THÈSE de DOCTORAT de l'UNIVERSITÉ PARIS 6
Spécialité : Informatique

présentée par ***Elpida Tzafestas***
pour obtenir le grade de DOCTEUR de l'UNIVERSITÉ PARIS 6

Sujet de la thèse :

VERS UNE SYSTÉMIQUE DES AGENTS AUTONOMES : DES CELLULES, DES MOTIVATIONS ET DES PERTURBATIONS

soutenue le 22 décembre 1995 devant le jury composé de :

Mr. Jean-Pierre Müller	*Rapporteur*
Mr. Bernard Victorri	*Rapporteur*
Mr. Jacques Ferber	*Directeur de recherche*
Mr. Yves-Marie Visetti	*Examinateur*
Mr. Dominique Duhaut	*Examinateur*

À mes parents

Ce n'est pas la science qui fait le bonheur,
mais l'acquisition de la science.

(Edgar Allan Poe)

Table des matières

Résumé

Ce travail se situe au carrefour de l'intelligence artificielle et des systèmes multi-agents avec la robotique autonome, la vie artificielle et les approches ascendantes aux sciences cognitives. Notre aspiration à long terme étant d'élaborer une théorie causale de l'émergence et de l'évolution à partir d'un ensemble de principes applicables à tous les niveaux d'organisation et d'une fonction récursive d'émergence, nous cherchons une réponse à la question "existe-t-il un ensemble de principes d'organisation d'agents autonomes qui soient indépendants de niveau d'organisation ?". Nous proposons donc un modèle abstrait d'agent autonome de niveau quelconque d'organisation : un agent autonome est un système couplé à son environnement qui démontre des propriétés de réactivité, de motivation, de socialité et d'adaptativité.

Ces principes sont illustrés par la résolution d'un ensemble de problèmes pratiques impliquant des agents autonomes à deux niveaux d'organisation : le niveau "agent-animat cellulaire" et le niveau "agent-cellule". En premier lieu, nous nous plaçons au niveau d'un agent-animat pour lequel nous développons une organisation cellulaire servant de base pour la résolution de deux problèmes inspirés de la robotique comportementale et de la productique. La cellularité est définie comme le mode d'organisation d'un réseau de composants ayant des fonctionnalités hétérogènes mais une syntaxe de connexions homogène, et le réseau cellulaire est couplé à une physiologie indépendante. Nous étudions les deux problèmes ci-dessus pour le cas d'un seul et de plusieurs agents cellulaires et nous montrons, d'une part le besoin d'autorégulation des paramètres organisationnels de l'agent et le besoin d'une socialité du type partage du besoin, et d'autre part la transcription d'un problème de planification dans une physiologie élaborée de l'agent.

Ensuite, nous descendons au niveau de la cellule pour montrer comment ces mêmes principes peuvent être utilisés pour assurer la plasticité et l'intégrité du réseau cellulaire face à des pannes imprévues. Nous montrons plus précisément comment un réseau de cellules étant à leur tour des agents motivés, sociaux et adaptatifs, peut démontrer de la dynamicité et de la plasticité topologique. Nous montrons également le besoin d'autorégulation des paramètres organisationnels de la cellule et le besoin d'un système cellulaire immunitaire complémentaire au précédent.

Finalement, nous proposons la sénescence comme force motrice de l'émergence de structures d'organisation d'ordre supérieur et de l'apprentissage et nous développons un modèle de sénescence qui répond à toutes les spécifications nécessaires. Plus précisément, nous montrons qu'une fonction de sénescence reposant sur des rétroactions négatives du métabolisme de l'agent appliquées à son programme favorise les modèles sociaux et plus "cognitifs" d'agents, sans préspécifier en détail leur durée de vie.

Dans toutes ces études, nous montrons l'importance des dynamiques d'interaction agent-monde et le besoin de réguler et d'intégrer plusieurs dynamiques au sein d'un système. Les perspectives de l'approche se situent principalement dans les voies de l'extension des modèles cellulaires développés et de l'apprentissage cellulaire ou social.

Abstract

This work is situated at the intersection of artificial intelligence and multi-agent systems with autonomous robotics, artificial life and comparative cognitive science. The long term objective being to elaborate a causal theory of emergence and of evolution out of a set of principles applied to all organizational levels and a recursive emergence function, we seek an answer to the question "is there a set of organizational principles of autonomous agents, independently of organizational level ?". We propose therefore an abstract model of autonomous agents at any organizational level : an autonomous agent is a system coupled to its environment that shows properties of reactivity, motivation, sociality and adaptivity.

These principles are illustrated through the solution of a set of practical problems involving autonomous agents of two organizational levels : the "cellular animat-agent" level and the "cell-agent". We develop first a cellular organization for an animat-agent and we demonstrate its properties on two problems inspired by behavior-based robotics and production engineering. Cellularity is defined as the mode of network organization in which the component functionalities are heterogeneous but the connection syntax is homogeneous, and the cellular network is coupled with an independent physiology. We study those problems for a single or multiple cellular agents and we show, on the one hand the need to self-regulate the agent's organizational parameters and the need for a "shared needs" sociality, and on the other hand the transcription of a planning problem into an elaborate agent physiology.

Next, we descend to the cell level and we show how the same general principles may be used to ensure the network's plasticity and integrity in front of unpredictable failures. More precisely, we show how a network of cells being themselves motivated, social and adaptive agents, may show topological dynamicity and plasticity. Furthermore, we show the need for self-regulation of the cell's organizational parameters and the need for an immune cellular system complementary to the previous one.

Finally, we propose senescence as the motor of emergence of higher-order structures and of learning and we develop a senescence model that meets all the specifications set. To this end, we show that a senescence function founded on negative feedback of the agent's metabolism upon its program favors social and more cognitive agent models without prespecifying in detail their life span.

Throughout those studies, we demonstrate the importance of the dynamics of the agent-environment interactions and the need to regulate and integrate multiple interaction dynamics within a system. The perspectives of the approach lie mainly in the directions of extending the cellular models studied so far and of cellular or social learning.

Remerciements

Un thésard ressemble à un apprenti de tennis. Avoir en face de soi un joueur comme Jacques Ferber m'a permis de tout bien apprendre (la défense, l'attaque, le service, le jeu contrôlé et prédictif ...) et de développer mon propre jeu. Merci !

Je remercie Bernard Victorri, Dominique Duhaut et Yves-Marie Visetti d'avoir commenté ce travail et de m'avoir aidée à éclaircir ces idées et à améliorer le texte. Je remercie également Jean-Pierre Müller d'avoir fait partie de mon jury de thèse. Je remercie vivement Rodney Brooks de m'avoir permis d'espionner dans son laboratoire pendant la période la plus perturbée de ma vie et d'avoir ainsi marqué mon développement scientifique. Je remercie ensuite tous ceux qui m'ont encouragée au cours de ces années, mais aussi tous ceux qui se sont opposés à mes idées et à moi-même : j'en suis sortie plus forte.

Je remercie toute ma famille de son soutien constant et mes parents de m'avoir appris à réfléchir, à travailler et à être indépendante. Je remercie plus particulièrement mon père de m'avoir convaincue au bon moment que je pourrais faire de l'art en faisant de l'intelligence artificielle. Je remercie également tous mes amis de leur compréhension, de leurs encouragements et de leur aide. Merci aux compagnons du bureau, Claude Delaye, Madeleine Girard, Steffen Lalande, Laurent Magnin et les autres, pour les discussions, les critiques et leur amitié. Merci encore à Philippe Darche pour avoir partagé avec enthousiasme mes préoccupations à l'égard d'une nouvelle robotique distribuée.

Je remercie tous les grands compositeurs, de Beethoven à Kottke, qui m'ont tenu compagnie pendant les longues heures d'étude, de réflexion et de rédaction. Je remercie finalement mon ancien maître de musique de m'avoir appris à ne jamais suivre les règles (des autres).

Pré-texte

Les choses, les lieux et les gens. Comme tout jeune scientifique, je suis entrée en thèse avec la volonté de faire des choses intéressantes, de contribuer et d'être créative. Suivant la conviction répandue que les choses ne se font que dans un contexte particulier, ma trajectoire est passée par plusieurs étapes et par plusieurs villes et elle m'a offert l'occasion de connaître beaucoup de gens d'origines, de tempéraments et de cultures très variés. Ces années de thèse m'ont encore donné beaucoup de joies, mais aussi beaucoup de peines, au niveau personnel. Un travail de plusieurs années ne peut qu'être le fruit de toute l'expérience et de toutes les préoccupations cumulées pendant ces années, il doit donc être lu et évalué ainsi, c'est-à-dire avant tout comme un travail personnel façonné dans son milieu.

À la recherche de soi-même. Au sortir de cette période, je regarde en arrière : il existe un fil que je ne voyais pas avant, un fil reliant beaucoup de choses qui m'ont inspirée ou impressionée depuis mon enfance et qui semblaient jusqu'à maintenant disparates. Cela ne peut pas être au hasard qu'en tant qu'élève ingénieur je me suis intéressée à la question de la liberté de l'esprit et de l'autonomie. Et le relativisme d'une phrase de Russell qui m'avait saisie lorsque j'avais seize ans (... la charge est le nom que nous donnons à une propriété de la matière que nous observons ...) est le même que celui qui m'avait fascinée bien des années plus tard dans les écrits de Brooks (... l'intelligence se trouve dans l'oeil de l'observateur ...). De même, la lecture (et relecture) préférée de mon enfance a été l'île mystérieuse de Jules Verne, mais seulement la partie avant l'apparition des petits cadeaux de capitain Némo : c'est justement une des citations favorites de Jacques Pitrat lorsqu'il parle à ses étudiants de l'importance de la métaconnaissance. Je ne vais pas énumérer d'autres exemples, mais je tiens à affirmer qu'au sortir de cette thèse j'en sais plus sur moi-même et sur ce mystère à mon intérieur qui m'a poussée à arriver jusqu'ici. Derrière les mots, les figures et les résultats, j'espère avoir réussi à décrire cette évolution d'idées et à présenter les fragments de ce rêve conducteur qui n'est pas encore entièrement sculpté.

Introduction

*" ΕΝ ΟΙΔΑ, ΟΤΙ ΟΥΔΕΝ ΟΙΔΑ." **
(ΣΩΚΡΑΤΗΣ)

La systémique des agents

> *"Il y a une cohérence dans les descriptions de la science, une unité dans les explications qui témoigne d'une unité sous-jacente dans les entités et les principes en jeu. Quel que soit leur niveau, les objets d'analyse sont toujours des organisations, des systèmes. Chacun d'eux sert d'ingrédient au suivant."*
> *(François Jacob, "La logique du vivant", 1970)*

La nouvelle vague de l'intelligence artificielle et de la robotique est une approche ascendante, souvent qualifiée de comportementale (*behavior-based AI, behavior-based robotics*). Le courant analogue dans le domaine des sciences cognitives au sens large est l'approche animat. Tous les deux reposent sur la conception d'artefacts dont la propriété principale est la possibilité de "survivre" dans un monde imprévisible pendant la poursuite de leurs propres buts. Ces artefacts sont appelés des agents autonomes. D'autre part, les systèmes multi-agents constituent l'avant-garde des recherches en intelligence artificielle distribuée et mettent l'accent sur une problématique d'intelligence collective. La discipline de la vie artificielle, de son côté, part de la même hypothèse de complexité émergente à partir d'interactions locales simples.

Ce travail se situe au carrefour de l'intelligence artificielle et des systèmes multi-agents avec la robotique autonome, la vie artificielle et les approches ascendantes aux sciences cognitives.

L'organisation d'un agent est l'ensemble des relations qui le définissent comme une unité (Varela 1979). Un agent peut être composé récursivement de plusieurs agents, auquel cas on parle d'organisations récursives à plusieurs niveaux. Par exemple, un agent vivant pluricellulaire est constitué de plusieurs autres agents vivants, les cellules. De même à une autre échelle, les sociétés sont constituées de plusieurs autres agents.

> ***Objectif :*** *Il s'agit d'élaborer des principes d'organisation d'agents autonomes qui soient indépendants du niveau d'organisation, donc des principes récursifs. Ces principes sont identifiés et illustrés par la résolution d'un ensemble de problèmes pratiques impliquant des agents autonomes à deux niveaux d'organisation : celui d'un agent-animat et celui d'un agent-cellule (composant du système de contrôle de l'agent-animat).*

Notre aspiration à plus long terme est d'étudier les organisations émergentes et évolutives à plusieurs niveaux et d'élaborer une théorie causale de l'émergence et de

* *"Je sais seulement que je ne sais rien."* (Socrate)

l'évolution. Nous partons alors de l'hypothèse qu'un ensemble de principes applicables à tous les niveaux d'organisation et une "fonction (ou force) d'émergence" récursive ou inductive faisant appel à ces principes sont les éléments constitutifs de cette théorie. Cette thèse se limite à la première partie de l'hypothèse, c'est-à-dire de montrer qu'il existe un ensemble de principes applicables à tous les niveaux d'organisation et se focalise, pour ce faire, sur l'étude de deux niveaux successifs d'organisation. Notre démarche est donc une démarche systémique (au sens de von Bertalanffy 1968 et Weinberg 1975) : il s'agit de résoudre un ensemble de problèmes particuliers, ensuite de comparer *qualitativement* ces solutions et élaborer des principes suffisamment abstraits et généraux pour s'appliquer à tous les problèmes étudiés. C'est encore une démarche systémique au sens de Varela (1979) : à tous les niveaux il y a des systèmes *autonomes* et l'autonomie phénoménale de ces systèmes est due à un ensemble de mécanismes de régulation et d'auto-organisation. Notre perspective est d'aborder dans le futur le problème complet et de valider notre hypothèse à tous les niveaux à partir des études présentées ici, ce qui explique le titre de la thèse.

Les principes en résumé

Avant de présenter brièvement les principes d'organisation d'agents autonomes, il est important de souligner qu'il s'agit d'agents autonomes *conçus* spécialement pour la résolution de quelques problèmes particuliers, donc des agents qui ne soient a priori pas autotéliques. Cependant, un observateur externe, qui n'a aucune idée de ce qui se passe à l'intérieur de l'agent ni du problème que celui-ci est en train de résoudre, aura probablement tendance à lui attribuer des propriétés et des mécanismes rencontrés chez les agents naturels, par exemple chez les animaux. Ainsi faut-il bien distinguer par la suite les trois cadres de référence (Clancey 1989) : celui du concepteur, celui de l'agent et celui d'un observateur externe. Ce qui est intéressant dans cette distinction, c'est le rapport entre la réalité "causale" de l'agent (c'est-à-dire le cadre de référence du concepteur) et le modèle de cette réalité telle qu'elle est perçue (le cadre de référence de l'observateur). Dans ce sens, des études comme celles présentées ici peuvent donner des modèles computationnels de phénomènes naturels, et cela dans un monde artificiel et dans un contexte d'ingénierie.

Un agent autonome est *réactif* : il répond aux aspects imprévisibles de son environnement.

Un agent autonome est *motivé* : il a une mission qui est "décrite" dans son organisation comme un besoin qu'il veut ramener à 0. La motivation de l'agent exprime à la fois la volonté de l'agent pour accomplir sa mission et l'image ou l'idée que l'agent a du monde. L'état du monde tel qu'il est perçu par l'intermédiaire de ses systèmes de perception constitue une *perturbation* à cette idée.

Dans un contexte multi-agents, un agent autonome est *social* : il a une socialité du type "partage du besoin".

Un agent autonome est *adaptatif* et *opérationnellement couplé* avec son environnement : il existe un couplage, une correspondance entre agent et monde, telle que l'agent doit s'autoréguler continûment pour accomplir sa mission et pour être opérationnel. L'autorégulation de l'agent concerne un ou plusieurs paramètres organisationnels qui déterminent la réaction de l'agent à ses perturbations. Ces

paramètres organisationnels autorégulés ont une signification temporelle — ils sont par exemple des taux ou des vitesses d'adaptation — et déterminent donc la dynamique de l'interaction de l'agent avec son monde, ou la dynamique de la perturbation.

Dans tous les cas, l'importance de l'intégration et de la régulation de plusieurs dynamiques d'interaction avec le monde est démontrée et soulignée.

Ces principes ont été extraits des solutions à un ensemble des problèmes (cf. ci-dessous) qui impliquent des agents autonomes à deux niveaux d'organisation. Nous n'avons cependant aucune preuve (et même aucune indication) que ces solutions et ces principes sont uniques — nous n'avons non plus aucun moyen logique ou analytique pour examiner s'il y en a d'autres alternatives. Il s'agit simplement de trouver *une* réponse à la question "existe-t-il un ensemble de principes applicables à tous les niveaux d'organisation ?".

Les principes d'organisation sont présentés et analysés dans le chapitre 1. La résolution des divers problèmes abordés fait référence à la terminologie et aux principes introduits dans ce chapitre ; pour un rappel continu de la définition des différents termes employés, un glossaire est mis à la disposition du lecteur.

Les problèmes

Problème 1 : *Quels sont les principes généraux d'organisation du système de contrôle d'un agent autonome situé ?*

Il s'agit d'une grande question ouverte pour les systèmes autonomes, la vie artificielle et la robotique. Nous n'aurons probablement jamais qu'une réponse partielle. Nous proposons un réseau cellulaire métabolique couplé avec une physiologie indépendante qui démontre un ensemble de propriétés importantes et de facilités en vue de la conception d'agents autonomes. L'étude porte aussi sur l'éclaircissement de certains points concernant l'action et sur l'identification de plusieurs principes de conception et d'organisation, tels que les principes de la séparation des différentes dynamiques. Cette première étude se situe à un niveau structurel d'implémentation et est présentée dans le chapitre 3. Les problèmes 2 à 4 concernent les rapports entre tâche de l'agent et principes d'organisation et sont présentés dans les chapitres 4 à 6 respectivement.

Problème 2 : *Quelle est la particularité du système de contrôle d'un agent devant "connaître" le paramètre critique du problème qu'il résout, c'est-à-dire celui dont la valeur dénote l'état d'avancement de la résolution, et donc celui qui est directement affecté par l'activité de l'agent ?*

Cette question est abordée dans un problème d'exploration, typique en robotique comportementale : par exemple, comment concevoir un robot ayant comme mission de ramasser toutes les pierres noires dans les couloirs d'une station de métro. Le paramètre critique est la quantité des pierres noires. L'intérêt de ce problème dans l'étude actuelle porte sur l'instanciation des principes généraux d'organisation, et plus particulièrement sur la notion de motivation et d'autorégulation de dynamique. L'agent doit posséder une représentation du paramètre critique et autoréguler continûment la dynamique de son interaction avec le monde afin de rester opérationnel, c'est-à-dire afin d'avoir des performances satisfaisantes.

Problème 3 : *Quelle est la particularité du système de contrôle de l'agent du problème 2 qui doit résoudre son problème collectivement avec d'autres agents ?*

C'est la suite du problème précédent dans le cas de plusieurs agents. Les agents doivent être sociaux pour assurer l'opérationalité du système multi-agents et cette socialité doit être conforme aux principes généraux décrits. Plusieurs variantes de comportements non opérationnels sont également analysées et contrastées avec les modèles de socialité présentés.

Problème 4 : *Quelle est la particularité du système de contrôle d'un agent chargé de résoudre un problème nécessitant un certain degré de "planification" ?*

Pour l'étude de cette question, nous choisissons un problème de productique vu sous l'angle de la robotique comportementale : comment concevoir un robot chargé de desservir et de réguler les demandes d'un ensemble d'unités de production dans un atelier industriel ? Ce problème est sujet à moins d'imprévisibilité et à davantage de besoin de planification que le problème de l'exploration (problème 2). Les mêmes principes généraux s'appliquent, mais cette fois la partie physiologique du système de contrôle doit être plus élaborée que celle de l'agent explorateur.

Problème 5 : *Quelle doit être l'organisation du système de contrôle de l'agent cellulaire du problème 1 lui permettant d'être robuste face à des pannes imprévues de matériel, c'est-à-dire face à une perte de cellules ?*

C'est un problème qui a un intérêt pratique : il concerne l'implémentation distribuée du système de contrôle cellulaire d'un robot, qui doit être robuste face à des pannes imprévues de matériel. La solution repose sur les mêmes principes généraux, et plus spécifiquement sur la motivation, la socialité et l'autorégulation de dynamique, s'appliquant cette fois aux composants de l'agent-animat, les cellules : les cellules sont donc des agents autonomes au même droit que les agents-animats. Il est aussi démontré qu'un processus de "dégradation naturelle" induisant des pannes au réseau peut jouer le rôle de stimulus de l'auto-organisation et de force motrice de l'apprentissage. Le besoin d'un système immunitaire est également identifié. Ce problème est présenté dans le chapitre 7.

Problème 6 : *À quoi pourrait ressembler une fonction de sénescence et quelles organisations favoriserait-elle ?*

Ce sixième et dernier problème est présenté dans le chapitre 8 et repose sur une argumentation concernant la nature et l'origine de l'autonomie de même que sur l'identification du besoin d'une fonction de sénescence comme force motrice de l'apprentissage et de l'émergence des structures de niveau d'organisation supérieur. Cette argumentation est donnée dans le paragraphe 1.3.6 et l'intérêt de la solution donnée réside dans le fait qu'elle favorise les relations sociales complexes et les mécanismes progressivement plus "cognitifs". De plus, le modèle de sénescence élaboré ne préspécifie pas la durée de vie d'un agent ; celle-là dépend des interactions avec les autres agents. La solution montre également le rôle des dynamiques d'interaction au sein de l'organisation.

Plan de lecture

Le premier chapitre présente la terminologie et les principes d'organisation d'agent autonome.

Le chapitre 2 passe rapidement en revue la littérature des agents autonomes tels qu'ils sont définis et utilisés dans des diverses disciplines.

Ensuite, le chapitre 3 présente le problème 1, celui de la description et de la spécification de l'organisation d'un agent-animat. Avec la cellularité pour hypothèse de base, l'intérêt d'avoir un ensemble de cellules et un ensemble de variables globales physiologiques est révélé. Un ensemble de principes de conception sont identifiés (principes de séparation des dynamiques, d'indépendance des actionneurs etc.), qui simplifient la spécification et la conception d'un système de contrôle cellulaire. Les principes sont illustrés sur deux exemples d'agents cellulaires, un agent explorateur et un agent manager dans un atelier industriel. Le problème de la sémantique et de l'espace de représentation dans la cellularité sont également discutés.

Les chapitres 4 et 5 présentent le problème de l'exploration, pour le cas d'un seul ou de plusieurs agents-explorateurs, respectivement (problèmes 2 et 3). Avec l'organisation cellulaire du chapitre 3 comme substrat d'implémentation informatique, les principes du chapitre 1 sont instanciés. Le chapitre 4 montre le besoin d'un système motivationnel et d'un mécanisme d'autorégulation de dynamique pour assurer l'opérationalité de l'agent dans des mondes inconnus et imprévisibles. Le chapitre 5 démontre le besoin d'une socialité du type "partage du besoin" entre agents explorateurs.

Le chapitre 6 est dédié au problème 4, celui de la régulation d'un atelier industriel par un agent manager. Si ce problème ne possède pas les caractéristiques des problèmes 2 et 3, il montre en revanche un plus grand besoin de planification qui se traduit par une physiologie plus élaborée de l'agent.

Le chapitre 7 présente le problème de l'opérationalité et de l'intégrité du réseau cellulaire face à des pannes de matériel (problème 5). Les cellules aussi doivent être des agents autonomes au sens du modèle général, c'est-à-dire des agents possédant un système motivationnel, un mécanisme d'autorégulation de dynamique et une socialité du type "partage du besoin". Le problème de l'espace de représentation dans le réseau est rediscuté et le besoin d'un système immunitaire est montré.

Ensuite, le chapitre 8 étudie le problème de la sénescence (problème 6), vue à la fois comme force motrice de l'apprentissage et comme base de l'émergence récursive de structures organisationnelles. Le modèle de sénescence élaboré répond à toutes les spécifications comportementales et repose sur un lien de rétroaction entre programme et métabolisme de l'agent. Ce modèle de sénescence favorise les mécanismes de régulation récursive, dans le sens où ceux-ci augmentent la durée de vie de l'agent.

Finalement, le chapitre 9 fait le bilan de toutes ces études. Les agents autonomes des deux niveaux étudiés suivent les mêmes principes généraux : réactivité, motivation, socialité et autorégulation. De plus, l'importance de l'intégration et de la régulation de plusieurs dynamiques au sein d'un même agent est soulignée.

Les chapitres 3, (4+5), 6, 7 et 8 sont indépendants et peuvent être lus dans un ordre aléatoire à la suite de la lecture du premier chapitre.

Chapitre 1 Les agents en général

1.1 Les agents autonomes

> *Agent (Dictionnaire Larousse) n.m. 1. Toute personne ou tout phénomène physique qui a une action déterminante. 2. Celui qui est chargé d'une mission par une société, un gouvernement, un particulier.*
>
> *Agent (Ferber 1989a, p. 249). On appelle agent une entité réelle ou abstraite qui est capable d'agir sur elle-même et sur son environnement, qui dispose d'une représentation partielle de cet environnement, qui, dans un univers multi-agents, peut communiquer avec d'autres agents, et dont le comportement est la conséquence de ses observations, de sa connaissance et des interactions avec les autres agents.*[1]

Un agent artificiel est un artefact créé par l'homme et destiné à accomplir une mission, résoudre un problème ou réaliser une tâche. Par définition alors, un agent doit être "autonome", c'est-à-dire il doit être capable d'accomplir sa ***mission*** et réaliser sa ***tâche*** sans supervision ou intervention de son concepteur humain. Tout comme un agent de police connaît son rôle suffisamment bien pour ne pas nécessiter la supervision des autres citoyens, un agent artificiel doit connaître sa mission suffisamment bien pour ne pas nécessiter la supervision de son maître ; il doit être suffisamment autonome pour que ce maître lui fasse confiance. Ce degré d'autonomie individuelle au sens de l'indépendance ne peut être assuré que si l'agent est réactif, c'est-à-dire s'il répond aux aspects dynamiques de son environnement.

Inversement, on a besoin de concevoir de tels artefacts-agents lorsque les missions en question sont mal définies ou nécessitent une interaction continue entre l'artefact-agent et son monde dont les détails ne sont pas connus d'avance. Il y a actuellement un courant important de pensée, selon lequel la propriété principale de l'intelligence est l'autonomie (cf. par exemple Covrigaru & Lindsay 1991, Meystel 1991)[2]; par conséquent, les problèmes impliquant des agents autonomes sont considérés comme une super-classe des problèmes de l'intelligence artificielle. Dans ce contexte d'ingénierie ou utilitariste, les agents autonomes représentent donc des solutions particulières à des problèmes spécifiques qui montrent ce besoin d'interaction continue. Deux exemples typiques d'agents autonomes sont un robot chargé de localiser et collectionner les boîtes vides de soda (Connell 1990) et un agent logiciel

[1] Une description plus détaillée des différentes possibilités d'un agent est donnée dans (Ferber 1995, p. 13).

[2] En effet, cette idée est le contrepied de la conception classique de la robotique, selon laquelle les machines autonomes sont une sous-classe des machines intelligentes (qui est due au caractère rudimentaire des premiers systèmes "intelligents" non situés).

d'interface (Maes 1994*b*) chargé d'explorer Internet afin de localiser un fichier qui porte certaines caractéristiques.

Dans beaucoup de cas, les "missions" nécessitent un *ensemble* ou une société d'agents autonomes ou un *système* multi-agents plutôt qu'un agent isolé. La raison en est triple :

(a) *Émergence de comportements ou structures complexes* : La complexité potentielle du comportement observable comme un résultat des interactions entre les agents est très grande. Un ensemble limité de tâches simples distribuées aux membres d'une société suffit pour donner naissance à des phénomènes complexes comparables à ceux rencontrés dans la nature (Huberman 1988; Lindgren 1991). Au lieu de créer un agent très sophistiqué pour répondre au besoin d'une mission complexe, il est donc souvent préférable de créer toute une société d'agents plus simples dont les interactions auront comme effet de bord l'accomplissement de la mission.

(b) *Robustesse* : Une société est robuste dans le sens où la détérioration ou même la disparition de certains de ses membres n'empêche pas sa survie et son évolution. Or, souvent cette redondance est un élément constitutif des tâches en question.

(c) *Automatisation de conception* : Il y a des tâches, et notamment des tâches de robotique, impliquant par définition plusieurs agents, par exemple l'exploration, la supervision ou la récolte. Une société d'agents, plutôt qu'un agent isolé, montre l'avantage de permettre une "production" massive, qui entraîne une diminution des coûts effectifs de production en même temps qu'une augmentation de qualité (Flynn 1987).

Dans un contexte moins utilitariste que celui de l'ingénierie, les agents autonomes sont utilisés comme des bancs d'essais de la vie artificielle (Gutowitz 1993; Maes 1994*a*; Steels 1994*a*) et des approches ascendantes aux sciences cognitives (plus particulièrement l'approche animat, (Meyer & Wilson 1991; Meyer 1995)). La différence ne réside pas tellement dans la nature des problèmes étudiés et des solutions recherchées, mais dans la démarche même : les systèmes multi-agents et l'intelligence artificielle cherchent à comprendre comment concevoir, tandis que la vie artificielle et les sciences cognitives cherchent à comprendre en concevant pourquoi les solutions naturelles sont telles qu'elles sont.

1.2 Terminologie

Avant de présenter et d'analyser les principes d'organisation d'agents autonomes, nous donnons quelques définitions de termes de base.

Structure. Une entité matérielle ou logicielle qui détermine le comportement d'un agent. Cette entité est généralement un système de composants, c'est-à-dire un ensemble de composants ayant un certain nombre de relations entre eux. [3]

Comportement. La façon dont un agent agit. Le comportement du même agent peut varier selon le contexte dans lequel il se trouve, c'est-à-dire la même structure peut donner naissance à plusieurs comportements différents. Le comportement est

[3] La différence entre organisation et structure : *"L'ensemble des relations qui définissent une machine comme une unité constitue son organisation. L'ensemble des relations effectives entre les composants présents sur une machine concrète dans un espace donné constitue sa structure."* (Varela 1979, p. 41)

également une notion relative : deux observateurs différents peuvent décrire le fonctionnement de la même structure dans le même contexte comme deux comportements différents. Nous allons par la suite utiliser souvent l'adjectif "comportemental" au lieu de "structurel", pour montrer précisément que la structure discutée peut donner naissance à plusieurs comportements.[4]

Opérationalité. Un agent est plus opérationnel qu'un autre s'il accomplit sa mission mieux que le deuxième. Dans le contexte des agents conçus, cette définition ne pose aucun problème. En revanche, elle est relativement contradictoire avec la notion de l'opérationalité telle qu'elle est rencontrée en éthologie, selon laquelle un agent opérationnel est un agent qui parvient à maintenir ses variables essentielles dans leur zone de viabilité. Cependant, un observateur externe pourra adopter l'attitude d'un éthologue et *analyser* un agent conçu en termes de viabilité. La notion de l'opérationalité, du point de vue du concepteur de l'agent, est donc une notion principalement relative et correspond à la notion de viabilité du point de vue de l'observateur.

Représentation. Image d'un objet. L'argument cognitiviste presque unanimement accepté après le rejet du behaviorisme et retrouvé ensuite dans les fondements de l'intelligence artificielle est que le comportement intelligent présuppose la faculté de représenter le monde d'une certaine façon. Dans la mesure où sa représentation de la situation est fidèle, le comportement de l'agent sera adéquat. L'hypothèse cognitiviste classique ne se limite cependant pas à cette vision générale mais postule en plus que la cognition consiste à agir sur la base de représentations qui ont une réalité physique sous forme de code symbolique dans un cerveau ou une machine.

Connaissance. Choses connues, le savoir. En termes plus techniques, la connaissance nécessite la présence des variables dans l'organisation (en l'absence de variables, il s'agit d'un automate), donc la connaissance doit être dynamique.

Couplage. Relation d'influence mutuelle entre deux entités différentes. Pour Varela (1979) une entité et son milieu sont couplés en certains points et l'activité endogène de l'un influence l'autre et vice versa. Ces points de couplage peuvent être perçus par un observateur externe comme une sorte de "connaissance" que chacune des entités a de l'autre sans impliquer des représentations au sens classique du terme. Il existe donc de la connaissance *sans* représentation (cf. (Varela 1989) pour une présentation concise et synthétique des sciences cognitives et une discussion de tous ces concepts).

(Comportement) cognitif. Le comportement d'un agent est appelé plus cognitif que celui d'un autre, si le premier agent est plus opérationnel que le deuxième. Un observateur externe sera ainsi tenté de dire que le premier agent possède plus de "connaissance" que le deuxième et cette possibilité d'utiliser de connaissances constitue son avantage opérationnel. Une fois de plus, qualifier un agent ou un modèle de "cognitif", n'a donc qu'un sens relatif et cette relation est transitive. Selon les définitions précédentes, un agent couplé à son environnement peut démontrer un comportement cognitif sans posséder des représentations.

---

[4] Dans le contexte de la robotique comportementale, il a été déjà souligné que l'utilisation du terme "comportement" peut conduire à une confusion : *"Sometimes the approach is called behavior-based as the computational components tend to be direct behavior producing modules. [Unfortunately, this clashes a little with the meaning of behavior as used by ethologists as an observed interaction with the world, rather than as something explicitly generated.]"*, (Brooks 1991*b*, p. 2 et note de bas de page 2).

Dynamique. Modification continue dans le temps. Le comportement de l'agent doit varier continûment en fonction de l'environnement ; sa réponse aux perturbations est donc la modification de soi-même. Pour van Gelder et Port (1995) l'étude de la cognition doit se faire en adoptant une approche dynamique. Dans notre modèle d'agent autonome, la dynamique constitue l'aspect du comportement le plus important ; deux propriétés semblent primordiales, l'intégration de diverses dynamiques dans le même système (cf. chapitres 3, 5, 6 et 8) et la régulation des paramètres des dynamiques (cf. chapitres 4, 7 et 8).

Émergence. L'apparition d'une structure nouvelle ou d'un comportement nouveau d'un niveau supérieur d'organisation. Qualification d'une structure ou d'un comportement comme émergent : *"[A behavior] is emergent if it can only be defined using descriptive categories that are not necessary to describe the behavior of the constituent components"* (Steels 1994*a*, p. 78).

Motivation. Si la réactivité constitue une propriété primordiale dans un monde imprévisible, une motivation innée de l'agent est tout aussi importante. Cette motivation exprime en effet la volonté individuelle de l'agent pour agir et résoudre le problème posé ; inversement, elle constitue "l'image" ou "l'idée" que l'agent a du monde et donc elle représente une mesure de l'état d'avancement de la résolution. L'état du monde tel qu'il est perçu par l'intermédiaire de ses systèmes de perception constitue une ***perturbation*** à cette idée.

Socialité. Si l'agent partage le problème avec d'autres agents, il faut une forme d'interaction qui va induire une perturbation supplémentaire de provenance sociale. Bien entendu, le rôle de la socialité est opérationnel : plusieurs agents doivent accomplir la mission mieux qu'un seul.

Paramètre libre de tâche. Un paramètre de la tâche de l'agent est appelé ***libre***, si sa valeur n'est pas connue d'avance et elle varie en cours de la résolution (cf. chapitres 4 et 7). S'il est possible "d'optimiser" les performances de l'agent pour une valeur spécifique de ce paramètre, il faudra l'équiper d'un moyen d'estimation et d'évaluation continue en cas de paramètres libres, autrement dit d'un moyen d'adaptation. Les paramètres libres posent donc un problème d'opérationalité.

Adaptation, apprentissage et autorégulation. Les deux notions de l'adaptation et de l'apprentissage sont utilisées avec plusieurs nuances et dans beaucoup de contextes différents. Ces nuances peuvent être regroupées en deux grands courants de pensée, celui des biologistes et celui des ingénieurs. Pour les biologistes et les autres scientifiques qui en raison de la nature de leur discipline se placent principalement en tant qu'observateurs des systèmes adaptatifs, l'adaptation signifie la possibilité de survivre en se maintenant dans les limites de viabilité et l'apprentissage signifie l'amélioration des capacités adaptatives : *"The behavior of an animat is adaptive so long as it allows the animat to survive or to fulfill its mission. This requires that the animat's essential variables be monitored and maintained within their viability zone, an ability which can be enhanced, should the animat be capable of learning which actions elicit a positive or negative reward from the environment"* (Meyer & Guillot (1994), p. 2). Pour les ingénieurs, qui se placent en tant que concepteurs de systèmes adaptatifs, l'adaptation signifie la modification d'une structure connue selon des critères tout aussi connus, tandis que l'apprentissage est l'apparition de nouvelles structures : *"Adaptation will refer to a known dynamic structure [...] and unknown parameters [...]. Learning will refer to the case where the structure itself is unknown*

[...], whether it be the dynamic structure [...] or the task structure [...]" (Slotine 1994, p. 31). Ainsi, pour les uns l'adaptation est le *phénomène central* dont l'apprentissage est un mécanisme, tandis que pour les autres l'adaptation est le *mécanisme central* dont l'apprentissage est un phénomène.

Par la suite nous adopterons le point de vue des ingénieurs et nous expliciterons le sens des termes tels qu'ils se retrouvent dans nos modèles (chapitres 3 à 8).

Adaptation : L'harmonisation d'un comportement aux exigences d'un environnement imprévisible.

Apprentissage : L'acquisition de nouvelles structures d'interaction avec le monde, autrement dit l'apparition de nouveaux comportements.

Pour s'adapter à l'environnement dans lequel il est situé, l'agent doit régler certains de ses paramètres architecturaux.[5] Nous montrons sur les problèmes 2 et 5 qu'une autorégulation, c'est-à-dire un réglage homéostatique des paramètres entre deux limites, suffit pour résoudre le problème des paramètres libres de tâche. En effet, l'autorégulation assure l'opérationalité de l'agent dans des mondes imprévisibles. La présence de limites aux valeurs des paramètres organisationnels homéostatiques signifie que l'opérationalité de l'agent est assurée seulement dans une gamme donnée des paramètres de tâche et de monde. Les paramètres autorégulés ont dans les deux cas ont un sens temporel, c'est-à-dire ils sont des taux ou des vitesses d'adaptation qui définissent des dynamiques lentes ou rapides d'interaction avec le monde ; van Gelder et Port (1995) discutent précisément l'importance de l'adaptation des paramètres temporels pour la cognition. ***Le temps constitue alors le paramètre de conception le plus important***.

Si l'adaptation est un ingrédient indispensable pour l'autonomie, l'apprentissage constitue ce moyen qui permettra à l'agent de se stabiliser dans de nouveaux environnements. Il n'y a pas de raison particulière fondamentale (théorique) pour croire que les deux processus diffèrent en leur nature ; il vaut mieux y voir une différence d'échelle plutôt qu'une différence conceptuelle. Par exemple, si le comportement de l'agent peut être décrit par une équation (une fonction de transfert ou une loi d'interaction avec le monde), l'adaptation correspondra à la modification de ses paramètres, tandis que l'apprentissage sera la modification de l'équation même (Ashby 1960; Slotine 1994). Cependant, cela n'est possible que s'il existe une autre loi fixe à un niveau supérieur qui dicte comment cette première équation doit être modifiée. Si ce mécanisme est connu, il n'y a aucune magie dans l'apprentissage (et, en fait, dans ce sens le produit du processus de l'apprentissage n'est pas vraiment "nouveau", Varela 1979, p. 206). Il existe ainsi une notion de niveau qui relie les deux concepts : l'adaptation se situe à un niveau d'interaction plus directe avec le monde. Ceci veut aussi dire que dans ce cas la dynamique de l'apprentissage doit être beaucoup plus lente que celle de l'adaptation (sinon il n'y aurait pas de sens à vouloir modifier les paramètres d'une équation dont la structure change plus rapidement). Ces deux relations de niveau et de dynamique entre les deux processus ont pour conséquence la possibilité de monter de niveaux récursivement, si le niveau plus haut est permis d'agir sur lui-même (Pitrat 1991). Les interactions prennent donc leur "signification" seulement à travers le niveau des interactions directes avec le monde,

[5] Si le réglage était exogène et hors ligne, on parlerait d'un système flexible, mais si le réglage est endogène et en ligne, il s'agit d'un agent autonome.

c'est-à-dire le niveau le plus bas avec la dynamique la plus rapide. Cette idée du "méta" récursif est analysée à fond par (Pitrat 1991) et (Minsky 1986*a*).[6] L'importance de la notion de "niveau" pour les capacités adaptatives et cognitives d'un animat est aussi soulignée par Roitblat (1991).

1.3 Principes d'organisation d'agent autonome

> *"Autonomie signifie loi propre ... [elle] représente la génération, l'affirmation de sa propre identité, la régulation interne, la définition de l'intérieur."*
> *(Francisco Varela, "Autonomie et connaissance", 1989)*

1.3.1 Agents, mondes et autonomie

Comme nous l'avons déjà vu, un agent doit être considéré autonome par définition, c'est-à-dire indépendant de supervision. L'autonomie de l'agent se manifeste aussi comme l'impossibilité (ou grande difficulté) d'une tierce personne (c'est-à-dire d'un observateur) de contrôler l'agent (*"freedom from control"*, McFarland 1992, p. 141). Dans les deux cas, autonomie signifie *loi propre* et *contrôle propre* de l'agent plutôt que loi et contrôle (commande) définis par l'extérieur.

Un agent a besoin d'un monde. Pour parler d'agent autonome, il importe de tracer des limites entre ce qui est l'agent et ce qui est le monde de l'agent, l'environnement dans lequel il est situé. Un agent autonome est donc un agent situé dans un monde délimité et qui "se pose des questions" sur ce monde (bien entendu, les autres agents font partie du monde de l'agent). La métaphore des questions veut dire que l'agent a une sorte de projet personnel qu'il veut mener à terme (sa mission) et à la réalisation duquel l'état du monde peut être favorable ou pas ; le monde est alors une entité *différente* de l'agent, c'est-à-dire une entité dont l'état est a priori indépendant de celui de l'agent. C'est précisément la relation entre les deux états (initiaux) qui détermine la co-évolution des deux entités.

Un agent a besoin d'un observateur. Un agent n'est un agent que quand un observateur externe peut lui attribuer une intentionnalité, c'est-à-dire un "projet" personnel.[7] On a une tendance à appeler intentionnel un processus ou phénomène ou acte qui ne suit pas, ou qui ne paraît pas suivre à première vue, une loi physique connue. Nous savons très bien qu'une pierre n'est pas un agent, un agent intentionnel : la pierre n'a aucune volonté particulière de suivre telle ou telle autre trajectoire, elle ne fait qu'obéir aux lois de la mécanique newtonienne selon lesquelles à un moment donné une et seulement une trajectoire est possible. Appeler un être vivant, tel qu'un animal ou un homme, un agent autonome et intentionnel, ne fait que montrer une fois de plus notre connaissance limitée des processus sous-jacents qui déterminent son comportement. Bien évidemment, un agent (autonome) artificiel ne peut pas être considéré autonome et intentionnel dans ce dernier sens puisque c'est nous qui l'avons créé (rappelons-nous qu'il faut distinguer les trois cadres de référence, celui de l'agent, celui de son concepteur et celui d'un observateur externe, cf. Introduction) ; cet agent est en effet un contrôleur particulier pour un problème de

---

[6] " *... as in any society, there must be watchers to watch each watcher, lest any one or a few of them get too much control of the rest.*" (Minsky 1991, p. 50)

[7] Un agent est appelé intentionnel si son comportement *peut* être prédit en lui attribuant des croyances, des désirs et des facultés rationnelles (Dennett 1987, p. 49, souligné par nous).

commande, comment conduire le système agent-monde à un état désiré (ou maintenir un état désiré ou suivre une trajectoire particulière etc.). La particularité de ce problème qui nous fait chercher des solutions non classiques réside dans la complexité des modèles en question ou l'absence totale de modèles analytiques.

Notons, finalement, que ***le monde de l'agent n'est pas nécessairement un monde "géométrique" et la situation ne veut pas nécessairement dire occupation d'espace, mais il peut s'agir de monde abstrait*** : par exemple, pour un agent connecté à d'autres agents par l'intermédiaire de connexions, le monde est l'espace (mathématique) des messages qui voyagent dans les connexions et il peut prendre des formes très diverses. Pour décomposer récursivement un agent en d'autres agents, il est donc nécessaire d'identifier les mondes des divers agents ou de choisir les actions élémentaires qui caractérisent chacun des agents.

1.3.2 Autonomie, motivation et perturbation

Nous avons vu qu'un agent autonome doit avoir une motivation qui exprime sa volonté d'effectuer la tâche qu'on lui a attribuée et que l'état de cette motivation constitue une mesure de l'état d'avancement de la réalisation de sa tâche. Puisque le monde n'a pas généralement un comportement "monotone" ou linéaire, il importe de disposer d'un ensemble de motivations qui détermineront le comportement, la "réaction" de l'agent aux différents états de son monde et de lui-même. S'il existe ainsi plusieurs motivations "parallèles", il faut en plus un composant d'arbitrage ou de régulation entre motivations. ***Un agent autonome est alors une entité qui possède un système motivationnel, c'est-à-dire un ensemble de motivations et un mécanisme central de régulation***. Notons que la régulation centrale équivaut à une *prise de décision* qui peut être binaire (agir ou pas agir) ou continue (comment agir) et qui se fait par rapport à un besoin premier présent explicitement ou implicitement dans le système motivationnel : le besoin premier est en réalité la tâche ou la mission de l'agent et la réalisation graduelle de la tâche se manifeste comme baisse graduelle et disparition du besoin. Toute l'activité de l'agent a alors comme "but" d'amener le(s) besoin(s) premier (s) de l'agent à 0, c'est-à-dire *l'état de satisfaction* ou le "*nirvana"* de l'agent. Si le système motivationnel exprime la prédisposition de l'agent face à son monde (et alors utilise des variables ou des "besoins" endogènes), l'activité actuelle de l'agent n'a de sens que par rapport à un monde particulier, dont l'état constitue, comme nous l'avons déjà vu, une perturbation à l'état motivationnel de l'agent. L'agent s'engage ainsi dans une activité "d'ajustement" de ses motivations à ses perturbations, qui a comme but de fond l'atteinte de son état de satisfaction. La motivation se manifeste donc comme la résistance de l'agent aux efforts extérieurs de manipulation (puisque la seule possibilité d'intervention au niveau des besoins endogènes de l'agent est indirecte par l'intermédiaire des stimuli externes). De son côté, perturbation veut dire différence.[8] ***L'ajustement de la motivation à la perturbation signifie alors que l'activité de l'agent a un double but : de modifier le***

[8] Maturana & Varela (1987) postulent qu'un agent autonome est structurellement couplé avec le monde dans lequel il est situé de façon que l'agent et son monde constituent deux systèmes dynamiques dont chacun sert de source de perturbation à l'autre.

monde pour le rendre moins hostile[9], et donc le besoin moins urgent, mais aussi de se modifier pour devenir moins sensible à la perturbation éventuelle future. Un agent autonome possède ainsi un système motivationnel : l'agent est "contrôlé" par son concepteur dans son espace motivationnel, il s'agit donc de *contrôle motivationnel* (la notion de l'espace motivationnel est analysée par McFarland & Bösser (1993, chapitre 4)).

