor A., Delgado-Frias Jose G., McGee Dennis L., Giordano Joseph V., Popyack Leonard J., Gorodetski Vladimir I., Tarakanov Alexander 0. BASIS: A Biological Approach to System Information Security, Information Assurance in Computer Networks: Methods, Models and Architectures for Network Security, p127-142, Springer 2001.

(Smets 98) Thibaut Smets, Nicolas Van den Berghe, 1998, Les agents intelligents. Travail réalisé dans le cadre du cours ELEC2920 : Réseaux de télécommunication.

(Stützle 98) Stützle, T., Hoos, H., 1998, Improvements on the ant-system: Introducing the max-min ant system. Artificial Neural Nets and Genetic Algorithms, Springer, 245–249.

(Stützle 02) Stützle, T., Dorigo, M., 2002, A short convergence proof for a class of Ant Colony Optimization algorithms. IEEE Trans on Evolutionary Computation, 358–365.

(Stützle 04) Stützle, T., 2004, Acotsp, version 1.0, Available from http://www.acometaheuristic.org/aco-code.

(Stützle 11) Stützle, T., López-Ibáñez, M., Pellegrini, P., Maur, M., Montes de Oca, M., Birattari, M., Dorigo, M., 2011, Parameter Adaptation in Ant Colony Optimization. Autonomous Search, Springer, 191–215.

(Sturm 14a) Sturm, A. and Shehory, O., 2014, Agent-oriented software engineering: revisiting the state of the art, Agent-Oriented Software Engineering, Springer, 13–26.

(Sturm 14b) Sturm, A. and Shehory, O., 2014, The landscape of agent-oriented methodologies, Agent-Oriented Software Engineering, Springer, 137–154.

(Talbi 00) Talbi E-G., 2000, Méthodes d'optimisation avancées, cours.

(Tarakanov 01) Tarakanov, A., 2001, Information security with formal immune networks, Information Assurance in Computer Networks Journal, LNCS, Springer, (2052), 115–126.

(Tveit 01) Tveit A., 2001, A survey of Agent-Oriented Software Engineering, First NTNU Computer Science Graduate Student Conference, NTNU-Trondheim, Norwegian University of Science and Technology.

(Tsang 05) Tsang, C.H. and Kwong, S., 2005, Multi-agent intrusion detection system in industrial network using ant colony clustering approach and unsupervised feature extraction, Industrial Technology, ICIT 2005. IEEE International Conference on, IEEE, 51-56.

(Wang 15) Wang, J., Cao, J., Li, B., Lee, S. and Sherratt, R. S., 2015, Bio-inspired ant colony optimization based clustering algorithm with mobile sinks for applications in consumer home automation networks, IEEE Transactions on Consumer

Electronics, IEEE, (61), (4), 438–444.

(Weyns 06) Weyns, D., Parunak, H.V.D. and Michel, F., 2006, Environments for Multi-Agent Systems II, Second International Workshop, E4MAS 2005, Utrecht, The Netherlands, July 25, 2005, Selected Revised and Invited Papers, Springer, (3830).

(Winikoff 04) Winikoff Michael and Padgham Lin. , 2004, The Prometheus methodology. In Federico Bergenti, Marie-Pierre Gleizes, and Franco Zambonelli, editors, Methodologies and Software Engineering for Agent Systems, chapter 11. Kluwer Academic Publishing (New York).

(Wooldridge 95) Wooldridge M., Jennings N.R., 1995, Intelligent agents: Theory and practice. The Knowledge Engineering Review, 10(2) pp 115-152.

(Wooldridge 98) Wooldridge Michael, Jennings Nicholas R., 1998, Pitfalls of Agent-Oriented Development; AGENTS '98: Proceedings of the second international conference on Autonomous agents.

(Wooldridge 99) Wooldridge Michael, 1999, Intelligent Agents; Cours.

(Wooldridge 00) Wooldridge M. J., Jennings N.R., and Kinny D., 2000, The Gaia Methodology for Agent-Oriented Analysis and Design. In Autonomous Agents and Multi-Agent systems, volume 3, pages 285–312, The Netherlands.

(Xiang 08) Xiang, W. and Lee, H.P., 2008, Ant colony intelligence in multi-agent dynamic manufacturing scheduling, Engineering Applications of Artificial Intelligence Journal, Elsevier, (21), (1), 73–85.