Il faut souligner ici une propriété importante des systèmes motivationnels, la ***sélectivité*** : un agent ne considère que les aspects de son monde qui sont relatifs à ses motivations et demeure indifférent au reste. Dans ce sens, l'agent *choisit* (ou fait croire qu'il choisit) ce qui importe pour la réalisation de sa mission, de son projet personnel. Nous discuterons l'importance de la sélectivité dans les paragraphes 7.1.3 et 7.5.3.

1.3.3 Autonomie et opérationalité

L'autonomie d'un agent artificiel a une importance utilitariste et relative à une tâche précise déterminée par son concepteur. Nous avons vu que la présence de paramètres libres de tâche pose un problème d'opérationalité, c'est-à-dire de degré de satisfaction des performances et qu'on a besoin des structures d'adaptation.

Couplage opérationnel. Un agent (autonome) artificiel ne doit pas seulement être structurellement couplé avec le monde dans lequel il est situé, mais il doit également être opérationnellement couplé avec lui : la loi de "réaction" à une perturbation doit être telle que, si l'on connaissait d'avance la perturbation exacte, la réaction serait optimale, c'est-à-dire les performances de l'agent seraient statistiquement meilleures que celles de toutes les autres réactions possibles avec la même organisation de base. Cette loi de réaction doit alors inclure des paramètres opérationnels, c'est-à-dire des paramètres de tâche. Le terme couplage signifie que les paramètres de tâche ne sont pas explicitement présents dans l'organisation de l'agent, mais l'activité de l'agent dépend des paramètres endogènes pour lesquels *on peut dire* qu'ils correspondent aux paramètres de tâche ou qu'ils ont une relation de dépendance avec eux. Dans les deux problèmes étudiés qui impliquent des paramètres libres, le problème 2 et le problème 5, les paramètres endogènes couplés à ceux de la tâche sont des paramètres temporels, c'est-à-dire des taux ou de vitesses de réaction qui déterminent des dynamiques d'interaction agent-monde. Le terme couplage signifie encore que l'agent ne *connaît* pas sa tâche ; il peut connaître éventuellement son projet personnel, son besoin individuel, mais il n'a aucun autre accès à la tâche que son concepteur lui a attribuée.

Pour assurer l'opérationalité de l'agent en présence des paramètres libres de tâche, il faut d'abord définir des paramètres endogènes de l'agent couplés à ces paramètres libres de tâche. Ensuite, il faut trouver un régime d'adaptation des paramètres endogènes qui conduit à une augmentation des performances. Vu que les paramètres libres changent continûment, l'adaptation des paramètres endogènes de l'agent doit être continue aussi. Étant donné de plus que les paramètres libres peuvent changer de façon capricieuse et non linéaire (par exemple, faire des sauts), nous avons trouvé naturel de définir des régimes d'adaptation qui sont en effet des régimes

[9] Le terme "hostile" est utilisé au sens figuré : un monde est hostile s'il perturbe l'agent, il lui pose des questions et lui induit des besoins. Cette perturbation n'est pas nécessairement "négative". L'agent modifie le monde pour ne pas voir des sources de perturbation.

d'autorégulation continue entre deux limites pour que les variables endogènes puissent aller d'une valeur quelconque permise à une autre valeur quelconque permise.[10]

La dernière remarque importante concerne ***l'homéostasie***. Un agent autonome qui possède un système motivationnel comme celui décrit et qui est structurellement et opérationnellement couplé avec son environnement a comme variables "essentielles" (Ashby 1960), c'est-à-dire des variables dont sa "survie" dépend, ces besoins premiers qui correspondent à sa tâche, à sa mission (elles sont essentielles car en leur absence il n'y a pas de notion d'agent!). Ces variables "essentielles" ne sont *pas* régulées entre limites, mais l'agent veut les amener à 0 (en effet, ces variables ont un caractère dual épicurien qui exprime à la fois douleur et plaisir). ***Ce qui est homéostatique et autorégulé n'est donc pas les besoins premiers de l'agent, mais son organisation même*** (Maturana & Varela 1980), les variables endogènes qui déterminent l'interaction de l'agent avec son monde, sa "réaction" face à ses besoins et à ses perturbations : ***la variable critique est la dynamique de l'interaction avec le monde, donc la dynamique de la perturbation***.

1.3.4 Agents autonomes sociaux : de la socialité à la coopération

Nous avons dit (paragraphe 1.2) qu'un système multi-agents a un avantage opérationnel par rapport à un seul agent et que la socialité dans ce système doit se manifester comme une forme d'interaction induisant une perturbation sociale pour chacun des agents. Intuitivement, la socialité signifie que les agents coopèrent pour la réalisation d'un but commun. Dans cette optique descendante, la modélisation des processus coopératifs et sociaux par la communauté de l'intelligence artificielle distribuée est fondée presque exclusivement sur le paradigme cognitiviste traditionnel qui utilise des représentations explicites de buts, de croyances et d'actions telles qu'un observateur extérieur au système les voit : Durfee et al. (1989) passent en revue les applications de ces mécanismes de coopération dans le domaine de la résolution de problèmes. Cette approche focalise sur les sous-problèmes d'identification de but, de communication, de résolution de conflits et de négociation etc. (cf. plusieurs articles dans (Bond & Gasser 1988), (Huhns 1987), (Gasser & Huhns 1989), (Demazeau & Müller 1990) et (Demazeau & Müller 1991)). Le même paradigme cognitiviste a également servi de base pour les recherches en communication et structure ou organisation sociales, toujours dans le contexte de l'intelligence artificielle (Werner 1989, 1990; Findler & Malyankar 1993).

Cependant, comme Castelfranchi l'a souligné (Castelfranchi 1990, 1993; Conte et al. 1991), la question fondamentale n'est pas "comment une population d'agents réalise une tâche sociale", mais "pourquoi un agent autonome s'engage-t-il dans des relations et des interactions sociales" (*le problème de la socialité ou de la dépendance*) et "comment un agent fait pour rendre son but social, c'est-à-dire pour le faire adopter par les autres agents" (*le problème de l'adoption de but*). Castelfranchi a ensuite déclaré que la coopération n'est qu'un cas particulier d'adoption de but, qui dans la plupart des systèmes d'IAD a lieu comme par magie, et que pour avoir une adoption de buts, la communication ou la requête ne sont pas nécessaires. Il propose alors une

[10] Il n'y a cependant aucune raison logique ou philosophique derrière l'autorégulation de ces variables ; la seule justification que je puisse trouver est la simplicité et l'efficacité du mécanisme. De même, il n'y a aucune raison particulière pour que ces variables aient un caractère temporel.

vision égoïste des relations sociales générales, fondée sur la réciprocité qui résulte des intérêts partagés des agents, plutôt que d'une instruction ou persuasion. La vision de Castelfranchi est une vision essentiellement ***ascendante*** et ***sélectiviste*** : ce qui se manifeste comme la socialité des agents, n'est que le partage des buts autrement individualistes et égoïstes des agents.[11] Dans notre modèle d'agent motivé, la socialité doit donc intervenir au niveau du système motivationnel des agents et se manifester comme modification de ces motivations, c'est-à-dire, la socialité doit induire un nouveau type de perturbation des motivations de l'agent, une ***perturbation sociale***. ***On appelle alors socialité le partage de buts ou de besoins des agents, c'est-à-dire la socialité intervient au niveau du système motivationnel des agents***. Notons que, partage de buts ne signifie pas toujours que les agents ont tous le *même* but ; des relations sociales beaucoup plus compliquées émergent si les buts des agents forment un réseau de dépendances à plusieurs niveaux. De plus, de même qu'avec le système motivationnel individuel, des mécanismes d'apprentissage sont permis d'agir sur la socialité des agents.

Ainsi la coopération est la socialité positive, c'est-à-dire le type de socialité qui induit une amélioration des performances individuelles de l'agent, en passant toujours par son système motivationnel. La différence avec la vision descendante est que maintenant la coopération devient relative : ***on appelle coopération le type de socialité qui conduit à une amélioration des performances individuelles d'un agent selon ses*** **propres** ***critères***. Ceci dit, ce sont les agents eux-mêmes qui décident de ce qui est coopératif ou pas, et une socialité qui est perçue comme coopérative par un agent, peut ne pas l'être pour un autre agent. Une population d'agents ayant une socialité ainsi relative et différenciée (donc une société d'agents) va donner naissance à des phénomènes et des structures émergentes, dont des standards sociaux. La socialité relative des agents aura comme résultat qu'un agent pourra décider de suivre ces standards ou pas, de "coopérer" ou pas, donc ces structures sociales émergentes et ces standards vont être dynamiques. Nous pouvons dire ainsi, que cette société est autonome, puisqu'elle évolue principalement grâce à de ses forces internes plutôt que par des "instructions" externes.

Cette conclusion de besoin de *réciprocité intrinsèque* dans les relations sociales, c'est-à-dire de *réciprocité motivée* stimulée par une *reconnaissance* des agents frères ayant le même but, sont en effet les deux hypothèses que les chercheurs en biologie de l'évolution ont formées pour expliquer les systèmes vivants (cf. Trivers (1971) pour la réciprocité et (Hamilton 1964) pour la parentèle) et sur lesquels toute une école d'étude est fondée, celle de l'évolution de la coopération (Axelrod & Hamilton 1981). Notons également que c'est ce type de socialité et de coopération rencontré chez les animaux les plus évolués, tels que les vertébrés, qui nous convient, et non pas celui rencontré chez les animaux qui forment des sociétés par différenciation, tels que les insectes sociaux (Wilson 1975). Sous cette optique, nous avons quantifié le modèle original de tit-for-tat (Axelrod & Hamilton 1981) de manière à pouvoir l'appliquer dans le contexte de relations sociales qui n'ont pas l'allure d'un jeu au vrai sens du terme, comme le dilemme du prisonnier où un agent extérieur (tel qu'un arbitre ou un manipulateur) a fixé les règles du jeu et les récompenses que les agents reçoivent sont statiques. Au lieu d'un tel jeu statique, nous avons voulu rechercher des situations où

[11] Cela ne l'a pas empêché de rester dans le cognitivisme lors de son analyse sur les engagements sociaux (Castelfranchi 1995), même si son raisonnement ne nécessite pas obligatoirement une description à l'aide d'états mentaux.

les agents forment ce "jeu" eux-mêmes dynamiquement et l'histoire des agents joue un rôle plus direct que dans un jeu. De plus, ce modèle met l'accent sur l'opérationalité des agents, c'est-à-dire sur leur satisfaction au cours de leur vie plutôt que sur leur valeur reproductive (fitness). Le modèle tit-for-tat quantitatif est présenté brièvement dans l'annexe A ainsi qu'un résumé des expérimentations que nous avons effectuées afin d'explorer ses propriétés.

Dans les problèmes étudiés ici, l'étude de la socialité n'est cependant pas complète, dans le sens où il n'existe pas de paramètre libre de socialité et donc pas de besoin de régulation de socialité. Nous ne montrons donc pas toute la puissance de la réciprocité mais simplement la nécessité du partage de besoin plutôt que de simples réactions à des stimuli sociaux. Nous montrons également que l'importance relative de la socialité est beaucoup plus grande que celle de l'individualité et que, par conséquent, un agent autonome est susceptible à une manipulabilité "sociale", ce qui dans le cas des agents-cellules du chapitre 7 se traduit par un besoin d'un système immunitaire. Nous ne montrons pas complètement non plus le rôle de la diversité pour la stabilité d'une société autonome, mais simplement son importance pour un aspect restreint d'opérationalité, également dans le chapitre 7.

1.3.5 Physiologie, temps et connaissance

Un agent possède deux parties "comportementales" : son ***métabolisme*** qui est responsable de la "consommation" des messages provenant de l'extérieur (et donc responsable de l'autorégulation de l'agent pour répondre à ses perturbations) et son ***programme*** qui détermine ce que le métabolisme fait. Dyson (1985) a dessiné une analogie entre métabolisme et matériel et entre programme et logiciel pour fonder sa théorie de la double origine de la vie. Ce qu'il définit comme programme (logiciel) est ce qui passe de génération à génération (et que l'agent ne peut pas explicitement contrôler ou modifier au cours de sa vie, c'est sa partie *inerte*), tandis que ce qu'il définit comme métabolisme est ce qui n'est important que pour un individu particulier (et alors la partie où l'apprentissage est permis). Par exemple, toutes les fonctions d'adaptation environnementale et sociale sont essentiellement des fonctions métaboliques, mais elles sont régulées par le programme de l'agent.

La ***physiologie*** de l'agent est le mode de couplage entre son programme et son métabolisme et l'agent n'y a pas accès pour le modifier, donc pour apprendre. La physiologie est alors un milieu d'interactions interne à l'agent qui simplifie la conception (grâce à sa globalité) mais qui contraint les formes possibles du métabolisme et par conséquent les possibilités d'apprentissage.

Le concepteur intervient au niveau du programme en essayant de l'adapter aux spécificités du médium d'implémentation qui restreignent les formes possibles de métabolisme.[12] Le programme de l'agent repose sur l'utilisation du temps comme un paramètre de conception sous forme de taux ou de vitesse de métabolisme, d'où nous pouvons conclure que ***toute la connaissance du concepteur réside dans la nature des***

[12] Dans une perspective évolutive, qui n'est pas la nôtre dans cette thèse, le programme de l'agent est la partie comportementale qui passe de génération à génération précisément parce qu'elle a été trouvée "stable" ou "opérationnelle" au niveau de l'espèce. Dyson (1985) postule alors que l'évolution de ce mécanisme de "compilation de connaissance" ou de "transmission de culture" est indépendante de l'évolution du métabolisme qui doit venir en premier.

dynamiques des différentes interactions entre agent et monde (des différentes fonctions de métabolisme). Cette connaissance se situe alors au niveau méta. Dans les problèmes 2 et 5, ces dynamiques sont des *dynamiques homéostatiques propres* à l'agent et indépendantes de son environnement, c'est-à-dire les taux de modification des dynamiques sont constants et indépendants de l'environnement.

1.3.6 Émergence, sénescence et autonomie

Un des principaux thèmes de recherche de la vie artificielle est l'évolution de structures et l'émergence d'organisations : "***Comment s'effectue le passage d'un niveau d'organisation au niveau supérieur et à quoi cela sert-il ? Qu'est-ce qui émerge ?*** Comment une société se forme-t-elle à partir des agents-composants et pourquoi ? Et comment détecte-t-on une société (un réseau de relations sociales) en n'observant que les agents-composants ?". Nous adoptons le point de vue de Brooks et Wiley (1988) selon lequel l'évolution est un processus irréversible capable de générer progressivement plus de complexité à des niveaux plus élevés d'organisation.[13] ***Notre hypothèse est que la force de l'évolution est intrinsèque au système vivant et qu'il s'agit en réalité d'une fonction continue de sénescence conduisant le système vivant progressivement et inévitablement à la mort.*** Une telle fonction de sénescence peut être une vraie force d'évolution et elle peut conduire à l'émergence d'organisations de niveaux supérieurs[14] si elle favorise la socialité des agents, c'est-à-dire si la durée de vie de l'agent asocial est plus courte que celle de l'agent social : une organisation pluricellulaire aura donc un "avantage sélectif" par rapport à une seule cellule, une société d'agents pluricellulaires aura un "avantage sélectif" par rapport à un seul agent pluricellulaire et ainsi de suite.[15] Un "réseau de relations sociales" émerge, autrement dit les agents bâtissent des relations sociales, suite à des engagements dans des relations de coopération (Ferber 1994*c*). Nous avons déjà vu auparavant que, de son côté, la coopération repose sur un partage des buts individuels.

Cette fonction de sénescence doit être suffisamment abstraite pour permettre une application récursive à des niveaux d'organisation progressivement plus élevés. Ceci dit, la fonction doit posséder des propriétés et reposer sur des principes indépendants du niveau d'organisation : nous devons pouvoir étudier et analyser une société d'agents comme un super-organisme conforme aux mêmes principes que chacun des agents séparément (effectivement, la forme exacte des structures va varier selon le niveau d'organisation, l'idée étant toujours que les principes de coopération seront les mêmes). Bien entendu, la fonction de sénescence doit être couplée avec un mécanisme de détection et de renforcement (ou de stabilisation) de propriétés émergentes afin de permettre l'apparition d'organisations de niveau supérieur. Nous

---

[13] *"Because we know of no natural hierarchical configurations not produced by an irreversible process, we are logically compelled to accept the proposition that the natural hierarchy of form we see is the result of historical and irreversible processing operating on discrete units"* (Brooks & Wiley 1988, p. 82).

[14] Atlan (1972) a fait l'association entre l'émergence de complexité, les processus irréversibles et la théorie de l'information. Il a également analysé l'irréversibilité du vieillissement du point de vue de l'information, mais sans parler explicitement d'une force d'évolution.

[15] Entre biologistes, la nécessité d'étudier la sénescence *simultanément* au niveau moléculaire et au niveau de la population, a été soulignée par Rose et Finch (1994).

nous intéressons à la première partie de cette hypothèse, c'est-à-dire d'élaborer une fonction de sénescence qui favorise les sociétés d'agents plutôt que les agents individuels ; Le passage de niveau à niveau et l'évolution d'organisations ne seront pas abordés.

À côté de cet argument descendant, il existe un deuxième argument, plus ascendant, qui a une signification plus pratique. Un agent motivé et perturbé par son environnement, qui suit les principes décrits jusqu'à maintenant, démontre une propriété intéressante : son histoire est réversible (ou répétitive), puisque les seules modifications qu'il subit (les effets de l'adaptation et de l'autorégulation) sont temporaires ; par exemple, l'autorégulation continue d'un paramètre entre deux limites ne modifie en aucun cas la gamme des valeurs possibles pour ce paramètre et les vraies possibilités de l'agent restent intactes. Cela n'est gênant que dans la mesure où il conduit à des agents relativement manipulables : l'agent explorateur asocial ou social des chapitres 4 et 5 et la cellule autonome du chapitre 7 sont manipulables au niveau de leurs besoins premiers, puisqu'on peut arriver à maintenir leur activité en modifiant leur monde et en leur offrant le stimulus nécessaire pour ce faire.

Dans un monde ainsi "vicieux" et anormal, il importe d'avoir des agents qui s'en rendent compte pour *modifier leurs besoins irrémédiablement*, c'est-à-dire pour *apprendre*, et au pire, si rien ne semble avoir un effet heureux, pour *mourir*.[16] Par conséquent nous pouvons dire que, quant à ses buts, un agent autonome doit être mortel donc sénescent, sinon quel intérêt aurait-il pour apprendre ?[17] L'autonomie est donc l'outil offert à un être mortel pour survivre autant que possible. ***Une fonction de sénescence constitue la force motrice de l'apprentissage des agents et donc de leur autotélie phénoménale.***

Puisque la fonction de sénescence doit favoriser les relations sociales indépendamment de niveau d'organisation, nous avons l'intention de l'utiliser dans le futur proche comme une force d'apprentissage au niveau cellulaire à l'intérieur d'un animat. Le chapitre 7 présente un réseau cellulaire qui s'auto-organise spontanément en cas de pannes en essayant de découvrir de nouvelles interactions sociales. Une force naturelle de dégradation, telle qu'une fonction de sénescence, pourrait perturber le réseau des relations sociales et maintenir les activités d'auto-organisation à leur maximum (et ainsi faire de sorte que le réseau reste actif et l'agent reste en vie). Comme un effet de bord, ces nouvelles interactions sociales dans le réseau cellulaire pourraient faire apparaître des comportements perçus par un observateur externe comme des comportements plastiques et appris. Cette vision de l'apprentissage cellulaire en tant qu'émergence de nouvelles structures sociales à l'intérieur de l'agent cellulaire implique que *l'apprentissage aussi est un processus irréversible* et que *la mémoire n'est pas cyclique*. L'agent cellulaire n'apprendra pas suite à des forces externes appliquées, mais pour une raison intrinsèque, de vivre plus longtemps.

Nous verrons dans le chapitre 8 qu'une fonction de sénescence fondée sur le principe d'une rétroaction entre métabolisme et programme remplit toutes les spécifications

---

[16] Cependant, au niveau le plus "haut" des besoins, il est nécessaire d'avoir des besoins fixes : "*If we could deliberately seize control of our pleasure systems, we could reproduce the pleasure of success without the need for any actual accomplishment. And that would be the end of everything*" (Minsky 1986*a*, section 6.13).

[17] La pierre non intentionnelle qui suit toujours la même trajectoire unique, le fait parce que suivre telle ou telle autre trajectoire ne la change *en rien*, la pierre n'apprend rien du tout et ne "meurt" pas.

nécessaires et que des mécanismes supplémentaires d'autorégulation augmentent la durée de vie des agents sénescents, c'est-à-dire que le prolongement progressif de la durée de vie est dû à l'ascension des niveaux méta de "raisonnement" (donc des niveaux cognitifs) : plus un agent est cognitif, plus il vivra, alors la sénescence n'est qu'une force qui pousse les agents à devenir progressivement plus cognitifs.

Parler des systèmes autonomes nous conduit donc à introduire une notion de sénescence et à identifier sa relation avec l'émergence des niveaux d'organisation supérieurs. Cependant, l'émergence n'est considérée ni comme un outil ni comme une méthodologie de conception, ce qui serait contradictoire par définition ; Havel (1993) discute pourquoi, dans la plupart des cas, l'utilisation du terme émergence dans un contexte de conception dissimule l'ignorance et comment l'émergence doit nous éloigner de la conception "intentionnelle", alors utilitariste. Si on parle d'émergence, ce n'est que dans le cadre d'un point de vue plus "ouvert" qui s'éloigne de la conception utilitariste en faveur d'une démarche plus évolutionniste au sein des approches ascendantes aux sciences cognitives (mais pour Delaye (1993), un seul système multi-agents conçu évolutif serait capable de s'adapter potentiellement à tous les environnements et de résoudre tous les problèmes).

1.3.7 Récapitulation

Le modèle général d'agent autonome dont les propriétés viennent d'être énumérées et analysées peut être résumé comme suit (cf. fig. 1.1 et tableau 1.1).

Un agent autonome est *réactif* : il répond aux aspects imprévisibles de son environnement.

Un agent autonome est *motivé* : il a une mission qui est "décrite" dans son organisation comme un besoin qu'il veut ramener à 0. La motivation de l'agent exprime à la fois la volonté de l'agent pour accomplir sa mission et l'image ou l'idée que l'agent a du monde. L'état du monde tel qu'il est perçu par l'intermédiaire de ses systèmes de perception constitue une perturbation à cette idée.

Dans un contexte multi-agents, un agent autonome est *social* : il a une socialité du type "partage du besoin", qui induit une perturbation de provenance sociale.

Un agent autonome est *adaptatif* et *opérationnellement couplé* avec son environnement : il existe un couplage, une correspondance entre agent et monde, telle que l'agent doit s'autoréguler continûment pour accomplir sa mission et pour être opérationnel. L'autorégulation de l'agent ne concerne pas le besoin premier de l'agent, mais un ou plusieurs paramètres organisationnels qui déterminent la réaction de l'agent à ses perturbations. Ces paramètres organisationnels autorégulés ont une signification temporelle — ils sont par exemple des taux ou des vitesses d'adaptation — et déterminent donc la dynamique de l'interaction de l'agent avec son monde ou la dynamique de la perturbation.

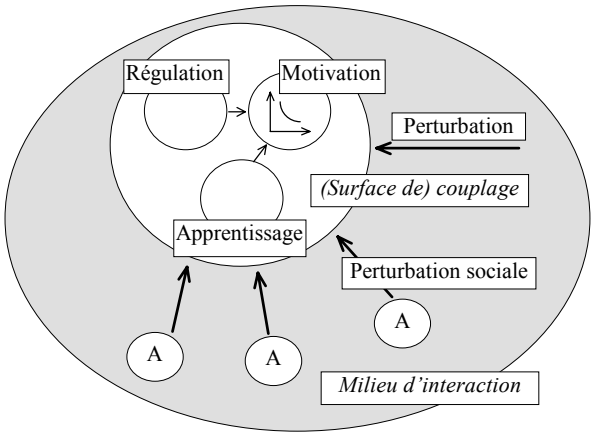

Figure 1.1 Modèle général d'agent autonome. Conceptuellement, il comprend trois composants : la régulation, la motivation et l'apprentissage. Les autres agents sont représentés par les petits cercles marqués comme A.

Concepteur	*Agent*	*Agent social*
Tâche (problème)	*Besoin premier (motivation)*	*... partagé*
Monde	*Perturbation*	*... + perturbation sociale*
Artefact	*Structure*	*... diversifiée*
Paramètre libre	*Paramètre temporel (taux ou vitesses)*	
Opérationalité	*Autorégulation de dynamique d'interaction*	
Tâche accomplie	*Besoin=0 (état de satisfaction)*	

Tableau 1.1 Mode d'emploi du modèle d'agent autonome

Chapitre 2 Les agents jusqu'ici : État de l'art

La notion d'agent a eu une double évolution dans les différentes disciplines : d'une part, un agent est contrasté par son autonomie et par son individualité avec une entité qui n'est pas un agent, et d'autre part un agent fait partie d'un système multi-agents, par opposition à un système mono-agent.

La première notion, qui recouvre les notions d'agent intentionnel et d'agent autonome, est rencontrée typiquement dans les disciplines de l'intelligence artificielle et de la robotique qui se sont traditionnellement intéressées aux artefacts complètement préprogrammés, tandis que la deuxième, agent social, est rencontrée dans l'intelligence artificielle distribuée et la vie artificielle et vise à exploiter la complexité émergente des systèmes composés d'entités relativement simples interagissant localement. On peut retracer les origines de la notion d'agent dans la cybernétique (Wiener 1961) et dans les travaux de von Neumann sur les automates cellulaires (von Neumann 1966).[18]

Dans ce chapitre nous présentons rapidement ces disciplines en mettant l'accent sur la notion d'agent telle qu'elle est utilisée dans chacune d'elles (cf. tableau 2.1 pour une récapitulation concise). Dans les chapitres suivants et au fur et à mesure de la présentation des différents problèmes abordés, nous reprendrons des travaux spécifiques parus dans la bibliographie pour décrire et comparer analytiquement avec les modèles que nous avons développés. Nous ferons également l'association entre certaines idées émergées lors de notre modélisation et des idées retrouvées dans la littérature.

	IA/Robotique	***Vivant alternatif***
Multi-agents (émergence, complexité)	*IAD* *Robotique cellulaire*	*Vie artificielle*
Agent "autonome" (adaptation, apprentissage)	*Agents situés* *Nouvelle IA* *Contrôle intelligent*	*Animats*

Tableau 2.1 La notion d'agent dans les différentes disciplines classées selon deux grandes catégories : IA/Robotique qui est en principe plus orientée à la résolution de problèmes d'ingénieur, et vivant alternatif qui est en principe plus orienté à l'exploration d'hypothèses scientifiques sur la nature du vivant.

[18] Mais je tiens aussi à citer un article largement méconnu : (Gullahorn & Gullahorn 1963). Leur modèle de comportement social est basé sur la réciprocité et le renforcement du comportement.

2.1 Les agents de l'intelligence artificielle et de la robotique

2.1.1 Du mono- au multi- : Intelligence artificielle distribuée et systèmes multi-agents

L'intelligence artificielle existe comme discipline depuis la conférence de Dartmouth en 1956 et son sujet est la création d'artefacts intelligents et plus particulièrement des machines intelligentes : "*Artificial intelligence was un attempt to build intelligent machines without any prejudice toward making the system simple, biological or humanoid ...*" (Minsky 1968, p. 7). Traditionnellement, l'intelligence artificielle s'est intéressée à la création de systèmes informatiques destinés à la résolution de problèmes tels que le diagnostic médical, les jeux et le traitement de langage naturel ; la résolution de ces problèmes nécessite une quantité importante de connaissances et un mode de raisonnement qualitatifs. L'intelligence artificielle distribuée (Bond & Gasser 1988; Huhns 1987; Gasser & Huhns 1989) est la branche de l'intelligence artificielle qui s'intéresse aux systèmes distribués plutôt que mono-agent et se sous-divise typiquement en résolution distribuée de problèmes (*distributed problem solving*, Durfee et al. 1989) et systèmes multi-agents (Demazeau & Müller 1990,1991; Ferber 1989*a*, 1995). Le premier domaine met l'accent sur la résolution d'un problème spécifique en le *distribuant* sur un ensemble d'autres modules qui "coopèrent" en échangeant des données et des solutions, tandis que le deuxième s'intéresse à la *synthèse* d'un comportement intelligent à partir d'un ensemble d'agents autrement indépendants qui interagissent dans un espace commun. Tous deux ont comme prédécesseur historique les systèmes des tableaux noirs (Engelmore & Morgan 1988), qui sont une évolution des systèmes des règles de production à des grandes sources de connaissances communiquant et coopérant par l'intermédiaire d'une structure commune, le tableau noir. Les thèmes de l'intelligence artificielle distribuée et des systèmes multi-agents incluent : description des tâches, décomposition, distribution et allocation, interaction et organisation, cohérence et coordination, modélisation d'agents, discordances entre agents, telles que l'incertitude ou les conflits etc. Si l'IAD maintient un degré de centralisation, il n'en est pas de même pour les systèmes multi-agents : le "but" global d'un système multi-agents est atteint comme un effet de bord de l'activité concurrente des agents, de façon qu'un agent particulier paraisse plus qu'un simple composant d'un système d'IAD. La différence réside dans l'autonomie des agents : chacun des agents possède ses buts individuels qui ne sont pas obligés d'être directement corrélés avec le but global du système.

L'intérêt de l'utilisation de systèmes multi-agents est une question de complexité : si l'espace d'états de chacun des agents peut rester limité, l'espace d'états du système multi-agents global peut croître de façon non linéaire au nombre des agents, c'est-à-dire que les systèmes multi-agents sont des systèmes synergiques. Si alors on dispose des représentations nécessaires, la modélisation multi-agents va montrer un avantage méthodologique par rapport aux systèmes mono-agent : la complexité des systèmes multi-agents est réduite lors de l'analyse, mais multipliée lors de la synthèse. Un deuxième avantage souvent discuté, est la possibilité de dimensionnement d'un système multi-agents par l'introduction d'agents supplémentaires. La modélisation multi-agents est ainsi un outil de gestion de complexité.

Les systèmes réactifs, dont on peut retracer les débuts dans (Firby 1987) (Georgeff & Lansky 1987) (Schoppers 1987), constituent une branche des systèmes multi-agents fondée sur la réactivité et l'émergence du comportement (Ferber & Drogoul 1992;

Ferber 1994*a*,*b*). Le comportement global émerge de l'interaction des individus entre eux et avec le monde, et l'organisation architecturale des agents peut prendre plusieurs formes : on y rencontre des mécanismes de régulation d'activation (Drogoul et al. 1992) et des relations d'agression et de fuite (Ferber & Jacopin 1991) etc. Les emprunts éthologiques n'y manquent pas non plus (Theraulaz et al. 1991; Deneubourg et al. 1991; Goss & Deneubourg 1992).

La problématique multi-agents se révèle utile aussi bien pour la conception d'artefacts que pour la modélisation de processus sociaux, naturels ou artificiels ; Wooldridge et Jennings (1995) et Ferber (1995) font l'état de l'art des deux domaines d'application. Notre objectif dans cette thèse se situe principalement du côté de la conception, mais nous abordons également la frontière avec la modélisation dans le chapitre 8.

2.1.2 De l'action à la réaction à l'interaction

Les travaux de l'intelligence artificielle traditionnelle ainsi qu'une vaste partie des travaux en intelligence artificielle distribuée se sont basés sur l'hypothèse du *système de symboles physiques* : "un système de symboles physiques consiste en un ensemble d'entités, appelées des symboles, qui sont des formes physiques rencontrées comme composants d'un autre type d'entité, l'expression ou structure de symboles. Alors cette structure est décomposée en un ensemble d'instances de symboles entre-liées à l'aide de relations physiques. Un tel système de symboles physiques possède les moyens suffisants et nécessaires pour produire des actions intelligentes". Tous les états de ce système constituent un espace, appelé l'espace de recherche, qu'on explore à l'aide d'algorithmes sophistiqués de recherche qui exploitent les connaissances du domaine ; ces connaissances sont décrites comme des heuristiques et sont souvent soumises à des limitations sérieuses telles que le *"frame problem"* (Hayes 1990). Qui plus est, ces connaissances ne semblent pas être liées à la réalité physique, mais plutôt à une autre réalité artificielle, symbolique et abstraite, ce qu'on appelle souvent un monde de jouets (*"toy world"*, Brooks 1986*b*) ou de manière plus formelle le *"symbol grounding problem"*. Il semble que c'est l'interaction routinière avec la réalité physique qui fait paraître ce qu'on appelle intelligence : *"In discussing "intelligence", people are usually too concerned with the peaks of human creative innovation. First-rate creativity is not the most common nor the most essential quality of intelligence. Newton, Einstein and Mozart were all very intelligent men, but we don't scorn other men as being "mere superpowered adding machines" because they don't routinely make monumental discoveries"* (Minsky 1968, Introduction, p. 12).

L'intelligence artificielle, se voyant ainsi restreinte dans des domaines et des problèmes loin de la réalité physique, a cherché à traiter des problèmes dans un monde réel, dynamique et incertain. Durant les années 80, une nouvelle approche révolutionnaire a vu le jour, la nouvelle I.A., selon laquelle *l'organisation de l'intelligence doit lui permettre d'être réactive aux aspects dynamiques de l'environnement et de générer un comportement robuste face à l'incertitude des capteurs et l'imprévisibilité de l'environnement* (Brooks 1991*a*). L'élaboration de ces principes fondamentaux est due surtout à Agre (1985,1988) qui s'est inspiré de la théorie de l'action située de Suchman (Suchman 1987). Les principes de cette nouvelle discipline sont maintenant assez clairement définis :

- Un agent intelligent est généralement engagé dans *une activité routinière située dans un monde dynamique*. Un ensemble de possibilités préprogrammées

permettent à l'agent d'agir de façon cohérente et apparemment intelligente. Ces possibilités sont opportunistes et par conséquent robustes (Agre 1985).

- *Représentations déictiques (ou représentations indexiques-fonctionnelles) et situation dans le monde* : Il n'y a pas de représentation globale du monde. Un agent n'a des représentations — locales — que pour les objets relatifs à son fonctionnement et ces représentations partielles ne reposent en aucun cas sur une interprétation sémantique de symboles, mais sont plutôt définies par l'intermédiaire des interactions fonctionnelles et/ou spatiales avec le monde (Agre & Chapman 1987, 1991). Il s'agit donc de représentations distribuées, non symboliques et non maniables. Ce principe est souvent rencontré sous le nom de *"physical grounding hypothesis"* et peut être résumé ainsi : *"The world is its own best model"* (Brooks 1991*c*, p. 15).
- *Un observateur peut parler des croyances, des plans et des buts d'un agent*, bien que ces croyances, ces plans et ces buts ne soient nulle part représentés et manipulés explicitement à l'intérieur de l'agent (Rosenschein & Kaelbling 1986; Kaelbling & Rosenschein 1991; Maes 1991*b*; Agre & Chapman 1987).
- *L'intelligence est déterminée par la dynamique de l'interaction avec le monde*. L'intelligence n'est pas une propriété d'un composant isolé, mais émerge de l'interaction entre l'agent et son monde, ainsi que des interactions entre les différents composants au sein de l'agent : "*Intelligence is in the eye of the observer*" (Brooks 1991*c*, p. 16).

L'I.A. traditionnelle est fondée sur l'utilisation d'heuristiques pour contrôler la recherche dans l'espace d'états. La nouvelle I.A. est fondée sur l'émergence d'un comportement global à partir des interactions d'un ensemble d'unités comportementales simples. De même qu'avec les heuristiques, cette approche n'offre *pas de garantie de fonctionnement*. Jusqu'à présent et dans les travaux les plus représentatifs du domaine (Brooks 1989; Steels 1991*a*; Beer 1990), on est parvenu à concevoir des systèmes qui exhibent des propriétés émergentes intéressantes et utiles sans essayer une analyse théorique dans le but d'optimiser la conception de ces systèmes.

La contribution importante de cette nouvelle discipline est d'avoir créé des systèmes et plus spécialement des robots mobiles autonomes bâtis sur les principes présentés auparavant et exhibant des propriétés comme la robustesse, l'adaptabilité et la cohérence de comportement, en un mot ***intelligence primitive*** : *"It is unfair to claim that un elephant has no intelligence worth studying because it does not play chess"* (Brooks 1991*a*, p. 13).

Notre approche se situe dans la même problématique d'action située et d'interaction avec un monde dynamique imprévisible comme la source de la complexité apparente.

2.1.3 Les robots de l'automatique

La robotique industrielle et de service se trouve actuellement à une phase de transition des tâches nécessitant une précision à celles nécessitant une importante flexibilité en ligne, et des robots-esclaves aux robots-experts qui doivent faire preuve d'un degré d'intelligence plutôt que d'un degré de dextérité (Coiffet 1993).

Les besoins de flexibilité et d'intégration ont rendu insuffisants les outils formels d'analyse et de synthèse de l'automatique traditionnelle et ils ont fait apparaître un besoin d'utilisation de connaissances qualitatives plutôt qu'analytiques, ils ont alors fait pencher l'automatique vers l'intelligence artificielle. La nouvelle automatique, qui s'est donnée le nom de "contrôle intelligent", a aussitôt reconnu le besoin *d'intégrer de la reconnaissance dans la boucle de commande* (Meystel 1985) et s'est donc focalisée dans la synthèse de méthodes ayant un composant de représentation de connaissances. La théorie de contrôle intelligent qui a réussi à capturer tous les aspects centraux et à montrer la possibilité et la nécessité d'intégration de composants très hétérogènes a été formulée par Saridis (1985,1987). Ces points principaux sont résumés ainsi :

- *Contrôle hiérarchiquement intelligent* : Le contrôle est distribué dans une hiérarchie selon *le principe de précision décroissante avec intelligence croissante*. Cette hiérarchie comprend trois niveaux de contrôle, chacun pouvant avoir une structure complexe : le niveau d'organisation (où se situent toutes les décisions de "haut niveau", la mémoire à long terme etc.), le niveau de coordination (prise de décisions et planification locale, mémoire à court terme) et le niveau d'exécution (le niveau plus bas de la hiérarchie, le niveau matériel).
- La variable principale du fonctionnement d'une telle machine intelligente est le *flux de connaissance* qui est perçue comme une information structurée qui vise à la réduction de l'incertitude présente au début. De façon similaire, *le rythme de connaissance est une mesure de l'intelligence de la machine*. Le principe de la réduction d'incertitude peut s'appliquer récursivement à tous les niveaux de la hiérarchie.
- La conception d'une machine intelligente est formulée comme *un problème de programmation mathématique*, dans lequel *la formulation de la tâche fait, elle aussi, partie du problème de contrôle* (Meystel 1985).

Par définition, le contrôle intelligent est le guide d'une machine intelligente qui doit atteindre son but de façon autonome, c'est-à-dire sans interagir avec un opérateur humain. Son application dans le domaine de la robotique est directe : on appelle robots intelligents ceux qui intègrent un contrôleur intelligent (Meystel 1988; Saridis 1985). La plupart de ces robots sont utilisés dans des tâches d'intervention et de service qui sont caractérisées par une a priori méconnaissance de l'environnement : la supervision, le gardiennage, la maintenance et l'exploration dans des environnements dynamiques et hasardeux ou hostiles pour l'homme, tels que les centrales nucléaires, l'espace, les mines et le fond des mers.

Ce que nous partageons avec la démarche du contrôle intelligent est le souci des solutions récursives et des principes applicables à tous les niveaux d'organisation d'un agent autonome.

2.1.4 Les robots de l'intelligence artificielle

L'approche réactive en robotique a émergé durant les années 80 de l'effort des chercheurs de trouver une alternative efficace à la planification. Les systèmes réactifs se situent alors à l'autre extrême de la planification, qui repose sur l'existence et la mise-à-jour d'un modèle de monde global ; au lieu de présupposer un modèle quelconque de monde, ils définissent une réaction directe à chacun des stimuli perçus

(schéma S-R ou stimulus-réponse). Si en principe ce mode d'organisation paraît extrêmement simple, sa puissance augmente de façon exponentielle lorsque plusieurs composants réactifs sont connectés et utilisés à l'intérieur du même système. Les interactions entre ces composants peuvent donner naissance à des phénomènes qu'un observateur externe caractériserait comme une activité délibérée (Chapman & Agre 1986). Pour un robot, ce qui intervient entre les perceptions et les commandes fournies aux actionneurs est ainsi un "algorithme réactif" qui ne dispose pas de connaissance explicite sur le monde et sur la tâche de l'agent. Par rapport aux robots traditionnels, les robots réactifs ont un avantage dans des environnements partiellement connus, dans lesquels les données des capteurs sont bruitées ou peu fiables et le fonctionnement doit se montrer robuste face aux événements imprévus.

Contrairement à l'approche traditionnelle de décomposition verticale du composant de raisonnement, une vaste partie des travaux en robotique réactive repose sur une décomposition horizontale : les modules hiérarchisés de perception, interprétation, planification etc. disparaissent ; désormais, la hiérarchie contient des unités élémentaires de fonctionnalité qui correspondent aux différents buts et tâches du système en question. Ces unités ne s'échangent plus d'information, mais perçoivent directement le monde et agissent sur lui conformément à leur rôle (Brooks 1986*a*). Le comportement global n'est donc pas préprogrammé, mais émerge de l'interaction de ces unités entre elles et avec l'environnement.

L'implémentation la plus importante de cette approche est l'architecture de subsomption (Brooks 1986*a*, 1989) qui a été testée et élaborée sur une série de robots mobiles pendant une période de dix ans.

D'autres efforts basés sur la même approche de décomposition en niveaux de compétence sont présentés dans (Anderson & Donath 1991) (Arkin 1991) (Kaelbling 1986) (Malcolm & Smithers 1988, 1991) (Smithers & Malcolm 1987, 1989) (Steels 1990).

Pour permettre l'arbitrage consistant d'actions, un mécanisme d'inhibition et suppression a été introduit (Brooks 1986*a*) selon lequel la place d'une unité dans la hiérarchie montre sa priorité en cas de conflit, c'est-à-dire en cas d'activation concurrente. Cette architecture a été sévèrement critiquée sur plusieurs points (cf. par exemple les critiques de Hartley et Pipitone 1991) :

- *L'organisation linéaire n'est pas suffisamment modulaire,* ce qui fait que l'intuition joue le rôle le plus important dans le choix et la conception des couches,
- *La détermination de priorités statiques est artificielle,* les priorités devraient être plutôt dynamiques,
- *Les interfaces sont définies de façon implicite,* l'inhibition et la suppression constituent des moyens trop brutaux, en effet il n'existe pas de partage ni de passage d'information, et
- *L'arbitrage ne convient pas dans le cas d'actions exclusives ou non corrélées.*

Une autre limitation est l'absence totale de formalisation qui permettrait d'aborder des questions telles que la convergence et la cohérence du comportement, l'adéquation et la synthèse automatique (Brooks 1991*b*).

La contribution majeure de cette approche reste d'avoir démontré que l'émergence est possible et parfois supérieure à l'approche traditionnelle de traitement séquentiel de l'information. Trois niveaux d'étude ont été identifiés : le niveau micro (étude et

détermination des unités élémentaires), le niveau macro (intégration d'unités au sein d'un robot) et le niveau multitude (étude sociologique d'interaction des robots entre eux et avec le monde). Enfin, un effort isolé dans le domaine de la formalisation est paru dans (Steels 1991*b*).

Les systèmes multi-robots. (Brooks et al. 1990) et (Angle & Brooks 1990) présentent quelques premières réflexions sur l'intelligence collective des robots et les spécifications impliquées ; le rapprochement avec le travail de (Reynolds 1987) est un des premiers signes que les recherches éthologiques peuvent servir de riche source d'inspiration et d'emprunt de méthodologies. Flynn et Brooks (Flynn 1987; Flynn & Brooks 1989*a*) donnent des spécifications et décrivent des applications éventuelles de micro-robots, appelés "gnats" (moucherons). Sugihara et Suzuki (1990) présentent des algorithmes distribués pour les formations des robots. La communication entre robots est discutée dans (Asama et al. 1991) (Matsumoto et al. 1990) (Shin & Epstein 1985).

Une caractéristique partagée par tous ces travaux, est qu'ils restent généralement à un niveau expérimental, sans formalisation, ce qui permettrait de passer de l'analyse à la conception des spécifications. Brooks (Brooks 1991*c*) résume de manière élégante ce *souci de prévisibilité* qui nous éviterait les longues sessions de modélisation et d'expérimentation qui sont actuellement la règle.

Le chapitre 3, qui aborde la question de l'organisation du système de contrôle d'un agent situé, est directement inspiré de la robotique comportementale.

Robotique hybride. À côté du purisme de la réactivité on retrouve des approches hybrides qui essayent de combiner les avantages de la réactivité avec ceux de la planification. Les travaux les plus représentatifs sont ceux de Payton (Payton et al. 1990, Payton & Bihari 1991), Gat (1992,1993) et Noreils (1993).

2.1.5 Robotique cellulaire

La première approche cellulaire en robotique constitue en effet une fusion des théories de calcul distribué et des automates cellulaires avec application sur les systèmes multi-robots (Wang 1990; Wang & Beni 1988, 1989, 1990; Hackwood & Beni 1992). À part la lacune de l'implémentation, la caractéristique principale de cette approche cellulaire est l'absence d'architecture interne de cellules ainsi que d'hétérogénéité.

Une autre approche cellulaire distincte de celle de Wang est celle des robots cellulaires reconfigurables (Fukuda et al. 1990*a*; Fukuda & Kawauchi 1990; Ueyama et al. 1992) : un robot consiste en un ensemble de cellules qui s'auto-organisent de façon quasi-optimale selon la tâche en question. Ces cellules ont une architecture interne assez complexe qui leur permet d'effectuer un ensemble de tâches primitives : percevoir l'état des cellules voisines, se déplacer dans l'espace et se raccrocher à une autre cellule, communiquer avec toutes les autres cellules, prendre une décision locale de choix de structure — et par conséquent de déplacement — selon un critère emprunté à la théorie de l'information etc. L'évaluation et l'analyse de communication s'effectuent à l'aide d'un ensemble de mesures de communication (Fukuda et al. 1990*b*). Les projets les plus récents dans le domaine comprennent entre autres une étude d'hétérogénéité de cellules en déterminant des cellules-maîtres conformément à un critère d'énergie de réseau et application d'un algorithme

génétique pour la production de connaissance (Kawauchi et al. 1992). Bien que cette approche se trouve à l'extrême opposée de la précédente en ce qui concerne l'architecture interne, elle ne procède toujours pas à une étude de l'organisation de l'intelligence, mais reste au contraire à un seul niveau presque homogène où les interactions sont fixes et les communications statiques — ce qui est d'ailleurs justifié par le choix du domaine. Elle n'intègre pas de notion d'adaptation non plus.

En fait, aucune de ces deux approches n'aborde le problème de la nature de l'intelligence et de sa compréhension. La première s'intéresse à explorer les structures émergentes dans un espace réel euclidien avec la perspective de spécifier et de concevoir des formations statiques ou dynamiques de robots homogènes ou hétérogènes chargés de la résolution distribuée de problèmes tels que la perception collective. Elle constitue alors l'approche complémentaire de son homonyme P.K.C. Wang (1991), où le même problème de formations de robots est abordé du point de vue analytique et descriptif. La deuxième approche cellulaire s'intéresse à la fabrication de robots-manipulateurs composés dynamiquement de plusieurs cellules afin d'atteindre des positions ou des configurations éloignées et met le poids sur la communication d'information spatiale entre les cellules.

Il semble que c'est justement ce *manque d'organisation et de structuration* qui constitue la limitation la plus sérieuse des approches cellulaires actuelles et que la définition d'une architecture interne sophistiquée ne suffit pas pour faire émerger le comportement demandé. Inversement, une architecture interne simple et rigide accompagnée d'une possibilité d'interaction dynamique ne suffit pas non plus. La cellularité telle qu'elle est définie dans le chapitre 3 concerne l'organisation du système de contrôle d'un agent situé (tel qu'un robot) et se situe dans une problématique connexionniste. Elle ne doit donc pas être confondue avec la robotique cellulaire.

2.2 Le vivant alternatif

2.2.1 Life as it could be

La vie artificielle, un domaine fascinant regroupant des recherches dans des disciplines aussi diverses que la biologie, la physique, l'anthropologie, l'économie et l'informatique, est née en 1987, lors du premier congrès à Santa Fe (Langton 1988*a*). Son objectif est "l'étude du vivant tel qu'il pourrait être au lieu du vivant tel que nous le connaissons" (*"life as it could be, instead of life as we know it"*). Elle partage avec les systèmes multi-agents réactifs les mêmes principes de réactivité et d'émergence, en ajoutant un *principe d'adaptation comme source principale et à la fois effet de bord du comportement* (Taylor 1991; Travers 1988).

Les thèmes de recherche de la vie artificielle incluent (cf. Langton 1988*a*; Langton et al. 1991; Varela & Bourgine 1992; Langton 1994; Brooks & Maes 1994) :

- *Chimie artificielle ou calculatoire et modélisation de l'origine de la vie* (Tamayo & Hartman 1988; Fontana 1991; Rasmussen et al. 1991; Morowitz 1994).
- *Évolution* (Dawkins 1988; Lindgren 1991; Ray 1991; Ackley & Littman 1994; Bak et al. 1994; Thearling & Ray 1994).
- *Modélisation du développement, embryologie et morphogénèse* (Lindenmayer & Prusinkiewicz 1988; De Boer et al. 1991; Prusinkiewicz 1994).

- *Systèmes auto-organisants et automates cellulaires* (Goel & Thompson 1988; Bersini & Detours 1994).

Un exemple de méthodologie inspirée de la biologie et importée dans la conception d'agents situés est la *"génétique"* : algorithmes (Goldberg 1989) et programmation (Koza 1991*a*). La méthodologie repose sur un mécanisme de sélection inspiré par l'évolution naturelle et se propose comme outil puissant d'exploration de l'espace des possibilités. Elle a été appliquée avec succès à la résolution de problèmes au niveau de l'individu (Koza 1992; Delaye & Ferber 1992) ainsi qu'au niveau de la société (Koza 1991*b*). Deux de ses applications les plus fructueuses sont l'évolution de réseaux connexionnistes (Belew et al. 1991; Ackley & Littman 1991) en général et l'évolution de systèmes de contrôle de robots autonomes (Cliff et al. 1993; Harvey 1994; Nolfi et al. 1994).

2.2.2 Animats

L'approche ascendante aux sciences cognitives s'intéresse à la question : "Pourquoi la nature a-t-elle inventé ces mécanismes cognitifs que nous observons chez nous et chez les animaux ? et quel est le rôle adaptatif de la cognition ?". La démarche caractéristique de cette approche est la conception et la synthèse d'animaux artificiels ou animats *complets* (des agents simulés ou des robots), qui démontrent des capacités cognitives comparables à celles rencontrées dans la nature (Meyer 1995) ; on espère que l'exploration des mécanismes artificiels nous conduira à mieux comprendre les mécanismes naturels.

À côté des systèmes typiquement multi-agents et des robots comportementaux qui peuvent être considérés comme des animats, les thèmes de recherche de l'approche animat incluent (cf. Meyer & Wilson 1991; Meyer & Guillot 1991; Meyer et al. 1992; Cliff et al. 1994; Meyer & Guillot 1994) :

- *Comportements adaptatifs.* Par exemple, comportements (neuro-)éthologiques ou connexionnistes (Teitelbaum et al. 1991; Halperin 1991; Cecconi & Parisi 1992; Blumberg 1994; Werner 1994) et comportements à base de classifieurs (Booker 1991; Iba et al. 1992; Donnart & Meyer 1994).
- *Mécanismes d'apprentissage.* Par exemple, conditionnement (Halperin & Dunham 1992; Klopf et al. 1993; Schmajuk 1994) ou apprentissage par renforcement (Sutton 1991; Peng & Williams 1993; Millán 1994; Colombetti & Dorigo 1994).
- *Évolution de comportements* (Ram et al. 1994; Colombetti & Dorigo 1992).
- *Étude dynamique du comportement* (Beer 1995).

Nous croyons que ces deux disciplines sont en réalité des incarnations modernes d'une biologie théorique au sens large du terme et nous partageons avec les deux la volonté d'explorer dans des problèmes particuliers des formes ou des solutions opérationnelles alternatives, c'est-à-dire sans faire nécessairement référence à des modèles et à des mécanismes naturels connus, si ce n'est que pour comparer. Nous espérons que l'étude des possibilités alternatives, s'il y en a, nous permettra de mieux comprendre le monde dans lequel nous vivons et de mieux nous comprendre nous-mêmes.

Chapitre 3 Les agents cellulaires

"Solving a problem means representing it so as to make the solution transparent."
(Herbert Simon, "The sciences of the artificial", 1981)

3.1 Introduction

3.1.1 Architecture et conception

Il s'agit de s'inspirer des recherches en robotique autonome comportementale pour élaborer un substrat de conception, autrement dit une architecture, destinée à la description et à l'implémentation des agents autonomes situés (en simulation ou en robotique) et conçus spécialement pour la résolution de problèmes particuliers. Nous utiliserons par la suite le terme ***"architecture"*** ou ***"organisation"*** pour nous référer à l'ensemble de relations et de principes auxquels les solutions aux problèmes particuliers doivent se conformer et le terme ***"système de contrôle"*** (ou même ***"structure"***) pour ces solutions. Les structures sont des instanciations de l'architecture dans le même sens où les programmes informatiques sont des instanciations des langages informatiques utilisés. L'architecture peut donc être vue comme un *langage de programmation* des systèmes de contrôle d'agents situés. L'architecture en soi ne résout aucun problème, parce qu'elle n'existe pas en tant qu'ensemble de modules réutilisables après instanciation et intégration, mais en tant qu'ensemble de principes de programmation. L'objectif à plus long terme serait de traduire les caractéristiques d'un grand nombre de problèmes abordés dans le contexte des agents autonomes en un ensemble de règles de programmation d'agents. Le problème est de trouver les bonnes abstractions qui minimisent le "bricolage" lors de la programmation.[19]

Le paradigme comportemental (*behavior-based*) démontre un ensemble de propriétés-clés pour la conception d'agents autonomes (Brooks 1991*a*,*b*) : pas de modèle ni de représentation centrale de l'environnement de l'agent, minimisation des goulots d'étranglement et des temps de réponse aux perceptions, résistance aux pannes et coût bas d'implémentation. En résumé, la puissance du paradigme réside dans sa simplicité : c'est à l'interaction agent-monde de donner naissance à des phénomènes complexes et c'est ce potentiel de complexité que les chercheurs dans ce domaine essaient d'apprivoiser et d'exploiter. Pour permettre la systématisation de l'étude et

[19] Un de mes premiers essais en Lisp était un programme d'unification ne comprenant que 80 lignes environ : j'étais surprise de voir à quel point ce programme était concis et élégant par rapport à son frère aîné que j'avais écrit en Fortran, tout simplement parce que la structure-clé, la liste, ainsi que les opérations associées, sont des primitives du langage.

par conséquent de la conception, une organisation comportementale doit exhiber les propriétés supplémentaires suivantes :

(a) Extensibilité : Nous désirons pouvoir ajouter de nouvelles structures, éventuellement développées séparément, pour étudier leurs effets individuels ou collectifs avec d'autres structures et pour élargir la classe des problèmes que la structure de départ permet de traiter. C'est-à-dire, *nous désirons disposer d'un degré de modularité permettant l'extensibilité* : la question devient, de quel genre de "modules" et de "relations" a-t-on besoin ?

(b) Spécifiabilité : Nous aimerions avoir une relation 1 à 1 entre une structure et une spécification (c'est-à-dire une entité abstraite représentant l'instanciation particulière de l'architecture), dans le double but de : (1) raisonner en termes de spécification lors de l'expérimentation et alors de trouver des correspondances entre structures et fonctionnalités observables ou performances, (2) "reproduire" éventuellement (ou automatiser) la conception des agents. *Cette spécification doit expliciter les paramètres architecturaux en question* afin de permettre l'évaluation qualitative et quantitative de leur impact. La question devient ainsi, *quel genre de paramètres doit-il y avoir dans une structure spécifique, quel genre de paramètres l'architecture doit-elle pouvoir "représenter" et comment* ?

(c) Simulabilité : Faute de méthodologie d'analyse du fonctionnement de structures si complexes, il est finalement nécessaire de pouvoir simuler le fonctionnement de ces structures sur ordinateur afin de les évaluer. Bien entendu, une structure *informatiquement simulable* est aussi *informatiquement implémentable* sur robot, à quelques détails de gestion de matériel près. L'implémentabilité de la structure n'induit pas pour autant une opérationalité absolue, puisque les problèmes matériels de la robotique ne se présentent pas, et donc ne peuvent pas être abordés, en simulation.

3.1.2 Architecture et tâche : Classification ou pas ?

Si les considérations de conception telles que l'extensibilité sont primordiales pour une étude comparative, il est tout aussi important de trouver des correspondances *fonctionnelles* entre l'architecture de base et les tâches étudiées. Quelle est la nature des tâches que nous étudions ? Et comment cela simplifie-t-il (ou restreint) la classe des structures/solutions que nous développons ? Steels (1994*a*) a identifié quatre types d'architectures/approches de robotique comportementale (et par extension d'agents situés) : les approches algorithmiques (Brooks 1986), les approches connexionnistes (Verschure et al. 1992), les approches de circuit (Chapman 1990) et les approches dynamiques (Steels 1994*b*). Cependant, on peut supprimer les deux dernières catégories, si l'on remarque que les approches de circuit sont des approches algorithmiques mais de très grande échelle[20]— dans le but de permettre une implémentation massive à l'aide des techniques VLSI —, tandis que les approches dynamiques sont en effet des approches algorithmiques mais dans un espace différent de celui de l'informatique traditionnelle, dans un espace continu. Les architectures de

[20] Chapman a cependant caractérisé son approche comme essentiellement connexionniste (ibid, p. 38, 185) du fait de reposer sur l'interconnexion d'un très grand nombre de composants (unités de calcul) faisant un traitement local d'information, sans se restreindre à la fonctionnalité élémentaire ni à l'uniformité de la topologie des connexions du neurone de McCulloch & Pitts.

robotique comportementale et d'agents situés peuvent donc à première vue être classées en deux grandes catégories : le connexionnisme et l'algorithmique.

Toutefois, cette première catégorisation découvre un confus conceptuel entre les propriétés structurelles et fonctionnelles des différentes architectures distribuées.[21] Dans les tableaux 3.1 et 3.2 sont données séparément les deux catégorisations, la catégorisation fonctionnelle et la catégorisation structurelle. D'une part, selon que la connectivité du réseau est fixe ou plastique, les problèmes abordés sont de nature "algorithmique" ou sont des problèmes de classification/catégorisation (généralement au niveau de la perception) : l'algorithmique présuppose que tant le problème que sa solution sont connus d'avance dans leur détail, tandis que la classification présuppose une a priori connaissance limitée qui fait que le système doit converger à la solution à partir d'une solution partielle ou plus générale. Cette souplesse ou plasticité est assurée d'un degré de redondance dans le réseau (dans le chapitre 7, nous discuterons davantage de la plasticité comme une propriété *endogène* au réseau).

Connectivité fixe	Algorithmique
Connectivité plastique	Classification (ou catégorisation)

Tableau 3.1 Catégorisation fonctionnelle des architectures distribuées selon leur connectivité.

	Fixe	***Plastique***
Homogène	Cliff, Beer	Verschure
Hétérogène	Brooks	Cellularité (cf. §3.1.3)

Tableau 3.2 Catégorisation structurelle des architectures distribuées selon deux axes : les fonctionnalités des composants (c'est-à-dire les fonctions de transfert) qui peuvent être homogènes ou hétérogènes, et la connectivité du réseau qui peut être fixe ou plastique.

D'autre part, les architectures peuvent être analysées structurellement selon deux axes, l'axe des fonctionnalités des composants (c'est-à-dire les fonctions de transfert) qui peuvent être homogènes ou hétérogènes, et la connectivité du réseau qui peut être fixe ou plastique. Le connexionnisme traditionnel (dont des architectures comme celle de Verschure et al. (1992) sont issues) repose sur une homogénéité des fonctionnalités couplée à une plasticité des connexions, tandis que la plupart des approches dites "algorithmiques" (par exemple, Brooks 1986; Chapman 1990; Steels 1994*b*) reposent sur une rigidité des connexions que les fonctionnalités soient homogènes ou pas. Notons également qu'il existe d'architectures neuronales de connectivité fixe, donc de nature algorithmique (par exemple, Beer et al. 1989; Beer & Chiel 1991; Cliff et al. 1993), dans lesquelles chacun des neurones accomplit une fonction très précise et les connexions sont spécifiques et certainement pas "uniformes". Les problèmes de classification ont été abordés jusqu'à maintenant à l'aide d'architectures possédant un degré de plasticité, donc d'architectures connexionnistes (mais il n'y a pas de justification théorique ni de preuve qu'une architecture a priori non plastique ne peut pas résoudre des problèmes de classification).

Pour mieux voir la relation entre l'approche algorithmique et la classification dans le contexte de la robotique comportementale, il est intéressant de discuter un exemple particulier d'architecture "hybride", qui combine des caractéristiques des deux approches : celle de Kube et Zhang (1994).

[21] À ce stade, je préfère utiliser le terme "distribuée" qui a un sens plus large que le terme "connexionniste" ; ce dernier a été adopté dans la littérature pour montrer que l'action a véritablement lieu au niveau de la connexion (ce qui n'est pas le cas de toutes les architectures distribuées).

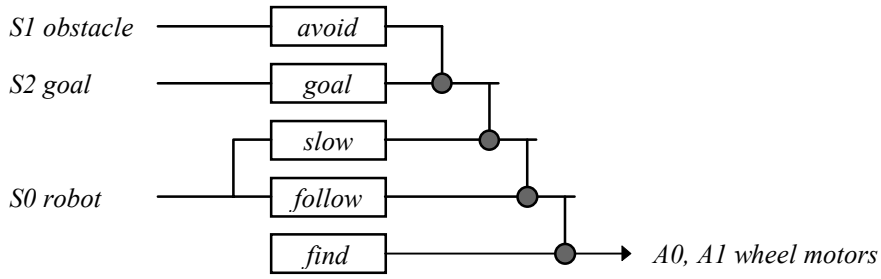

Figure 3.1 Architecture de Kube et Zhang (1994). Les cercles gris représentent des liens de suppression.

Étant donné les comportements algorithmiques de base de la figure 2.1, nous constatons qu'il existe moins de capteurs que de comportements et que ces capteurs peuvent être actifs en même temps. "L'information" produite par les capteurs ne peut pas être fournie directement aux actionneurs après transduction comportementale, mais il faut une étape intermédiaire de filtrage ou combinaison ou arbitrage. Flynn et Brooks (1989*b*) ont remarqué que le contrôle comportemental repose sur la présence de "nombreux capteurs", c'est-à-dire sur la présence d'un ensemble suffisant de capteurs pour donner l'information nécessaire pour l'activation d'un comportement. De leur côté, Kube et Zhang ont reformulé le problème de l'arbitrage entre comportements et ont utilisé un composant spécial du type ALN (Adaptive Logic Network) *pour faire la classification des patterns comportementaux* (ils ont également implémenté sur un robot réel un composant alternatif d'arbitrage : il s'agit d'un circuit numérique qui correspond à un arbitrage statique linéaire du type subsomption, comme sur la fig. 3.1). Cela correspond à une étape de classification des perceptions repoussée juste avant les actionneurs, ce qui ne serait pas le cas si l'on disposait de capteurs spécialisés "dédiés" à chacun des comportements autrement routiniers. Mais à côté de l'insuffisance des capteurs il existe une raison plus importante derrière le besoin de classification : les comportements tel qu'ils ont été définis ne sont pas corrélés. À part ceux d'évitement et de but, les trois autres n'ont en principe aucune relation, ils sont indépendants et ils peuvent être actifs en même temps. Dans ce cas, donc, il a *fallu* passer par une étape de classification nécessaire pour deux raisons : (a) insuffisance de capteurs par rapport aux comportements (problème de perception) et (b) absence de relation entre les comportements.

Jacopin (1993), en discutant les routines de Agre et les systèmes Pengi et Sonja de Chapman (Agre & Chapman 1987; Agre 1988; Chapman 1990), souligne que le problème de la perception est souvent passé sous silence et que la perception y est une véritable délibération au sens de l'IA classique. Tsotsos (1995) caractérise le problème de la recherche visuelle comme un problème de catégorisation qui nécessite des représentations intermédiaires dynamiques et donc qui ne peut pas être abordé de façon strictement behavioriste et algorithmique, il nécessite de la plasticité. Et il est bien connu que le connexionnisme constitue le meilleur choix s'il s'agit de faire de la classification. *Les approches connexionnistes seront donc avantageuses dans tous les cas où le problème de contrôle implique un problème de classification des perceptions* (ce qui n'est pas le cas de la subsomption, par exemple, qui présume une pré-catégorisation ad hoc). Le cas des perceptions fixes et connues constitue un cas dégénéré, celui des problèmes d'algorithmique. Inversement, un réseau plastique et auto-organisant, tel qu'un réseau neuronal traditionnel, pourra paraître sous

perturbation persistante comme classificateur et alors comme un réseau qui apprend (dans le chapitre 7 nous discutons la forme qu'un tel réseau pourrait prendre).

> ***Proposition :*** *Les problèmes de classification constituent une sous-classe des problèmes algorithmiques (cf. paragraphe 7.1).*

3.1.3 Principe architectural de base : La cellularité

Dans un premier temps, nous nous plaçons dans le paradigme comportemental (Brooks 1991*a*,*b*) : du point de vue fonctionnel, nous abordons des problèmes algorithmiques, tandis que du point de vue structurel nous adoptons une approche distribuée non neuronale. Il s'agit de chercher une instanciation plus structurée du paradigme comportemental avec l'objectif de permettre à la fois une efficacité face à un problème donné et une souplesse quant au développement de structures de complexité incrémentale, c'est-à-dire de structures algorithmiques hiérarchiques et évolutives.[22] L'objectif d'un ordre supérieur est ensuite de permettre et de comprendre le passage des problèmes algorithmiques aux problèmes de classification. Dans cette perspective, il nous faut donc une architecture qui combine les avantages de l'approche comportementale avec un degré de plasticité quant à son connectivité.

Les approches connexionnistes (neuronales) présupposent une fonctionnalité homogène et une connectivité homogène (en termes de sémantique des messages) pour tous les composants du réseau. Les approches non neuronales, quant à elles, reposent sur l'interconnexion de composants spécialisés[23] à l'aide de connexions ayant des sémantiques variées.[24] Nous combinons donc l'hétérogénéité des fonctionnalités des approches non neuronales avec l'homogénéité de la sémantique des connexions des approches neuronales. Étant donné l'uniformité des interfaces entre composants du réseau, notre problème de conception peut être reformulé comme suit : *quelle est la fonctionnalité élémentaire qui permet la définition des structures "représentant naturellement" les problèmes en question ?*

> *On appelle* ***cellularité*** *le mode d'organisation d'un réseau de composants ayant des fonctions de transfert hétérogènes mais une syntaxe de connexions homogène.*

Syntaxe de connexions homogène veut dire que tous les messages qui voyagent sur les connexions sont de même nature et que la cellule ne peut pas distinguer entre ses connexions ; elle doit, par exemple, traiter ses messages d'entrée comme un ensemble non ordonné de valeurs et non comme un vecteur. Les messages voyageant dans le

---

[22] On retrouve cette idée de structures hiérarchiques de contrôle dans la littérature des systèmes motivationnels en éthologie (cf. par exemple Toates 1986), ainsi que dans certains travaux qui portent sur les animats du point de vue à la fois éthologique et comportemental (Tyrrell 1993*a*, 1993*b*; Balkenius 1995). De son côté, l'incrémentalité correspond aux besoins de modularisation et de compositionalité, rencontrés par exemple dans les recherches connexionnistes de deuxième génération (Chandrasekaran et al. 1988).

[23] Mais pour Chapman, et dans une perspective d'implémentation matérielle, ce n'est pas la question de l'homogénéité des composants qui vient en premier mais celle de leur limite de complexité.

[24] Par exemple, dans l'architecture de la subsomption (Brooks 1986*a*), il existe trois *types* de connexions : la connexion "classique" (envoi de message), la connexion inhibitrice et la connexion de suppression. Au contraire, dans le paradigme neuronal, c'est la variation des poids des connexions qui permet une variation de "sémantique", les connexions étant structurellement homogènes.

réseau sont a priori des nombres réels qui peuvent être interprétés d'une façon ou d'une autre dans les différentes cellules. Des considérations supplémentaires quant à la forme des messages sont présentées dans le paragraphe 3.6.

Qu'est-ce qui n'est pas conforme à ce critère de cellularité ? Un module algorithmique, tel qu'un composant implémenté en logique combinatoire, ne l'est généralement pas. Soit, par exemple, une cellule ayant deux entrées et une sortie. La fonction de transfert (1) n'est pas conforme à la contrainte de la cellularité parce que la cellule doit distinguer les deux entrées. Au contraire, la fonction de transfert (2) est bien permise.[25]

si (entrée_1 = 0) alors sortie=entrée_2, sinon sortie=une_constante (1)

x=entrée_1+entrée_2, si (x > une_constante) alors sortie=x,
sinon sortie=une_constante (2)

Ainsi la plupart des processus présentés dans l'annexe de (Brooks & Flynn 1989), et implémentés comme des AFSM, c'est-à-dire des machines à états finis augmentées avec des variables, ne sont pas permises non plus (cela ne signifie pas que les AFSM *en général* ne sont pas permises, mais seulement que ces AFSM qui présupposent des entrées ordonnées ne le sont pas). La fonction de transfert du neurone McCulloch-Pitts (si $\sum x_i \cdot w_i \geq T$, alors 1, sinon 0) est un cas particulier. En principe, on considère que les poids "appartiennent" aux connexions et se renforcent positivement ou négativement comme un effet de bord physico-chimique de l'activité cellulaire, sans que ce soit les cellules elles mêmes qui les *commandent* à ce faire, les cellules ne font que *lire* leurs entrées après pondération : la fonction de transfert est alors permise. Cependant, lorsqu'on considère la connexion comme un objet sans activité propre, c'est la cellule qui doit associer à chacune de ses entrées un poids de connexion différent et donc distinguer ses entrées. Dans ce deuxième modèle, la fonction de transfert cesse d'être permise.

Mais si la vraie fonction de chaque cellule doit être spécialisée, y a-t-il une raison pour que la communication entre cellules le soit aussi ? Il vaut mieux imaginer que la cellule existe dans un monde où elle reçoit des messages de tous les côtés : si elle sait traiter ces messages, elle les traite, sinon elle ne fait rien.[26] Elle ne connaît certainement pas la source de ces messages : c'est au réseau de trouver la bonne sémantique de connexions afin de se maintenir et de donner naissance à un comportement adéquat.[27] La plasticité se manifestera donc comme la possibilité de

---

[25] L'observation que (2) peut être implémentée à l'aide d'une diode, tandis que pour (1) on a besoin d'un circuit numérique, m'a initialement fait penser qu'on cherche des fonctions de transfert analogiques. Cela n'est pas nécessairement vrai, cependant, car les circuits numériques sont implémentés "au fond" à l'aide de sémiconducteurs — pour revenir à la question de la complexité de la cellule, il est nécessaire de choisir le bon niveau d'abstraction et la bonne échelle de temps. Par exemple, les recherches en génie neuromorphe (Sejnowski & Koch 1994) se placent à un niveau de matériel analogique.

[26] Il y a une difficulté pratique de programmation qui a stimulé la définition du principe de cellularité : si tout est hétérogène, fonctions de transfert et types de connexions, la complexité des programmes informatiques correspondant aux réseaux cellulaires s'accroît énormément. Dans l'annexe C sont analysés plusieurs problèmes d'implémentation qui ont stimulé le raffinement des principes organisationnels.

[27] Une extension du modèle cellulaire que j'envisage de faire concerne la discrimination entre entrées et sorties : dans un modèle à connexions je peux imaginer qu'il n'existe pas d'entrées et de sorties, la cellule consulte toutes ses connexions et parmi tous les messages traite ceux qu'elle sait traiter. Ensuite elle envoie le résultat aux mêmes connexions et elle repète le cycle. L'idée est que l'activité des

découvrir de nouvelles sémantiques des connexions dans le réseau. Dans ce sens, la cellularité contraint à première vue la classe des fonctions de transfert considérées pour élargir en revanche les possibilités sémantiques. Comment fait-on pour passer de la syntaxe de l'architecture à la sémantique d'une structure particulière ? Ce problème est abordé dans le paragraphe 3.6 sous le titre du problème de l'espace de représentation.

3.2 Les réseaux cellulaires

3.2.1 Cellularité et physiologie

Un réseau cellulaire est composé d'un ***ensemble de cellules*** connectées entre elles à l'aide de ***connexions locales*** et d'un ensemble de ***structures physiologiques*** régulant l'opération du réseau *globalement*. La structure du réseau représente le savoir-faire ("know-how") du système de contrôle tandis que la partie physiologique est la partie "inerte" de l'agent. Qu'est-ce qui est représenté comme une cellule et qu'est-ce qui est représenté comme structure physiologique ? Ou, à quoi cela sert-il d'avoir des structures physiologiques dans un réseau connexionniste ou pseudo-connexionniste ?

Imaginons deux parties A et B du réseau cellulaire totalement distinctes, c'est-à-dire sans connexions entre elles, qui doivent être mutuellement exclusives. Une solution consiste en la définition d'une structure cellulaire d'arbitrage. Cette structure doit avoir accès en sortie aux sorties de A et B : cela se révèle souvent d'une grande difficulté pratique du point de vue de la complexité du réseau, puisque la structure d'arbitrage doit pouvoir reconnaître les différentes sorties et rediriger les messages sélectivement. C'est plus simple de définir une variable globale binaire qui sera utilisée par A et B de façon que la valeur "vrai" stimule A mais inhibe B et la valeur "faux" inhibe A mais stimule B. Cette variable est donc une *variable physiologique* dont la valeur a une *signification d'état global* dans le réseau et dont le rôle est celui de la *régulation* de A et B. Bien entendu, il peut également exister des variables physiologiques continues.

Imaginons maintenant une structure qui "réagit" simplement à une condition environnementale ou interne sans se faire commander par d'autres parties du réseau, comme, par exemple, dans le cas d'un robot agricole, un détecteur de fumée déclenchant l'émission d'un signal d'alerte. Le détecteur et l'émetteur constituent une structure "réflexe" qui n'a pas de connexions avec le reste du réseau, donc une *structure physiologique*. Tout comme pour une variable, le rôle de cette structure est un rôle de régulation (il est logique de supposer que la détection de fumée va à son tour inhiber ou stimuler des parties cellulaires du réseau et jouer un rôle de régulation).

Dans tous les cas de structures physiologiques, que ce soient de simples variables ou des structures plus complexes, la définition d'une *boucle* est indispensable. Une structure dans le réseau doit donner la valeur "vrai" à la variable binaire et une autre structure doit lui donner la valeur "faux". Pour le détecteur de fumée, l'environnement va généralement jouer ce rôle.

---

cellules doit être autorégulatrice et qu'elles doivent maintenir une certaine densité des messages de certains types autour d'elles. La même extension est envisagée dans le cadre du modèle sans connexions du chapitre 7.

La raison de la séparation des structures physiologiques des cellules est donc double. D'une part, ces structures sont globales (par exemple, les variables globales d'état) et leur localisation dans des cellules spécialisées conduirait à des patterns de connexions obscures (cf. le problème de la synchronisation discuté dans l'annexe C), et d'autre part, si les cellules sont permises de s'adapter à leur environnement et donc "agir pour apprendre", le système physiologique ne l'est pas[28] — ce dernier apparaît alors comme la frontière, les limites, ou le vrai programme de l'agent.

> ***Principe de séparation des dynamiques spatiales*** *: Ce qui est local et qui agit, c'est une* ***cellule*** *; ce qui est global et qui n'agit pas, c'est une* ***structure physiologique****.*

Les variables physiologiques peuvent avoir une dynamique propre indépendante de l'interaction agent-monde (par exemple une dynamique périodique ou de dégradation) et peuvent en plus être reliées en un réseau de dépendances (par exemple, l'affectation d'une variable peut stimuler un processus de dégradation chez une autre variable). Les variables physiologiques sont, soit des variables continues, soit des interrupteurs binaires (on/off).

> ***Variables physiologiques :*** *Les variables physiologiques sont des variables continues ou des interrupteurs binaires (relais on/off) et sont responsables de la régulation et de la coordination des parties éloignées du réseau. Elles peuvent avoir à la fois une dynamique propre (telle que la dégradation) et être affectées, ou simplement utilisées, au niveau cellulaire.*[29]

Notons au passage l'analogie entre les cellules sans entrées et un système physiologique qui suit sa propre dynamique ; une cellule sans entrée fonctionne pour l'agent comme une limite, une contrainte, un système de régulation qui peut représenter un monde ésotérique.

3.2.2 Cellules

La cellule est l'unité élémentaire de stockage et de traitement d'information. Une cellule représente et correspond à un composant doué d'une possibilité élémentaire de stockage et de traitement. Elle possède un ensemble de connexions d'entrée, un ensemble de connexions de sortie, un vecteur-mémoire statique ou dynamique et une unité de traitement qui correspond à une fonction de transfert ayant comme effet de bord la possibilité de mise à jour du vecteur mémoire (adaptation). La cellule possède aussi un temps interne d'exécution qui peut être contrôlable par le concepteur et qui représente son rythme d'exécution.[30] Les cellules sont ainsi des primitives

---

[28] Mais il n'existe aucune justification théorique pourquoi cela *doit* être ainsi, il s'agit d'une limitation de complexité : permettre un apprentissage global n'est en principe pas impossible, mais il est certainement plus compliqué de maintenir la viabilité des relations globales dans un réseau que celle des relations plus localisées.

[29] La deuxième version de l'architecture de subsomption (Brooks 1990) utilisait des registres partagés par toutes les AFSM faisant partie du même comportement ainsi qu'un système hormonal artificiel qualifié de "non procédural", mais la globalité du contrôle étant considérée à l'époque comme rétrograde, le rôle, l'importance et la nécessité du système physiologique ont disparu dans la réticence. En revanche, quatre ans plus tard, Brooks (1994) a utilisé un analogue biochimique pour soutenir sa proposition de quantités globales régulatrices et en dissipation.

[30] L'importance de la diversité des dynamiques temporelles au sein d'un système de contrôle d'agent situé n'a été révélée que très récemment et surtout avec l'apparition des architectures connexionnistes

structurelles différenciées par conception ou par adaptation dans un milieu particulier et définissent un substrat uniforme à partir duquel des agrégats complexes peuvent être construits. La complexité de la cellule peut être considérée comme comparable à celle d'un neurone artificiel (sauf que les entrées des cellules ne sont pas ordonnées et que toutes les cellules n'ont pas la même fonction de transfert) mais inférieure à celle d'une AFSM, puisque les cellules n'ont pas plusieurs "états symboliques", *elles exécutent toujours la même fonction de transfert sur l'ensemble de leurs messages en entrée*. La question de la complexité des cellules est discutée plus analytiquement dans le paragraphe 3.6.

Chaque cellule appartient à une catégorie fonctionnelle de cellules. On distingue trois types de cellules, les cellules-capteurs, les cellules-actionneurs et les cellules de traitement. Cette catégorisation est fonctionnelle et en aucun cas elle ne concerne les *modes* d'exécution et de communication entre cellules, qui sont des invariants de l'organisation cellulaire.

- *Cellules-capteurs* : Chacune est responsable d'une opération primitive de perception. Les capteurs peuvent être externes ou internes à l'agent. Nous avons prévu lors de la définition de l'architecture une possibilité d'allumer ou éteindre ces cellules, mais cette possibilité n'a pas été utilisée dans les deux solutions-exemples (en effet, allumer et éteindre les cellules serait intéressant seulement pour des raisons de minimisation des dépenses énergétiques). Nous avons utilisé exclusivement des capteurs de signaux locaux ou dans un rayon de perception, mais d'autres possibilités peuvent être envisagées.
- *Cellules-actionneurs* : Chacune est responsable d'une opération primitive de commande aux actionneurs. Au cours de ces simulations, nous avons utilisé exclusivement des cellules-actionneurs qui correspondent à des moteurs de navigation.
- *Cellules de traitement* : Ce sont les cellules des niveaux intermédiaires entre capteurs et actionneurs qui ont pour rôle la transduction des signaux en entrée. Elles peuvent être catégorisées selon leurs fonctions de transfert : des cellules *min*, *max* et *find-a-stimulus* (qui renvoient respectivement l'entrée minimale, maximale ou la première présente), des cellules *timeout* et des *cellules-priorité* (dont les entrées sont ordonnées de façon statique ou dynamique et qui renvoient l'entrée de la plus grande priorité). Il y a en plus trois types spéciaux de cellules de traitement : les *cellules-décodeurs* (qui font partie des systèmes périphériques des actionneurs), les *cellules d'éveil* et les *cellules-d'observation* (ces deux derniers types ont été utilisés dans les systèmes d'éveil de tâche, voir plus bas).

Cette conception de la cellule diffère radicalement de celle rencontrée dans les systèmes de robotique cellulaire (CRS, Wang 1990) où les robots situés dans l'espace sont considérés comme des cellules et il n'existe pas de notion d'arbitrage ou d'adaptation, ainsi que de celle des robots reconfigurables (CEBOT, Fukuda et al. 1991), où les cellules possèdent une structure interne complexe et une possibilité de communication, mais il n'existe pas de notion d'organisation de l'intelligence (cf. la description des deux approches dans le paragraphe 2.1.5).

---

de temps discret et asynchrones qui incorporent plusieurs échelles de temps : c'est la discontinuité dans le temps qui démontre un potentiel de phénomènes dynamiques complexes (cf. par exemple les connexions retardées de Cliff et al. 1993).

3.2.3 Structure du réseau

Pour gérer la complexité du réseau, on a défini une structure à trois niveaux imbriqués (figure 3.2). Les cellules sont regroupées dans des *agrégats* cellulaires et les ensembles d'agrégats forment des *tâches* ; les tâches correspondent abstraitement aux comportements (behaviors) rencontrés dans la littérature. Les agrégats sont des ensembles de cellules qui accomplissent collectivement une "fonction" particulière de perception, de commande ou de traitement intermédiaire ; on a alors respectivement des *systèmes de perception*, des *systèmes de commande-d'actionneurs* et des *systèmes de traitement*.

- Les systèmes de perception consistent en un ensemble de cellules-capteurs corrélées logiquement mais non interconnectées et peuvent appartenir à une tâche particulière ou être partagés par plusieurs tâches (les signaux en sortie d'un système de perception peuvent être utilisés différemment par des systèmes de traitement différents).
- Les systèmes de commande-d'actionneurs sont en principe partagés et leur structure générale est donnée dans la figure 3.3. La cellule-décodeur combine les entrées des capteurs-réflexes[31] avec les entrées des tâches connectées et envoie la commande calculée à l'actionneur correspondant. En effet, le décodeur a généralement comme rôle d'assurer que la commande en entrée n'est pas incompatible avec les entrées des capteurs réflexes qui agissent comme des contraintes ; par exemple, dans les deux systèmes cellulaires de contrôle implémentés (cf. paragraphe 3.3), le décodeur du système de navigation compare la commande en entrée, qui est une direction de navigation, avec les entrées des 4 capteurs d'obstacle, et si la direction commandée est bloquée par un obstacle, une autre est sélectionnée.

Cette structure permet de ***découpler la commande du réflexe et donc de séparer les différentes dynamiques d'interaction avec l'environnement***, c'est-à-dire découpler les commandes de haut niveau, émises par les tâches, des contraintes de bas niveau créées par les capteurs-réflexes. Ce schéma est différent des autres approches comportementales rencontrées dans la littérature (Arkin 1987, Brooks 1986*a*) et il a pour conséquence la disparition du besoin de "subsumer" les réflexes et donc de compliquer les interfaces entre tâches. La résolution de ces conflits élémentaires est ouverte au concepteur du système de commande d'actionneurs. Les systèmes de commande d'actionneurs sont définis comme des systèmes périphériques :

> ***Définition*** *: Un système* ***périphérique*** *est un système de perception ou de commande qui est irrépressible, c'est-à-dire dont on ne peut pas empêcher le fonctionnement, et dont le fonctionnement est indépendant des autres parties du réseau cellulaire.*

Enfin, chaque tâche comporte un système d'éveil (non visualisé dans la figure 3.2) dont le rôle est d'activer la tâche quand il faut et de jouer éventuellement le rôle

---

[31] Une structure est appelée une *structure réflexe* si elle n'est pas commandée à l'intérieur de l'agent, c'est-à-dire si elle n'a pas de connexions avec la partie cellulaire. Un capteur réflexe est un capteur au signal duquel l'agent doit répondre aussi vite que possible, par exemple un capteur d'obstacle. La solution idéale serait de connecter ce capteur directement aux actionneurs correspondants, d'où l'absence de connexions avec la partie cellulaire.

d'interface avec les autres tâches. Dans une version antérieure du système de contrôle de l'agent explorateur du paragraphe 3.3.1 (cf. paragraphe 4.2), nous avions expérimenté avec un mécanisme de flot d'activation inspiré de celui de Maes (1989). Les systèmes d'éveil comprenaient alors une cellule d'éveil (ayant pour fonction de transfert une fonction de seuil au-dessus duquel la tâche se réveillait et s'activait) et plusieurs cellules d'observation connectées à la cellule d'éveil. Ces dernières renvoyaient à chaque stimulation comme message la quantité d'activation qui participait ensuite au cumul d'activation de la cellule d'éveil (fig. 3.4). Ces cellules étaient connectées aux cellules d'éveil des autres tâches ou à d'autres cellules de la même tâche. Dans la version actuelle des deux systèmes cellulaires de contrôle, les systèmes d'éveil sont beaucoup plus simples (figures 3.6 et 3.7) : ils comprennent une seule cellule de calcul de motivation dont le message de sortie est composé avec d'autres messages et utilisé pour la sélection de la commande de navigation (cf. paragraphes 3.3.1 et 3.3.2).

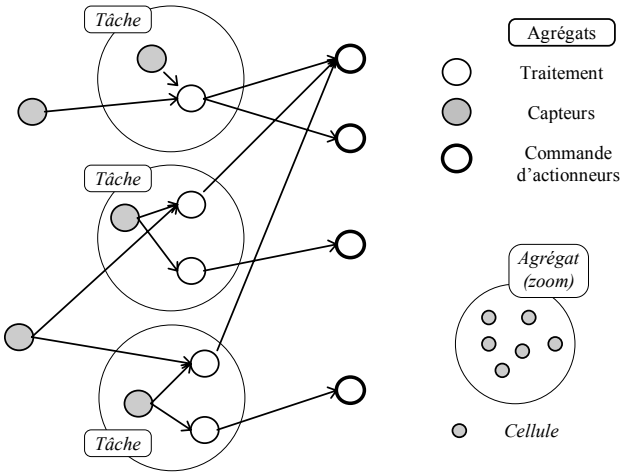

Figure 3.2 Le réseau cellulaire. Les petits cercles sont des agrégats de cellules, tandis que les grands cercles sont des tâches. Les liens correspondent à des ensembles de connexions reliant des agrégats (c'est-à-dire des cellules appartenant aux divers agrégats). Les cellules et les connexions à l'intérieur des agrégats ne sont pas visualisées, mais le petit nombre des liens sur la figure montre précisément que les connexions entre agrégats (ainsi qu'entre tâches) ne sont pas denses.

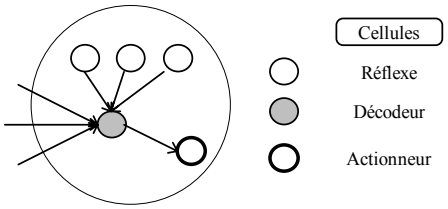

Figure 3.3 Un système de commande-d'actionneurs. À côté des connexions aux capteurs réflexes et aux actionneurs, la cellule décodeur est aussi connectée en entrée aux sorties des tâches (connexions à gauche sans entrées).

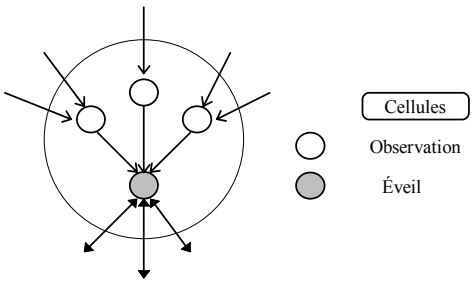

Figure 3.4 Un système d'éveil. La cellule d'éveil active ou inhibe les autres cellules de la tâche, tandis que les cellules d'observation sont connectées aux systèmes d'éveil des autres tâches ou à d'autres cellules de la même tâche.

Une tâche possède un système d'éveil, des systèmes de traitement et éventuellement des systèmes privés de perception. Les systèmes de traitement sont connectés en entrée aux systèmes de perception ou à d'autres systèmes de traitement et en sortie aux systèmes de commande-d'actionneurs. À noter que *le niveau de la cellule est le seul niveau d'exécution, tandis que les niveaux de l'agrégat et de la tâche sont utiles uniquement pour structurer la conception*.

La structure de l'activité (tâche) est généralement acyclique (feedforward) ; de leur côté, les variables physiologiques permettent une cyclicité implicite et globale. Généralement, la structure de chaque tâche reproduit un algorithme réactif à couches d'activité.

3.2.4 Grain d'activité

Nous avons identifié comme grain d'activité celui de l'activité homéostatique qui permet un arbitrage simple et localisé dans le réseau (fig. 3.5). Les "tâches" sont des activités complexes qui dépendent d'une variable motivationnelle interne (la "pulsion", qui est une mesure de "faim") et qui ont au moins trois actions différentes, *l'action consommatoire* (s'il existe de la nourriture sur ma place, je consomme[32]), *l'action appétitive* (si je perçois de la nourriture dans la proximité, je m'y dirige) et *l'action aléatoire ou désespérée* (je cherche de la nourriture au hasard). Il peut encore y avoir *des actions instrumentales* supplémentaires (je vais dans une direction où j'estime pouvoir trouver de la nourriture, même si je ne la perçois actuellement pas). Ce qui est intéressant si l'on regroupe ces actions dans une activité "motivée", c'est que l'arbitrage entre les actions élémentaires devient statique et peut se faire à l'aide d'une cellule de priorité statique (l'action consommatoire est plus prioritaire que l'action appétitive qui est à son tour plus prioritaire que l'action aléatoire), tandis que l'arbitrage entre activités motivées est dynamique et peut se faire à l'aide d'une cellule de priorité dynamique (la priorité ou la motivation de chacune des tâches est une fonction de la pulsion et du stimulus présent, elle est donc dynamique). Au sein de la même tâche, chacune des actions consommatoire, appétitive et aléatoire donne une mesure de son stimulus, telle que l'action consommatoire donne la valeur maximale, l'action aléatoire donne la valeur minimale et l'action appétitive donne une valeur dans la gamme entre le minimum et le maximum. Si alors les motivations des différentes tâches sont normalisées, pour le même niveau de la pulsion des deux

[32] L'action consommatoire est l'action qui diminue la valeur de la pulsion, sans nécessiter littéralement une consommation (cf. par exemple la tâche de retour à la base de l'agent explorateur du paragraphe 3.3.1). La même analogie s'applique à l'action appétitive.

tâches, l'action consommatoire d'une tâche est plus prioritaire qu'une action appétitive d'une autre tâche (Tyrrell (1994) discute le problème de l'arbitrage déséquilibré entre tâches qui est assez souvent retrouvé chez les mécanismes de sélection d'action). Pour la même raison, l'action aléatoire d'une tâche peut devenir plus prioritaire que l'action appétitive d'une autre tâche et par conséquent l'agent peut démontrer de la spontanéité, c'est-à-dire une tâche peut s'éveiller en l'absence de stimulus, tandis que par exemple l'architecture de Beer et Chiel (1990) ne le permet pas.[33] Dans tous les cas, tout mécanisme du type persistance ou spontanéité peut se traduire en une fonction de calcul de motivation à partir du niveau de la pulsion et de la valeur du stimulus.

Notons cependant que la tâche homéostatique, telle qu'elle a été définie, dépend d'une variable "essentielle" (selon la terminologie de Ashby (1960)), la variable de la pulsion, qui ne *peut* pas dépasser une limite physique. Le but de l'agent est d'amener et de maintenir la valeur de cette variable à 0 (à l'état de satisfaction).

Cela n'impose aucune contrainte quant à la profondeur d'une tâche homéostatique ou la forme des différentes actions ; on peut très bien avoir dans le même réseau des tâches rapides et simples de "survie" et des tâches lentes et complexes, de grande profondeur de traitement.