(Zambonelli 15a) Zambonelli, F., 2015, Engineering Environment-Mediated Coordination via Nature-Inspired Laws, Agent Environments for Multi-Agent Systems IV, Springer, 63–75.

(Zambonelli 15b) Zambonelli, F., Omicini, A., Anzengruber, B., Castelli, G., De Angelis, F.L., Serugendo, G.Di M., and Dobson, S., Fernandez-Marquez, J.L., Ferscha, A., Mamei, M. and others, 2015, Developing pervasive multi-agent systems with nature-inspired coordination, Pervasive and Mobile Computing, Elsevier, (17), 236–252.

(Zeghida 03) Zeghida, D., 2003, Une architecture multi-agents d'un système d'apprentissage avec compagnons "MALCS: Multi-Agent Learning CompanionS System", Magister Thesis, Badji Mokhtar Annaba University, Algeria.

(Zeghida 04) Zeghida, D., 2004, Une architecture multi-agents d'un système d'apprentissage avec compagnons, Séminaire National en Informatique Biskra, SNIB'04, Biskra, Algeria, 21-32.

(Zeghida 12) Zeghida, D., Michel, F., Meslati, D., 2012, An agent-based model for biomorphic software systems. in: International Conference on Complex Systems, ICCS'2012. IEEE, 1–6.

(Zeghida 18a) Zeghida, D., Meslati, D., Bounour, N., 2018, Bio-IR-M: A Multi-Paradigm Modelling for Bio-Inspired Multi-Agent Systems. Informatica, International Journal of computing and Informatics, Slovenian Society Informatika, 42(3), 451-466.

(Zeghida 18b) Zeghida, D., Meslati, D., Bounour, N., and Allat Y., 2018, Agent Influence/Reaction Ant System Variants: An Experimental Comparison, International Journal of Artificial Intelligence, CESER Publications, 16(2), 60-77.

(Zhou 09) Zhou, Y., 2009, Runtime Analysis of an Ant Colony Optimization Algorithm for TSP Instances. IEEE Trans. Evolutionary Computation, 13(5), 1083–1092.

A PROPOS DE L'AUTEUR

Biographie

Djamel ZEGHIDA rejoignit l'université de Constantine pour suivre une formation en informatique et obtint le diplôme d'Ingénieur d'état, option Système d'Information en 1992. Il obtint son diplôme de Magister option Intelligence Artificielle en 2003 de l'université Badji Mokhtar-Annaba. Actuellement, il est enseignant chercheur (Maître assistant classe A) à l'université 20 Août 1955 Skikda et membre du **L**aboratoire d'**I**ngénierie des **S**ystèmes **CO**mplexes ***LISCO*** de l'université Badji Mokhtar-Annaba, qui s'active autour des thèmes de modélisation des systèmes complexes et intégration des paradigmes. Ses intérêts de recherche comprennent: l'ingénierie orientée agent, la méta-modélisation, les systèmes bio-inspirés, heuristique d'optimisation.

Courriel: dj.zeghida@gmail.com; d.zeghida@univ-skikda.dz.
Adresse:
Département d'Informatique, Université du 20 août 1955-Skikda, BP 26, 21000 Skikda.
Laboratoire LISCO, Département d'informatique, Université Badji Mokhtar, BP 12, 23000 Annaba.

Publications Internationales

Zeghida, D., Meslati, D., Bounour, N., 2018, Bio-IR-M: A Multi-Paradigm Modelling for Bio-Inspired Multi-Agent Systems. ***Informatica***, International Journal of Computing and Informatics, Slovenian Society Informatika, 42(3), 451-466.

Zeghida, D., Meslati, D., Bounour, N., and Allat Y., 2018, Agent Influence / Reaction Ant System Variants: An Experimental Comparison, ***International Journal of Artificial Intelligence***, Ceser Publications, 16(2), 60-77.

Communications Internationales

Zeghida, D., Michel, F., Meslati, D., 2012, An agent-based model for biomorphic software systems. in: International Conference on Complex Systems, ICCS'2012. IEEE, 1–6.

D. Zeghida, F. Michel, D. Meslati, 2013, Bio-IR: A model for bio-inspired multi-agent systems, International Conference on Computational Science, ICCS 2013, June 05-07, 2013 – Barcelona, Spain.