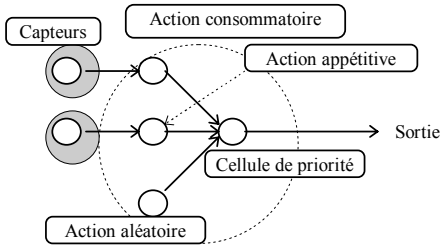

Figure 3.5 Une tâche homéostatique constituée de trois actions élémentaires, l'action consommatoire, l'action appétitive et l'action aléatoire. En réalité, l'action consommatoire n'existe pas sous forme cellulaire mais sous forme de réflexe physiologique (cf. paragraphe 3.4.3).[34]

L'arbitrage entre tâches obéit au principe de séparation qui nécessite deux niveaux de composition : le traitement des perceptions relatif à une tâche particulière est séparé aussi bien de la structure d'arbitrage ou d'éveil (premier niveau de composition) que de la structure de commande (deuxième niveau de composition) qui incorpore éventuellement des capteurs réflexes. Ainsi, divers algorithmes réactifs peuvent être développés séparément et intégrés dans des contextes différents en définissant seulement les interfaces nécessaires à chaque fois, c'est-à-dire les systèmes d'éveil. L'arbitrage se fait soit au niveau du système d'éveil soit au niveau du système de commande-d'actionneurs : au niveau du système d'éveil, l'arbitrage concerne des interactions qui peuvent être vues et gérées de façon distribuée par les tâches (telles que flot d'activation, des tâches mutuellement exclusives ou activables

[33] C'est ce qu'ils ont écrit dans leur article, mais peut-être qu'il n'y a pas de sens à parler de spontanéité en dehors d'un contexte de compétition entre tâches.

[34] Pour McFarland et Bösser (1993), l'essence des systèmes comportementaux (behavior-based) par opposition aux systèmes orientés-but (goal-oriented) est précisément la présence de *deux* capteurs différents, dont un sert pour l'action appétitive et l'autre pour l'action consommatoire : ce qui amène l'agent à son but ne doit pas être une représentation de ce but qui est ensuite comparée à ses perceptions, mais une propriété secondaire de son environnement. Il suffit ensuite de disposer d'un moyen de détecter l'accomplissement du but : *"once you get there, you know you are there"*.

en série etc.), tandis qu'au niveau du système de commande-d'actionneurs il concerne des interactions qui doivent être gérées de manière plus centralisée (par exemple des tâches ayant des priorités différentes ou dont les commandes doivent être fusionnées etc.).

Notons que le découplage entre l'activité proprement dite et les réflexes, renforce le rôle de ces réflexes sans compliquer la structure des tâches. Celles-là n'ont pas besoin de connaître les détails d'implémentation et de fonctionnalité des réflexes, ce qui permet de voir les systèmes de commande-d'actionneurs comme de vrais systèmes périphériques. La séparation du traitement de l'arbitrage et des réflexes permet donc la recombinaison de divers algorithmes réactifs ou règles réactives à l'intérieur de la même tâche (un exemple avec l'algorithme de dispersion est donné dans le paragraphe 5.3). La conception des mécanismes d'arbitrage doit reposer sur la modélisation et le contrôle des lois d'interaction entre les différents composants/structures arbitrés.

3.3 Exemples

Les deux exemples implémentés ont à peu près la même structure cellulaire qui repose sur un grain d'activité homéostatique (paragraphe 3.2.4), mais des physiologies très différentes.

3.3.1 L'agent explorateur

Le système de contrôle de l'agent explorateur, qui cherche à ramasser tous les échantillons dans un monde délimité, repose sur la définition de deux tâches motivées couplées entre elles qui peuvent être décrites comme suit (l'analyse du point de vue motivationnel et opérationnel sera présentée dans les chapitres 4 et 5) :

Comportement de chasse :
Si je suis sur un échantillon, je le ramasse (action consommatoire),
sinon si je perçois des échantillons je marche dans la direction de la plus grande densité (action appétitive),
sinon je marche au hasard : tout droit, mais avec une petite probabilité je change de direction (action aléatoire).

Comportement de retour à la base :
Si je suis à la base je dépose tous les échantillons (action consommatoire),
sinon je suis un signal d'orientation que la base émet (action appétitive).

Notons que si la base d'exploration se déplace dans l'espace, le comportement de retour à la base va paraître comme un comportement de poursuite.

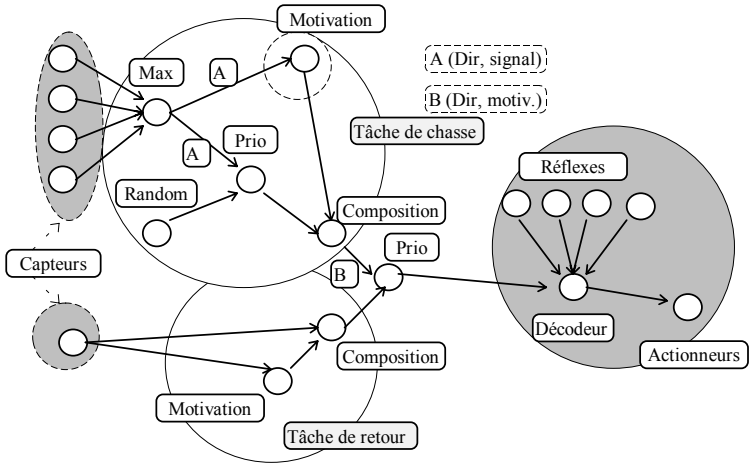

Figure 3.6 Le système cellulaire de contrôle de l'agent explorateur. Les systèmes de perception des deux tâches, celle de chasse et celle de retour à la base, ainsi que le système périphérique de navigation, sont marqués.

Le système cellulaire de contrôle de l'agent et ses réflexes sont donnés dans les figures 3.6 et 3.7. L'action appétitive de la tâche de chasse, qui est de trouver la direction de la plus grande densité d'échantillons, nécessite quatre capteurs de densité pour les quatre directions possibles de voyage en simulation et une cellule du type "max" qui choisit la direction de densité maximale. La cellule "random" émet la direction "aléatoire" de chasse. Les cellules de motivation calculent la motivation de la tâche correspondante selon les formules données dans le chapitre suivant et nécessitent en entrée la valeur du stimulus (d'où la connexion directe ou indirecte avec les capteurs), tandis que les cellules de composition doivent composer les deux messages d'entrée (la direction commandée et la motivation) en un pour permettre aux tâches d'entrer en compétition (cf. paragraphe 3.6 pour la composition des messages). La dernière cellule de priorité est une cellule de priorité dynamique qui est généralement considérée comme faisant partie du système périphérique de navigation. Elle pourrait cependant commander simultanément d'autres parties du réseau, auquel cas on dirait qu'elle fait partie d'une super-tâche de commande, c'est-à-dire d'une tâche qui raisonne à un niveau *supérieur* à celui des deux tâches de chasse et de retour (puisqu'elle prendrait comme entrée leurs sorties). Les fonctions de transfert des cellules sont discutées plus analytiquement dans le paragraphe 3.6.

Les réflexes de l'agent explorateur nécessitent trois capteurs "d'état" : un capteur qui détecte quand l'agent est de retour à la base, un capteur qui détecte quand l'agent se trouve sur un échantillon et un capteur binaire de l'état de la plate-forme de transport (remplie ou pas). Il faut en plus deux actionneurs réflexes : celui qui dépose et celui qui ramasse.

- *Si à la base, déposer tout*
- *Si sur un échantillon et plate-forme de transport non remplie, ramasser l'échantillon*

Figure 3.7 Les réflexes de l'agent explorateur

Le système d'adaptation (cf. paragraphe 4.3) de l'agent explorateur solitaire ou social nécessite une ***variable physiologique représentationnelle*** p_a et une structure complexe de mise-à-jour, dont les principaux éléments sont : (a) un compteur des échantillons ramassés qui est incrémenté à chaque ramassage et remis à zéro à chaque déchargement (il est donc connecté aux deux actionneurs réflexes), (b) un compteur

des pas de navigation effectués ou de distance parcourue qui est incrémenté à chaque mouvement et remis à zéro à chaque déchargement (il est donc connecté au moteur de navigation et à l'actionneur de déchargement), (c) une structure de calcul qui utilise les valeurs des deux compteurs ainsi que celle de p_a, (d) deux variables pour les paramètres d'adaptation (la fenêtre et le taux) et (e) une structure qui équivaut au critère de terminaison du paragraphe 4.3 et qui bloque le calcul de la motivation de chasse. L'agent social nécessite en plus un capteur de la moyenne sociale de p_a (paragraphe 5.4) qui est connecté à la structure de calcul (c). Le réseau physiologique de l'agent est visualisé dans la figure 3.8.

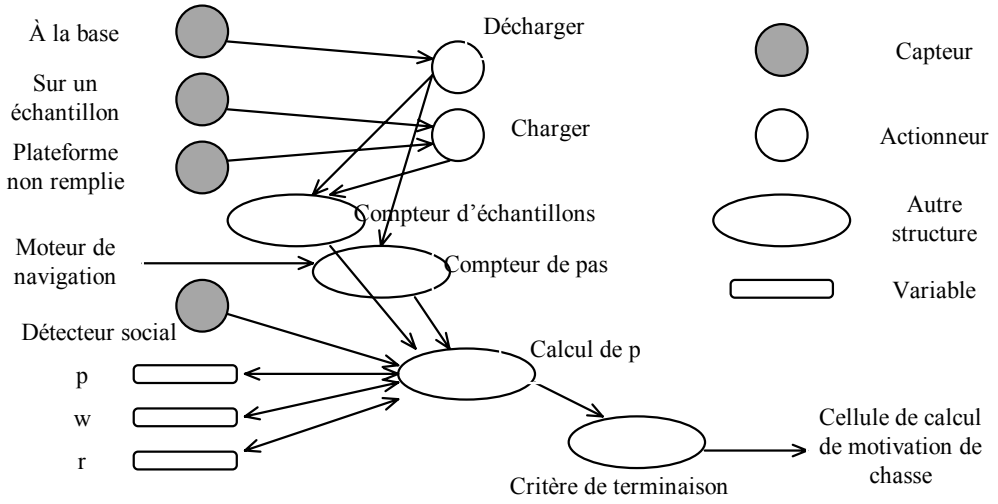

Figure 3.8 Le réseau physiologique de l'agent explorateur. Les liens représentent les dépendances entre les diverses structures physiologiques (cf. chapitre 4).

3.3.2 L'agent manager

Le système de contrôle de l'agent manager, qui cherche à réguler les demandes de desserte d'un ensemble d'unités industrielles de service, repose sur la définition d'autant de tâches motivées couplées que de nombre d'unités de service. Ces tâches peuvent être décrites comme suit (l'analyse du point de vue motivationnel et opérationnel sera présentée dans le chapitre 6) :

> ***Comportement de desserte d'unité de service :***
> *Si je suis sur l'unité de service, je la dessers (action consommatoire),*
> *sinon si je suis passé par le centre de l'atelier, je marche dans la direction de l'unité de service (action appétitive),*
> *sinon je marche dans la direction du centre (action aléatoire).*

Ce comportement n'a pas de composant aléatoire (désespéré), parce qu'il est toujours possible d'exécuter une action appétitive ou instrumentale (aller vers l'unité elle-même ou vers le centre de l'atelier).

Le système cellulaire de contrôle de l'agent et ses réflexes sont donnés dans les figures 3.9 et 3.10. L'action appétitive de chaque tâche de desserte, qui est de trouver la direction de l'unité de service, nécessite un capteur spécialisé (on peut imaginer qu'il s'agit d'un capteur de signaux radio et que chaque unité émet son signal d'orientation à une fréquence spécifique). Les cellules de propagation transforment le signal des capteurs (la direction d'orientation) en une forme utilisable par les cellules

de priorité (cf. paragraphe 3.6), les autres cellules de composition et de priorité étant comme celles de l'agent explorateur.

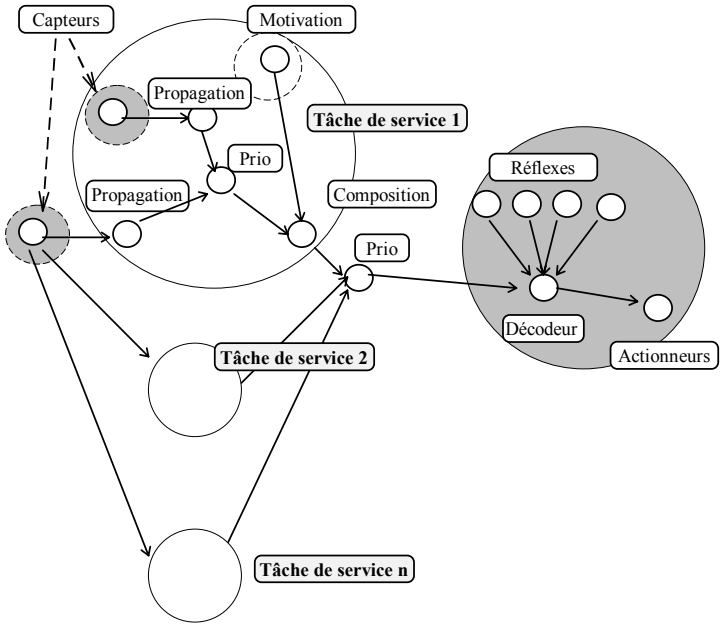

Figure 3.9 Le système cellulaire de contrôle de l'agent manager

Les réflexes de l'agent manager nécessitent un capteur qui détecte quand l'agent se trouve sur une unité et un actionneur de desserte d'unité.

- *Si sur une unité, la desservir*

Figure 3.10 Les réflexes de l'agent manager

Le système physiologique de l'agent contient plusieurs variables : (a) une variable physiologique binaire pour le centre de l'atelier qui prend la valeur "vrai" à chaque fois que l'agent passe par le centre et "faux" à chaque fois qu'il désélectionne la tâche courante pour sélectionner une autre (un capteur qui détecte quand l'agent se trouve sur le centre est donc nécessaire), (b) une variable physiologique de "marquage" qui prend comme valeur un identificateur de l'unité courante de service pour dénoter la tâche courante (et donc l'état global) de l'agent et qui est mise-à-jour lors de chaque sélection ou désélection de tâche (il faut en accompagnement un ensemble de constantes, les identificateurs des différentes unités), et (c) une variable physiologique continue du type "timer" qui est mise-à-jour à chaque sélection ou désélection de tâche et dénote la durée de la desserte (cette variable "dégrade" dans le temps puisque elle représente un timer). Quelques structures supplémentaires nécessaires liées au système motivationnel de l'agent sont présentées et analysées dans le paragraphe 6.4 (cf. fig. 6.2, fig. 6.3), qui est dédié à l'étude opérationnelle du problème et où la relation entre physiologie et planification ou décision est illustrée.

3.4 Sélection, action et autres illusions assorties

3.4.1 Sélection d'action

Nous avons légèrement abordé la question de la sélection d'action en parlant de découplage d'action et de réflexes, de disparition du besoin de subsomption, de deux

niveaux de composition et de définition de systèmes d'éveil comme interfaces entre tâches. Nous venons maintenant analyser cette question en détail.

> ***Le problème de la sélection d'action*** *: Si plusieurs actions sont possibles simultanément, comment l'agent fait-il pour choisir la plus pertinente (celle qui le mène plus près de son but/ses buts) ?*

Comme d'habitude, on doit la formulation de ce problème à une suite d'heureux accidents historiques. À première vue, il s'agit en effet d'un problème typique d'éthologie (n'oublions pas que les pionniers des agents autonomes ont été des lecteurs avides de textes éthologiques et les premiers agents autonomes ont été des insectoïdes). Mais il représente également l'alternative de la planification : sélection d'action signifie décision *en ligne*. Il est plus précisément supposé qu'au lieu de suivre un plan rigide établi hors ligne, l'agent évalue continûment plusieurs possibilités pour choisir la meilleure, *comme s'il* possédait une représentation de ses buts et/ou de ses actions ainsi que de leurs conséquences au sens classique du terme.[35] La formulation suivante pose alors le problème à l'envers ("*inside out"*) : en supposant que l'agent soit un système de composants dont chacun connaît ses conditions d'activation ainsi que l'action à entreprendre, le comportement global de l'agent émerge de leurs interactions dans un monde particulier. Au lieu d'imaginer un agent qui choisit parmi ses alternatives, on imagine un système d'alternatives dont les interactions nous font penser qu'il s'agit d'un agent (il suffit que les alternatives "sachent" s'arbitrer entre elles). Le problème de la sélection d'action est donc reformulé comme un problème d'arbitrage :

> ***Le problème de l'arbitrage entre comportements :*** *Si plusieurs comportements deviennent actifs simultanément, comment font-ils pour s'arbitrer entre eux ?*

La différence de la deuxième formulation est conceptuelle, puisque le point de vue est synthétique et ascendant : les actions ou comportements élémentaires *préexistent* et doivent être suffisamment organisé(e)s afin que la cohérence dans le réseau soit préservée. Si l'on dispose d'un grand nombre d'actions "élémentaires" redondantes ou en contradiction ou avec d'autres relations plus subtiles, le problème de l'arbitrage évolue donc vers un ***problème de cohérence*** (Brooks 1994).[36] Cependant, la question fondamentale n'est toujours pas abordée : qu'est-ce qu'une action ou un comportement ?

3.4.2 Action

Généralement, on parle d'action et on entend commande des comportements aux actionneurs (telle que *aller_tout_droit*, *tourner_à_gauche_35_degrés*, ou même *manger*, *dormir* etc.). Première constatation : cette définition présuppose des architectures "plates", des réseaux à un seul niveau ou des processus parallèles (*"flat*

---

[35] On observe clairement la confusion conceptuelle à cet égard dans des travaux se proposant comme hybrides entre IA traditionnelle et IA comportementale, tels que ceux de Gat (1992,1993).

[36] Brooks (ibid., p. 23) a écrit que le problème de la cohérence apparaît dans le cas où il existe plusieurs systèmes d'actionneurs ("ressources") non corrélés — et non seulement un système de navigation comme d'habitude. Nous verrons plus loin que certains de ces systèmes d'actionneurs peuvent être implémentés comme des réflexes qui ne se font pas commander — cela bien évidemment complexifie le système des réflexes et pose un problème de cohérence des structures non cellulaires.

networks" ... "one can substitute the term competence module or behavior for action", Maes 1994*a*). À l'intérieur d'un comportement il peut y avoir tout un réseau complexe (par exemple les réseaux des AFSM de Brooks 1986*a*) mais la sélection se fait *à l'extérieur* des comportements. Les architectures hiérarchiques parues dans la littérature (l'approche opportuniste "do whatever works" de Payton et al. 1992 et celle de Tyrrell 1993*b*) n'ont pas abordé le problème de l'action non plus. La première, qui suppose un réseau d'arbitrage de comportements exprimant une participation et de préférences locales aux actions primitives, a simplement résolu avec succès un problème de fusion de commandes dans un contexte particulier de navigation sous-marine. De son côté, la deuxième, qui est une version plus élaborée de la première, s'applique à un problème de comportement animal impliquant plusieurs motivations indépendantes de l'animal. Deuxième constatation : si l'architecture est plate, tous les comportements ont la même dynamique d'interaction avec le monde. Dans les architectures hiérarchiques, les noeuds intermédiaires sont considérés comme des points de fusion intermédiaire et les actions correspondent aux noeuds feuilles : comme dans le cas des architectures plates, toutes les actions suivent la même dynamique d'interaction avec le monde. Dans tous les cas, il y a donc un seul niveau d'action et de sélection d'action.

Quel est alors le problème avec l'action ? Je n'ai pas l'intention de reprendre les longs débats à ce sujet parus dans la littérature (cf. par exemple le recueil d'articles dans (Situated Action 1993)), mais seulement de prendre position.

> *Une action est ce qui est utilisé pour modifier une entité (Jacopin 1993, p. 1)*

Puisque nous nous intéressons aux agents autonomes et situés, ***par action nous entendons action située***. L'action située est celle où la modification en question ne concerne pas le monde, comme est maintenu dans l'IA classique (il ne s'agit pas d'amener le monde d'un état à un autre, d'une situation à une autre), mais il s'agit d'une modification interne à l'agent. Cette modification interne affecte le monde, mais l'action elle-même n'est pas une action à distance à partir des représentations non matériellement fondées. Ainsi, la transition du monde d'état à état n'est qu'un phénomène qui peut être intéressant ou pas pour un observateur externe. Cette vision de l'action constitue simplement un changement conceptuel, un changement de point de vue (Agre 1993, 1995). Jacopin (ibid., chapitre 1) a argumenté sur la modularité et la compositionalité de l'action et a insisté sur le fait qu'une action peut se décomposer récursivement autant de fois qu'on veut si c'est intéressant pour la description, donc pour un observateur ou concepteur particulier — et que finalement toute action est un réflexe déterministe et mécaniste. Il suffit donc de trouver comment décomposer pour faciliter la conception.

> *Le problème de la sélection d'action ou d'arbitrage ou de cohérence est étroitement lié à celui du grain d'action ou d'action primitive : à quel niveau de granularité d'action doit-on se placer pour décrire et concevoir le système de contrôle d'un agent autonome ?*

Si l'action est modulaire et compositionnelle et si elle peut être décomposée conformément à nos besoins de description et de conception, nous pouvons donner la définition suivante :

> ***Définition*** *: Chaque cellule est une primitive d'action.*

Puisque l'action est située, elle est considérée comme une ***action métabolique*** : la cellule a *besoin* de métaboliser ses messages d'entrée, elle n'est pas commandée à le

faire (le concept de métabolisme sera plus évident dans le chapitre 7). D'autre part, puisque le niveau de granularité est au choix, *nous choisissons comme grain d'action celui qui simplifie la structure du réseau autant que possible (selon le nombre des cellules et le nombre des connexions)*. Étant donné l'uniformité des mécanismes de métabolisme et d'interaction pour toutes les cellules, ce choix se voit dans la diversité des fonctions de transfert des cellules.

> ***Le principe de la minimalité :*** *Les fonctions de transfert des cellules sont choisies de façon que le nombre de cellules et des connexions nécessaires soit minimal.*

Nous parlons bien entendu de structures hiérarchiques plutôt que plates, qui ont l'avantage de permettre la compositionalité des actions et leur description à des niveaux différents de granularité : ***tout ensemble connecté de cellules, toute structure peut être considéré comme une action***. Plus ces actions sont complexes, plus la structure du réseau est simple.

La question de l'arbitrage devient ainsi : à quels points dans la hiérarchie les actions d'arbitrage se placent-elles ? Autrement dit, parmi toutes les actions, quelles sont les actions d'arbitrage ?

> ***Le principe de la composition des dynamiques fonctionnelles :*** *Les actions (cellules ou sous-structures) ayant des relations connues (statiques ou dynamiques) sont arbitrées par l'intermédiaire de cellules de priorité, de façon que la structure du réseau reflète la structure de la tâche (activité).*

Les cellules de priorité reçoivent en entrée des messages composés du type (priorité, message) et renvoient en sortie le message qui correspond à la plus grande priorité (éventuellement, après l'avoir métabolisé). Les priorités peuvent être statiques ou dynamiques. Un exemple de priorité statique est celui des actions consommatoires, appétitives et instrumentales qui concernent la même activité, tels qu'un comportement lié à la faim ou la soif (cf. fig. 3.5) : l'action consommatoire est toujours plus prioritaire qu'une action appétitive qui est à son tour plus prioritaire qu'une action instrumentale et cela est connu d'avance, lors de la conception. Cependant, entre deux comportements de faim ou de soif, la relation est dynamique et dépend du niveau "d'urgence" de chacun, des stimuli présents etc. Cette relation est généralement décrite à l'aide du concept de motivation qui est une mesure généralisée et continue d'urgence de la tâche/activité correspondante. On remarque que dans ce cas le système de sélection d'action est un système motivationnel. ***Un système motivationnel est donc nécessaire chaque fois que la "décision" entre deux actions (en effet, deux structures) se fait dans un espace continu au lieu d'un espace binaire*** : il ne suffit pas de savoir s'il existe des actions proposées, il faut de la connaissance supplémentaire et dynamique (généralement exprimée à l'aide de variables motivationnelles).

Notons que la motivation peut dépendre des facteurs internes ou externes à la tâche en question, tels les stimuli présents ou l'échec d'autres tâches dans le réseau, d'où le ***besoin de séparer à l'intérieur des tâches les systèmes d'éveil des autres systèmes***. La justification de cette séparation est triple : (a) possibilité d'échange du système d'éveil contre un autre système d'éveil, afin d'intégrer d'autres structures dans le réseau, (b) possibilité de complexifier le système d'éveil indépendamment ou en parallèle avec d'autres structures, par exemple pour inclure des mécanismes d'apprentissage (pour Toates (1986), le problème de l'apprentissage ne peut être

abordé qu'en relation avec un système motivationnel) et, (c) corollaire de (a), plus grande "robustesse" puisque certaines pannes peuvent être localisées (cf. chapitre 7 pour l'importance des pannes).

Le grain "idéal" d'action cellulaire est donc celui qui minimise les besoins d'arbitrage dynamique, et donc la complexité des interconnexions entre tâches : les tâches doivent être les parties du réseau ayant des connexions internes relativement denses, mais les connexions entre les tâches doivent être moins nombreuses (cf. (Minsky 1991), p. 46). Il y aurait alors un degré d'opacité (puisque beaucoup d'information spécifique à une tâche particulière ne serait pas disponible aux autres), mais ce n'est pas nécessairement un problème, car Marvin Minsky soupçonne que *"our societies and hierarchies of subsystems have evolved ways to evade the problem by arranging for some of our systems to learn to model what some of our other systems do"* (Minsky 1991, p. 48).[37]

3.4.3 Dynamiques

Les besoins de spécifiabilité et de simulabilité des structures induisent un besoin de séparation des dynamiques temporelles d'interaction avec le monde, afin de permettre au concepteur de mieux contrôler leurs relations.

> ***Le principe de la séparation des dynamiques temporelles :*** *Les structures qui suivent des dynamiques différentes sont séparées.*

Les structures-réflexes, qui ne sont pas commandées à l'intérieur de l'agent, sont séparées des structures cellulaires, ce sont des structures physiologiques. Par exemple, les systèmes de commande-d'actionneurs spécialisés à une tâche sont des réflexes (cf. par exemple le système de ramassage d'échantillons de l'agent explorateur cellulaire du paragraphe 3.3.1) ; la navigation n'est pas une structure-réflexe parce qu'elle est commandée par d'autres parties du réseau (la navigation est une action instrumentale ou appétitive pour la plupart des tâches). De nombreux systèmes de commande-d'actionneurs spécialisés peuvent être actifs en même temps : par exemple, un robot peut émettre un signal tout en se déplaçant dans l'espace, si la condition d'émission du signal est indépendante de la navigation (un exemple d'une telle condition est la baisse du niveau de l'énergie).

C'est une pratique de la robotique comportementale de définir comme réflexes les systèmes de commande-d'actionneurs spécialisés pour certaines actions consommatoires. Connell (1990, chapitre 3) a résolu le problème de l'arbitrage sur le robot Herbert qui devait à la fois naviguer et ramasser des boîtes vides de soda avec son bras manipulateur en introduisant dans le système de manipulation un simple réflexe binaire au niveau le plus bas : un rayon lumineux entre les doigts de Herbert se brisait ce qui déclenchait la fermeture des doigts.[38] Les robots de Steels (1994*c*), qui doivent aller recharger leurs piles dans une station de rechargement, n'ont pas de système de rechargement qu'ils commandent (il n'y a pas de comportement de

---

[37] L'architecture non hiérarchique de Brooks est souvent accusée de réplication inutile de structures de base dans les différentes couches de fonctionnalité parallèles ou d'incorporation de connaissance sur les comportements subsumés (cf. par exemple Payton et al. 1992, p. 235-236). Il faut croire que la réplication et l'opacité sont inévitables à un certain degré : le problème est de trouver quel est ce degré.

[38] Connell lui-même a appelé ce réflexe un *modèle* simple de boîte de soda (*"... anything that fits between the robot's fingers counts ..."*, p. 51).

rechargement)[39] : à sa place, ils ont une structure-réflexe, une paire de barres qui ferment le circuit de rechargement, lorsqu'elles entrent en contact avec deux disques embarqués sur les pôles de la station.

> ***Le principe de l'indépendance des actionneurs :*** *Les actionneurs spécialisés sont des réflexes.*

Pourquoi des actionneurs réflexes ? S'ils n'étaient pas des réflexes, et pour éviter un problème d'incohérence, toutes les tâches devraient y avoir accès pour les commander d'agir ou de se taire. Imaginons, par exemple, un système d'émission de signal commandé d'une tâche A. Nous supposons que la tâche A sait quand commander le début et quand la fin de l'émission. Une deuxième tâche B "sélectionnée" aurait alors à avoir accès à ce système pour commander la fin de l'émission ou pour inhiber son émission. Il est plus simple d'avoir un actionneur qui dépend d'un relais binaire "commandé" d'une condition environnementale ou interne : à chaque changement de l'état de la condition, le relais inverse son état, ainsi les transitions inutiles sont éliminées. Toutes les variables physiologiques suivent le même principe de conditions-réflexes pour permettre la régulation éloignée dans le réseau.

Regardons maintenant l'architecture incrémentale de Kube & Zhang (1994) de plus près (fig. 3.1). On voit que, les actions intuitivement de plus bas niveau (l'évitement des obstacles) ont la plus grande priorité, ce qui contourne le problème de la subsomption, puisque les couches de haut niveau n'ont plus besoin d'avoir accès à l'information des capteurs de bas niveau. Cet ordre de priorités rend l'extensibilité descendante (on doit ajouter des niveaux bas d'action !). À propos de niveaux et de priorités, Kaelbling (1986, p. 404) a écrit que le niveau le plus bas, c'est-à-dire celui qui nécessite l'information la plus primitive, doit toujours être actif et avoir une priorité inférieure à celle des autres niveaux. Or, ce n'est pas le cas des réflexes d'évitement d'obstacle : lorsqu'ils deviennent actifs, ils sont plus prioritaires que les autres actions. C'est cette observation qui m'a fait penser très tôt à des structures plus hiérarchiques et au besoin de déplacer ces réflexes dans un système périphérique de navigation. Alors les autres actions n'ont plus besoin d'avoir de l'information sur l'état de ces réflexes et n'ont plus besoin de subsumer les réflexes pour l'acquérir ; le système périphérique de navigation s'occupe de manière décentralisée de la "fusion" de cette information avec les commandes de haut niveau.

3.4.4 Extensibilité compositionnelle

Si toutes les structures sont en effet des actions, combien de niveaux conceptuels d'action faut-il ? Jusqu'ici nous en avons défini trois : cellule, agrégat, tâche. Le premier niveau est celui de l'exécution, il est donc indispensable. Le deuxième niveau est celui de la coordination (les cellules-membres du même agrégat sont corrélées de façon fonctionnelle). Finalement, le troisième niveau est celui de l'organisation même de l'agent. Cette décomposition est celle rencontrée dans la nature (avec les trois niveaux de cellule, tissu et organe), mais aussi celle proposée dans le cadre du contrôle intelligent théorique (Saridis 1987).[40] Chaque niveau récursif constitue un

---

[39] Il y faut cependant un comportement d'inhibition de la navigation quand le robot est en cours de rechargement.

[40] Je suis tentée de dire que le niveau d'organisation (ou celui de coordination) est redondant et qu'il suffit de disposer d'un seul niveau et d'un mécanisme de méta récursif pour reproduire un système de

niveau de compétition et de composition pour les actions du niveau inférieur. Le niveau plus global de l'agent est donc le niveau de compétition et composition entre les tâches. On peut monter des niveaux de composition et alors de conception d'action en permettant de connexions réflexives entre les tâches sans nécessiter un niveau conceptuel supplémentaire. Avoir par exemple une tâche qui observe une autre en cours d'exécution, auquel cas on pourra parler de capteurs et d'actionneurs internes ou virtuels. Cette idée de développement récursif hiérarchique se retrouve régulièrement dans pratiquement toutes les disciplines. En ce qui concerne l'intelligence artificielle, les robots et les animats, on peut citer le méta-raisonnement de Pitrat (1991), le contrôle cognitif de Meystel (1991), la théorie de l'action cognitive de Roitblat (1991) et la cognition hiérarchique de Toates (1994).

L'idée de la séparation des différentes dynamiques est que la commande aux actionneurs n'a pas toujours besoin d'être couplée avec la perception et à l'action locale à une tâche. Il paraît que c'est précisément ce couplage parfois inutile qui a empêché les architectures comportementales d'être extensibles, puisqu'un seul mécanisme de sélection d'action était utilisé *universellement* en n'étant en réalité adapté qu'à une sous-classe de problèmes de sélection. Même les mécanismes de sélection d'action les plus élaborés, les mécanismes hiérarchiques de Payton et de Tyrrell, n'abordent toujours pas les problèmes complémentaires de l'extensibilité et de la compositionalité architecturale.

> ***Le problème du dimensionnement ou de l'extensibilité :*** *Si on complexifie le problème original, par exemple en introduisant de nouveaux "comportements" ou de nouvelles contraintes, est-on certain de pouvoir étendre respectivement le système de contrôle de l'agent cellulaire et comment ?*

Une réponse typique est celle de Maes (1994*a*, p.150) : la reconfigurabilité est possible par l'intermédiaire de réutilisation de composants. On a donc besoin d'un répertoire de composants spécifiables et paramétrables ainsi que d'un mécanisme de composition.

> ***Le problème de la compositionalité architecturale :*** *Est-il possible de réutiliser et de composer des structures élaborées séparément dans des contextes différents, si oui dans quel cas et comment ?*

Si toutes les structures sont des actions et si à tous les niveaux on peut avoir de la sélection, les deux problèmes deviennent en effet un problème unifié : celui de l'action et de sa composition. Conséquence immédiate de cette formulation : si le problème de l'extensibilité est en effet ***le problème de composition de l'action***, et vu que l'action est algorithmique, l'action et la composition seront *hétérogènes*. Pour (Maes 1994*a*), le problème de sélection d'action ou d'arbitrage à un niveau plat est déjà assez complexe pour vouloir passer à une architecture hiérarchique et compositionnelle ; elle voit donc l'utilisation de méthodes évolutives ou d'apprentissage comme la seule voie prometteuse pour le dimensionnement d'une architecture.

---

niveaux imbriqués d'une profondeur quelconque. La seule raison en faveur de trois niveaux au lieu de deux est qu'au niveau de la coordination les relations entre actions sont statiques, tandis que au niveau d'organisation les relations entre actions sont dynamiques — mais ce n'est qu'une raison pratique et non conceptuelle, tout comme la distinction entre tissu et organe n'a de sens que pour la description d'un organisme, les seules entités vivantes étant toujours l'organisme et ses cellules.

L'extensibilité compositionnelle s'oppose à une extensibilité incrémentale au niveau conceptuel mais pas nécessairement au niveau pratique : par exemple, l'extensibilité incrémentale de l'architecture de la subsomption augmentée par des mécanismes globaux de régulation hormonale peut certainement servir de base pour tout un éventail de systèmes de contrôle et de modes de sélection assez complexes (Parker (1992, 1994) est même parvenu à définir des systèmes motivationnels). Il suffit de complexifier le système hormonal. La question n'est donc pas "est-il possible d'avoir une structure d'un type ou d'un autre dans un réseau incrémental ?", mais si oui "est-ce la meilleure façon de le faire ?". Tsotsos (1995) croit que pour certaines structures impliquant des représentations intermédiaires, la réponse à la première question est négative dans ce cadre.

3.5 Une note sur l'adaptation

Puisque l'adaptation affecte certains paramètres organisationnels de l'agent (dans la plupart de cas il s'agit d'autorégulation, paragraphe 1.3.3), il est naturel d'implémenter et de décrire les systèmes d'adaptation comme des systèmes physiologiques non cellulaires, sinon ils seraient des cellules spécialisées sans entrées (leurs sorties seraient toutes les cellules qui utilisent les paramètres régulés). Par exemple, supposons que l'adaptation concerne la valeur d'un paramètre utilisé par une cellule de calcul de motivation (p_a, cf. paragraphes 3.3.1 et 4.3). Une solution est de localiser ce paramètre dans une cellule spécialisée responsable de l'adaptation. Cette solution présente la même difficulté pratique qui a été discutée dans le paragraphe 3.2.1 à propos des structures physiologiques : si les parties du réseau cellulaire qui dépendent de la valeur du paramètre sont éloignées ou séparées, la gestion des connexions peut devenir impossible. De plus, une structure d'adaptation ressemble plutôt à une structure réflexe, qui "réagit" à une condition du réseau ou de l'environnement, qu'à une structure cellulaire commandée par d'autres parties du réseau. Il est donc préférable de l'implémenter comme une structure physiologique. Comme toute autre structure physiologique, cette structure n'est pas permise de changer, donc d'apprendre (une fois de plus, il n'existe pas de justification théorique pour cette limite fonctionnelle, elle correspond simplement à une limite de complexité que nous nous posons).

> *Les systèmes d'adaptation sont des systèmes de régulation physiologique : l'agent n'y a pas accès.*

Dans la version antérieure de l'agent explorateur (paragraphe 4.2), nous avions expérimenté avec un mécanisme d'adaptation à l'intérieur des cellules d'éveil, d'observation et d'une cellule *min*. L'adaptation concernait une valeur numérique (telle qu'un seuil) et se faisait par renforcement (positif ou négatif) ou par remplacement avec une autre valeur, elle était donc une opération physiologique *à l'intérieur* de la cellule, c'est-à-dire une opération différente et indépendante de celle de l'action de la cellule.[41] Il suffit de retenir que dans tous les cas ***l'adaptation est une opération physiologique qui apparaît comme un effet de bord de la "vraie action"***

[41] Puisque, dans cette implémentation, le mécanisme d'arbitrage de base n'était pas opérationnel (paragraphe 4.2), le mécanisme d'adaptation ne l'a pas été non plus. Cela ne signifie pas que les mécanismes comme ceux-là ne sont pas opérationnels en général, seulement qu'ils ne sont pas opérationnels dans ce cas particulier.

métabolique, c'est-à-dire de la commande de l'animat ou de la fonction de transfert de la cellule.

3.6 Le problème de l'espace de représentation et la sémantique

Nous avons déjà dit que l'action ne peut pas être dissociée de la structure : toute structure peut être vue comme une action. Nous avons également dit que la syntaxe des connexions est homogène. Pour une fonction quelconque de transfert qui transforme un signal A en un signal B, A et B sont des points dans un espace et la fonction de transfert est une opération dans le même espace. Il s'agit donc de ***l'espace de représentation de l'action***, c'est-à-dire l'espace dans lequel le concepteur définit et conçoit les actions.

Question : Quel est l'espace de représentation pour un système de contrôle d'agent autonome ?

Quel est l'espace dont la forme convient aux messages qu'on veut pouvoir traiter et quelles sont les opérations primitives nécessaires (ou, quel doit être la chimie des agents autonomes artificiels) ? Suivant un peu aveuglément la consigne "pas de symboles, ils ne sont pas résistants au bruit" (cf. par exemple Schöner & Engels 1994), je me suis donnée la contrainte de l'espace numérique : tous les messages sont des nombres réels. J'ai pourtant maintenu en principe toutes les libertés quant aux opérations, ce qui m'a permis d'identifier un certain nombre de problèmes et qui m'a conduite à repenser la nature de l'espace de représentation.

Le problème de la concaténation et de la séparation. Prenons une des cellules de priorité de la figure 3.6. Tout l'intérêt d'une cellule de priorité, qui correspond à une action primitive de sélection, est de pouvoir choisir entre plusieurs entrées selon un critère qui ne dépend pas de l'entrée elle-même. Il faut donc des messages composés du type (entrée, critère), en l'occurrence l'entrée est la direction commandée et le critère est soit la valeur du signal (message composé du type A, fig. 3.6), soit la motivation de la tâche (message composé du type B, fig. 3.6). J'ai implémenté le mécanisme de concaténation/séparation dans l'espace des nombres réels comme *un mécanisme de codage/décodage* : le message composé *(x,y)* est défini comme le message (numérique) $z=B*x+y$, où B est une puissance de 10 telle que $y<<B$, et z se décompose ensuite comme $x=z \text{ div } B$, $y=z-(x*B)$. Cette implémentation présente deux problèmes. En premier lieu, x doit être un nombre entier sinon il doit être traduit en un nombre entier, par exemple en l'amplifiant par C où C est une puissance de 10 telle que $x<<C$. Deuxièmement, les messages composés peuvent monter rapidement vers des nombres très grands si plusieurs messages sont concaténés ($z = (x_1,x_2,\ldots,x_n) \Rightarrow z = (B_1 * x_1 + x_2) * B_2 + \ldots) * B_{n-1} + x_n$) et notamment des messages réels qui peuvent prendre une grande gamme de valeurs. Notons que la concaténation/séparation des messages peut être opérationnelle seulement en présence des bonnes interfaces entre les cellules intéressées : les paramètres B et C du codage/décodage doivent être communs aux cellules, autrement dit les cellules doivent disposer d'une ***sémantique commune*** leur permettant de communiquer correctement (au contraire, dans les réseaux neuronaux artificiels traditionnels, il n'y a aucune "sémantique" particulière, tous les messages sont traités ou interprétés de la même manière).

Le problème de l'identification des messages. Si les messages à composer proviennent de différentes cellules-sources, il faut des actions intermédiaires de composition[42] (cf. les cellules de composition de fig. 3.6). Mais alors la composition est en contradiction avec le principe de la syntaxe homogène : on ne peut pas distinguer les différentes entrées. Que fait-on ? Le mécanisme de codage/décodage permet une solution directe : pour composer x et y comme précédemment, nous introduisons deux cellules, dont la première amplifie x par B et la deuxième somme la sortie de la première cellule et l'entrée y originale. Bien qu'aucun principe ne soit violé, cette solution conduit à des actions ou des structures de composition linéaires au nombre des entrées en composition et même de profondeur linéaire au nombre des entrées en composition. J'ai alors essayé de trouver une structure alternative de composition moins complexe. L'idée-clé est ***d'identifier les messages sans distinguer les entrées***. Il suffit de composer localement les messages avec des identificateurs de cellule de provenance, c'est-à-dire renvoyer des messages composés du type (identificateur, message) qui seront décomposés dans la cellule destinataire. Ces identificateurs sont encore une fois des nombres réels et au niveau de la reconnaissance une opération de comparaison (*matching*) suffit : dans l'espace des nombres réels, regarder si la différence absolue des deux valeurs est inférieure à un petit seuil de tolérance, constitue une opération de comparaison efficace et résistante au bruit.[43] Ce besoin d'identification des messages apparaît pour les cellules de priorité, les décodeurs et la cellule max de la figure 3.6.

Les mêmes considérations de sémantique sont également pertinentes dans le problème de l'identification. ***La propriété la plus importante de l'espace de représentation semble être la possibilité de permettre la définition (et même l'émergence) de telles sémantiques hétérogènes.*** Nous avons vu que l'espace des nombres réels pose un problème de complexité lors de la définition d'une sémantique compositionnelle. Un espace discret et/ou symbolique ne démontrerait pas cet inconvénient : le message 'ab' est le message obtenu si l'on met simplement l'atome 'a' à côté de l'atome 'b'. L'opération de la concaténation et son alter ego, la séparation, sont des opérations primitives dans cet espace. La chimie organique définit de son côté un espace très riche en opérations discrètes de ce type-là : il existe par exemple des structures cycliques, linéaires, hiérarchiques etc. Je ne vois aucun problème conceptuel quant au traitement symbolique, si ce n'est qu'il n'est pas bien adapté à notre matériel informatique : l'état symbolique des AFSM (Brooks 1986*a*) est un état constitué de quelques bits indépendants, de sorte que la moindre erreur conduit à une distorsion sémantique énorme. Il faudrait imaginer un espace où le traitement "symbolique" serait résistant au bruit et aux mutations ; une fois de plus, la chimie organique me paraît un espace parfait sous cette optique. Notons que le traitement symbolique a été très tôt appliqué à la simulation de processus biologiques ainsi qu'à la recherche d'analogues biologiques[44] et que la seule contrainte des fonctionnalités étudiées est une contrainte de complexité (Stahl & Goheen 1963). Cette même contrainte de

---

[42] Varela (1979, p. 50) distingue entre la *composition* qui implique une transformation chimique et nécessite un catalyseur (et dont la *désintégration* est le complément) et la *concaténation* qui est la formation d'une chaîne.

[43] Puisqu'il faut une opération de reconnaissance par entrée, ce modèle de composition favorise une cellule constituée d'un ensemble de primitives de reconnaissance qui s'exécutent en parallèle.

[44] Sauf que, *"[r]ather surprisingly, Turing wrote a paper on chemical morphogenesis [1952] but did not cite his machine or algorithm theory in this work"* (Stahl & Goheen 1963, p. 268).

complexité que Chapman (1990) discute à propos de ses réseaux électroniques compilés. Le problème de l'espace de représentation est évoqué de nouveau dans le chapitre 7 avec une discussion sur la sémantique de la symétrie, les formes topologiques et la possibilité de mutation.

Finalement, au niveau physiologique (non cellulaire) l'espace de représentation comprend, comme nous l'avons déjà dit, des variables continues ou binaires (des relais) qui maintiennent et qui représentent des états globaux non commandés à l'intérieur de l'agent. Le rôle de cette mémoire cyclique et répétitive est d'assurer la cohérence ou de réguler le réseau globalement.

3.7 Conclusion

Ce chapitre a présenté un ensemble de principes architecturaux pour la conception des systèmes de contrôle d'agents autonomes et situés. Nous avons justifié notre choix de base, qui était de rester dans la problématique connexionniste algorithmique plutôt que de classification et de considérer l'architecture comme un langage de programmation plutôt que comme une structure panacée, passe-partout. Le premier principe est celui de la cellularité, qui est le mode d'organisation d'un réseau dans lequel les fonctions de transfert des différentes cellules sont hétérogènes, mais la syntaxe est homogène ; la plasticité se manifestera comme la découverte de nouvelles sémantiques de connexions. La partie cellulaire de l'organisation est couplée avec une partie physiologique qui contient des variables binaires ou continues commandées par plusieurs endroits dans le réseau et des structures réflexes. La forme hiérarchique de l'organisation a été ensuite décrite et la tâche homéostatique a été définie comme le quantum de tâche dans l'organisation. Nous avons implémenté informatiquement deux systèmes de contrôle d'agents dont les détails omis ici sont présentés dans les chapitres 4 et 6, respectivement.

Nous sommes ensuite passés à la considération du problème de l'action et de la sélection, pour parler d'unification structure-action, de sélection d'action qui est reflétée dans la structure, de composition d'action et de séparation des différentes dynamiques dans l'organisation. Le besoin de définir la plupart des actionneurs comme des réflexes non commandés à l'intérieur de l'agent a été analysé. Le problème du dimensionnement a été discuté sous l'angle de la compositionalité de l'action, de même que le besoin d'avoir des structures d'adaptation physiologique dans l'organisation. Finalement, le problème de l'espace de représentation a été présenté et une brève analyse a été tentée à base d'exemples précis concernant les problèmes de la concaténation, de la séparation et de l'identification des messages et de la sémantique dans le réseau.

En ce qui concerne les critères posés, les principes de la séparation des différentes dynamiques assurent la simulabilité et la spécifiabilité de l'organisation. Quant à la question de l'extensibilité, et puisque nous ne disposons pas d'outils formels d'analyse, nous ne pouvons pas juger du degré de satisfaction du critère. Dans tous les cas, toute la puissance des principes présentés n'a pas été exploitée. Nous croyons que l'implémentation des systèmes de contrôle d'autres agents autonomes révélera plus de rapports entre problèmes abordés et principes d'organisation et conduira éventuellement à un enrichissement et un raffinement de ces principes.

Le problème	Architecture des agents autonomes. Spécifiable, simulable et extensible
Les applications	Agent explorateur solitaire ou social Agent manager
La solution	*Cellularité* (fonctionnalités des cellules hétérogènes, syntaxe des connexions homogène) ⇒ Efficacité, incrémentalité, plasticité sémantique (cellules et connexions) • Cellules : Capteurs, actionneurs, cellules de traitement • Structure du réseau : Trois niveaux imbriqués (cellule, agrégat de cellules, tâche ou comportement) *Physiologie* : Milieu interne d'interactions, incapable d'apprendre (variables et structures globales) *Couplage* entre cellularité et physiologie
Les enseignements	• Grain d'activité homéostatique • Unification structure-action • Sélection d'action reflétée dans la structure (sélection à plusieurs niveaux) • Principe de minimalité • Séparation des dynamiques temporelles et spatiales, indépendance des actionneurs • Composition des dynamiques fonctionnelles • Dimensionnement/extensibilité ⇔ Compositionalité • Adaptation physiologique • Espace (abstrait) de représentation des actions ⇔ Sémantiques hétérogènes (concaténation, séparation, identification des messages)

Tableau 3.3 Tableau récapitulatif du chapitre 3

Chapitre 4 L'explorateur solitaire

"Ce jeu mortel qui mène de la lucidité en face de l'existence à l'évasion hors de la lumière, il faut le suivre et le comprendre."
(Albert Camus, "Le mythe de Sisyphe", 1942)

4.1 Introduction

Un problème typique de robotique comportementale est celui de ***l'exploration*** : un ensemble d'agents (robots) débarque sur une planète[45] avec la mission d'explorer sa surface pour des échantillons de minerai ayant certaines propriétés. Les robots arrivent dans un vaisseau spatial qui sert de base planétaire tout au long de la mission. La mission est accomplie quand toute la surface dans un certain rayon de la base est explorée. À noter que ce problème d'exploration se présente comme un problème ***d'échantillonnage***, où les agents doivent ramasser *quelques* échantillons des sources de minerai intéressantes, tandis que, en pratique dans la littérature correspondante, on traite du problème comme un problème de ***balayage***, où les agents doivent *épuiser* ces sources d'intérêt (cf. par exemple Brooks et Flynn (1989), Beckers et al. (1994)). Dans les deux cas, il s'agit d'un problème de ***couverture de champ*** : le critère de terminaison est que *toute* la surface impliquée doit être explorée. Dans le même ordre d'idées que les chercheurs en robotique comportementale, nous avons adopté la variante du balayage qui est susceptible d'une instanciation plus "mondaine" : imaginons un ensemble d'agents (robots) lancés dans un garage ou un autre espace délimité, avec la mission de nettoyer toutes les instances d'objets d'un certain type.[46] Les agents sont supposés de rentrer à la base lorsque leur mission est finie : les agents-nettoyeurs de métro vont s'éveiller pour entrer en activité en dehors des heures d'ouverture, par exemple pendant la nuit, et rentrer à leur base définitivement lorsqu'ils auront tout nettoyé, avant la re-ouverture de la station. Par rapport à l'échantillonnage, le balayage paraît ainsi un problème plus "primitif", puisqu'il présuppose toute la fonctionnalité de navigation, de détection et de localisation, sans nécessiter un raisonnement spatial sophistiqué : dans le cas de l'échantillonnage, le robot doit "se souvenir" d'une manière ou d'une autre de toutes les sources de minerai qu'il a déjà explorées pour ne pas ramasser des échantillons redondants, tandis que,

[45] Ce problème doit son nom et sa formulation à un des premiers projets de robotique comportementale qui visait — de manière un peu futuriste — à l'exploration de Mars (Angle & Brooks 1990).

[46] Des auditeurs naïfs auxquels j'explique le fonctionnement du système, imaginent d'habitude "un robot qui ramasse les sacs noirs dans les couloirs des stations de métro".

dans le cas du balayage, le fait d'avoir visité une source ne doit pas être enregistré en mémoire en tant que tel[47].

Le problème du balayage a été abordé jusqu'à présent du point de vue "fonctionnel" :

> ***Balayage (1) - Point de vue fonctionnel :*** *Comment un ou plusieurs agents balayent un espace délimité pour épuiser les sources d'intérêt ?*

La réponse à cette question est un système de contrôle, une architecture, qui permet à l'agent de naviguer, percevoir, détecter du minerai etc., afin de balayer tout l'espace en question.

Une solution, comme celles rencontrées dans la bibliographie (par exemple, Mataric 1992*c*) et comme celle que nous avons présentée dans le chapitre 3 avec un composant aléatoire et même sans apprentissage ou raisonnement spatial, assure statistiquement la couverture du champ d'intérêt et l'épuisement des sources de minerai. Mais d'un point de vue plus "cognitif", cette fonctionnalité seule ne répond pas à la question essentielle :

> ***Balayage (2) - Point de vue cognitif :*** *Comment les agents savent-ils qu'ils ont balayé tout l'espace, ou qu'ils ont accompli leur mission ?*

Pour répondre à cette question, il faut reformuler la description de la tâche de balayage, de manière à y inclure une expression, analytique ou autre, qui représente le critère de terminaison, c'est-à-dire l'épuisement des sources de minerai. Il suffit alors de définir une variable environnementale, la densité des sources de minerai, qui caractérise l'état du monde à un instant donné. Le but de l'agent explorateur-balayeur devient donc de ramener la valeur de cette variable à 0. Nous verrons qu'un agent ayant une représentation de cette variable constitue une solution simple à ce problème de description.

> ***Balayage - Variable environnementale :*** *La variable critique qui décrit la tâche de balayage est la densité des sources d'intérêt (par exemple, minerai) dans le monde, dénotée par la suite comme* p_m.

Troisièmement, nous cherchons à étudier l'opérationalité du système, c'est-à-dire la relation entre l'architecture interne des agents et leurs performances, dans le but de trouver une architecture qui "optimise" ces performances-là. Le critère d'opérationalité qui s'applique à la tâche de balayage est, bien évidemment, la durée de la mission : les agents sont plus performants s'ils se rendent compte de la terminaison de leur mission plus rapidement.

> ***Balayage - Critère opérationnel :*** *La mesure des performances de l'agent (des agents) est la durée de la mission : un agent A est plus opérationnel qu'un agent B, si pour les mêmes conditions environnementales initiales, il accomplit sa mission plus vite.*

Le paramètre libre de la tâche de balayage est la densité (initiale) des sources de minerai dans le monde. Une "optimisation" des performances consiste ainsi en une linéarité de la durée de la mission selon la valeur de la variable environnementale. La durée de la mission est la somme de la durée du balayage proprement dit (c'est-à-dire

[47] La mémorisation ou le "marquage" des positions visitées, explicitement ou implicitement à l'aide de mécanismes tels que les traces-phéromones (par exemple, Drogoul et Ferber 1992), induit une augmentation des performances et une augmentation de la vitesse de balayage, mais ne fait pas partie de la description de la tâche de balayage.

de la durée jusqu'au moment du ramassage du dernier échantillon) plus le temps supplémentaire jusqu'à ce que l'agent comprenne que tout l'espace est balayé et retourne définitivement à la base. Le premier terme est une fonction statistiquement linéaire de $p_m(0)$, qui dépend exclusivement de la fonctionnalité élémentaire de l'agent, c'est-à-dire de la façon dont il balaye l'espace en ramassant les échantillons. Le système "cognitif" de l'agent selon la définition précédente est responsable du deuxième terme. Les deux systèmes, celui de ramassage et le système "cognitif", fonctionnent en parallèle, comme nous le verrons dans le paragraphe 4.3.

> ***Balayage (3) - Point de vue opérationnel :*** *Quel est le modèle comportemental d'agent-balayeur qui "optimise" les performances du système pour toutes les conditions environnementales initiales ? C'est-à-dire, quel est le modèle qui permet à l'agent de terminer sa mission presque aussi vite après l'épuisement des sources pour toute valeur de* $p_m(0)$ *?*

Dans les simulations effectuées, le monde sous exploration est défini comme un carré autour de la base centrale : la taille du monde est alors la longueur du carré (sauf indication contraire, les résultats reportés par le suite ont été pris dans un monde 25x25). Nous supposons que la base émet régulièrement un signal d'orientation, que les agents perçoivent et qu'ils utilisent pour rentrer à la base. Nous supposons également que la capacité de transport des agents est limitée (ici elle a été fixée au 30) — c'est ce qui les oblige de faire des voyages aller-retour à la base pour déposer la quantité de minerai ramassée. Finalement, puisque les simulations sont discretisées dans le temps et dans l'espace, nous supposons que toute "action" (mouvement, chargement d'un échantillon, déchargement complet à la base) dure une unité de temps de simulation.

Nous décrivons dans les deux paragraphes suivants le système motivationnel de l'agent balayeur qui assure les fonctionnalités de base et le composant cognitif élémentaire qui assure la terminaison de la mission. Après avoir présenté et comparé les deux alternatives d'adaptation envisagées, nous étudions dans le paragraphe 4.5 le couplage opérationnel de l'agent avec son environnement et nous montrons le besoin d'un mécanisme d'autorégulation. Finalement, nous discutons le cas des mondes des tailles diverses.

4.2 Niveau fonctionnel : Système motivationnel

La première implémentation du niveau fonctionnel de l'agent explorateur a été basée sur les approches antécédentes rencontrées dans la bibliographie, dont l'exemple le plus complet est celui de (Mataric 1992*c*). Il y avait deux tâches par agent : la tâche de fourragement (ou chasse) et la tâche de retour à la base. Il y avait aussi un mécanisme de flux ou de propagation d'activation entre tâches inspiré de celui de Maes (1989, 1991*b*,*c*,*d*,*e*,*f*).

Cette implémentation a révélé deux problèmes. Premièrement, le comportement de l'agent démontre un phénomène de ***cycle opérationnel*** avec les allers-retours à la base, qui n'est *pas* explicitement prescrit dans la description des tâches, c'est-à-dire ces cycles phénoménaux étaient émergents. Cela ne serait guère gênant si notre but était simplement de développer le système fonctionnel de l'agent ; cependant, dès qu'on s'intéresse à l'opérationalité du système, c'est-à-dire à ses performances,

l'émergence ne suffit pas. Ce qui paraît nécessaire, c'est la possibilité de contrôler ces phénomènes autrement émergents par l'intermédiaire d'un système motivationnel relativement central.

Deuxièmement, si le flux d'activation entre les différentes règles (ou actions individuelles) qui constituent les deux tâches est permis d'augmenter/diminuer au cours de la propagation, la comparaison des niveaux d'activation et la compétition entre les tâches devient non équilibrée, ce qui entraîne comme conséquence que la conception du mécanisme de propagation d'activation devient problématique. Ce point a été révélé et analysé en détail par Tyrrell (1993*a*, 1993*b*, 1994) qui a suggéré une conception hiérarchique des architectures et éventuellement la définition de systèmes centraux de sélection d'action.

Les deux problèmes de cette première implémentation se traduisent ainsi par un problème d'arbitrage entre comportements et de sélection d'action. Un des principes de l'architecture cellulaire du chapitre 3 est précisément l'existence de deux niveaux de sélection d'action : celui des "actions" qui peuvent être arbitrées hors ligne et celui des actions pour lesquelles le critère d'arbitrage est dynamique, c'est-à-dire dépendant de paramètres dynamiques. Dans la figure 3.6, a été décrite l'architecture de l'agent explorateur et il a été expliqué pourquoi la moitié du problème de la sélection d'action (la sélection statique) était résolue. Nous donnons ici la deuxième moitié de la solution, celle de la sélection dynamique.

Les deux tâches doivent calculer localement leurs motivations qui seront ensuite comparées et arbitrées au niveau des systèmes d'actionneurs, ici le système de navigation. Ces motivations doivent dépendre à la fois d'une variable (ou "pulsion") interne ainsi que d'un ou plusieurs stimuli externes (qui correspondent à la "récompense" future attendue). Le "but" de l'agent hédoniste est de ramener les deux motivations à 0. Selon la littérature éthologiste (cf. par exemple McFarland & Bösser (1993)), la formule de calcul de motivation doit être additive ou multiplicative selon les deux mesures, les niveaux de la pulsion interne et des stimuli externes :

Tâche motivée :

$motivation = besoin * f(stimulus)$ (1)

$ou,\ k * besoin + (1-k) * f(stimulus)$ (2)

$f(stimulus) = (stimulus + a)/(stimulus_{max} + a)$ (3)

Pour permettre une compétition équilibrée entre les différentes motivations, il faut une normalisation de toutes les variables (ici elles sont normalisées entre 0 et 1). Pour une règle additive de calcul de motivation (2), la normalisation nécessite encore un paramètre supplémentaire normalisé (*k*) exprimant le poids relatif de la pulsion et du stimulus. En plus, la normalisation du stimulus (*f(stimulus)*, (3)) nécessite un facteur d'amplification (*a*) exprimant la préférence par défaut de l'agent pour cette tâche ($a/(stimulus_{max}+a)$), c'est-à-dire exprimant la possibilité d'une activité à vide (en l'absence de stimulus). Le paramètre $stimulus_{max}$ est un paramètre endogène de l'agent : c'est la valeur de saturation du capteur correspondant (la valeur maximale de signal qu'il donne).[48]

[48] Comme nous l'avons vu dans le chapitre 3, la situation réelle est légèrement plus compliquée que cette formule laisse comprendre, parce qu'il existe *deux* capteurs, un capteur pour le composant appétitif du comportement et un capteur pour le composant consommatoire. Il suffit alors de définir

Notons, finalement, que la formule multiplicative de calcul de motivation (1) s'applique dans le cas des motivations dont les pulsions et les stimuli ont une relation OU (la motivation est 0 si la pulsion *ou* le stimulus sont 0), tandis que la formule additive (2) s'applique dans le cas des motivations dont les pulsions et les stimuli ont une relation ET (la motivation est 0 si la pulsion *et* le stimulus sont 0).

Évidemment, la pulsion de la tâche de retour à la base est le ratio charge/capacité de transport (plus un agent est chargé, plus il est motivé de rentrer à la base), tandis que la pulsion de chasse est complémentaire à la précédente. À l'initialisation, la charge de l'agent est à 0, alors sa motivation de retour est 0 et celle de chasse 1 ; l'agent abandonne donc la base et part à la chasse aux échantillons de minerai. La tâche de retour à la base est du type (2), puisque l'agent doit être à la base *et* avoir une charge 0 pour que la motivation de retour soit 0, et ne démontre pas d'activité à vide ($a=0$). De son côté, la tâche de chasse est du type (1), puisque l'agent a besoin d'une pulsion 0 ou d'un stimulus 0 pour que la motivation de tâche soit 0, sinon il va recourir à une activité à vide (d'où $a>0$). Le réglage des paramètres a et k a été fait de manière que l'agent ne rentre pas à la base prématurément, c'est-à-dire quand il n'est que peu chargé. Le stimulus de retour à la base est le signal d'orientation émis par la base, et son intensité à un point donné dans l'espace est la longueur de la grille moins la distance de perception ; ceci entraîne comme conséquence que, plus un agent est près de la base, plus il sera prêt à y rentrer pour déposer sa charge, sinon, s'il se trouve loin d'elle, il devra être pratiquement plein pour décider de rentrer. Le stimulus de chasse repose sur la perception de la densité des sources de minerai à une distance, dans un rayon maximal de perception qui a été fixé à 5 cases.

Retour à la base :
$besoin_{retour}$ = *charge/capacité,* *(2) k=0.9, a=0*
stimulus = signal/rayon (max=1)

Chasse :
$besoin_{chasse} = 1 - besoin_{retour}$, *(1) a=10*
stimulus = min(densité/distance, 10) *(max=10)*

Nous pouvons remarquer que le paramètre a exprime à la fois l'activité de l'agent à vide et la persistance de la tâche (sa dominance au-delà du point où elle est absolument prioritaire). La persistance est aussi renforcée par un facteur de bruit quant à la perception des densités des sources de minerai (avec une probabilité de 5%, la densité perçue à une position est incrémentée). Cette persistance n'est pas prescrite dans l'architecture de l'agent, elle est donc émergente ; (Beer & Chiel 1990) ont montré sur leur architecture neuronale une persistance tout aussi émergente, mais dissipative au cours du temps. La notion d'une persistance dissipative est discutée dans l'application du chapitre 6 ; ici, une dissipation n'est pas nécessaire pour l'arbitrage entre tâches, puisque, contrairement à l'architecture de Beer et Chiel, les besoins sont explicitement représentés dans l'architecture, c'est-à-dire il existe un système motivationnel.

Une dernière observation est que ***les besoins des deux tâches sont couplés de façon que la satisfaction de l'un fasse automatiquement monter l'autre. Ainsi, en l'absence d'autres mécanismes l'activité de l'agent et ses voyages aller-retour à la base vont se maintenir perpétuellement, c'est-à-dire l'état de satisfaction de l'agent***

---

comme signal de sortie du capteur consommatoire (qui est par définition binaire) la valeur de saturation du capteur appétitif.

est un état non atteignable. Inversement, si l'on veut maintenir l'activité de l'agent pour toujours, il suffit de coupler ses besoins de cette manière.[49]

La figure 4.1 montre un monde en cours d'exploration par un agent.

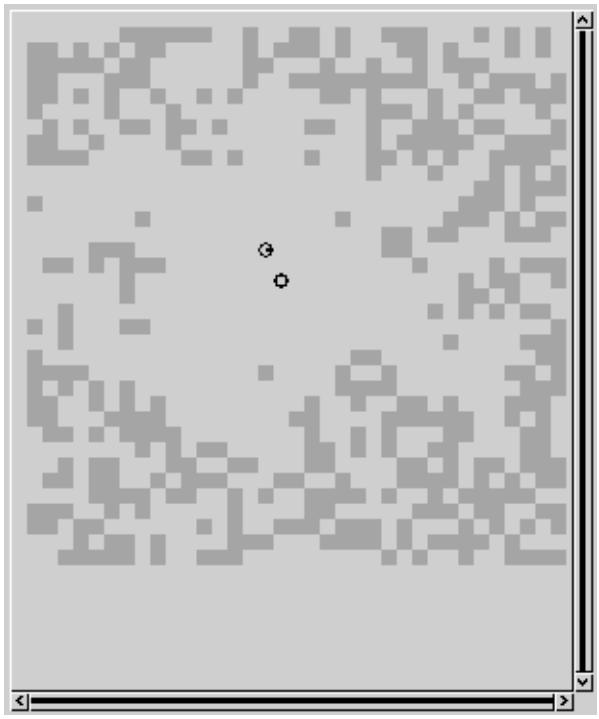

Figure 4.1 Un monde 35x35 en cours d'exploration. L'agent ramasse d'abord les échantillons qui sont près de la base (ce qui paraît normal, puisqu'il tombe sur eux en premier).

4.3 Niveau représentationnel : Satisfaction récursive et adaptation

Nous abordons maintenant la deuxième question : "Comment les agents savent-ils qu'ils ont balayé tout l'espace pour rentrer définitivement à la base ?". Ils ont besoin d'un moyen de détection du degré de complétion de la tâche ou d'un critère de terminaison (balayage complété). Les motivations de l'agent doivent donc dépendre récursivement de ce critère. Lorsque sa valeur tombe à 0, les deux motivations, celle de retour à la base et celle de chasse, seront 0 et l'agent retournera définitivement à la base.[50] Rappelons-nous que, selon ce schéma, les deux composants comportementaux, le système de ramassage et le système "cognitif", sont indépendants et fonctionnent en parallèle, donc il peuvent être étudiés et évalués séparément.

> ***Motivation récursive :***
> *motivation = besoin*f(stimulus)*term*
> *ou, (k*besoin + (1-k)*f(stimulus))*term*
> *term = critère de terminaison*

[49] Steels (1994*c*) a décrit une tâche d'élimination de parasites de la même manière — à la seule différence que le couplage se situait à l'extérieur de l'agent, il ne concernait pas les pulsions internes mais les stimuli externes, donc cet agent me paraît plus manipulable (cf. l'agent du chapitre 6).

[50] En réalité, dans le cas du balayage, ce n'est que la motivation de chasse qui est définie ainsi récursivement. Celle de retour à la base reste additive et indépendante du critère de terminaison, pour forcer l'agent à rentrer à la base depuis sa position courante lors de la détection de la terminaison.

Le seul paramètre de la tâche qui peut être utile pour le développement d'un critère de terminaison est la densité des sources dans le monde $p_m(t)$. Si l'agent connaissait d'avance sa valeur initiale $p_m(0)$, on pourrait définir comme critère de terminaison une formule du type *{$p_m(0)$ * sqr(r) échantillons ont été ramassés}* (où r est la longueur du côté du carré, ici 25). Cependant, ce critère n'est pas sûr, parce que, si un échantillon n'est pas détecté, l'agent ne terminera jamais (mais on pourrait laisser tomber un ou deux échantillons qu'on n'a pas trouvés). Un deuxième critère plus hybride qu'on peut imaginer est du type *{$p_m(0)$*n*sqr(r) échantillons ont été ramassés, ou $p_m(0)$*m*sqr(r) unités de temps sont passées}*. La première partie du nouveau critère de terminaison est celle du critère précédent avec un facteur supplémentaire de tolérance *n*, tandis que la deuxième partie constitue un critère de timeout. La durée de ce timeout dépend bien évidemment de la taille du monde balayé, d'où le facteur *m* qui doit être suffisamment supérieur à l'unité pour permettre plusieurs aller-retour à la base ainsi qu'un degré de comportement erratique et exploratoire. Le premier problème d'un tel critère est qu'on ne sait pas d'avance la valeur de $p_m(0)$. Le deuxième problème concerne le réglage des paramètres *m* et *n*. La solution très simple aux deux problèmes est d'estimer continûment la valeur de $p_m(t)$ et, étant donné qu'elle tombe à 0 comme effet de bord de l'activité de l'agent, de prendre comme critère de terminaison $p_m(t)=0$. L'estimation de la valeur de $p_m(t)$ nécessite alors une variable représentationnelle locale à l'agent ($p_a(t)$) et peut se faire par l'intermédiaire d'une formule simple d'adaptation proportionnelle :

Variable représentationnelle : $p_a(t)$

Adaptation proportionnelle :
fenêtre d'observation w, taux r
$p_a(t) = p_a(t\text{-}w) + diff * r$
$diff = p_{calc} - p_a(t\text{-}w)$
p_{calc} = *nombre des échantillons ramassés / nombre des pas effectués (pendant la fenêtre de l'adaptation)*

Critère de terminaison :
$p_a(t) < e_p$
où e_p *un petit seuil (ici,* $e_p=0.001$*)*

Le p_{calc} exprime l'estimation de l'agent lors de sa fenêtre d'observation et la loi proportionnelle assure que la mise-à-jour de l'estimation de l'agent ne se fait pas trop rapidement. Ce système de représentation et d'adaptation présente l'avantage de robustesse face aux perturbations/manipulations du type réinitialisation de la variable $p_m(t)$ au cours du balayage, ce qui n'est pas le cas des autres critères de terminaison définis auparavant — cela est dû au fait que l'estimation est continue et ne repose pas sur des variables ad hoc de timeout ou autres. Dans la figure 4.2 est illustrée la co-évolution des deux variables $p_m(t)$ et $p_a(t)$ dans le temps. Comme on peut le voir sur la figure, ***la variable représentationnelle permet à l'agent de résoudre son problème de terminaison dans tous les cas sans jamais prendre la valeur réelle qu'elle représente*** (sauf un point de croisement). Les deux variables tombent progressivement à 0 sans jamais prendre la même valeur — on pourrait dire que celle de $p_a(t)$ "suit" celle de $p_m(t)$. En effet, la montée rapide de $p_a(t)$ au début du balayage est la conséquence de l'utilisation du capteur de détection d'échantillons à distance qui fait orienter l'agent vers les sources de minerai en minimisant son comportement erratique de façon que la plupart des places visitées soient des places contenant des échantillons (pour la même raison le degré de couverture du champ de balayage à la

fin du balayage se situe typiquement entre 45% et 75%). La valeur de $p_a(t)$ baisse ensuite puisque celle de $p_m(t)$ baisse comme effet de bord de l'activité de l'agent qui trouve de moins en moins des échantillons.

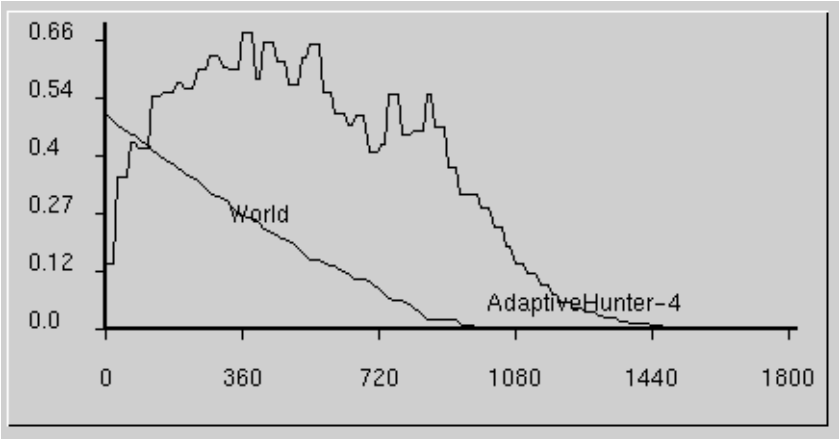

Figure 4.2 Co-évolution agent-monde : agent rentré à t=1787 ($p_m(0)$=0.5, $p_a(0)$=0.15). La courbe qui baisse de façon linéaire est celle de $p_m(t)$. La courbe cloche est celle de $p_a(t)$.

Nous pouvons également remarquer que ce couplage entre agent et monde (qui se manifeste comme un couplage entre $p_a(t)$ et $p_m(t)$) avec la définition d'une motivation et d'un état récursif de satisfaction, peut conduire l'agent à ne jamais pouvoir arriver à son état de satisfaction si le monde est continûment perturbé (nous pouvons maintenir l'activité de l'agent infiniment, si nous réinitialisons systématiquement le monde chaque fois qu'il est presque vide).

Pour revenir à l'exemple utilitariste du robot-ramasseur des sacs noirs dans les couloirs du métro, nous pouvons imaginer un mécanisme supplémentaire qui réinitialiserait la valeur de $p_a(t)$ à des instants donnés (par exemple tous les matins à 3 heures) pour forcer l'agent à partir balayer. Un tel mécanisme donnerait alors naissance à un phénomène périodique qui pourrait être perçu comme une "petite mort" du robot toutes les nuits.

4.4 Adaptation endogène et exogène

Un deuxième mécanisme d'adaptation a été également étudié. Ce mécanisme est plus "exogène" que le précédent, c'est-à-dire il dépend d'une mesure perçue, plutôt que d'une mesure d'activité calculée à son intérieur :

Adaptation exogène :
p_{calc} = *moyenne perçue des échantillons / nombre de pas effectués (pendant la fenêtre de l'adaptation)*

Les résultats comparatifs entre ce mécanisme et celui de l'adaptation endogène sont donnés dans la figure 4.3. Le mécanisme d'adaptation endogène (par activité) donne des résultats plus idiosyncratiques, tandis que le mécanisme d'adaptation exogène (par perception) rend l'agent plus "manipulable", puisque $p_a(t)$ est beaucoup plus près de $p_m(t)$. De plus, le critère d'adaptation exogène rend l'agent moins robuste aux perturbations à court terme : nous avons pu maintenir l'activité de l'agent en lui donnant de faux stimuli réguliers (des illusions), c'est-à-dire en plaçant des

échantillons dans des endroits qu'il pouvait percevoir et en les enlevant lorsqu'il était arrivé à côté d'eux.[51]

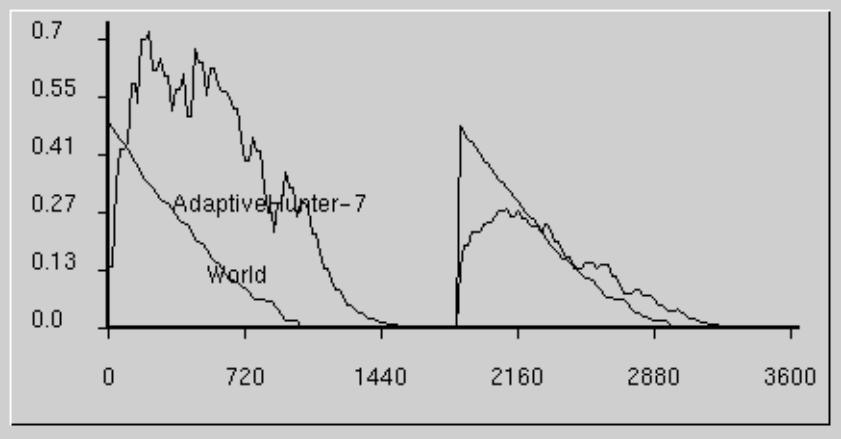

Figure 4.3 Adaptation endogène (par activité) versus adaptation exogène (par perception). $p_m(0)$=0.5, $p_a(0)$=0.15, (moments de terminaison des deux expériences : t_1=1845, t_2=3546) durées des deux expériences $dt_1=t_1-0=1845$, $dt_2=t_2-t_1=1701$, w=30, r=0.2. Dans le deuxième cas, l'agent se montre beaucoup plus manipulable. La deuxième partie de la courbe est prise après réinitialisation du système avec le même paramétrage qu'avant mais avec le deuxième mécanisme d'adaptation. Les petits plateaux de $p_m(t)$ correspondent aux périodes de temps pendant lesquelles l'agent fait des tours dans une région vide (statistiquement, il est assuré d'en sortir).

4.5 Couplage opérationnel : Méta-adaptation et autorégulation

Nous avons ensuite voulu étudier la relation — s'il en existe une — entre les paramètres *w* et *r* du système d'adaptation et la valeur $p_m(0)$. Le système a été alors simulé pour des différentes valeurs de *w* et *r* et dans des différentes densités initiales de monde. Les résultats de ces simulations pour trois ensembles des paramètres d'adaptation (adaptation rapide, moyenne ou lente) sont donnés dans les figures 4.4a, 4.4b et 4.4c.

L'adaptation rapide est plus opérationnelle que l'adaptation moyenne qui est à son tour plus opérationnelle que l'adaptation lente (toujours selon le critère du paragraphe 4.1). Cependant, plus l'adaptation est rapide, plus elle démontre des fluctuations, et plus l'adaptation est lente, plus elle démontre des retards. Qui plus est, le même paramétrage donne des résultats différents dans les différentes densités de monde : la différence des résultats se voit dans la forme des courbes. Plus particulièrement, la réponse de l'agent aux différentes perturbations (la forme de la courbe d'évolution de $p_a(t)$) diffère selon le paramètre libre de la tâche ($p_m(0)$) : pour le même paramétrage d'adaptation, l'agent finit plus ou moins vite sa mission selon la valeur de $p_m(0)$, c'est-à-dire l'intervalle entre le moment du ramassage du dernier échantillon et le retour définitif de l'agent à la base est d'une durée très variable. Il semble donc que, pour assurer l'opérationalité de l'agent dans les différents mondes, il faut trouver un moyen de combiner les avantages de l'adaptation rapide en termes d'opérationalité avec les avantages de l'adaptation lente en termes de régularité de courbe.

[51] La plupart des tortures de la mythologie grecque ancienne sont centrées autour de ce sujet de manipulation, où la personne qui subit la torture est éternellement manipulée sans pouvoir modifier son destin. Une des pires tortures est celle de Tantale, condamné pour avoir défié la puissance des dieux à avoir éternellement faim et voir les fruits délicieux arriver devant sa bouche pour y disparaître.

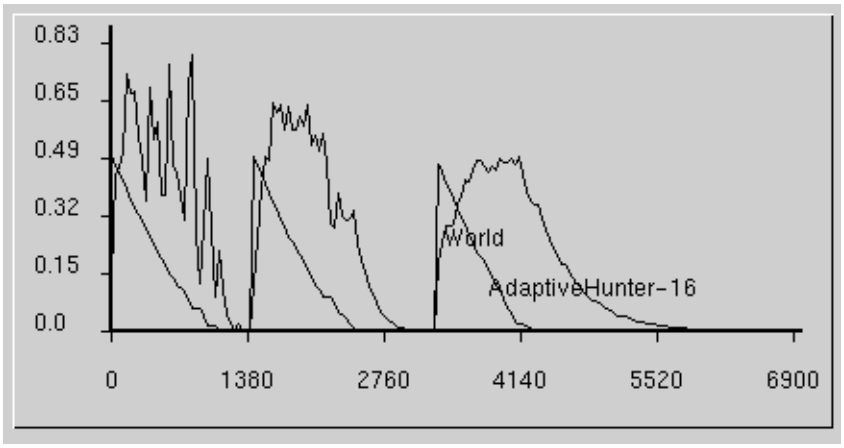

Figure 4.4a Performances de l'agent pour différents paramétrages dans une moyenne densité initiale de monde, $p_m(0)$=0.5 ($p_a(0)$=0.15), t_1=1437, t_2=3278, t_3=6821.
Première partie (adaptation rapide) : w=15, r=0.3, dt_1=1437.
Deuxième partie (adaptation moyenne) : w=30, r=0.2, dt_2=1841.
Troisième partie (adaptation lente) : w=45, r=0.1, dt_3=3543.

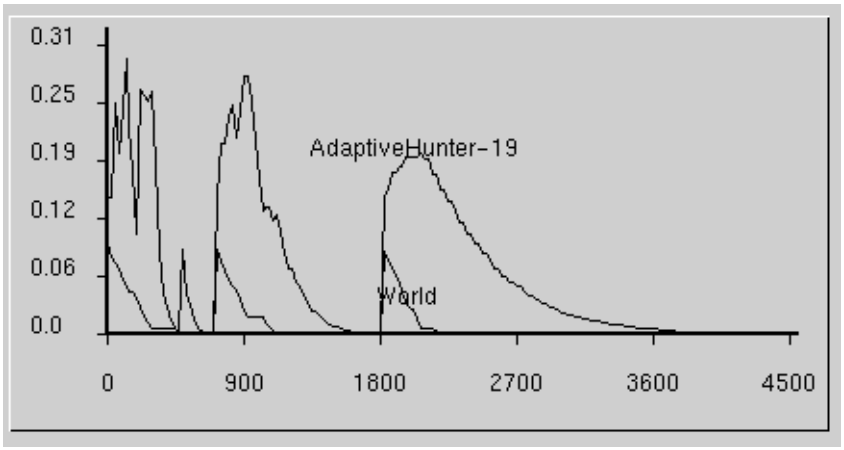

Figure 4.4b Même expérience que celle de la figure 4.4a pour une faible densité initiale de monde, $p_m(0)$=0.1 ($p_a(0)$=0.15). dt_1=711, dt_2=1097, dt_3=2599 (t_1=711, t_2=1808, t_3=4407)

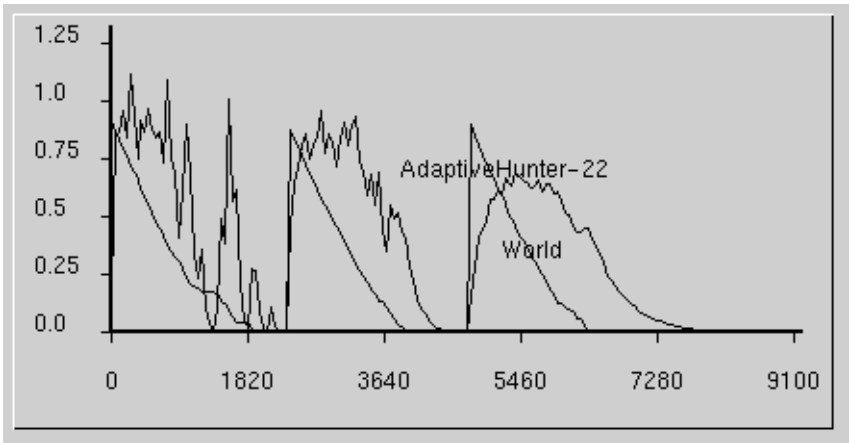

Figure 4.4c Même expérience que celle de la figure 4.4a pour une grande densité initiale de monde, $p_m(0)$=0.9 ($p_a(0)$=0.15). dt_1=2353, dt_2=2425, dt_3=4243 (t_1=2353, t_2=4778, t_3=9021).

Plus précisément, il faut une adaptation rapide à la fin (pour terminer rapidement), mais lente lors du ramassage (pour éviter les fluctuations). Il s'agit alors de trouver un moyen de se stabiliser au bon paramétrage *en ligne*. La solution est d'avoir un intervalle de valeurs possibles pour chacun des deux paramètres du système d'adaptation et un critère supplémentaire de modification (d'autorégulation entre les limites) de ses paramètres en ligne. Autrement dit, ***il faut un système de méta-adaptation***.

La méta-adaptation doit affecter les paramètres w et r de façon que l'adaptation devienne plus rapide quand le p_{calc} est suffisamment près de $p_a(t)$ et plus lente quand il est plus loin. Cette loi de méta-adaptation signifie que le monde paraît plus fiable quand il ne diffère pas trop de l'idée que l'agent en a, sinon il est pris moins au sérieux. Quelques expérimentations avec la loi inverse (adaptation plus rapide quand le monde est très différent) ont montré que cette loi serait contre-intuitive et non opérationnelle puisqu'elle rendrait l'agent plus manipulable dans des mondes capricieux.

Méta-adaptation :
Si $|diff|$ $(= |p_{calc}-p_a(t-w)|) \leq f_p$,
alors adaptation plus rapide
$r \Rightarrow r_{max}$, $w \Rightarrow w_{min}$
sinon adaptation plus lente
$r \Rightarrow r_{min}$, $w \Rightarrow w_{max}$
$r = r + r_r * (r_{max}-r)$, $w = w + r_w * (w_{min}-w)$

Dans la figure 4.5 sont donnés les résultats de l'application du système de méta-adaptation pour les trois densités du monde ; comme on peut facilement voir sur la figure, la réponse de l'agent (la forme de la courbe) est la même pour les trois densités exemplaires, autrement dit le résidu de la durée de la mission après le ramassage du dernier échantillon est approximativement le même dans les trois cas.

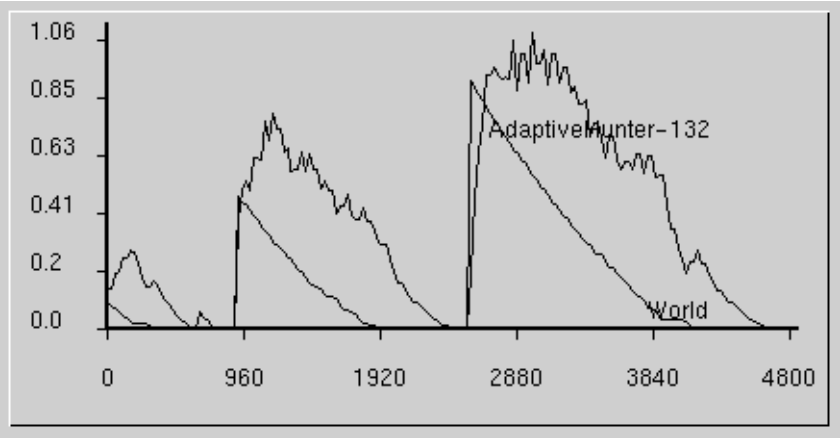

Figure 4.5 Performances de l'agent avec un système de méta-adaptation pour les trois densités initiales de monde, faible ($p_m(0)$=0.1), moyenne ($p_m(0)$=0.1) et rapide ($p_m(0)$=0.9) ($p_a(0)$=0.15). dt_1=897, dt_2=1663, dt_3=2211 (t_1=897, t_2=2560, t_3=4771). (f_p=0.1, w_{min}=15, w_{max}=40, r_{min}=0.15, r_{max}=0.3, r_r=r_w=0.2)

Les courbes de l'évolution de la fenêtre et du taux d'adaptation pour la simulation de la figure 4.5 sont données dans les figures 4.6 et 4.7. On constate que l'agent devient plus adaptatif (son adaptation est plus rapide) vers la fin du balayage : cela est dû à la baisse considérable de $p_a(t)$ qui se rapproche de 0, et donc le critère de méta-adaptation se satisfait et l'adaptation devient de plus en plus rapide jusqu'à son

maximum. Pour la même raison, l'agent reste peu adaptatif lors de son activité de balayage/ramassage, ce qui lui permet de mieux suivre $p_m(t)$ et de faire paraître la courbe de $p_a(t)$ assez régulière, sans les fluctuations ou les retards importants des figures 4.4a, 4.4b et 4.4c.

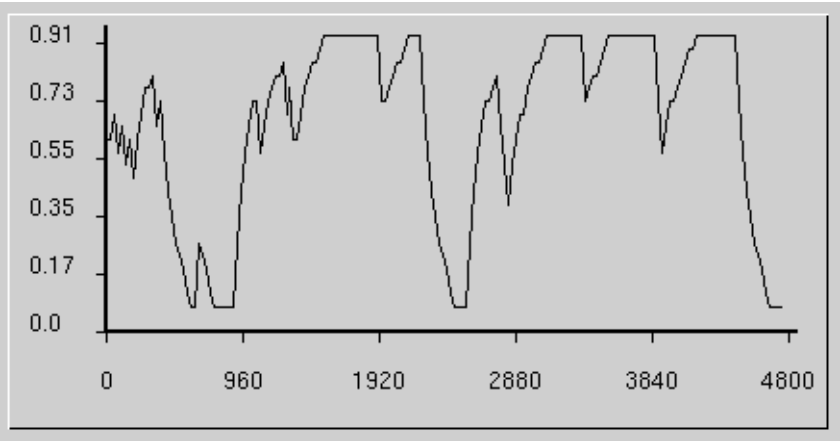

Figure 4.6 L'évolution de la fenêtre d'adaptation w de l'agent lors de l'expérience précédente. Sa valeur est normalisée entre 0 et 1, où 0 correspond à la valeur minimale (ici w_l=15) et 1 à sa valeur maximale (ici w_u=40).

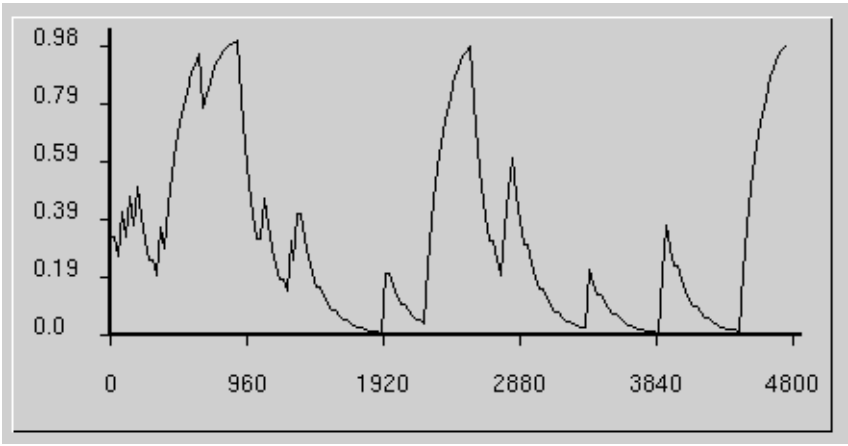

Figure 4.7 L'évolution du taux d'adaptation r de l'agent lors de l'expérience précédente. Sa valeur est normalisée entre 0 et 1, où 0 correspond à la valeur minimale (ici r_l=0.15) et 1 à sa valeur maximale (ici r_u=0.3).

Dans la figure 4.8 sont donnés les résultats de l'application de la loi de méta-adaptation pour des conditions initiales ($p_a(0)$) variées. La figure montre que *les conditions initiales ne jouent aucun rôle quant à l'opérationalité du système*. En effet, un $p_m(0)$ varié va se manifester comme un $p_a(t_0)$ varié, où t_0 est le moment du ramassage du dernier échantillon ; l'étude de la diversité de $p_m(0)$ est ainsi équivalente à une étude de la diversité de $p_a(0)$ dans un monde vide ($p_m(0)$=0). Plus généralement, l'étude du système peut se faire dans une condition de limite, telle que $p_m(0)$=0. L'individualité de l'agent, qui se manifeste dans ses réponses aux diverses perturbations, est justement sa réponse dans une condition de limite (autrement dit, pour voir la différence entre deux agents ayant des paramétrages différents d'adaptation, il suffit de les étudier dans ce cas dégénéré plutôt que de les étudier exhaustivement dans tous les cas possibles de perturbation).

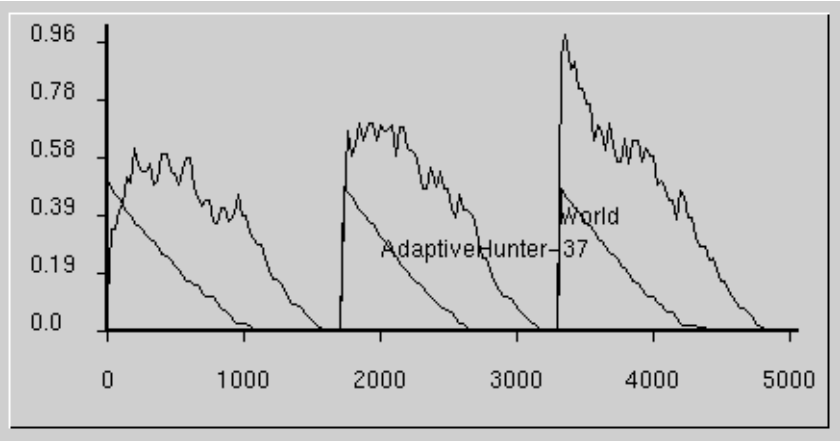

Figure 4.8 Performances de l'agent pour des conditions initiales ($p_a(0)$) variées. $p_m(0)=0.5$. Première partie : $p_a(0)$=0.1, dt_1=1723 (t_1=1723). Deuxième partie : $p_a(1723)$=0.5, dt_2=1585 (t_2=3308). Troisième partie : $p_a(3308)$=0.9, dt_3=1669 (t_3=4977).

D'autres expériences ont été menées pour explorer l'importance relative du taux d'adaptation et de la fenêtre d'adaptation : le système a été simulé dans le cas où la méta-adaptation agit seulement sur le taux ou seulement sur la fenêtre. La première alternative conduisait à un système beaucoup plus opérationnel que la deuxième et seulement un peu moins opérationnel que le cas par défaut.

La méta-adaptation a été ensuite testée dans le cas de l'adaptation exogène pour le premier niveau : les résultats étaient analogues à ceux obtenus dans le cas de l'adaptation endogène (meilleure opérationalité et indépendance de la densité initiale du monde), mais les désavantages en termes de manipulabilité sont restés intactes.

Finalement, dans la figure 4.9 sont donnés les résultats comparatifs d'une dynamique propre (auto-catalytique) de meta-adaptation et d'une dynamique dépendante de la perturbation :

Dynamique dépendante de méta-adaptation :

$$r = r + r_r * |diff| * (r_{max} - r)$$
$$w = w + r_w * |diff| * (w_{min} - w)$$

On voit que ***les dynamiques dépendantes sont sous-optimales par rapport aux dynamiques propres*** (sur la figure l'agent devient trop idiosyncratique vers la fin de sa tâche et par conséquent son opérationalité est détruite).

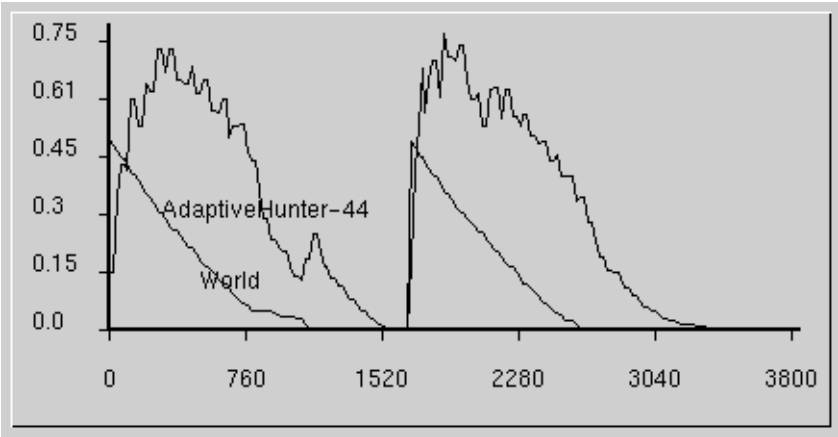

Figure 4.9 Dynamiques d'adaptation propres (première courbe) versus dynamiques dépendantes (deuxième courbe). $p_m(0)$=0.5, dt_1=1669, dt_2=2049 (t_1=1669, t_2=3718). La dynamique dépendante est sous-optimale (l'agent devient trop "idiosyncratique" à la fin)

4.6 Peut-on monter des niveaux méta ?

Nous avons ensuite étudié le cas des mondes de taille plus grande (de 30x30 à 70x70) en espérant trouver un mécanisme générique d'autorégulation/adaptation qui rendrait l'agent opérationnel pour toute taille de monde dans ces limites (le mécanisme par défaut donne des courbes frisées (cf. figure 4.10) quand il s'applique à des plus grands mondes). Nous avons alors considéré les alternatives suivantes :

- Autorégulation des paramètres de méta-adaptation (w_{min}, w_{max}, r_{min}, r_{max}, r_w, r_r)
- Régulation de la probabilité de maintien de direction de voyage.
- Combinaison des deux.

La deuxième alternative doit son existence à l'observation que dans des mondes plus grands, l'agent a tendance à "isoler" des régions plus grandes, c'est-à-dire à laisser des forêts entières d'échantillons non visités, alors une régulation de la probabilité de maintien de direction de voyage (composant aléatoire de la tâche de chasse) pourrait s'avérer opérationnelle. Aucune de ces alternatives n'a donné de résultats qualitativement différents de ceux du cas par défaut.

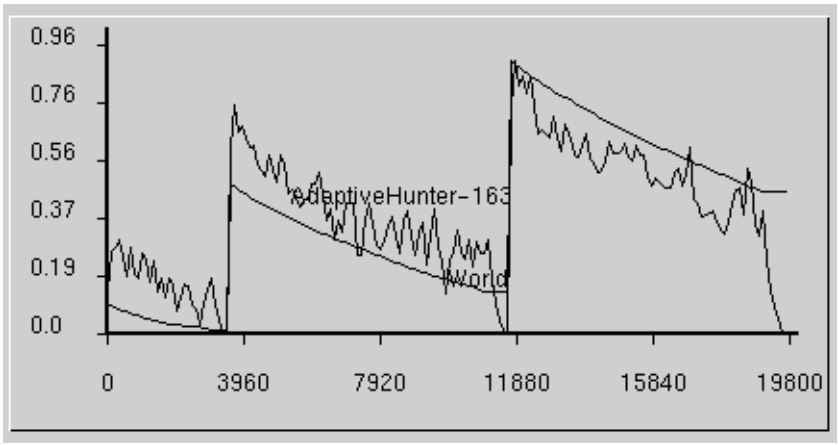

Figure 4.10 Performances de l'agent dans la même expérience que celle de la figure 4.5 mais dans un monde de taille 70x70.

En regardant de plus près, on peut constater que ***la taille du monde n'est pas un paramètre libre de la tâche de balayage*** : elle est constante depuis le début jusqu'à la

fin et connue d'avance (elle fait partie de la description de la tâche). Il suffit alors de régler d'avance les paramètres de l'agent de manière à le rendre opérationnel dans la taille du monde en question. Notons que dans un problème légèrement différent où la taille du monde serait libre, il serait sans doute possible de trouver un mécanisme approprié d'autorégulation (mais dans ce cas, le critère de terminaison de la tâche serait certainement très différent aussi). Comme ces dernières expériences ont révélé, l'ascension des niveaux méta de régulation ne se fait pas par l'intermédiaire des mécanismes récursifs (tels que la régulation des paramètres de régulation), mais plutôt par l'intermédiaire des mécanismes *parallèles* de régulation d'autres paramètres (tels que la régulation de la probabilité de maintien de direction en parallèle avec les autres régulations), car ***les mécanismes récursifs n'amplifient pas les effets de régulation***.

Dans le cas actuel du balayage, ce qui paraît non opérationnel dans les tailles de monde plus grandes, c'est le mécanisme d'adaptation de base qui fait que l'adaptation est trop rapide. Nous avons donc défini un mécanisme alternatif d'adaptation qui donne à peu près les mêmes performances que le mécanisme précédent dans de petites tailles de monde, mais qui est plus opérationnel dans de grandes tailles. Ce mécanisme "bloque" l'adaptation de la variable environnementale dans les limites [$-f_p, f_p$] (en automatique, on dirait qu'il s'agit d'une saturation du signal de contrôle) :

> ***Modèle alternatif d'adaptation :***
> $p_a(t) = p_a(t\text{-}w) + signe(diff)*min(|diff|,f_p)*r$

Le paramètre f_p devient ainsi le paramètre critique à régler pour initialiser l'agent explorateur dans des mondes de taille différente. Dans le cas du monde 25x25, la valeur de f_p doit être autour de 0.3-0.4 pour assurer une convergence aussi rapide que celle de l'ancien mode d'adaptation avec un f_p autour de 0.1. Pour des tailles de monde plus grandes (petites), le modèle alternatif d'adaptation doit être initialisé avec un f_p inférieur (supérieur) à 0.3.

4.7 Conclusion

Dans ce chapitre a été étudié le système de contrôle d'un agent explorateur-balayeur qui doit visiter et épuiser toutes les sources d'intérêt dans un espace donné et délimité. Il a été montré qu'un agent possédant deux tâches qui expriment des motivations couplées et ayant une motivation récursive et une représentation de son besoin constitue une solution au problème du balayage. L'état du monde dans lequel l'agent se trouve constitue une source de perturbation persistante pour l'agent, qui doit modifier la dynamique de son interaction avec le monde afin de mieux répondre à son besoin, et donc mieux résoudre notre problème. La modification de la dynamique d'interaction concerne les taux d'adaptation de l'agent et est homéostatique ; le couplage agent-monde est alors opérationnel.

La version opérationnelle de l'agent-balayeur a été comparée avec d'autres versions non opérationnelles : l'adaptation exogène (selon les perceptions de l'agent au lieu de son signal de rétribution interne) et la méta-adaptation dépendante (dans laquelle le taux d'adaptation est proportionnel à la perturbation). Finalement, nous avons mené une étude du problème dans des mondes plus grands pour aboutir à la conclusion que, puisque la taille du monde n'est pas un paramètre libre du problème du balayage,

aucune autorégulation d'aucun paramètre, ne pourra donner un système de contrôle capable de résoudre le problème du balayage dans un monde de taille quiconque.

Cependant, et puisque l'agent a des besoins, des représentations et des modes de couplage avec l'environnement fixes, nous pouvons dire qu'il n'est pas vraiment autonome au sens d'être "auto-déterminé", mais reste notre contrôleur pour le problème du balayage. L'individualité de ce contrôleur se voit dans sa réponse dans des conditions de limite (telles que $p_m(t)$=0), c'est-à-dire l'étude du contrôleur peut se faire en dehors d'un environnement particulier et d'un contexte particulier d'interaction.[52]

Le problème	Système de contrôle d'un agent devant "connaître" le paramètre critique du problème qu'il résout et dont la valeur dénote l'état d'avancement de la résolution
L'application	Agent explorateur solitaire
La solution	*Système motivationnel* (motivations de retour et de chasse) *Variable représentationnelle* *Estimation de la variable (adaptation)* *Autorégulation de dynamique d'interaction avec le monde* (paramètres d'adaptation)
Les enseignements	• Adaptation endogène vs. exogène • Couplage opérationnel (méta-adaptation) • Pas de sensibilité aux conditions initiales • Dynamiques propres vs. dépendantes • Monter des niveaux méta à l'aide de plusieurs paramètres libres

Tableau 4.1 Tableau récapitulatif du chapitre 4

[52] Dans ce sens, l'agent ne dépend pas de l'interaction, il agit grâce à ses motivations internes, selon ses lois internes et indépendamment de son environnement qui ne fait que le perturber temporairement ; il est donc un agent autonome.

Chapitre 5 Les explorateurs sociaux

5.1 Introduction

Après avoir étudié le problème de l'exploration/balayage pour le cas d'un seul agent explorateur, nous avons considéré le cas de plusieurs agents explorateurs. Les nouvelles questions qui se posent sont : la nature du problème change-t-elle qualitativement ? À quel degré les performances du balayage s'améliorent-elles ? Les agents ont-ils besoin d'une action sociale, d'un mécanisme supplémentaire d'adaptation/régulation ou autre pour se rendre compte de la terminaison de la tâche, c'est-à-dire de l'épuisement des sources d'intérêt ? Dans tous les cas, nous n'ajoutons pas des fonctionnalités supplémentaires du type transport d'échantillons en commun entre agents ou des traces/marqueurs de champ (Steels 1990; Drogoul & Ferber 1992). De même que pour l'explorateur solitaire, nous n'ajoutons pas de mécanismes d'apprentissage non plus, *le centre d'intérêt de l'étude est la terminaison du balayage*. Dans la figure 5.1 est donnée l'image d'un monde en cours d'exploration par 10 agents balayeurs.

> ***Balayage (4) - Point de vue social :*** *Faut-il une action sociale supplémentaire quand on passe de un à plusieurs agents balayeurs pour qu'ils se rendent compte collectivement de la terminaison de tâche ?*

Nous étudions en premier lieu les performances du balayage selon le nombre des agents et contrairement à ce qui est écrit dans la bibliographie il est conclu que ces performances ne montrent pas une amélioration auto-catalytique et donc super-linéaire mais présentent un niveau de saturation et se stabilisent pour les grandes tailles de population d'agents. Ensuite, sont étudiées les performances d'une population d'agents ayant un comportement supplémentaire de dispersion instrumentale dans l'espace et il est révélé que non seulement la dispersion n'améliore pas les performances mais les fait détériorer à cause de l'instabilité de la formation des configurations spatiales. Dans le paragraphe 5.4 nous comparons plusieurs modèles de socialité "réactive" pour arriver à la conclusion que la socialité coopérative et la socialité tit-for-tat sont les types de socialité les plus opérationnels pour le problème de l'exploration/balayage. Quelques résultats curieux obtenus avec une variante du modèle tit-for-tat sont également discutés.

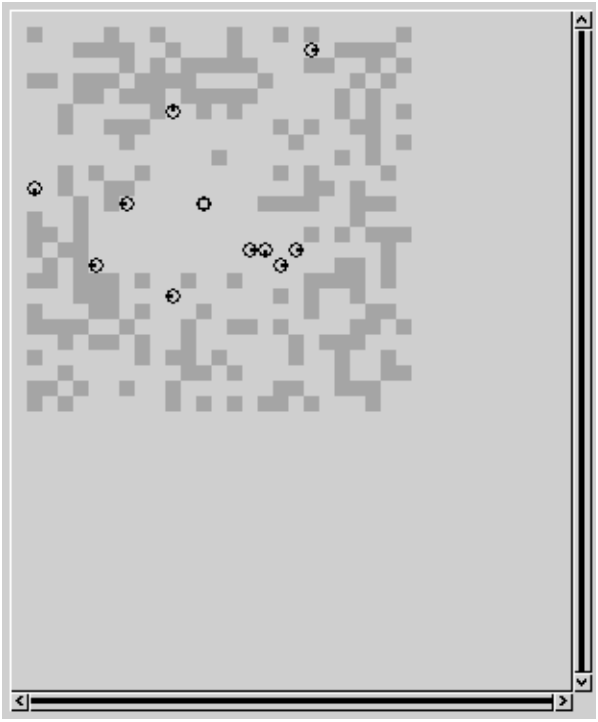

Figure 5.1 Un monde 25x25 en cours d'exploration par 10 agents ($p_m(0)$=0.5). De même que dans le cas de l'explorateur solitaire, ils consomment d'abord ce qui est près de la base.

5.2 Effets de population

Nous avons simulé le système pour 1 à 100 agents explorateurs du type du chapitre précédent sans autre fonctionnalité ou action sociale supplémentaire et nous avons enregistré les temps de complétion de tâche, c'est-à-dire les temps depuis le début jusqu'au moment du ramassage du dernier échantillon, pour montrer que les performances en l'absence de socialité explicite n'augmentent pas de façon super-linéaire comme (Mataric 1992*b*) et (Arkin et al. 1993) ont écrit.[53] En effet, le mécanisme de dispersion n'est pas auto-catalytique, par conséquent le seul gain qu'il puisse y exister c'est un gain de parallélisme, donc au mieux linéaire. Drogoul et Ferber (1992) ont montré que *même avec des mécanismes auto-catalytiques, les performances augmentent jusqu'à un niveau de saturation*.[54] Pour des populations plus grandes, il y a beaucoup d'encombrements.

Dans la figure 5.2 on voit l'évolution des performances et comment elles se stabilisent éventuellement sans montrer des améliorations super-linéaires : dans cette figure sont donnés les résultats pour 1 à 20 agents — au-delà de 20 agents les performances restent stables. Nous chercherons ensuite à accélérer l'exécution de la tâche (balayage

[53] Nous avons adopté comme critère de terminaison de la simulation le moment du ramassage du dernier échantillon plutôt que le moment du retour du dernier agent à la base pour éliminer le problème "cognitif" du chapitre précédent ("comment savent-ils qu'ils ont fini ?") ; en l'absence d'autres fonctionnalités collectives, le moment du dernier ramassage est indépendant des capacités cognitives des agents, autrement dit le ramassage des échantillons est parallèle avec le fonctionnement du système représentationnel/cognitif (cf. paragraphes 4.1 et 4.3).

[54] Le mécanisme auto-catalytique utilisé repose sur un système de marquage et un système de détection et de "vol" des agents voisins et profite de la structure non uniforme de l'environnement.

et retour des agents à la base) en introduisant une fonctionnalité supplémentaire de dispersion ou un mécanisme régulateur de coopération.

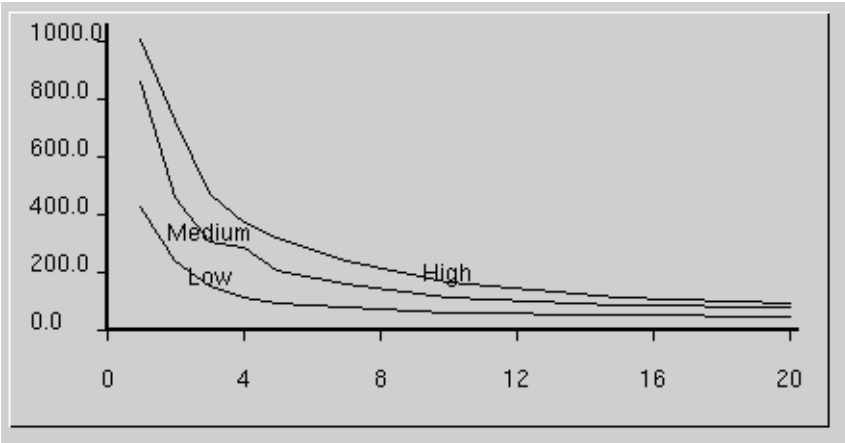

Figure 5.2 Temps de complétion de tâche selon le nombre d'agents explorateurs ($p_m(0)$=0.5) pour trois densités initiales de sources d'intérêt, faible (0.1), moyenne (0.5) et haute (0.9).

5.3 Le rôle de la dispersion

Mataric (1992*c*) a introduit dans ses robots explorateurs un comportement de dispersion dans l'espace pour accélérer la couverture du champ et par conséquent la vitesse d'exécution de la tâche.

> ***Règle de dispersion (Mataric 1992c, p. 437)***
>
> *Si un robot est perçu, tourner pour l'éviter et avancer pour une période de temps fixe.*
>
> *Si plusieurs robots sont perçus, tourner loin de leur centre de masse local et avancer pour une période de temps fixe.*

Mataric (ibid.) a séparé la dispersion des autres comportements de fourragement et formation de patrouille et a utilisé un mécanisme de propagation d'activation pour arbitrer entre eux ; elle a pourtant écrit (Mataric 1994, p. 47) que cette configuration n'est peut-être pas minimale.

Lors de nos premières expérimentations (cf. chapitres 3 et 4), nous avons cherché un comportement générique de dispersion, qui donnerait une répartition optimale des agents dans l'espace ; optimale dans le sens où, laisser ensuite les agents fourrager localement et séparément suffirait.[55] Cependant, comme le paramètre critique de la tâche est la densité des échantillons dans l'espace qui peut prendre une valeur arbitraire, il y aura éventuellement besoin de beaucoup de va-et-vient entre la base et les frontières de l'espace balayé. Alors, d'une part la répartition sera statistiquement perturbée de manière asynchrone par les agents se dirigeant vers la base ou rentrant d'elle, et d'autre part la nouvelle répartition ne devra plus être la même qu'avant. En plus, un tel comportement de dispersion n'aura de sens que dans un monde relativement petit par rapport à la taille de la population, où la dispersion pourrait assurer une répartition efficace.

[55] Nous avons également implémenté la dispersion comme une activité pendant laquelle l'agent simulé marche à grande vitesse tandis que pendant le fourragement il marche à faible vitesse (car pour fourrager, il doit consulter ces capteurs assez attentivement).

Pour éliminer le problème de l'arbitrage entre fourragement et dispersion, mais également pour permettre une répartition dynamique, nous avons introduit la dispersion comme une ***action instrumentale*** de fourragement/chasse dans le réseau cellulaire de la figure 3.6 qui devient ainsi celui de la figure 5.3.

> ***Règle de dispersion instrumentale***
>
> *S'il n'existe pas de source d'intérêt perçue :*
>
> *Si un ou plusieurs agents sont perçus, tourner loin de leur centre de masse local (c'est-à-dire, choisir comme direction de marche celle de densité minimale d'agents perçus).*

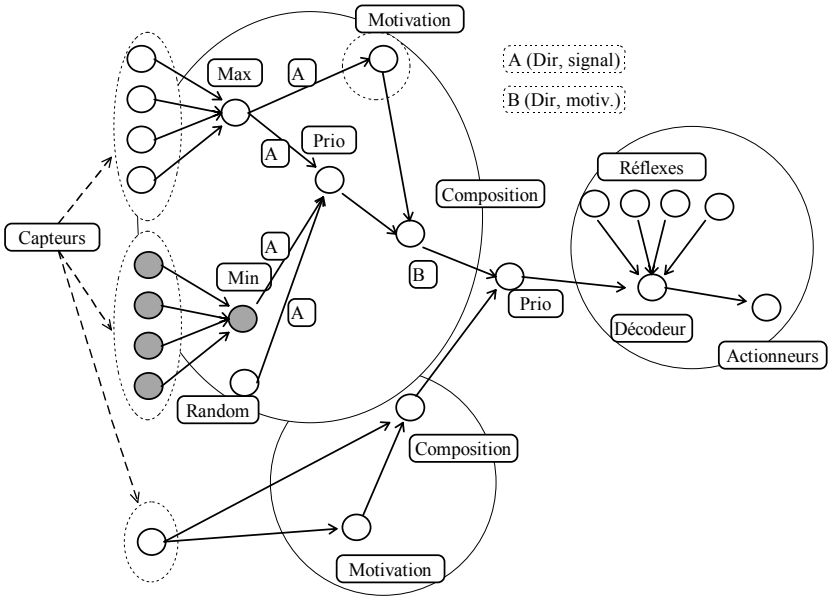

Figure 5.3 Le nouveau système cellulaire de contrôle de l'agent explorateur. La dispersion est une action instrumentale pour le fourragement/chasse (les cellules supplémentaires par rapport à la figure 3.6 composant le système de dispersion sont visualisées en gris). À remarquer que, si l'on remplace la cellule min d'une cellule max, l'agent choisira la direction de densité maximale d'agents perçus, et le programme de dispersion deviendra un programme de regroupement (flocking). La complémentarité entre certains comportements tels que l'agression et la peur a été soulignée par Braitenberg (1984).

Nous avons simulé le système avec et sans dispersion pour une gamme de tailles de population d'agents et les résultats comparatifs sont donnés dans la figure 5.4. On constate que, contrairement aux prédictions intuitives, la dispersion est sous-optimale : elle perturbe les performances des petites populations et devient plus favorable en donnant des performances du même ordre que celles sans dispersion pour une population de taille supérieure à 10. Notons que ce résultat est indépendant du rayon de dispersion, c'est-à-dire de la distance maximale de détection d'autres agents qui déclenche la dispersion : nous avons simulé le système pour un rayon de dispersion de 3 et de 7 et ses performances dans les deux cas étaient inférieures à celles sans dispersion.[56]

[56] Ces deux valeurs sont choisies la première inférieure et la deuxième supérieure au rayon de fourragement, c'est-à-dire à la distance maximale de détection d'échantillons qui déclenche le mouvement appétitif de l'agent (dans le problème du balayage, nous avons fixé ce rayon de perception d'échantillons au 5,cf. paragraphe 4.2). Comme prévu, les performances du balayage sont pires si le

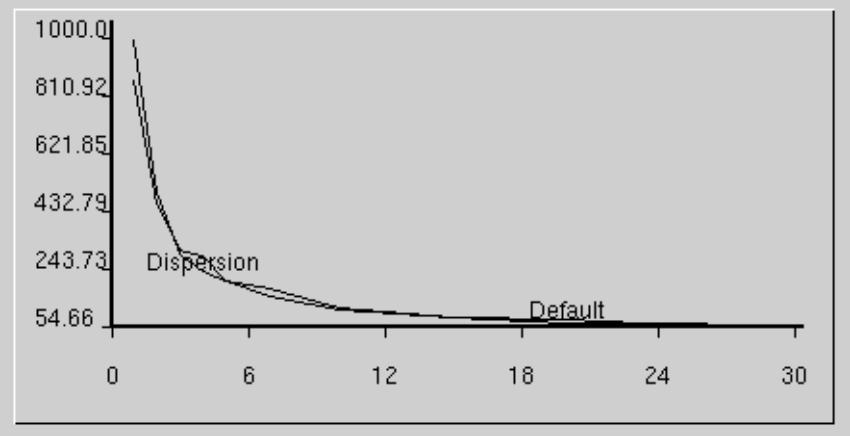

Figure 5.4 Temps de complétion de tâche sans et avec dispersion selon le nombre d'agents explorateurs ($p_m(0)$=0.5, rayon de dispersion=3).

Comme nous l'avons vu dans le paragraphe 1.3.4, une réaction à un stimulus "social" hors le système motivationnel d'un agent ne constitue pas une socialité. Alors, le comportement de dispersion que nous venons d'étudier ne constitue pas une socialité, il fait seulement partie d'un algorithme qui accélère la résolution du problème, ce qui se voit dans sa définition comme instrumental dans l'architecture de l'agent.[57] Le comportement du robot docker qui forme des chaînes de robots (Drogoul & Ferber 1992), le mécanisme de coopération modeste de Premvuti et Yuta (1990, 1993) et les architectures de navigation coopérative ou coordonnée (Kagawa 1992; Zeghal 1994) ne constituent pas une socialité non plus. Dans tous ces cas, le comportement "coopératif" en question n'est qu'un comportement supplémentaire qui accélère la réalisation d'une tâche *sans* intervenir explicitement au niveau motivationnel des agents. Pour utiliser la terminologie de Mataric (1992*b*), il s'agit d'une coexistence "informée" des agents, plutôt que d'une coexistence "intelligente" qui se manifesterait comme *un compromis différentiel entre les divers comportements*, c'est-à-dire qu'elle interviendrait au niveau motivationnel.

5.4 Modèles de socialité "réactive"

Ayant éliminé la dispersion pour manque d'opérationalité et adoptant comme point de départ que la socialité doit intervenir au niveau du système motivationnel des agents, nous avons cherché un mécanisme supplémentaire qui modifierait la motivation des agents en prenant en compte les motivations des autres agents. Pour ce faire, il est nécessaire d'ajouter des émetteurs et des détecteurs de motivation ; vu que les motivations prennent des valeurs réelles continues, des capteurs de signaux radio suffiraient pour une implémentation robotique distribuée. Cette socialité est une socialité "réactive" dans le sens où les agents ne l'ont pas "découverte" eux-mêmes,

---

rayon de dispersion est supérieur au rayon de perception que s'il est inférieur (car, dans le premier cas, si un échantillon se trouve entre deux agents, ils vont se disperser et s'en éloigner sans s'en apercevoir).

[57] Mais la vraie raison pour laquelle la dispersion ne constitue pas une socialité, c'est que la seule vraie motivation de l'agent explorateur/balayeur est de ramasser les échantillons et rentrer à la base, qui se décompose à son tour en une motivation de fourrager et une motivation de rentrer ; si ce n'est que pour accélérer ce ramassage, il n'a aucune vraie volonté de s'éloigner des autres agents.

ce n'est pas eux qui ont décidé qu'il serait mieux de partager leurs besoins, mais c'est le concepteur qui les a dotés du mécanisme nécessaire pour ce faire.

Quelle motivation les agents doivent-ils échanger ? Évidemment, il ne peut s'agir ni de la motivation de chasse/fourragement, ni de la motivation de retour à la base, mais de la super-motivation de p_a. Nous cherchons alors de nouvelles formules de calcul et d'adaptation de cette motivation du type :

Motivation contenant un composant de socialité

$p_{est} = f(p, p_{calc}, p_{soc})$

où p_{soc} est la moyenne perçue des motivations des autres agents.

Nous avons implémenté et comparé trois modèles de socialité : une socialité relative, une socialité coopérative et une socialité tit-for-tat. Le premier modèle de socialité réactive est celui de la socialité relative : la densité estimée d'échantillons est donnée par la formule $p_{est} = a*p_{soc} + (1-a)*p_{calc}$, où le p_{soc} inclut la valeur propre de l'agent ; cette formule signifie que l'estimation de la densité du monde est considérée comme étant quelque part entre celle perçue individuellement (p_{calc}) et la moyenne des croyances de tous les agents. Le reste du modèle est identique à celui de l'agent explorateur solitaire.

Socialité relative

$p_{est} = a*p_{soc} + (1-a)*p_{calc}$, $diff = |p_{est} - p|$, $0<a<1$
$p = p + r*diff$
Si $diff \leq f_p$, *adaptation plus rapide,*
sinon adaptation plus lente

Le deuxième modèle est celui d'une socialité coopérative, où chaque agent considère que la valeur actuelle de la densité du monde est la moyenne entre la densité calculée et la densité moyenne des agents (on considère que la "vérité" se trouve au milieu des deux valeurs, l'estimation personnelle et le standard social).

Socialité coopérative

$p_{est} = (p_{soc} + p_{calc})/2$ $(a=0.5)$, $diff = |p_{est} - p|$,
$p = p_{est}$
Si $diff \leq f_p$, *adaptation plus rapide,*
sinon adaptation plus lente

Finalement, la socialité tit-for-tat est un peu plus compliquée que les deux autres et elle est analogue au modèle alternatif du chapitre précédent : elle est comme la socialité coopérative en ce qui concerne l'estimation de la densité du monde, mais il y a en plus un seuil utilisée par le mécanisme tit-for-tat pour détecter la coopération. L'adaptation est alors un effet de bord de la coopération ou de la défection.

Dans une première version non opérationnelle de la socialité tit-for-tat, l'agent ne met à jour son estimation que lors de détection de coopération. Dans la deuxième version au contraire, lors de la coopération l'agent adopte comme nouvelle valeur de son estimation propre la valeur estimée, sinon il s'adapte vers cette valeur selon la même loi proportionnelle qu'avant, c'est-à-dire la formule de mise-à-jour de p_a est *différente* pour la coopération et la défection, mais l'agent s'adapte *continûment*.

Socialité tit-for-tat (version 1)
$p_{est} = (p_{soc} + p_{calc})/2$ *(a=0.5),* $diff = |p_{est} - p|$
Si $diff \leq f_p$,
$p = p_{est}$ *et adaptation plus rapide,*
sinon adaptation plus lente

Socialité tit-for-tat (version 2)
$p_{est} = (p_{soc} + p_{calc})/2$ *(a=0.5),* $diff = |p_{est} - p|$
Si $diff \leq f_p$,
$p = p_{est}$ *et adaptation plus rapide ,*
sinon $p = p + f_p$ * *signe (diff)*
et adaptation plus lente

Le problème avec le premier modèle de socialité tit-for-tat a été qu'il conduisait très souvent à une stabilisation de la population à une valeur de la densité estimée telle que plus aucun agent ne modifiait sa valeur propre et l'activité de la population serait en principe maintenue pour toujours malgré l'épuisement des échantillons ; Huberman & Glance (1993) et Nowak et May (1992) ont présenté des résultats de simulations qui montrent que la diversité spatiale et l'asynchronisme dans certains systèmes distribués conduisent à des configurations globales qui sont ou bien chaotiques ou bien uniformes. Pour remédier à ce problème, nous avons effectué des extensions quant au critère de coopération, du type bruit (autour de 0.1), générosité au sens de probabilité non nulle de coopération lors de détection de défection (autour de 0.1) et autorégulation soit de l'égoïsme *a* (autour de 0.5) soit de la tolérance/seuil *T* (autour de 0.5). L'autorégulation de la tolérance ou de l'égoïsme était opérationnelle tant que le niveau méta (celui du critère de coopération) restait "coopératif", c'est-à-dire que la détection de coopération doit induire une hausse plutôt qu'une baisse de la tolérance et une baisse plutôt qu'une hausse de l'égoïsme. Dans tous les cas, la régulation de la tolérance a donné de meilleures performances que la régulation de l'égoïsme (il semble que le paramètre critique dans le tit-for-tat quantitatif est la tolérance — cf. des résultats supplémentaires dans l'annexe A) mais en aucun cas on n'a éliminé complètement le besoin d'un degré de bruit ou de générosité. La raison en est que toutes ces variantes du modèle tit-for-tat reposent sur une défection inactive (il n'y a pas d'action et donc pas d'adaptation non plus si le monde est trouvé hostile). Cependant, comme il est discuté dans le chapitre 1, l'essence de l'autonomie, et pour ce qui nous concerne ici de l'opérationalité, c'est le *changement continu* de réponse au monde, d'où la version actuelle de la socialité tit-for-tat qui définit une action pour la défection comme pour la coopération et donc qui ne présente pas le problème d'aliénation (stabilisation sociale comme décrit auparavant).

La première implémentation de tous ces modèles de socialité reposait sur des interactions localisées : les moyennes perçues des motivations des autres agents concernaient seulement les agents à l'intérieur d'une zone limitée (un disque autour de l'agent avec un rayon ou bien fixe pour tous les agents ou bien varié). Cette implémentation a donné des résultats à peine différents de ceux sans socialité, malgré l'amélioration quantitative lors des premiers cycles des simulations. En regardant de plus près, nous avons en effet constaté que, puisque la taille du monde (25x25) était relativement importante par rapport à la taille de la population, les agents ne

socialisaient vraiment que pendant les premiers cycles quand ils étaient en train de se disperser loin de la base. Dès lors, leurs trajectoires divergeaient et leurs retours à la base étaient statistiquement non synchronisés, alors pratiquement ils ne se rencontraient que très rarement.

La deuxième implémentation des modèles de socialité reposait sur des interactions globales, c'est-à-dire sur une propagation des besoins individuels dans toute la population. Nous avons également implémenté plusieurs variantes de ces trois modèles de socialité selon que l'adaptation sociale est synchrone avec l'adaptation individuelle ou parallèle (dans laquelle les deux adaptations se font de manière indépendante et avec leur rythme propre) ou probabiliste (dans laquelle l'adaptation sociale se fait statistiquement avec une certaine probabilité). Les dernières variantes ont été trouvées non opérationnelles, c'est-à-dire il n'y a pratiquement pas eu de différence de performances avec le cas d'asocialité. La seule possibilité opérationnelle est l'adaptation synchrone donnée auparavant. Le modèle de la socialité relative a été éliminé tôt dans l'expérimentation pour manque d'opérationalité : en effet, la socialité relative donne une estimation de la densité du monde qui a trop d'inertie par rapport à l'activité des agents et donc par rapport à la vraie évolution de la densité du monde (ce résultat a été vérifié dans toute la gamme de valeurs pour le paramètre a de 0 à 1 : le problème reste et doit sa présence au facteur proportionnel r*diff).

Notons que le passage du système motivationnel de l'agent explorateur solitaire (formule d'adaptation du paragraphe 4.3) à l'agent explorateur social (une des formules d'adaptation de ce paragraphe) n'est pas continu : ici le p_{est} dépend de p_{calc} *ainsi que* de p_{soc}. Le système a été simulé dans le cas où l'agent social est seul dans son monde (auquel cas $p_{soc}=p$) et il a été précisément trouvé qu'il n'est pas opérationnel (son adaptation est trop rapide et irrégulière) ; en conséquence, un agent doit pouvoir "reconnaître" s'il est seul et s'il ne l'est pas pour exécuter le programme correspondant. Nous appelons cette propriété ***le saut social***.

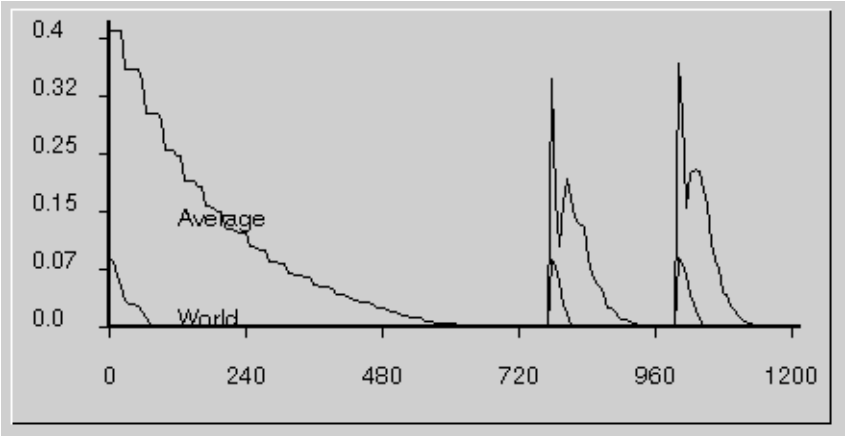

Figure 5.5a Asocialité vs. socialité coopérative vs. socialité tit-for-tat dans un monde de faible densité. 10 agents, $p_m(0)$=0.1, dt_1=774, dt_2=226, dt_3=200 (dt_1=774, t_2=1000, t_3=1200).

Dans les figures 5.5a, 5.5b et 5.5c sont donnés les résultats comparatifs d'application dans des différentes densités du monde des deux derniers modèles ainsi que du modèle par défaut qui est celui du chapitre précédent, c'est-à-dire le modèle asocial. L'évolution des courbes montre clairement l'augmentation des performances avec socialité ; la socialité tit-for-tat est légèrement supérieure à la socialité coopérative quant à ses performances.

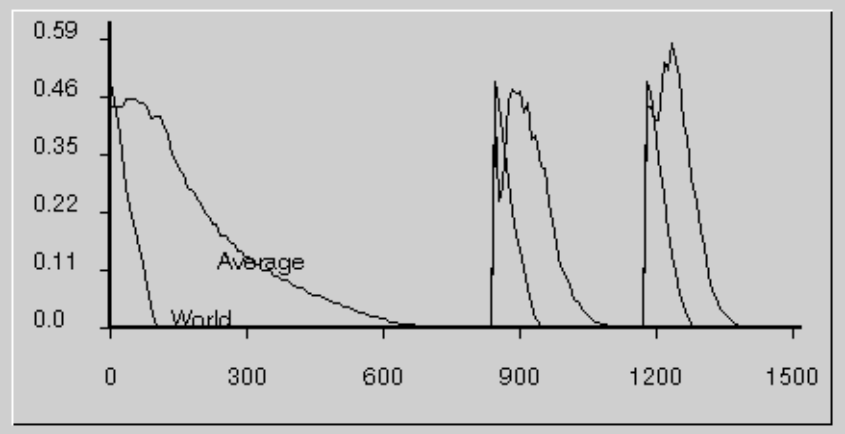

Figure 5.5b Asocialité vs. socialité coopérative vs. socialité tit-for-tat dans un monde de moyenne densité. 10 agents, $p_m(0)$=0.5, dt_1=844, dt_2=335, dt_3=287 (t_1=844, t_2=1179, t_3=1466).

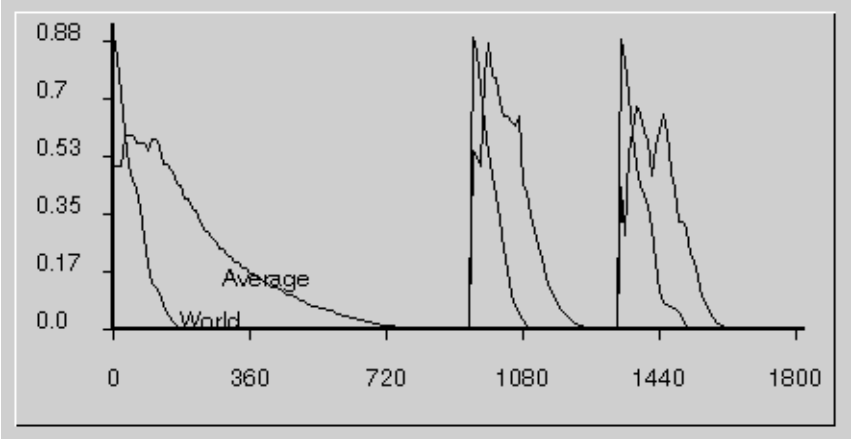

Figure 5.5c Asocialité vs. socialité coopérative vs. socialité tit-for-tat dans un monde de grande densité. 10 agents, $p_m(0)$=0.9, dt_1=946, dt_2=391, dt_3=364 (t_1=946, t_2=1337, t_3=1701).

Cette légère supériorité de la socialité tit-for-tat par rapport à la socialité coopérative est due à la variation des vitesses d'adaptation. Nous avons simulé le système avec ou sans diversité des besoins ($p_a(0)$), avec ou sans diversité des paramètres de tolérance (f_p) et avec ou sans diversité des paramètres d'adaptation. En gardant les besoins initiaux et les paramètres d'adaptation uniformes et constants dans les deux cas, mais avec une diversité des tolérances, la socialité tit-for-tat s'est avérée légèrement plus opérationnelle que la socialité coopérative. Pourquoi ? ***Il semble que le système des agents tit-for-tat génère de la diversité, puisque les agents ne font pas tous la même chose à chaque moment, et que cette diversité est responsable de l'augmentation des***

performances.[58] Cependant, cette augmentation n'est que marginale, puisque l'accélération ou le ralentissement de l'adaptation ne repose pas sur un vrai critère de coopération/défection. L'agent tit-for-tat est un agent essentiellement coopératif ; il modifie seulement sa dynamique d'adaptation, mais il n'est pas vraiment défectif (il n'a aucun intérêt à être défectif, si sa coopération ne peut jamais être vraiment détrimentale). Le besoin de défection naît dès qu'on introduit des facteurs qui peuvent rendre la coopération détrimentale, par exemple des agents trompeurs. Nous avons expérimenté avec une proportion d'agents qui "trichent" quant au besoin qu'ils communiquent aux autres : nous avons plus particulièrement défini des agents émettant toujours la même valeur constante ou une valeur "perturbée" (mutée avec une probabilité de bruit) ou une valeur totalement aléatoire. Dans tous les cas, les agents tit-for-tat étaient clairement plus opérationnels que les coopératifs. Cependant, puisque les agents trompeurs n'ont pas de volonté particulière pour tricher (c'est comme si leurs émetteurs étaient en panne), il n'existe pas de deuxième "force sociale" (la seule force présente est celle du balayage) et donc il n'y a pas de besoin de compromis et de régulation de socialité. Pour montrer le besoin de régulation de socialité, il faudrait modifier le problème social pour inclure une deuxième force sociale, par exemple en mettant deux types d'agents, dont les uns chercheraient à balayer et les autres à re-salir, ou en ajoutant un comportement social supplémentaire du type partage d'une ressource (comme dans Steels (1994*c*) et McFarland (1994)) ou achat/vente d'un objet (Ali Cherif (1993) a introduit un mécanisme d'achat/vente ad hoc sans montrer ni exploiter le besoin de régulation). Comme est formulé le problème, il n'y a pas de besoin de régulation de socialité parce qu'il n'existe pas de paramètre libre intervenant dans la socialité.

5.5 Conclusion

Le problème du balayage du chapitre précédent a été étudié dans le cas de plusieurs agents au lieu d'un seul. Premièrement a été étudié l'évolution des performances en fonction de la taille de la population et il a été montré que ces performances arrivent à un point de saturation. Ensuite a été étudié l'effet de l'introduction d'une possibilité supplémentaire de dispersion comme une action instrumentale de fourragement/chasse. Cette fois il y a eu une baisse des performances, en raison de l'instabilité des formes spatiales dynamiques. L'implémentation et la comparaison de plusieurs modèles de socialité réactive ont conduit à la conclusion que la socialité nécessaire est du type "partage global du besoin". Une première version non opérationnelle de socialité tit-for-tat a été analysée : le manque d'opérationalité semble être dû à la stabilisation de la dynamique de l'interaction agent-monde. Nous avons présenté également d'autres observations assorties, le phénomène du saut social, le rôle de la diversité et le comportement du système en présence d'agents-trompeurs. Finalement, le manque de besoin d'autorégulation a été attribué à l'absence de paramètre libre lié directement à la socialité et à la présence d'une seule force sociale dans le système multi-agents.

[58] Le rôle opérationnel de la diversité dans les populations n'a commencé d'être reconnu que très récemment (Nowak & Sigmund 1992; Lumer & Faieta 1994).

Le problème	Système de contrôle d'un agent devant accomplir sa tâche en commun avec d'autres agents
L'application	Agent explorateur social
La solution	*Solution pour agent explorateur solitaire* + *Socialité* (lors de l'estimation de la variable représentationnelle prendre en compte les estimations des autres agents)
Les enseignements	• Saturation des performances des populations • Dispersion ⇒ Baisse et instabilité des performances • Socialité coopérative vs. socialité relative vs. socialité tit-for-tat • Un phénomène de non opérationalité à cause de stabilisation des dynamiques d'interaction • Le saut social • Le rôle de la diversité • Présence d'agents trompeurs

Tableau 5.1 Tableau récapitulatif du chapitre 5

Chapitre 6 L'agent manager

6.1 Introduction

Dans ce chapitre nous étudions un deuxième problème au niveau de l'animat, inspiré de la productique. La particularité de ce problème est que, les systèmes industriels étant plus exigeants du point de vue de la connaissance a priori et des contraintes d'exécution, le système physiologique de l'agent autonome s'avère beaucoup plus complexe que celui des agents explorateurs des chapitres précédents, puisque le degré de pré-planification inhérent dans la fonctionnalité du système doit se transcrire dans la physiologie même de l'agent. En outre, cette étude est inspirée du robot Pedro (Darche 1994), ce qui a posé quelques contraintes supplémentaires lors de la formulation du problème.[59]

6.1.1 Robotique industrielle autonome et industrie cellulaire

Les robots industriels classiques ont été utilisés dans le secteur de la production des produits finaux ou intermédiaires et leur opération impliquait la manipulation des pièces et des outils, l'assemblage, le soudage, etc. (Coiffet 1986) Toutes ces opérations ont généralement lieu dans un environnement connu et structuré, par conséquent les recherches conventionnelles en productique se sont fondées sur deux hypothèses interdépendantes : *l'hypothèse de la presque parfaite connaissance*, selon laquelle tous ou presque tous les paramètres qui déterminent l'opération et les performances du système sont connus d'avance, et *l'hypothèse de la décision centralisée*, selon laquelle — et puisqu'il existe presque parfaite connaissance sur la tâche — le concepteur peut trouver d'avance une méthode complète pour contrôler le système afin d'atteindre les buts opérationnels, autrement dit prendre d'avance les décisions qui concernent l'opération du système. Jusqu'à présent, les robots ont été utilisés dans l'industrie presque exclusivement dans des tâches mécaniques et de manipulation nécessitant une haute précision, telles que l'assemblage et le soudage. Cependant, pendant la dernière décennie, les recherches en productique se sont décalées vers l'intégration des systèmes hétérogènes, la flexibilité de la production, l'adaptativité des processus industriels face aux événements imprévus, la gestion de l'incertitude et la tolérance aux pannes (Jones 1992; Ranky 1986; Rolstadås 1988; Wright & Bourne 1988). Le concept de l'industrie cellulaire (Bjorke 1979) a surgi comme le paradigme unificateur derrière les besoins diversifiés pour les systèmes de production orientés-produit plutôt que orientés-processus. Le paradigme repose sur l'allocation de la production complète d'une famille de produits à une unité de production relativement "autonome" qui se compose d'un ensemble de machines et/ou robots. Ces unités individuelles de production doivent répondre à leurs besoins

[59] Je suis redevable à Philippe Darche pour les longues discussions au sujet de cette application.

opérationnels locaux et sont soumises aux contraintes de flexibilité et d'adaptativité. La *flexibilité* implique la possibilité de réinitialiser ou reconfigurer le système sur un nouvel ensemble de besoins opérationnels, y compris des dates limites de production, des limites des coûts, un nouvel ordre des entrées etc., tandis que l'*adaptativité* correspond à la possibilité de récupérer des fautes de divers types. La flexibilité concerne alors principalement la connexion d'une unité aux autres composants de l'atelier de production, tandis que l'adaptativité se traduit par opérationalité individuelle. Baldwin (1989) a spéculé que ces systèmes cellulaires de production évoluent vers des systèmes *autonomes* de production, "dans lesquels les buts et l'intelligence du système seront distribués sur tout l'ensemble des entités (robots, outils, ressources, etc.) faisant partie du système", c'est-à-dire l'industrie sera un "écosystème" d'espèces industrielles.

L'hypothèse de la presque parfaite connaissance a ainsi progressivement reculé puisqu'une a priori connaissance sur le processus global n'est pas disponible, ou, si elle l'est, son utilisation pour la prise d'une décision parfaite (que ce soit un plan ou un séquencement) est souvent d'une complexité énorme. On a dû donc remplacer le principe de l'optimisation par des techniques plus qualitatives et plus heuristiques capables de manipuler une connaissance moins formelle ; ces techniques sont dérivées dans leur majorité de l'intelligence artificielle et leur intégration dans les systèmes industriels (Almgren 1989) (Kusiak 1990) a été synchrone avec l'émergence du nouveau domaine du contrôle intelligent qui combinait les techniques de la commande traditionnelle avec celles de l'intelligence artificielle (Meystel 1989). D'autre côté, des doutes sur la nature centrale des décisions n'ont vu le jour que très récemment et quelques systèmes de production consistant de plusieurs tâches, robots et ressources ont été modélisés en termes de composants (ou agents) coopératifs ou compétitifs fonctionnant en parallèle (par exemple, Fischer 1993).

Dans ces systèmes de production moderne, des diverses décisions doivent être prises pratiquement à chaque point pendant l'opération du système, puisque son état global exact ne peut pas être connu d'avance et une planification ou ordonnancement à long terme devient rapidement obsolète. La présence de règles locales et d'heuristiques réactives paraît plus indispensable qu'un pré-traitement analytique compliqué. L'introduction de robots autonomes à ces points de décision ou re-décision, c'est-à-dire à ces points où un "traitement" local intelligent, ou autonome, devient nécessaire, semble bénéfique. Les avantages d'une telle décomposition du processus en un nombre d'agents en interaction sont la facilité de conception (puisque la synthèse et l'intégration multiplient la complexité) et la possibilité de reconfiguration ou de dimensionnement. À côté des techniques heuristiques et qualitatives, une deuxième approche quant au traitement de l'information incomplète est la surveillance qui demande une re-décision (Desrochers & Silva 1994) : re-planification, re-ordonnancement, décisions floues, etc.; cette deuxième approche est aussi basée sur l'utilisation de techniques mathématiques analytiques d'une complexité énorme. La robotique industrielle moderne, qui doit se conformer aux contraintes de flexibilité et d'adaptativité (Coiffet 1993), est ainsi plus près de la robotique de service (Schraft et al. 1993), qui concerne par défaut les applications telles que nettoyage, transport, sécurité, etc. dans lesquelles l'espace de travail est un monde peu ou partiellement connu et le schéma classique de contrôle "sense-model-plan-act" n'est plus efficace, surtout pour des raisons de retard de réponse aux événements imprévus. Il paraît que tous ces problèmes de description analytique précise nécessitent un degré "d'intelligence réactive".

Les robots comportementaux, réactifs ou autonomes, sont précisément ceux plus aptes à être utilisés dans des environnements inconnus ou partiellement connus ; ils paraissent alors comme les candidats idéaux pour un environnement industriel partiellement connu et partiellement structuré. De plus, le concepteur du système peut exploiter la connaissance implicite qui existe et qui concerne le fonctionnement du système : les hypothèses opérationnelles de même que les contraintes. L'idée est que l'utilisation de robots comportementaux dans l'industrie peut démontrer à la fois une robustesse et une flexibilité, ainsi qu'une simplicité conceptuelle et pragmatique, tout en offrant un coût relativement bas. L'utilisation de robots comportementaux dans des applications similaires a déjà été proposée (cf. par exemple (Doty & Van Aken 1993) et (Arkin & Murphy 1990)) et plusieurs études en simulation ont été menées (cf. notamment l'étude dynamique de (Levi & Will 1994)).

6.1.2 L'application

Soit un atelier de production qui consiste d'un nombre d'unités indépendantes de production (robots, machines-CNC ou autres ateliers), chacune étant spécialisée à la production d'un ou de plusieurs types de produits (fig. 6.1). Une unité a besoin d'un certain nombre d'outils pour fonctionner ; nous supposons que ces outils sont relativement chers et doivent être partagés entre les différentes unités, de façon qu'un mécanisme de compétition ou allocation soit nécessaire. Nous supposons aussi que l'ordre exact de l'arrivée des demandes n'est pas connu d'avance. Dans ce cas, la solution classique d'une machine à outils avec un plan prédéterminé, ou presque déterminé, ne donne pas toujours des résultats acceptables. En effet, cette "machine" doit être plus qu'une simple machine, elle doit prendre une véritable décision quant à la desserte des diverses demandes de service, autrement dit elle doit démontrer un degré d'intelligence. La machine à outils évolue ainsi vers un robot manager d'outils.

Nous étudions un atelier de production qui comprend quatre unités de service ; des ateliers d'assemblage, par exemple. Chaque unité de production (appelée également unité de service, par la suite) émet une demande périodiquement incrémentée jusqu'à une limite supérieure, qui correspond à la demande maximale d'outils (tous les outils dont l'unité de production a besoin). Au-delà de ce point, la demande ne croît plus ; la production de l'unité est momentanément interrompue. Évidemment, ce processus d'incrémentation des demandes seul conduit à une saturation du système où toutes les unités sont bloquées. Notre but est de trouver des spécifications de robots réactifs (comportementaux, en tout cas) qui retardent ce processus de saturation autant que possible, afin de maximiser la production des unités de service (nous supposons que le robot ne peut pas être suffisamment rapide pour desservir toutes les unités instantanément). Les demandes de service représentent des durées de desserte. L'agent se situe initialement au centre de l'atelier et il voyage continûment entre cette position et les positions des unités de service (nous supposons que, pour des raisons de maintien ou autres, l'agent doit passer par le centre après chaque visite). Chacun de ces voyages nécessite T_M unités de temps (cycles de notre simulation). Nous avons imposé ***une limite supérieure à la durée de desserte*** (cette limite est fixée à T_S unités de temps) pour donner périodiquement la chance aux unités non desservies de prendre la main et pour obtenir ainsi une configuration de demandes plus équilibrée dans le cas de fluctuations irrégulières de demandes.

Tous les résultats présentés à la suite sont des moyennes de 10 simulations de 1000 cycles chacune, avec T_M=10 et T_S=3, et toutes les simulations ont été effectuées sans et avec bruit. La présence de bruit signifie que, avec une certaine probabilité, fixée au cours de ces simulations à 5%, on incrémente la demande d'unité de 1. Ce bruit représente en effet des pannes ou d'autres événements imprévus qui retardent la production en augmentant la demande d'une unité. La plupart des simulations ont utilisé une période d'incrémentation de demande de 3 à 5, auquel cas la saturation du système n'est pas trop rapide et alors les conséquences opérationnelles des différentes variantes de système motivationnel de l'agent sont plus visibles.

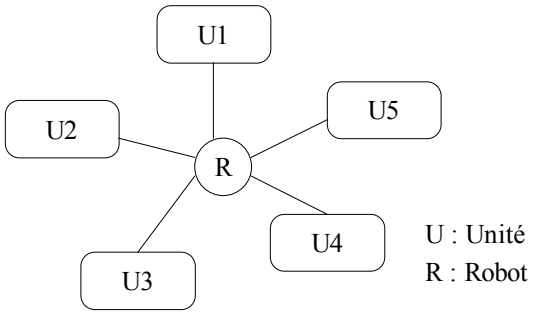

Figure 6.1 L'organisation de l'atelier

Notons que ce problème n'est *pas* un problème d'allocation de ressources (cf. par exemple plusieurs articles à ce sujet dans (Desrochers & Silva 1994)) : nous supposons que les unités de production disposent chacune d'un répertoire d'outils dont elles ont l'usage exclusif et qu'il existe un ensemble supplémentaire d'outils partagés entre les unités et dont l'usage est sporadique. Le rôle de l'agent manager est donc de réguler les demandes des unités pour ces outils "rares", ce qui paraît globalement beaucoup moins coûteux que la réplication de cet inventaire pour chacune des unités. Une autre hypothèse derrière cette configuration d'atelier est que les unités de service font un usage assez "optimisé" de ces outils précieux, par exemple en regroupant leur utilisation dans le temps ; cette flexibilité fonctionnelle est généralement assurée par l'utilisation de représentations distribuées des processus et des produits, telles que les réseaux de Petri d'assemblage (Suzuki et al. 1993). Un tel degré d'intelligence locale aux unités permet à l'agent manager de "décider" quand reculer d'une unité pour aller en desservir une autre, auquel cas il retirera tous les outils alloués pour les rendre disponibles aux autres unités. Ainsi, l'efficacité du système repose sur l'existence d'un tel *couplage coopératif* entre agent et unités, c'est-à-dire entre toutes les entités dont l'atelier se compose.[60]

Dans le paragraphe 6.2 nous présentons le programme de base de l'agent avec une motivation "naïve" qui fait que l'agent n'est pas opérationnel et qu'il a besoin d'un mécanisme de persistance. Plusieurs modèles de persistance sont ensuite présentés et comparés dans le paragraphe 6.3. Le système physiologique de l'agent est présenté dans le paragraphe 6.4 tandis que le paragraphe 6.5 présente les résultats d'une étude sur la spécialisation et son application au cas de plusieurs agents managers. Le chapitre conclut avec quelques observations sur l'implémentation et les résultats.

[60] On peut imaginer diverses extensions de ce problème de pseudo-allocation, par exemple divers types d'outils dont l'allocation est relativement indépendante et pour lesquels l'agent aura des motivations séparées.

6.2 Modèle de base : L'agent motivé

Conformément aux principes des chapitres 1 et 3, nous avons formulé la tâche de gestion d'outils comme une tâche de régulation : chacune des unités de service correspond à une tâche de l'agent, de façon que l'agent, en essayant de réguler ses propres besoins, régule en effet l'opération de ces unités et par conséquent de tout l'atelier de production. La modification des demandes des unités de service provoque une modification des besoins de l'agent qui essaie de mieux répondre à ces nouveaux besoins. Toutes les tâches de l'agent sont structurellement identiques et leur fonctionnalité peut être décrite comme suit :

- Aller à l'unité de service.
- S'installer et la desservir.
- Rentrer au centre de l'atelier.

Chacune de ces tâches exprime une motivation locale dont le niveau dépend de la nature et du degré d'urgence de la demande correspondante ; ces motivations sont finalement comparées et arbitrées au niveau du système de navigation et la tâche la plus motivée exécute.[61] Pour permettre un arbitrage efficace, les valeurs maximales de demande des différentes unités ont été normalisées à une valeur d_{max}, qui, lors des simulations effectuées, a été fixée au 40. La motivation que chacune des tâches exprime est une fonction de la demande correspondante telle que, si demande=0, alors motivation=0. Ainsi, l'agent atteint son état de satisfaction lorsque toutes ses motivations sont à 0, sinon il essaie de les ramener toutes à 0. La présence du processus de production qui tend à saturer les unités de service, perturbe l'état motivationnel de l'agent et le force à agir ou réagir. Le pseudo-code de chacune des tâches de l'agent, selon la fonctionnalité décrite ci-dessus est (cf. paragraphe 3.3.2) :

- Si l'agent est à l'unité de service sélectionnée, alors desservir la demande pour une durée de temps calculée (au plus T_s).
- Si l'agent est au centre, alors aller à l'unité de service.
- Si l'agent est ailleurs, alors aller au centre.

Les motivations des tâches sont recalculées après chaque pas d'exécution (mouvement ou desserte). Une "astuce" du modèle est que ce pas d'exécution est d'une durée variable pour pouvoir appliquer les règles ci-dessus qui définissent un mode d'arbitrage simple *sans besoin de redécision continue* (qui induirait une perte d'efficacité). Nous verrons dans le paragraphe 6.4 comment cela se transcrit dans le système physiologique de l'agent. L'important est que le système motivationnel ainsi que le système physiologique peuvent être écrits et complexifiés indépendamment des systèmes de perception et de navigation de l'agent, l'étude opérationnelle peut alors se faire séparément et en simulation.

Dans un premier ensemble d'expériences, nous avons utilisé une motivation naïve[62] pour chacune des tâches de l'agent :

---

[61] Cette transcription du problème est dérivée directement de la structure de l'environnement, c'est-à-dire de la structure de l'atelier de production et la bonne conséquence est que ce robot peut être réutilisé dans des ateliers de structure différente simplement en réinitialisant le nombre des tâches.

[62] En effet, cette formule, ainsi que les deux premières formules de persistance du paragraphe suivant, ne donnent pas une motivation au sens classique du terme, puisque la seule variable présente est la demande correspondante qui constitue le *stimulus* de la tâche. Il ne s'agit d'une motivation que par rapport à un robot complètement réactif, qui ne prendrait pas en compte la valeur de la demande, mais

Motivation naïve : *motivation=demande*

Ce schéma a conduit l'agent à ne jamais desservir une unité, parce que, lors de chaque évaluation des différentes motivations, et en raison de l'asynchronisme des demandes, une autre tâche était sélectionnée. La solution à ce problème est d'introduire un mécanisme de persistance : une fois une tâche sélectionnée, son exécution doit persister au-delà du point où elle est prioritaire. Ce mécanisme est nécessaire parce que les coûts de transition (switching costs) de tâche à tâche sont trop élevés par rapport au taux de "satisfaction" de chaque tâche (ici, le paramètre T_M ne peut pas être ignoré, sinon l'état de satisfaction serait atteignable instantanément).

6.3 Modèles alternatifs de persistance

Le premier modèle de persistance introduit utilisait un facteur additif appliqué seulement à la tâche sélectionnée.

Persistance additive : *motivation=demande+facteur*

Une expérimentation avec des différentes valeurs pour ce facteur a montré que souvent l'agent se bloquait définitivement à la première unité desservie en ignorant obstinément les autres unités. En analysant l'expression ci-dessus, nous avons constaté que cela peut arriver seulement lorsque

$$facteur\text{-}T_S+(T_S \text{ div période})>0$$

(le deuxième et le troisième terme sont les quantités de demande desservie et renouvelée, respectivement, durant la période que la tâche est active). Pour $période>T_S$, l'expression ci-dessus est vraie si $facteur>T_S$. L'utilisation d'un facteur inférieur à cette valeur ne donne qu'une amélioration marginale, puisque l'agent abandonne toujours une tâche trop vite. Quelques résultats avec ce modèle de persistance sont résumés dans le tableau 6.1.

	Période=3	*Période=4*	*Période=5*
facteur=2	110(57) 36.74	109(57.75) 35.336	44(23.3) 35.019
facteur=3	110(57) 36.74	65.8(34.95) 35.808	1(0.5) 35.844

Tableau 6.1 Résultats comparatifs pour le modèle de persistance additive (sans bruit) : les trois mesures dénotent le nombre de mouvements, la moyenne de temps de service par unité (entre parenthèses) et la demande moyenne par unité (à la ligne). Notons que les unités sont saturées pour la plupart du temps, de façon que, même si l'agent est bloqué depuis le début à une unité, les performances moyennes ne se dégradent pas trop.

Le modèle suivant utilisait un facteur multiplicatif, toujours appliqué seulement à la tâche sélectionnée.

Persistance multiplicative : *motivation=demande*facteur*

Évidemment, ce facteur doit être supérieur à 1. Une expérimentation avec des différentes valeurs de ce facteur a montré que les performances du système étaient nettement meilleures de celles avec une persistance additive et se stabilisaient rapidement avec une tendance descendante après *facteur=période*. Dans le pire des

qui raisonnerait seulement en termes de présence/absence de demande. Ceci dit, dans cette application, seule la réactivité d'un robot ne suffit pas.

cas, si toutes les unités sont saturées à leur demande maximale d_{max}, il paraît logique de les desservir l'une après l'autre jusqu'à ce que leurs demandes tombent à 0 (entre-temps bien entendu, les demandes des autres unités seront remontées de nouveau). Dans ce cas, une tâche sélectionnée doit persister pour autant que sa demande est supérieure à 0, donc *facteur*>d_{max}. Des résultats comparatifs pour *facteur=période* et *facteur=1+*d_{max} sont présentés dans le tableau 6.2. ***La persistance est alors le mécanisme qui rend le système opérationnel et elle est basée sur une étude du pire cas (worst case analysis)***.

	Période=3	*Période=4*	*Période=5*
facteur= période	37(156.75) 30.893	35.9(156.25) 26.498	37(155.93) 22.127
facteur= 1+d_{max}	29(177) 26.975	31(171.5) 23.205	34(164) 19.342

Tableau 6.2 Résultats comparatifs pour le modèle de persistance multiplicative (sans bruit). À comparer avec le tableau 7.1 pour voir l'amélioration importante des performances.

La persistance est toujours impérative, même si le système n'est pas surchargé (c'est-à-dire, même si sa vitesse de saturation est basse), parce qu'elle réduit les coûts de transition et elle améliore ainsi les performances : des résultats dans les cas de *période*=25 et 30 sont présentés dans le tableau 6.3.

	Période=25	*Période=30*
Sans persistance	90(17.825) 11.301	88.8(19.25) 7.415
Avec persistance	82(37.5) 1.793	84(31.25) 1.507

Tableau 6.3 Résultats comparatifs pour une grande période d'incrémentation de demandes avec ou sans persistance (sans bruit). Notons que par comparaison aux tableaux 6.1 et 6.2, la saturation des unités de service dans ce cas est beaucoup moins fréquentes.

Pour écrire le programme cellulaire correspondant, nous avons utilisé une fonction continue pour le facteur de persistance, la formule exponentielle[63] :

$$facteur=1+d_{max}(1-e^{-t/Tp})$$

qui monte de 1 à $d_{max}+1$ (t est le temps écoulé depuis la sélection de la tâche). Puisque la montée du facteur à son maximum doit être quasi-instantanée après la sélection de la tâche, la constante T_p doit être telle que $1+d_{max}(1-e^{-1/Tp})>d_{max}$, qui donne $T_p<1/ln(d_{max})$. Pour ces expérimentations, nous avons utilisé une valeur "sûre" de T_p, à peu près $1/2ln(d_{max})$, T_p =0.13 (puisque d_{max}=40).

Nous avons présenté jusqu'ici un ensemble de mécanismes de persistance qui améliorent les performances de l'agent motivé de base. Cependant, comme nous l'avons déjà mentionné, l'agent n'est pas vraiment autonome quant à ses motivations, car ces motivations ne dépendent que des stimuli présents. Est-ce gênant ? Imaginons une ou plusieurs unités de service qui sont défectueuses (ou vicieuses ou capricieuses!) et qui émettent toujours la demande maximale ; dans cette situation, l'agent va être "manipulé" et va se bloquer à une de ces unités sans jamais se rendre compte que son activité ne donne pas de résultat (la demande de l'unité ne baisse pas). Ce qui semble nécessaire est alors un mécanisme d'ennui ou un facteur dissipatif qui force le facteur de persistance à tomber progressivement à 1 après une période

[63] Une formule exponentielle qui converge rapidement constitue une approximation d'une fonction de pas (*step function*) discontinue.

écoulée T_R (pour laquelle, $T_R>2T_M+d_{max}(1+1/période)$ doit être vrai). En termes de théorie des systèmes motivationnels, il y a besoin d'une variable supplémentaire représentant une pulsion interne : cette variable doit être initialisée à T_R lorsqu'une tâche est sélectionnée et baisser progressivement avec le temps.

> ***Persistance dissipative :*** *facteur=var(t)*(1+d_{max}),*
> *var(0)=T_R .*

Comme c'est la règle avec les variables motivationnelles, cette variable ressemble à une variable hédoniste qui exprime le degré de plaisir futur attendu ; ce degré baisse avec le temps, comme si l'agent arrivait petit à petit à un état de satiation.[64]

6.4 Le manager : Système physiologique

Partie réflexe

- *Constantes : Un identificateur par unité de service (releaser(i)).*
- *Variables d'état : Une variable physiologique par unité de service (unit(i)), un marqueur-identificateur global (releaser), une variable pour le centre (centre), un timer (timer).*
- *Capteurs-réflexes : sur_le_centre, sur_unité(i) pour chaque unité.*
- *Si sur l'unité-i, alors catalyser unit(i) et décatalyser unit(j), pour tous les autres j.*
- *Si sur le centre, alors centre := vrai.*
- *Décrémenter timer. De plus, si sur une unité dont la demande est 0 ou releaser = nil, alors timer := 0.*
- *Si timer = 0, alors releaser := nil.*

Activité cellulaire qui affecte la physiologie

- *Cellule de priorité globale : Mise à jour du timer lorsque timer = 0. Mise à jour du releaser. De plus, si le releaser de la tâche sélectionnée n'est pas le releaser courant, alors centre := faux.*
- *Cellule de composition de motivation : elle exécute seulement si le releaser courant est nil ou celui qui correspond à la tâche dont la cellule fait partie, et le message de sortie contient le releaser correspondant.*
- *La première cellule de propagation de stimulus (celle qui donne le signal aller-à-l'unité) exécute seulement si centre = vrai.*

Figure 6.2 Description du système physiologique de l'agent manager. Le programme cellulaire est décrit dans le paragraphe 3.3.2 (cf. figures 3.9 et 3.10).

[64] Cette remarque constitue aussi la base de réflexion sur un mécanisme de gestion des pannes à long terme, pour lesquelles un facteur dissipatif supplémentaire serait nécessaire, mais cela dépendrait du type des pannes en question ainsi que de la gestion qu'on voudrait en faire.

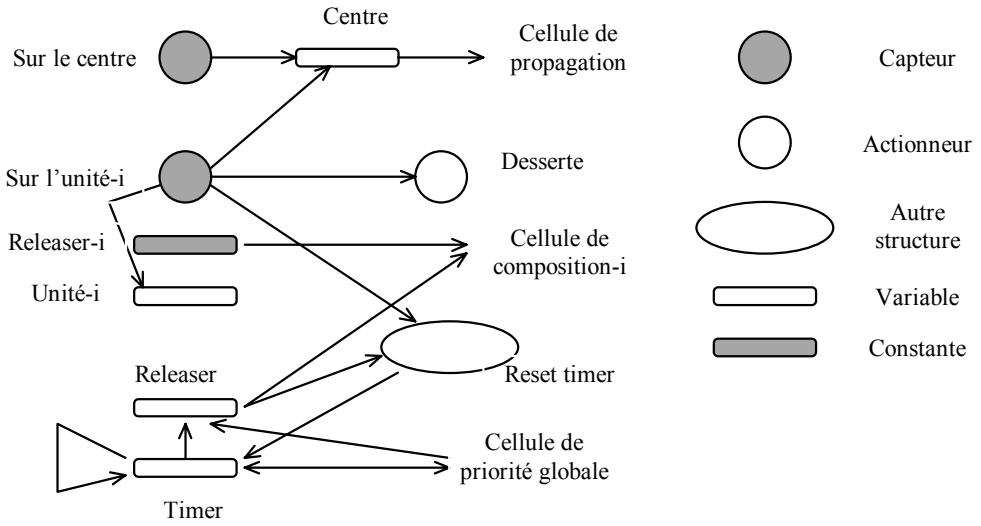

Figure 6.3 Le réseau physiologique de l'agent manager. Les liens représentent les dépendances entre les diverses structures physiologiques (cf. figures 3.9 et 3.10). La boucle du timer signifie la présence d'une dynamique propre indépendante des autres structures et dépendances, en l'occurrence la décrémentation.

L'agent doit passer par le centre avant d'aller à une unité de service, il faut alors mémoriser le fait du passage au centre (et plus tard l'oublier). Cela est possible si une variable binaire d'état (*centre*) est introduite : à chaque fois que l'agent passe par le centre, son réflexe de reconnaissance du centre s'active et la variable prend la valeur *vrai*. La première cellule de propagation de chaque tâche, celle qui décide d'aller vers l'unité correspondante, est inhibée par la valeur *faux* de *centre*. La variable prend la valeur *faux* lorsque l'agent se trouve sur une unité de service (ainsi elle a déjà la valeur *faux* lors de la désélection de la tâche courante).

Pour chacune des unités de service, nous avons également introduit une constante supplémentaire, le *releaser (i),* ainsi qu'une variable globale, le *releaser* global.[65] Les cellules de composition de chaque tâche composent en effet trois messages (selon le mécanisme de composition et séparation du paragraphe 3.6) : la direction de voyage, la motivation et la valeur de leur releaser. Pour exécuter, elle a simplement à comparer le releaser choisi et le releaser courant (le releaser global). De son côté, la cellule de priorité globale met à jour la valeur du releaser courant. *Ces variables constituent ainsi la solution à un problème de planification et redécision*.

Le dernier système de variables physiologiques concerne le système motivationnel de l'agent : comme nous l'avons déjà dit, il faut une variable "hédoniste" par tâche, *var(i)*, qui suit sa propre dynamique auto-catalytique et qui exprime "le suivi de l'activité de desserte". Il faut encore une variable *timer* qui bloque la redécision pour la durée déterminée par l'activité en cours, T_S ou T_M (rappelons-nous l'intérêt d'avoir un pas d'exécution variable, cf. paragraphe 6.2). Le *timer* prend la valeur 0 lorsque l'agent est sur une unité dont la demande est 0 ou lorsque le *releaser* devient nil (qui signifie qu'il y a eu redécision). Lorsque le *timer* devient 0, le *releaser* devient nil aussi — pour permettre la redécision, puisque cela libère les cellules de composition de motivation (cf. fig. 6.2). En effet, avec ce mécanisme de timer, on bloque les

[65] J'ai utilisé le terme "releaser" qui est celui que Brooks (1990) a emprunté à la terminologie des systèmes hormonaux pour dénoter les conditions globales, c'est-à-dire celles qui expriment des interactions entre des parties éloignées dans le réseau.

cellules de composition de toutes les autres tâches sauf celle activée, afin de permettre la redécision locale à l'intérieur de la tâche : par exemple, si l'agent est en train de se diriger vers une unité et entre-temps la demande émise par la tâche disparaît, la cellule de priorité s'en aperçoit (puisque la motivation exprimée devient 0) et elle met le *releaser* à nil. ***Le rôle du timer est donc de réguler une interaction entre les tâches qui se manifeste comme une planification de l'agent***. Le rôle de tout le système physiologique est de permettre la description des interactions globales dans le réseau : *l'idée de la description simple de ce plan a été de traduire les relations temporelles (quand décide-t-on?) en relations spatiales globales (qui décide à un moment donné?) qui sont directement implémentables au niveau de la physiologie*.

6.5 Multi-robots : Une étude sur la spécialisation

Nous avons répété la même étude pour plusieurs agents, à la recherche d'un mécanisme de spécialisation qui donnerait une allocation agents-unités relativement stable. Nous avons étudié l'option de l'auto-catalyse, où la motivation de chaque tâche dépendrait d'un facteur qui serait renforcé durant l'exécution de la tâche. Nous avons testé les mêmes possibilités, de facteur additif, multiplicatif et multiplicatif avec un seuil supérieur. La deuxième possibilité s'est avérée beaucoup plus opérationnelle que la première, mais le facteur multiplicatif pourrait éventuellement prendre des valeurs prohibitives en raison de renforcement continu. L'introduction du seuil supérieur a résolu ce problème, mais elle a conduit à des phénomènes semblables à ceux de la motivation individuelle naïve. La combinaison du facteur renforcé ayant un seuil supérieur avec un mécanisme de persistance a donné les meilleurs résultats parmi tous ces modèles, mais ces résultats n'étaient qu'à peine meilleurs du cas des agents sans spécialisation et seulement avec persistance. La légère amélioration des performances peut être attribuée à l'asynchronisme du système qui induit par exemple une régulation légèrement meilleure pour un atelier de deux unités et un seul agent, que pour un atelier de quatre unités et deux agents (dans le chapitre 5, on a décrit un résultat semblable à propos de l'asynchronisme et de la diversité).

La leçon que nous pouvons en tirer est que la spécialisation n'est peut-être pas la meilleure idée dans ce cas et la raison en paraît double. Premièrement, les tâches qui correspondent aux unités de service sont indépendantes, de manière que, à part les coûts de transition, il n'y a pas besoin d'équilibrage de charges (ce qui serait le cas, si l'on voulait par exemple maintenir un certain flux de tâche à tâche). Si les tâches sont indépendantes, la relation entre elles est ouverte : les agents n'ont aucun critère social (rétribution ou punition) pour arrêter ou reprendre service, c'est-à-dire ils ne fonctionnent pas en boucle fermée. Deuxièmement, l'auto-catalyse seule conduit à un système où un agent serait spécialisé à deux tâches maximum et ainsi elle n'est pas opérationnelle dans le cas de 3 agents et 8 unités de service (dans tous les cas, la persistance seule semble suffire). Puisque tous les agents ont accès *simultanément* à la même information, une spécialisation ne peut pas se faire par auto-catalyse : il faut, ou bien une sorte de communication/coordination, ou bien une sorte de "préférence" ou de spécialité innée, qui reposerait sur la présence de causes plus "génétiques", plutôt que des mécanismes simples d'auto-catalyse, c'est-à-dire une spécialisation plus sociobiologique que simplement un mécanisme de stimulus-réflexe aveugle. Deux exemples de spécialisation à succès sont (Deneubourg et al. 1987) et (Drogoul 1993), où les systèmes qui se spécialisent sont en effet des systèmes auto-organisants

où les agents n'ont pas accès à la même information et à toute l'information simultanément : des facteurs "spatiaux", tels que des rayons de perception limités, induisent une topologie et une mémoire spatiale non uniforme entre les agents. De plus, dans (Drogoul 1993) il existe une dépendance entre les tâches, même s'il manque une régulation explicite. En plus, dans les deux cas le ratio {nombre d'agents / nombre de tâches} est très grand par rapport à l'unité, c'est pourquoi ces sociétés survivent et sont opérationnelles.

6.6 Conclusion

Dans ce chapitre a été étudié le système de contrôle d'un agent chargé de la régulation d'un atelier industriel. Il a été montré que le système motivationnel de l'agent nécessite un mécanisme de persistance. L'étude d'un ensemble de modèles de persistance a révélé que le modèle opérationnel est celui d'une persistance dissipative dont les paramètres sont couplés avec ceux du processus industriel ou de la tâche de la régulation. Le séquencement de l'activité de l'agent a été transcrit dans une physiologie élaborée qui consiste en un ensemble de variables binaires ou continues régulant le système cellulaire de contrôle globalement. Une dernière étude au niveau d'une population d'agents managers a montré que la spécialisation n'est pas avantageuse à cause de l'indépendance des unités de production et de l'uniformité de "l'information" accessible aux agents.

Puisque le système de contrôle de l'agent a été décomposé en un ensemble d'activités autrement indépendantes arbitrées au niveau des actionneurs (ici au niveau du système de navigation), les détails des systèmes de perception et de navigation ont été étudiés séparément du système motivationnel et du système physiologique. Nous avons montré encore une fois le rôle du temps comme paramètre de conception, dans la définition du facteur dissipatif de persistance, ainsi que dans la durée variable de l'activité de la desserte. Cette continuité dans le temps a une autre conséquence bienvenue : non seulement elle renforce la résistance au bruit, mais elle en permet aussi l'exploitation pour l'obtention de meilleures performances. En effet, toutes les expérimentations ont montré que les performances du système étaient légèrement meilleures en présence de bruit.[66] De plus, la présence de bruit au sens de l'asynchronisme entre événement et perception de l'agent s'est révélée nécessaire pour assurer l'opérationalité du système motivationnel : par exemple, nous avons étudié le cas d'une période d'incrémentation de demande de 2 avec un incrément de 2 et nous avons conclu que l'agent se bloquait toujours à la première unité desservie, car les opérations de l'atelier et de l'agent étaient synchrones, donc l'agent ne pouvait jamais percevoir des différences significatives aux autres unités pour aller les desservir.

Une fois de plus, l'étude a montré le besoin de couplage entre l'agent (le robot) et le processus, dans le sens de l'utilisation de paramètres de processus (tels que d_{max} et T_M) lors du développement du système de contrôle de l'agent et plus particulièrement lors du développement de son système motivationnel. L'identification des lois *exactes* de ce couplage a été basée sur une analyse du pire cas du système atelier-agent. L'opérationalité dans le contexte de cette étude a le sens d'une optimalité statistique

[66] On peut remarquer que le bruit dans ce contexte équivaut à la variation ou diversité dans un système distribué.

pour l'ensemble des paramètres opérationnels critiques. Cependant, puisqu'il n'existe pas de paramètre libre (c'est-à-dire de paramètre qui est d'avance inconnu et qui varie au cours de l'opération du système), il n'existe pas non plus de besoin de régulation des paramètres internes du système de contrôle de l'agent.

Dernière remarque, étant donné ce couplage agent-atelier, l'opération de l'agent peut être vue comme une opération suivant un "plan prédéfini", mais adaptatif et robuste. Cependant, l'agent ne forme et ne manipule pas de plans au sens que l'IA classique donne au terme ; il "réagit" simplement à la situation rencontrée. S'il y a un plan, il n'est que dans la tête de son concepteur (ou dans l'oeil de son observateur) : le rôle du concepteur est ainsi d'identifier les paramètres opérationnels critiques et de trouver des lois de couplage appropriées pour assurer l'exécution efficace de la tâche et la résolution du problème. Dans cette application, l'agent lui-même n'a pas besoin de planifier parce que son concepteur l'a déjà fait ; l'idée est que la "réactivité" constitue la solution optimale dans le cas de tâches mal-définies comme celle-ci, mais elle doit être basée sur une connaissance opérationnelle. La structure du système de contrôle, et plus particulièrement celle du système motivationnel et du système physiologique, correspond au composant hors ligne d'un système hybride de planification, tandis que l'interaction avec le monde va fournir les données actuelles qui seront utilisées pour la "décision" en ligne.

Le problème	Système de contrôle d'un agent devant résoudre un problème qui nécessite un degré de "planification"
L'application	Agent manager, régulation d'un atelier industriel
La solution	*Système motivationnel* (une motivation de desserte par unité de service) *Mécanisme de persistance* *Couplage entre paramètres de persistance et paramètres de la tâche de régulation* *Physiologie élaborée*
Les enseignements	• Persistance additive vs. multiplicative vs. dissipative • Transcription du séquencement de l'activité de l'agent dans sa physiologie • Spécialisation non avantageuse

Tableau 6.1 Tableau récapitulatif du chapitre 6

Chapitre 7 La cellule est également un agent

"ΤΑ ΠΑΝΤΑ ΡΕΙ." [*]
(ΗΡΑΚΛΕΙΤΟΣ)

7.1 Introduction

Dans ce chapitre, nous discuterons le besoin du réseau cellulaire pour un degré de plasticité et nous montrerons comment cette plasticité observable se traduit en effet par *autonomie intrinsèque* des cellules. Nous implémenterons donc un mécanisme qui rend le réseau plastique en rendant les cellules autonomes (ou, *plus* autonomes qu'avant) et le comparerons aux mécanismes des chapitres précédents qui concernent des agents autonomes d'un niveau supérieur, celui de l'animat. Cette comparaison à travers les niveaux se fera par rapport aux principes qualitatifs introduits dans le chapitre 1, puisque les deux niveaux, et les problèmes correspondants dont la résolution sert de base d'illustration, ne sont pas corrélés.

7.1.1 Qu'est-ce que la plasticité et pourquoi en veut-on ?

Jusqu'à maintenant, nous avons décrit et présenté des modèles comportementaux algorithmiques (au niveau motivationnel ainsi qu'au niveau cellulaire). Une telle structure algorithmique donne naissance à des comportements "programmés". Si on vient de montrer la nécessité de l'existence de tels mécanismes préprogrammés qui peuvent assurer "l'opérationalité" des agents face à des tâches données, il reste tout aussi important d'équiper ces mêmes agents des mécanismes d'auto-programmation capables de modifier les structures préexistantes (c'est-à-dire les réseaux cellulaires) afin de mieux répondre aux besoins particuliers que l'interaction avec un environnement donné génère. Du point de vue de la conception, ce besoin d'auto-programmation résulte à la fois du besoin de minimisation de l'effort humain lors de la conception ainsi que de l'impossibilité de déterminer d'avance l'activité de l'agent, soit parce que son monde est a priori inconnu soit parce que les interactions agent-monde sont d'une complexité incontrôlable. Guillot et Meyer (1994) ont identifié trois classes de mécanismes de "programmation automatique" : apprentissage, évolution et développement. L'apprentissage concerne l'auto-programmation d'une structure autrement fixe selon des critères qui visent à optimiser plus ou moins les performances de l'agent ; cette auto-programmation est en effet une modification ou adaptation des paramètres internes des composants-noeuds du réseau (dans le cas des réseaux neuronaux, le pattern de connectivité dans une topologie fixe est reflété dans les poids associés aux connexions, qui pourraient être considérés comme des

[*] *"Tout est fluide."* (Héraclite)

paramètres internes des neurones). L'évolution concerne l'optimisation de ces performances à travers une succession de générations, tandis que le développement correspond à l'auto-modification de la structure même grâce à un mécanisme de croissance ou équivalent. Par la suite, nous nous intéresserons uniquement au domaine d'interaction de l'apprentissage avec le développement à l'intérieur d'un seul agent cellulaire et nous ignorerons l'option de l'évolution qui fait appel à une population d'agents et à un critère de sélection. Ces deux mécanismes partagent une propriété fonctionnelle : plasticité, c'est-à-dire *possibilité auto-organisationnelle du réseau*, que la structure du réseau reste fixe (auquel cas on l'appelle *topology preserving self-organization*) ou pas. On dit alors que pour permettre l'apprentissage, il faut de la plasticité :

> ***Plasticité - Point de vue fonctionnel :*** *La plasticité est la propriété d'un réseau de s'auto-organiser (apprendre) face à des situations imprévues ou sa propriété de s'adapter à des conditions d'entrée imprévues.*

On constate que, le passage de l'agent algorithmique à l'agent adaptatif au sens donné, modifie le problème original de conception : le concepteur ne centre plus ses efforts sur le "programme" qu'il doit mettre dans l'agent mais sur la méthode d'apprentissage/adaptation nécessaire. *C'est-à-dire le problème de conception devient un méta-problème et la solution recherchée reste algorithmique!* Qu'est-ce que cela signifie ? Du point de vue fonctionnel, l'agent est conforme à un ***modèle instructif***, selon lequel son environnement, ou milieu des interactions, donne des instructions d'apprentissage (fig. 7.1). Il existe bien entendu un lien de rétroaction non présent dans le schéma, celui de l'action de l'agent-cellule et de l'effet que cette action a sur le milieu d'interaction (autrement dit, la rétroaction est à son tour une instruction de la cellule à son monde).

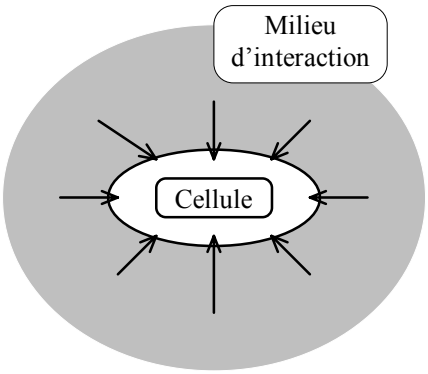

Figure 7.1 Modèle instructif (Interaction = Instruction)

Une conséquence immédiate en est que le raisonnement du concepteur est celui d'un observateur qui attribue une valeur à ce qu'il observe selon *ses* propres intentions et prédispositions. Une théorie causale du comportement "plastique" d'un agent observé peut ne pas avoir une relation directe avec les observations d'un tiers et nous allons tenter d'en esquisser une par la suite.

7.1.2 La poule et l'oeuf : Algorithmique et plasticité

La plasticité est souvent vue comme une propriété de base qui sert pour la génération des catégories de haut niveau, tels que les "symboles" utilisés dans les langues vivantes. Par exemple, des recherches connexionnistes se sont centrées à l'émergence des catégories linguistiques dans un réseau neuronal ayant des possibilités élémentaires d'auto-organisation ou d'apprentissage ; cependant, Clark (1993) plaide les rapports entre les structures de base et la fonctionnalité (ou "connaissance") émergente et s'oppose aux modèles "tabula rasa" en proposant à leur place des modèles de "rationalisme minimal". Varela et al. (1991) identifient la démarche connexionniste orthodoxe (Anderson & Rosenfeld 1988; Anderson et al. 1991) comme le contraire du cognitivisme classique qui a été le paradigme prédominant dans les recherches en intelligence artificielle jusqu'aux années quatre-vingt. Ils proposent alors une voie encore plus radicale, celle de l'enaction, où il existe un niveau élémentaire de catégorisation, c'est-à-dire une "taxonomie préexistante d'objets" se situant à la frontière de l'interaction entre agent et monde (ibid., p. 177). Varela et al. (ibid.) citent les travaux de Rodney Brooks et de son équipe comme une démonstration robotique de l'importance d'une théorie basée sur l'activité et non pas sur la fonctionnalité déterminée par l'extérieur. Or, une solution simple à un problème classique d'apprentissage spatial (Mataric 1990, 1991, 1992*a*) révèle le rôle que joue la présence de telles catégories élémentaires (mur, couloir) ; à noter la remarquable ressemblance avec des méthodes de raisonnement spatial qualitative de "haut" niveau (Kuipers & Byun 1991).

Cette démonstration nous amène à nous intéresser de nouveau aux rapports qui existent entre les différentes architectures de robotique et d'I.A. comportementale. Comme on l'a déjà vu dans le chapitre 3, ces architectures peuvent être classifiées dans deux grandes catégories : l'algorithmique et le connexionnisme.[67] Grosso modo, l'approche algorithmique (Brooks 1986*a*,1990; Arkin 1987,1989) présuppose que le problème (tâche robotique) est donné d'avance et que le système de contrôle peut être bâti là-dessus, tandis que l'approche connexionniste (Verschure et al. 1992; Husbands et al. 1993; Yamauchi & Beer 1994) présuppose une "connaissance a priori" limitée qui fait que le système de contrôle (alors l'agent!) doit apprendre tout seul les spécificités du problème et — d'une manière ou d'une autre — converger vers sa solution. Cependant, dans le cas de l'algorithmique, l'opération de l'agent n'est pas pré-planifiée : au contraire, l'activité de l'agent est le résultat d'une évaluation continue des données sensorielles et du feedback "envoyé" par le monde, ce qui fait qu'elle paraît suivre un véritable algorithme, adaptatif/réactif et robuste aux situations imprévues. De son côté, le connexionnisme part des structures et des règles d'apprentissage assez spécifiques à la tâche en question : il ne doit donc pas être vu comme une solution passe-partout. Pour revenir à l'algorithmique, elle permet l'implémentation (programmation) des systèmes de contrôle distribué extrêmement efficaces et peu coûteux (Brooks 1991*a*,*c*), tout en restant suffisamment compréhensibles par leur concepteur. Le prix à payer est la rigidité/inflexibilité des réseaux impliqués : bien que les performances du système se dégradent gracieusement

[67] En effet, la distinction est entre algorithmique et classification. Le connexionnisme n'est opposé à l'algorithmique que dans la mesure où, contrairement à l'algorithmique, il permet le développement des structures plastiques et la résolution de problèmes de classification (cf. paragraphe 3.1.2).

quand des composants individuels dégradent, il n'existe pas de possibilité de récupération lors d'une panne.[68] Notons que, cette discussion sur les pannes n'a de sens que dans le cas d'une implémentation distribuée des cellules en matériel ; l'approche habituelle de simulation du système distribuée de contrôle du robot sur un ordinateur embarqué ne présente pas ce besoin de gestion de pannes mais en revanche nécessite de plus grands efforts de synchronisation. Il est alors pertinent de réfléchir sur ce qui se passe en cas de pannes si l'on veut comparer les solutions inventées ou conçues avec celles de la nature.

Contrairement à l'algorithmique, le paradigme connexionniste est fondé sur des structures redondantes avec des fonctionnalités uniformes pour tous les composants, ce qui fait que les pannes n'affectent pas la fonctionnalité du réseau complet. Cependant, et à part l'inefficacité, la redondance et l'uniformité induisent un manque de compréhension du fonctionnement interne et de la causalité du système, ce qui rend le dimensionnement une possibilité hors portée.

En nous penchant de plus près sur cette différence entre l'algorithmique et le connexionnisme, nous constatons qu'en effet il ne s'agit que d'une différence d'échelle : l'algorithmique cherche la minimalité en vue d'un problème donné, tandis que le connexionnisme cherche la minimalité en vue d'une classe de problèmes. Cette différence reste large tant qu'on ne sait pas encore caractériser exactement ce qu'il faut ajouter dans le cas de l'algorithmique pour résoudre un problème voisin de celui traité et ce qu'il faut supprimer dans le cas du connexionnisme pour résoudre une instance particulière de la classe des problèmes. L'algorithmique reste ainsi efficace mais structurellement rigide, tandis que le connexionnisme démontre de la plasticité structurale au détriment de l'efficacité.

Ayant adopté l'approche algorithmique, nous avons essayé de comprendre la nature d'une plasticité à un sens inverse de celui qu'on lui a donné jusqu'à présent, c'est-à-dire une plasticité qui est nécessaire pour des raisons *intrinsèques* au réseau ne coïncidant pas nécessairement avec la vision d'un observateur. Cette plasticité doit finalement être capable d'élargir le domaine "d'application" de l'algorithme que le réseau implémente, c'est-à-dire la classe des problèmes qu'il sait résoudre, et être ainsi la *cause* de la plasticité fonctionnelle.

7.1.3 De l'apparence au mécanisme : Algorithmique, causalité et sélectivité

Nous définissons alors la plasticité comme la propriété du réseau de récupérer des pannes locales, afin de se maintenir en tant qu'unité d'activité algorithmique.

> ***Plasticité - Point de vue causal :*** *La plasticité est la propriété d'un réseau de s'auto-organiser afin de se maintenir en tant qu'unité cohérente d'activité algorithmique innée, c'est-à-dire la propriété du réseau de maintenir les relations sociales.*

[68] Ferrell (1993) a implémenté sur le robot héxapode Hannibal un algorithme *ad hoc* de récupération sur des pannes qui repose sur une décomposition hiérarchique du processus de détection de pannes et sur une distinction claire entre matériel (capteurs/actionneurs) et logiciel (programme compilé dans les processeurs). Ici j'explore la notion de pannes dans un réseau où une telle distinction n'existe pas : tout est de même nature, tout est cellule, et notamment le "logiciel" est lui aussi un réseau de cellules matérielles!

La plasticité est alors considérée comme une sorte de redondance et est analysée ici du point de vue "négatif" : quelle genre de redondance nous faut-il pour que le système de contrôle réussisse à résoudre le problème donné en dépit de pannes de matériel et quelle cascade de pannes devrait-on avoir pour anéantir l'opérationalité du système de contrôle ? N'oublions pas que dans les organismes vivants sains, comme dans un corps humain ou animal, il y a des cellules qui meurent tous les jours : malgré ce mécanisme de dégradation naturelle, les organismes arrivent à survivre et même à améliorer leur fonctionnement.[69] Pour un observateur de haut niveau, tel qu'un éthologiste, l'algorithme de fonctionnement n'est pas atteint par ces "pannes" de bas niveau. L'implémentation d'un système de contrôle sur un matériel distribuée devrait démontrer la même propriété. Dans le futur nous envisageons aussi d'analyser cette même redondance du point de vue "positif" : s'il n'y a pas de pannes, à quoi cette redondance pourrait servir et que faudrait-il pour que le système de contrôle "découvre" de nouveaux problèmes à résoudre, c'est-à-dire qu'il élargisse la classe de problèmes qu'il sait traiter ? La redondance paraîtra ainsi comme le potentiel cognitif qui est petit à petit "exploité" par l'agent lui-même (je suis attirée par l'idée que l'apprentissage pourrait être tout simplement la réaction des cellules autrement "indolentes" qui veulent participer à la fête, c'est-à-dire au processus de contrôle[70]).

La tâche que nous allons étudier sous ce prisme de plasticité est la navigation d'un agent dans un environnement contenant des obstacles à éviter et une trace à suivre éventuellement afin d'arriver à une position-but donnée. Notre but dans les paragraphes qui suivent sera de *développer* un mécanisme permettant la plasticité telle que nous l'avons définie ainsi que d'explorer les conséquences de ce mécanisme en termes de structure de réseau. Les simulations et les résultats obtenus ont alors une valeur qualitative ; ils montrent pourquoi et comment un mécanisme de plasticité ainsi matérielle peut être utile et élucident ses particularités. Dans une implémentation ultérieure nous comptons *utiliser* ce mécanisme et ce modèle cellulaire modifié comme base d'apprentissage spatial de la part de notre agent simulé. Actuellement, l'agent simulé n'apprend rien de nouveau sur son environnement, mais seulement comment mieux utiliser son potentiel cellulaire pour "implémenter" son algorithme de navigation.

Résumé du modèle plastique. L'extension du modèle cellulaire algorithmique consiste à équiper chacune des cellules de multiples "motivations" de traiter/consommer les messages de types différents et à supprimer toutes les connexions fixes de manière que les cellules soient en compétition pour la consommation des messages. Les motivations individuelles sont exprimées comme des variables continues excitées par les valeurs d'identification des messages en entrée, c'est-à-dire comme des fréquences propres. Parallèlement, on définit deux types de messages : les messages normaux qui sont consommés par les cellules et les messages catalytiques qui déclenchent la consommation des autres messages chez les cellules excitées. Comme les résultats de simulation laisseront voir, la variation génétique permet au réseau cellulaire de s'auto-organiser en "découvrant" des voies alternatives de flot de messages et plusieurs pannes ralentissent la réactivité du

[69] Effectivement, certaines de ces cellules sont remplacées par d'autres, mais ceci n'est pas le cas des cellules nerveuses.

[70] Dans un contexte différent, celui de l'étude de l'auto-organisation au sein des colonies des fourmis, Drogoul (1993) a discuté la nécessité de la présence de fourmis paresseuses pour assurer la robustesse de la société aux diverses perturbations.

système aux événements extérieurs, puisque les cellules des niveaux intermédiaires ne sont plus spécialisées au traitement d'un seul type de message, mais partagent leur temps entre plusieurs motivations internes.

Nous introduisons également un facteur supplémentaire développemental /ontogénétique pour permettre l'auto-organisation même dans le cas de pannes majeures dans le réseau, lorsqu'une des fréquences propres nécessaires pour le fonctionnement du système disparaît complètement du réseau. Mais la conséquence la plus spectaculaire de ce modèle est que, si l'on suppose que les messages circulant dans le réseau peuvent être faux (par exemple, ils pourraient être mutés ou trop bruités), alors naît le besoin d'un système immunitaire. Le modèle de la cellule motivée est donc modifié une deuxième fois pour inclure la fonctionnalité nécessaire à un système immunitaire élémentaire : deux types de cellules sont définies, les cellules "productrices" et celles du système immunitaire. La propriété importante de cette configuration à deux "forces sociales" est que les deux populations de cellules ont des fréquences propres complémentaires ! Avant de procéder à la description détaillée du mécanisme ainsi que des résultats de simulation obtenus, nous souhaitons discuter une propriété importante du nouveau modèle cellulaire : la ***sélectivité***.

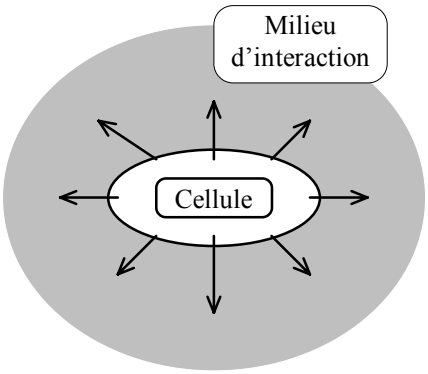

Figure 7.2 Modèle sélectif (Interaction = Sélection)

Le fait d'avoir des cellules motivées, c'est-à-dire des cellules qui *décident* si elles vont traiter un message ou non et de ce qu'elles vont en faire, implique que ces cellules ne sont plus *directement instruites* par le milieu où elles sont situées, mais que, en effet, elles sont *sélectives* envers leur monde ; il ne s'agit plus d'*information* qui est transmise du monde à l'agent, mais de *perturbation* personnalisée de l'agent par un sous-ensemble des propriétés du monde. Comme la figure 7.2 montre, les interactions ne sont plus des instructions envoyées aux cellules par le monde, mais elles sont au contraire des actes de véritable sélection de la part des cellules. Ainsi la plasticité n'est plus une propriété essentiellement *synaptique* mais une propriété *nodale*. Mais soyons francs : une modélisation sélectionniste plutôt que instructionniste ne change guère la forme et la nature des phénomènes émergeants. Du point de vue de la conception, elle peut simplifier la tâche de programmation/construction des systèmes de contrôle à la fois algorithmiques et plastiques, de même qu'elle paraît prometteuse pour une exploration plus causale de ces mêmes phénomènes (Varela 1979). L'importance des systèmes de sélection pour l'avenir des sciences cognitives a été soulignée par Edelman (1992) qui se fait le champion des *sciences de la reconnaissance* (il y inclut l'évolution, l'immunologie et la neurobiologie) qui étudient les systèmes de reconnaissance, c'est-à-dire les systèmes ayant un générateur de diversité, un mécanisme d'hérédité et un moyen

d'amplification des événements sélectifs. Nous reprendrons la notion de la sélection dans le paragraphe 7.5.3.

Une dernière question que nous avons négligée jusqu'ici concerne la nature et l'origine des pannes :

> ***Question** : Si la plasticité est en réalité la possibilité du réseau de s'auto-organiser face aux pannes imprévues, d'où viennent ces pannes ?*

Nous postulons alors que ces pannes sont en effet le résultat d'un processus naturel de *dégradation* de l'agent (même si cette dégradation n'est pas au sens de l'usure des machines fabriquées par l'homme), c'est-à-dire d'un processus de vieillissement ou de sénescence ; nous allons analyser cette hypothèse dans le chapitre suivant et implémenter un tel mécanisme de sénescence (en effet, la force de la sénescence est une force de développement).

Nous présenterons premièrement le modèle de la cellule autonome et nous discuterons ces propriétés avant de passer à une démonstration dans un problème de navigation. Ensuite nous montrerons le besoin d'un système immunitaire comme défense contre les perturbations persistantes et nous appliquerons ces idées sur le même problème de navigation.

7.2 La cellule autonome : Motivation et socialité

Avant de passer à la description et l'analyse du modèle de cellule autonome, nous donnons une esquisse du modèle cellulaire algorithmique introduit dans le chapitre 3.

- *La cellule est l'unité élémentaire de stockage et de traitement d'information.* Une cellule représente et correspond à un composant doué d'une possibilité minimale de stockage et de traitement. Elle possède un ensemble de connexions d'entrée, un ensemble de connexions de sortie, un vecteur-mémoire statique ou dynamique et une unité de traitement qui correspond à une fonction de transfert plus la possibilité de mise-à-jour du vecteur mémoire comme effet de bord (adaptation). Les cellules sont alors des primitives structurelles différenciées par conception ou par adaptation dans un milieu particulier et définissent un substrat uniforme à partir duquel des agrégats complexes sont construits. La fonction de transfert et la taille du vecteur-mémoire peut varier de cellule à cellule.
- *Chaque cellule appartient à une catégorie fonctionnelle de cellules.* On distingue trois types de cellules, les cellules-capteurs, les cellules-actionneurs et les cellules de traitement. Les cellules-capteurs et actionneurs sont responsables des opérations primitives de perception ou de commande aux actionneurs. Les catégories des cellules de traitement selon leurs fonctions de transfert incluent : cellules *min*, *max* et *find-a-stimulus* (qui sortent respectivement l'entrée minimale, maximale ou la première non vide), cellules *timeout* et *cellules-priorité* (dont les entrées sont ordonnées de façon statique ou dynamique et celle de la plus grande priorité est sortie). Il y a en plus plusieurs types spécialisés de cellules de traitement utilisées dans des sous-structures spécialisées du réseau, telles que les *cellules-décodeurs* qui font partie des systèmes périphériques des actionneurs.

7.2.1 La cellule motivée

L'architecture cellulaire est organisée de façon hiérarchique par des couches successives de cellules qui correspondent à des étapes algorithmiques différentes. Soit le niveau cellulaire intermédiaire de la fig. 7.3. Supposons maintenant que la cellule A tombe en panne. L'algorithme ne peut continuer à fonctionner que si une autre cellule, par exemple B, entreprend le rôle de A. Cela est possible si les trois propositions suivantes sont vraies.

- B peut exécuter la fonction de transfert de A,
- B peut reconnaître et traiter les messages traités par A auparavant, et
- B peut continuer à accomplir sa fonction précédente après avoir entrepris également le rôle de A.

Pour que les deux premières propositions soient vraies, les cellules doivent inclure plus d'une seule ***pulsion*** interne (terme emprunté à la littérature de comportement animal, *drive* en anglais)[71], c'est-à-dire plus d'un composant — ces composants doivent être indépendants et correspondent aux divers rôles cellulaires. Chacune de ces pulsions doit posséder une "template" d'identification, un ***identificateur***, qui lui permettra de reconnaître les messages intéressants parmi tous ceux disponibles en entrée. Les messages doivent alors comporter deux parties : la partie d'identification et la partie donnée. La troisième proposition est vraie si les cellules peuvent contenir plusieurs pulsions exécutant en parallèle.

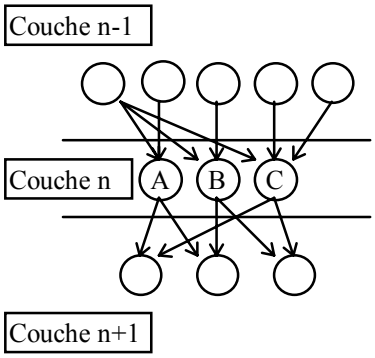

Figure 7.3 Une structure cellulaire de trois niveaux

Dans la même problématique de continuité que dans le chapitre 3 (c'est-à-dire pour contourner la rigidité des messages symboliques), nous avons défini les valeurs des deux parties comme des nombres réels et nous avons introduit un degré de résistance au bruit en ce qui concerne la reconnaissance des identificateurs. On peut adopter un analogue physique pour dire que les identificateurs reconnus par les diverses pulsions sont des ***fréquences propres*** situées dans l'échelle des valeurs permises (dans l'exemple qui suit il s'agit de l'intervalle de 0 à 1). Une cellule qui, parmi tous les messages présents en entrée, sélectionne ceux qu'elle veut traiter, est alors une cellule motivée.

---

[71] Aucun des deux termes (drive, en anglais, ou pulsion, en français) ne correspond exactement à la notion d'un composant indépendant réactif. D'autant plus, ils sont tous deux connotés avec des significations psychologiques absentes du contexte d'organisation cellulaire étudié ici. Leur utilisation ne doit donc pas confondre le lecteur puisqu'aucune analogie n'est tentée, les termes sont utilisés avec leur sens grammatical plutôt que scientifique.

Pour gérer l'éveil et l'exécution parallèle des pulsions, nous avons introduit un paramètre supplémentaire de ***productivité/vitesse*** pour chaque pulsion, qui exprime le rythme de balayage des messages en entrée. En effet, il s'agit de la vitesse de la réaction définie par la fonction de transfert de la pulsion. Une pulsion ayant une vitesse élevée équivaut à un canal de communication rapide, de manière que son exécution (réaction) est prioritaire par rapport aux pulsions moins rapides. Une conséquence de ce régime de "communication" dans le réseau est que, dans le cas de réactions à plusieurs parties, où il y a besoin de fusionner des messages de provenance de pulsions ayant des vitesses variées, la réaction dépend du message le plus tardif, c'est-à-dire de la plus lente pulsion du niveau précédent d'activité. Ainsi la vitesse totale de réponse de l'agent cellulaire est la somme des vitesses de chaque niveau, où la vitesse de chaque niveau est le minimum des vitesses de réaction des pulsions appartenant au niveau en question.

Chaque pulsion a une vitesse minimale, une vitesse maximale et un taux de modification de vitesse (augmentation ou diminution) selon la présence ou l'absence d'excitation :

$$vitesse(t+1)=vitesse(t)+s\_taux*(vitesse\_finale-vitesse(t))$$

où *vitesse_finale* est la vitesse maximale si la pulsion est stimulée (c'est-à-dire s'il existe des messages qu'elle sait consommer), sinon la vitesse minimale. Selon cette loi simple d'adaptation, la vitesse de chaque pulsion converge vers la vitesse maximale ou minimale si elle est constamment stimulée ou non stimulée, ou oscille entre ces deux limites si la stimulation est irrégulière.

Nous supposons également que la cellule possède une productivité/vitesse totale maximale, ce qui fait que des pulsions vivement stimulées et actives ayant une tendance à augmenter leur vitesse locale vont entrer en compétition pour les ***ressources cellulaires***, c'est-à-dire pour des portions de la productivité totale de la cellule. La productivité/vitesse maximale de la cellule correspond ainsi à un dépôt d'énergie (tel qu'une pile) de la cellule d'où toutes les pulsions dépendent. Ce paramètre est nécessaire pour forcer la compétition entre pulsions, sinon une pulsion stimulée pourrait s'auto-catalyser sans limites et il n'y aurait pas de possibilité de compétition. Les pulsions consomment des messages selon leurs vitesses internes, de manière que, étant donné suffisamment de variation génétique en vitesses, les pulsions les plus fortes (ou productives ou rapides) vont dominer la compétition en consommant les premières les messages qui les excitent tandis que les autres pulsions vont tomber à leur limite inférieure.

Tâchant de rendre tous les messages également disponibles à toutes les cellules, nous pouvons supposer à première vue une connectivité tous-à-tous (*all-to-all*) entre niveaux successifs : cela conduirait à des effets d'inconsistance et des goulots d'étranglement importants, puisque le même message serait généralement traité à plusieurs endroits, et le mécanisme de compétition ne fonctionnerait pas correctement. Pour pallier ce défaut nous avons supprimé les connexions entre niveaux et à leur place nous avons défini des ***buffers partagés de messages*** où les instances uniques des messages produits par les cellules du niveau précédent sont placées. De plus, les messages qui nécessiteraient vraiment d'être répliqués dans les différentes cellules ou pulsions (ceux qui dans le cas minimal algorithmique excitent en même temps plusieurs cellules) sont définis comme des ***catalyseurs***, c'est-à-dire des messages non consommés mais simplement "consultés" par les pulsions : en réalité, les pulsions font des copies locales de ces messages. En plus, et cela correspond à la notion

chimique de catalyse, les messages consommables sont ignorés si les catalyseurs ne sont pas présents.

7.2.2 La cellule sociale : Adaptation et auto-organisation

Finalement, pour permettre au réseau de récupérer de multiples pannes de plus grande échelle, nous avons introduit un paramètre supplémentaire développemental/ ontogénétique[72] qui "pousse" lentement les fréquences propres des pulsions non actives vers les fréquences propres des messages non traités. Dans les implémentations suivantes, le repère d'adaptation développementale est l'identificateur (la fréquence propre) le plus proche :

fréquence_propre(t+1)= fréquence_propre(t)
*+e_taux*plus_proche_des_fréquences_en_entrée_non_utilisées(t)*

Le paramètre *s_taux* doit être très élevé par comparaison au *e_taux* pour éviter "l'aliénation" des pulsions en cas de non excitation, c'est-à-dire pour assurer que le décalage développemental est très lent par rapport à la stabilisation des vitesses des pulsions. Ce facteur permettra aux cellules de s'auto-organiser entre elles en s'adaptant "socialement", c'est-à-dire de se stabiliser à des fréquences communes.

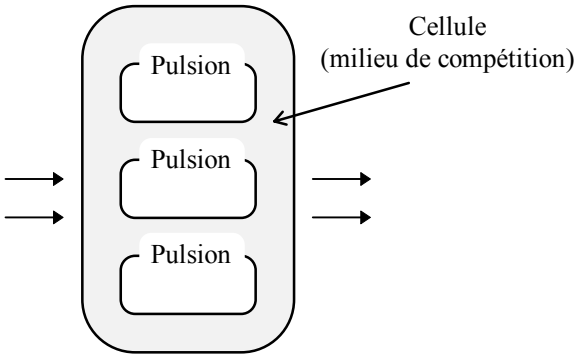

Figure 7.4 Le système de la cellule

7.2.3 Récapitulation et propriétés du modèle

On peut résumer le modèle précédent comme suit (cf. figures 7.4 et 7.5) :

- ***Les messages ont deux parties*** : une partie ***"donnée",*** utilisée par la fonction de transfert de la cellule qui traite le message, et une partie ***"identificateur",*** qui est utilisée pour la reconnaissance par les diverses pulsions candidatant pour son traitement. L'identificateur des messages en entrée est comparée par la pulsion à ses fréquences propres d'entrée pour détecter la stimulation. Par la suite nous allons donc utiliser les termes identificateur et fréquence (propre) presque sans distinction. Ce modèle démontre la propriété de ***gratuité des identificateurs***, ce qui signifie que la sémantique des connexions peut être dynamique. Il n'y a pas de connexions entre les cellules mais les messages *flottent* dans un milieu d'interaction des cellules avant d'être saisis et traités. Cela correspond à un buffer

[72] J'utilise le terme "développement" dans le même sens que Verschure & Pfeifer (1992), Almássy & Verschure (1992), pour montrer que, si l'environnement est stable, c'est-à-dire s'il ne change pas qualitativement, le réseau va se stabiliser progressivement à une configuration permanente et performante, sans croître.

partagé par niveau de connexions. Il y a deux types de messages : les ***messages consommables*** et les ***messages catalyseurs***.

- ***Chaque cellule est un système de pulsions indépendantes*** qui sont en compétition pour l'utilisation des ***ressources cellulaires*** ; la cellule a une productivité maximale bornée et constante.[73] Chaque cellule possède en plus un mécanisme physiologique ou régulateur qui est le même pour tout type de cellule (c'est-à-dire qui est indépendant de la fonction de transfert) et qui sert pour la gestion des relations de compétition entre pulsions : les "demandes" d'augmentation de vitesse sont traitées de manière proportionnelle selon la vitesse courante absolue des pulsions émettrices et sont normalisées selon la vitesse maximale de la cellule. Notons que les besoins de l'agent sont liés à la consommation de ces messages : un agent veut consommer aussitôt que possible les messages qui le perturbent, donc les besoins équivalent à *volume_de_messages_en_entrée=0*. Puisque, contrairement à l'agent explorateur du chapitre 4, il n'y a aucune nécessité de reconnaître la disparition des besoins et l'accomplissement de la mission, ces derniers ne sont alors pas explicitement décrits dans le système de contrôle de l'agent.

- Chaque pulsion est excitée par des messages consommables ou catalyseurs dont les identificateurs sont identiques (ou correspondent) aux fréquences propres, c'est-à-dire chaque pulsion ne reconnaît que les messages qui portent des identificateurs correspondant à ses fréquences propres.[74] Lorsqu'une pulsion est excitée, elle exécute sa fonction de transfert qui peut être vue comme une ***fonction métabolique*** transformant les messages d'entrée en messages de sortie. Il est important de souligner que ***le comportement "motivé" de la cellule émerge de la compétition d'un ensemble de composants "réactifs" indépendants, les pulsions***.

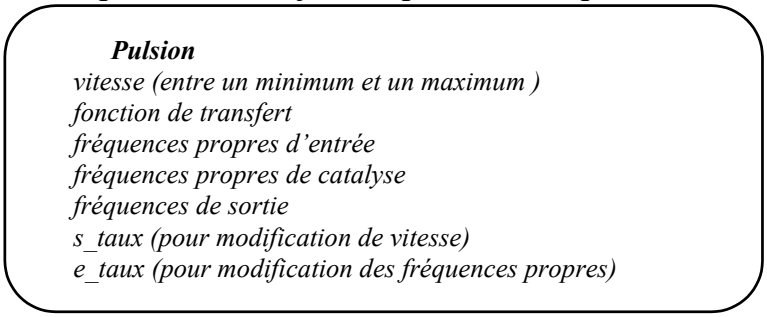

Figure 7.5 Le modèle de pulsion

- ***Les pulsions se catalysent elles-mêmes quand elles exécutent***, en augmentant leur productivité locale jusqu'à un maximum tant que leur stimulation persiste. Leur productivité baisse avec le même taux quand elles ne sont pas stimulées.[75] Ce mécanisme de modification de vitesse suit une dynamique *propre* à la pulsion et *rapide* par rapport à la dynamique de la tâche (c'est-à-dire par rapport à la

[73] Dans une future version, je compte introduire un facteur de dégradation de cette productivité maximale (cf. la fonction de sénescence dans le chapitre suivant).

[74] Dans l'exemple qui suit, toutes les fréquences d'entrée/sortie d'une pulsion sont identiques. Ceci n'est qu'une particularité de l'exemple en question ; en principe, le modèle cellulaire n'impose pas de telles contraintes.

[75] Pour la définition et simulation de ce modèle de cellule motivée et sociale, j'ai été beaucoup inspirée de la description par Monod (1970) du mécanisme de régulation catalytique des vraies cellules biologiques qui se fait à l'aide de protéines allostériques.

dynamique du fonctionnement du réseau entier) : contrairement à la situation du chapitre 4, ici il ne serait pas possible de trouver un critère de couplage de la dynamique de la pulsion à celle de la tâche, car cette dernière concerne tout le réseau et est alors *émergente*, ce qui fait que la dynamique propre est la seule solution possible. Dans ce qui suit, nous adopterons souvent un analogue chimique et nous nous référerons aux vitesses locales des pulsions comme des ***vitesses de réaction***. Le mécanisme d'autorégulation des vitesses de réaction est donc responsable de la *différenciation temporaire* des cellules face aux messages en entrée.[76]

- Un facteur supplémentaire développemental/ontogénétique pousse lentement les fréquences propres non excitées vers la plus proche des fréquences propres des messages non traités. Le taux de cette modification est extrêmement bas, afin d'éviter l'aliénation des pulsions. Ce facteur développemental induit un ***comportement social*** des cellules, puisque l'adaptation des fréquences propres se fait par rapport aux fréquences présentes dans le milieu d'interaction (dans les buffers) et alors par rapport aux fréquences provenant des autres parties du réseau.
- Conformément à la définition donnée dans le chapitre 1, ***la physiologie de la cellule*** est l'ensemble des mécanismes qui relient le "programme" de la cellule (c'est-à-dire ce qu'elle cherche à faire) avec son métabolisme (qui "implémente" ce programme). Le système de compétition entre pulsions, l'auto-catalyse et l'adaptation des pulsions sont donc les fonctions physiologiques de la cellule.
- Comme on le verra dans le paragraphe suivant, la robustesse du système aux pannes et au bruit est principalement due à la présence de potentiellement grandes ***variations génétiques*** dans les variables qui expriment les fréquences propres, les vitesses et les taux d'adaptation *("The real cause of stability in a distributed system is sufficient diversity"*, Hogg & Huberman (1993)).

7.3 Illustration : Apprendre à naviguer

7.3.1 L'algorithme de navigation

Nous avons démontré en simulation le potentiel auto-organisationnel du modèle pour le cas d'un système de navigation d'un robot autonome équipé de 4 capteurs d'obstacle et de quatre capteurs de trace et devant se diriger vers un point but donné en suivant une trace sur le sol, si cette trace existe.[77] Le monde simulé du robot est une grille bidimendionnelle, par conséquent le robot simulé a seulement quatre directions possibles de navigation (devant, derrière, à droite et à gauche). Ainsi, chacun des quatre capteurs d'obstacle et chacun des quatre capteurs trace correspond et est dédié à une de ces quatre directions de navigation.

---

[76] Morin (1980) trace l'analogie entre la rétro-différenciation cellulaire et la déspécialisation/ respécialisation dans les sociétés d'insectes. Par exemple, le modèle comportemental éthologique de Drogoul (1993) repose sur un mécanisme de spécialisation/déspécialisation dynamique par modification des diverses "motivations" des agents-fourmis. *"L'aptitude à la déspécialisation, là où elle se manifeste, est une qualité individuelle proprement régénératrice bénéfique à la communauté"* (Morin, ibid., p. 307).

[77] Ce problème a été étudié dans le cadre de l'étude du chapitre 6 et les contraintes robotiques ont été posées en collaboration avec Philippe Darche dont le robot Pedro (Darche 1994) a servi de base de réflexion.

Le système de contrôle algorithmique est donné dans la figure 7.6 et fusionne l'information sur le but avec l'information des capteurs de trace et d'obstacle. Nous supposons que la direction du but est fournie par d'autres parties du réseau (les systèmes de traitement des tâches, tels qu'ils ont été décrits dans le chapitre 3). Dans la figure 7.6, les quatre cellules "diff" de gauche comparent la direction d'entrée à la sortie du capteur correspondant et donnent une valeur de saturation si un obstacle est perçu ; les quatre cellules "diff" de droite font la même chose mais pour les capteurs de trace. Les cellules du niveau suivant calculent la somme des différences fournies par le niveau plus haut ; cette somme exprime la déviation totale de la direction correspondante depuis la direction désirée et celle de la trace. À ce niveau, il faut pouvoir éliminer tout d'abord les directions bloquées par un obstacle et ensuite favoriser celles où il existe une trace. Finalement, la cellule-décodeur qui gère le moteur choisit parmi les messages du niveau précédent celui qui propose la déviation minimale. Pour ce faire, les différentes fonctions de transfert sont définies comme suit :

capteur_obstacle (dir) : si un obstacle est perçu, alors valeur_de_saturation, sinon dir
capteur_de_trace (dir) : si une trace est perçue, alors dir, sinon dir + punition
*diff : si la différence est inférieure ou égale à 0.8 * valeur_de_saturation, alors différence, sinon valeur_de_saturation*
sommation : faire la somme numérique
*décodeur : éliminer les entrées supérieures à 0.8 * valeur_de_saturation, ensuite choisir la minimale*

Les valeurs de saturation de capteur d'obstacle et de punition sont 1000 et 5 respectivement.

Une première version de cet algorithme de navigation utilisait un seul capteur de trace qui donnait la direction de trace la plus favorable (devant, droite/gauche ou derrière, dans l'ordre) qui était ensuite comparée aux capteurs d'obstacle. Cette version n'était pas opérationnelle, parce que l'agent en arrivant au bout de la trace faisait demi-tour et retournait sur ses pas (Resnick & Martin (1990) ont eu le même résultat avec un programme symétrique de suivi de ligne sur un robot Lego). Pour résoudre ce problème, il aurait fallu un capteur de trace qui serait aveugle à la présence de trace derrière, auquel cas la trajectoire de l'agent serait en revanche sous-optimale pour un point-but derrière. Cela a révélé qu'il y avait besoin de comparer la direction d'entrée (but) avec *chacune* des directions où une trace pourrait être perçue.

Les messages qui voyagent aux connexions sont des nombres réels : les directions prennent une des quatre valeurs 0.5, 0.25, 0.75, 1 qui correspondent au devant, à gauche, à droite et derrière respectivement. La valeur de saturation est définie comme 1000, afin que les messages qui arrivent au décodeur et dont les valeurs dépassent 1 puissent être éliminés.

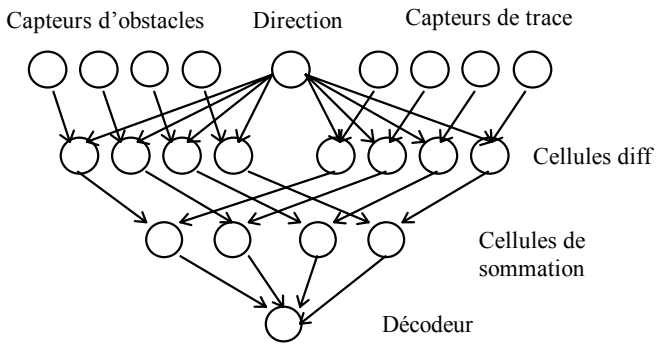

Figure 7.6 Le système algorithmique de navigation

Le réseau auto-organisant correspondant est visualisé dans la figure 7.7. Les ensembles des connexions entre les trois premiers niveaux ont été remplacés par des buffers partagés, tandis que les connexions entre les cellules de sommation et le décodeur sont restées intactes (parce que nous n'avons pas introduit des cellules décodeurs redondantes).

Nous avons adopté comme des fréquences propres les mêmes valeurs utilisées pour la discrimination des directions : 0.5, 0.25, 0.75 et 1 (c'est-à-dire, toutes les pulsions traitant les messages de devant ont comme fréquence propre 0.5, et de même pour toutes les autres directions de navigation). Ce schéma présente un avantage dans notre cas : la direction envoyée au moteur par le décodeur est la fréquence propre du message choisi, c'est-à-dire du message qui correspond à la déviation minimale (nous rappelons quand même que la fréquence du message et la donnée ne sont généralement *pas* corrélés). Les pulsions des cellules "diff" tout comme celles des cellules de sommation nécessitent deux ***identiques*** fréquences propres d'entrée, afin de fusionner deux messages en entrée en prenant la différence ou la somme respectivement. Pour éviter que des messages obsolètes arrivent au décodeur, chaque fois qu'un mouvement est fait tous les messages dans les buffers partagés ou privés sont "tués" (cela correspond à une entrée supplémentaire de *"reset"* de cellule).

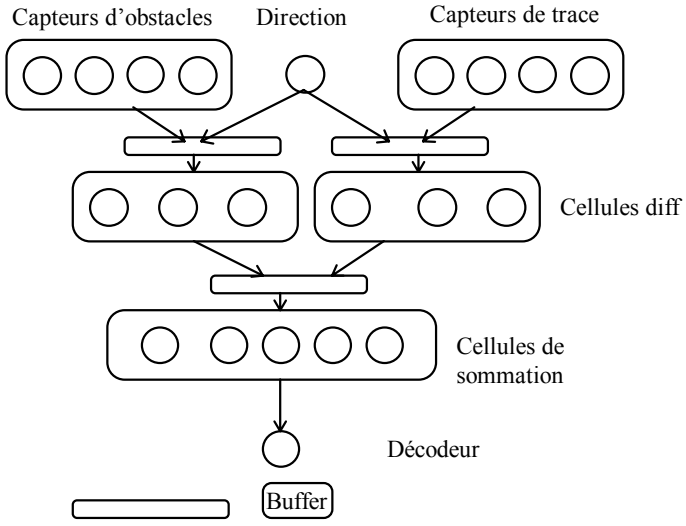

Figure 7.7 Le système auto-organisant de navigation

Puisque le message de direction doit être partagé entre toutes les cellules "diff", il a été défini comme un message catalyseur. Dans notre cas de navigation, cela signifie que les messages des capteurs d'obstacle et de trace sont ignorés s'il n'existe pas de

direction donnée de but. Cet algorithme d'attraction-par-un-but-en-suivant-une-trace est conçu pour le cas d'obstacles statiques, c'est-à-dire dans un monde inconnu mais pas hostile. Par construction, l'algorithme est tel que les directions (fréquences propres) pour lesquelles il n'existe pas de messages sont ignorées ; si aucun des messages qui concernent le côté gauche n'arrive au décodeur à temps, il n'y aura aucun mouvement vers la gauche, même si c'est permis.[78] Dans ce cas, l'agent peut faire des détours ou zigzaguer (par exemple, un agent avec le but d'aller tout droit mais ne répondant pas à la direction de devant, fera un zigzag pour y arriver).

Un dernier point concerne le problème de l'espace de représentation et sa métrique (cf. paragraphe 3.6). Le problème de cet algorithme de navigation est que l'espace des directions doit être un anneau de façon que l'opération de la soustraction donne, par exemple, *devant-droite=devant-gauche*, ce qui n'est pas possible avec les nombres réels qui sont un espace linéaire et ouvert. De plus, cette propriété cyclique doit se conserver pendant la transposition de l'origine, par exemple 5+0.75 doit être le même que 5+0.25. Effectivement, nous avons résolu ce problème en remplaçant les différences de 0.75 par 0.25, puisque seules les valeurs de 0, 0.25 et 0.5 sont permises et celle de 0.75 équivaut à 0.25 si l'on parcourt l'anneau au sens inverse. Cependant, le problème est plutôt un problème conceptuel qu'un simple problème d'implémentation : ***la forme de l'espace de représentation doit être couplée avec celle de l'interaction agent-monde***, pour ne pas dire la même. Si le problème nécessite une configuration des capteurs ayant la forme d'un anneau, les opérations primitives doivent être celles de l'anneau. La solution est de définir comme espace de représentation un espace mathématique ayant toutes les propriétés nécessaires. Il est certainement possible de définir un tel espace à base de l'espace des nombres réels (comme nous l'avons fait ici), mais, en vue d'une implémentation matérielle, il serait peut-être plus pratique de trouver un espace matériel ayant par définition toutes ces propriétés désirées. Justement, il me semble que la chimie organique est un espace de représentation très riche en propriétés telles que la ***symétrie***, la ***circularité*** et la ***récursivité*** et les réactions chimiques sont des opérations dans cet espace qui peuvent devenir favorablement complexes. En tous cas, les considérations de métrique et de topologie méritent toute l'attention des chercheurs en vie artificielle : quelle est l'espace (mathématique) de représentation le mieux adapté à un problème donné et quelles sont ces propriétés minimales ? Jusqu'à maintenant il n'y a eu que des efforts isolés et partiels : Thomas Ray utilise un mécanisme d'adressage par template inspiré de la biologie moléculaire (Ray 1991) et souligne la géométrie non euclidienne des ordinateurs (Ray 1994), tandis que Kephart (1994) insiste sur l'importance des formes topologiques pour les dynamiques des populations.

7.3.2 Réseaux sains

Nous avons simulé le fonctionnement du système de navigation pour 3 à 5 cellules dans chacun des ensembles "diff" et sommation. Chaque cellule était équipée de 1 à 3

[78] En effet, le décodeur n'a par défaut aucun critère pour savoir si tous les messages disponibles sont arrivés (il se peut qu'il n'existe pas de message correspondant à une direction particulière). C'est alors la seule cellule où il peut y avoir d'activité avec un nombre d'entrées variable (de 1 à 4). Pour ce permettre, nous avons introduit une variable qui exprime l'espace maximal permis dans le temps pour les messages en entrée : si tous les messages (4) sont arrivés ou si la durée maximale s'est écoulée depuis l'arrivée du premier message, la réaction a lieu avec les entrées présentes. Dans les simulations, cette variable est initialisée à une valeur aléatoire entre 0.3 et 0.5.

pulsions avec des fréquences propres choisies parmi les 4 fréquences fondamentales. Les vitesses minimales et maximales ont été prises au hasard entre (0, 0.2) et (0.8, 1) respectivement, tandis que les taux d'adaptation de vitesse et de fréquence propre ont pris des valeurs aléatoires entre (0, 1) et (0, 0.05) respectivement. Une trajectoire typique de l'agent est visualisée dans la figure 7.8. Dans toutes les expériences, et gardant un taux constant et uniforme d'échantillonnage des capteurs d'obstacles et de traces, le réseau s'est stabilisé rapidement (dans 10-20 cycles) à une configuration où 2 ou 3 cellules dans chaque ensemble étaient actives, c'est-à-dire qui avaient des pulsions exécutant à des vitesses près de leur maximum, avec le reste des cellules étant dormantes, c'est-à-dire à vitesse minimale. L'auto-catalyse des vitesses des pulsions dominantes conduit à une auto-catalyse de la vitesse totale de réponse comme la figure 7.9 montre.[79] La courbe de l'évolution de la vitesse d'une cellule du type "diff" est donnée dans la figure 7.10 : on voit que, tant que le réseau est stimulé, c'est-à-dire tant qu'une direction de but est fournie au réseau, les cellules stimulées se catalysent positivement et elles essaient de se stabiliser à leur vitesse maximale, mais lorsque la stimulation se retire — ici c'est parce que le but est atteint et la cellule de calcul de direction de navigation ne donne plus de message à sa sortie — les cellules se catalysent négativement et leurs vitesses de réaction tombent de nouveau à leur minimum. Finalement, la figure 7.11 donne les vitesses de réaction de deux pulsions appartenant à la même cellule du type diff et montre la compétition entre les pulsions qui conduit à une stabilisation *partielle*, c'est-à-dire à une configuration des vitesses dans laquelle aucune des deux pulsions n'a atteint sa vitesse maximale, mais les deux pulsions ont trouvé une sorte de "compromis" à ce sujet.

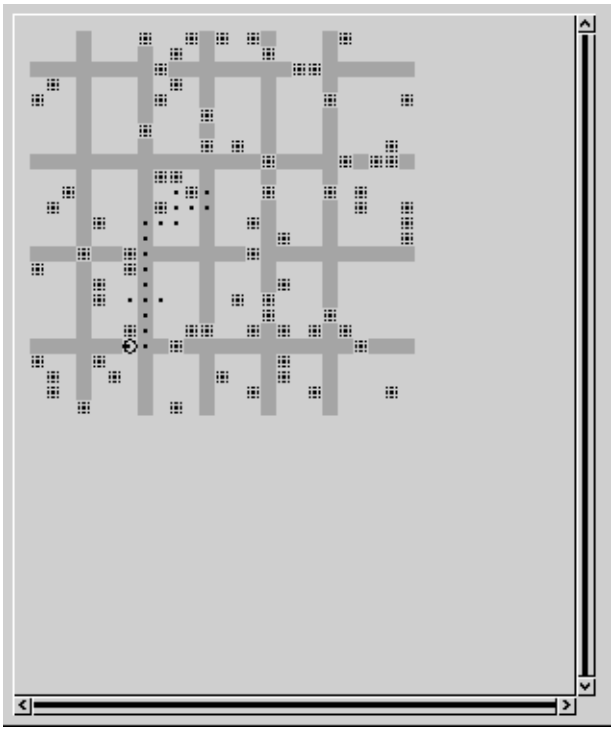

Figure 7.8 L'agent dans son univers de navigation. Suivi des traces avec évitement simultané des obstacles. La trace est la partie plus foncée du "sol" et les obstacles sont les

[79] Cette vitesse totale est calculée comme la somme des vitesses des réactions des différents niveaux, où la vitesse d'un certain niveau est en effet la vitesse minimale de toutes les réactions qui y ont lieu (cf. paragraphe 7.2.1).

petites grilles. La ligne pointillée est la trajectoire de l'agent. Les quelques déviations de l'agent de la trajectoire "optimale" sont dues à la présence d'un degré de bruit (ici 10%) dans la génération de la direction de voyage qui fait que la direction choisie n'est pas toujours la direction localement optimale ; ce bruit permet à l'agent de ne pas se bloquer dans des impasses.

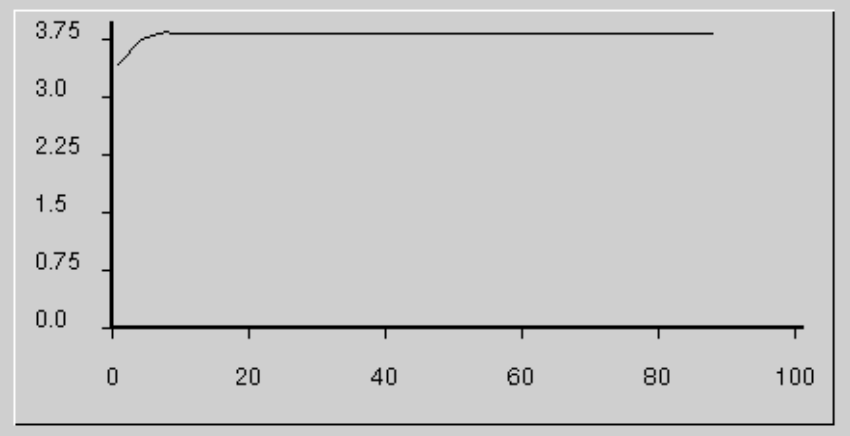

Figure 7.9 Stabilisation de vitesse totale de réponse. Cette vitesse est en effet la vitesse totale de réaction de l'agent, c'est-à-dire une mesure interne de l'agent, et ne correspond pas nécessairement à une vraie vitesse telle qu'elle serait perçue par un observateur externe (cf. paragraphe 7.2.1).

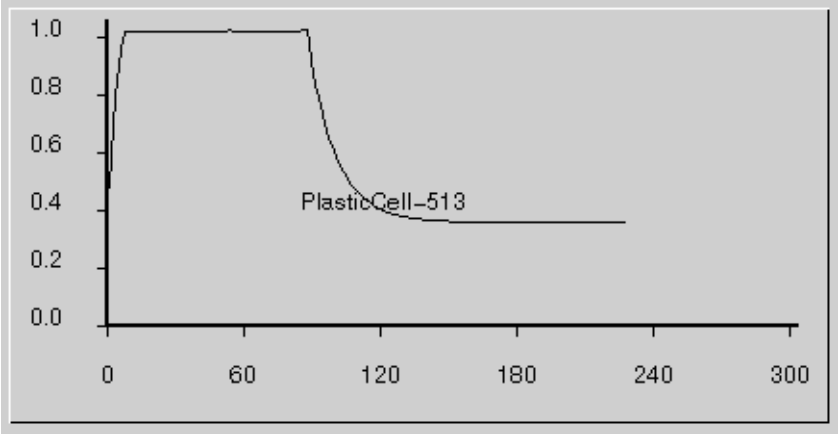

Figure 7.10 Évolution de la vitesse d'une cellule du type "diff". Dans la première partie de la courbe la cellule est stimulée, alors sa vitesse monte à son maximum ; dans la deuxième partie la cellule n'est plus stimulée et sa vitesse retombe à son minimum.

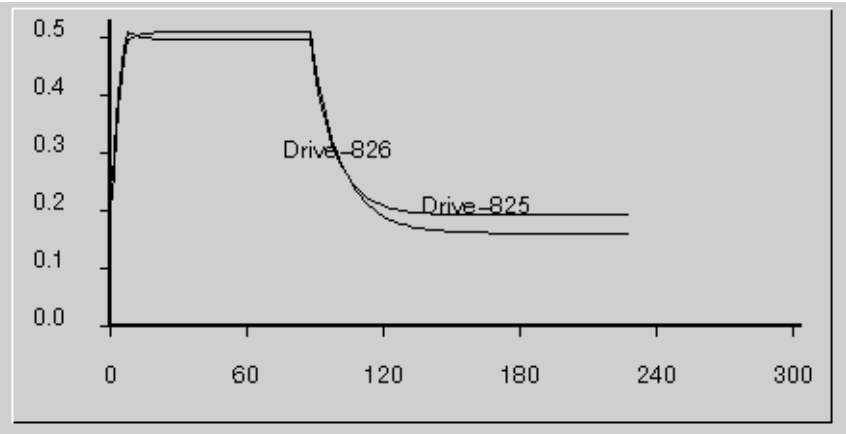

Figure 7.11 Compétition des vitesses des pulsions de la cellule de la figure 7.10. Les vitesses des deux pulsions ne sont pas leurs maximales, elles sont loin de 1.

7.3.3 Des lésions légères

Ensuite l'expérience suivante a été réalisée : une fois le réseau s'est stabilisé à une configuration des pulsions (à t=100), nous avons enlevé une des cellules diff qui contenait des pulsions actives. Encore une fois, il a fallu à l'agent seulement quelques cycles pour se re-stabiliser à une nouvelle configuration ; en fait, *le processus d'auto-organisation n'est guère distinct de l'activité normale du réseau*, alors le réseau n'a pas arrêté pour apprendre la nouvelle configuration avant de continuer. Au lieu de cela, le système de navigation a continué de fonctionner comme avant, même si la direction affectée par la blessure n'arrivait plus au décodeur à temps. Il a fallu quelques cycles de plus avant que le réseau redevienne sensible aux messages correspondant à la direction affectée. La variation des vitesses minimales et maximales des pulsions, ainsi que de leur configuration initiale, fait que le réseau ne perd pas toujours la direction affectée par une blessure comme celle-ci, mais assez souvent reste sensible à la direction malgré la vitesse sous-optimale (figure 7.12a). Une autre remarque pertinente est que la physiologie de la cellule est telle que, si soudain une pulsion inactive devient active à cause de stimulation, les pulsions actives vont trouver un compromis de vitesses qui permettra la re-stabilisation de la vitesse globale à une valeur plus opérationnelle. Ainsi dans la figure 7.12b, lorsqu'une pulsion devient active, la deuxième pulsion active de la même cellule "cède" une fraction de sa vitesse pour trouver un compromis de vitesse avec la première.

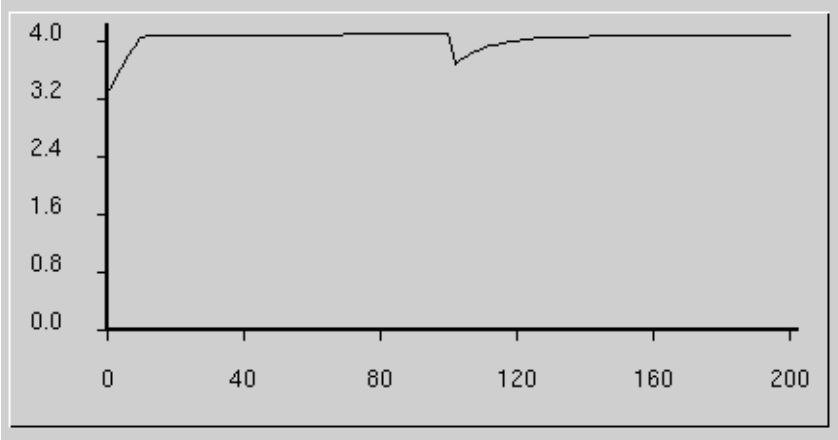

Figure 7.12a Re-stabilisation de la vitesse totale. À t=100 coupure d'une cellule diff qui traitait de la direction de droite (0.25). La vitesse totale baisse après la blessure et se re-stabilise plus tard (t=200).

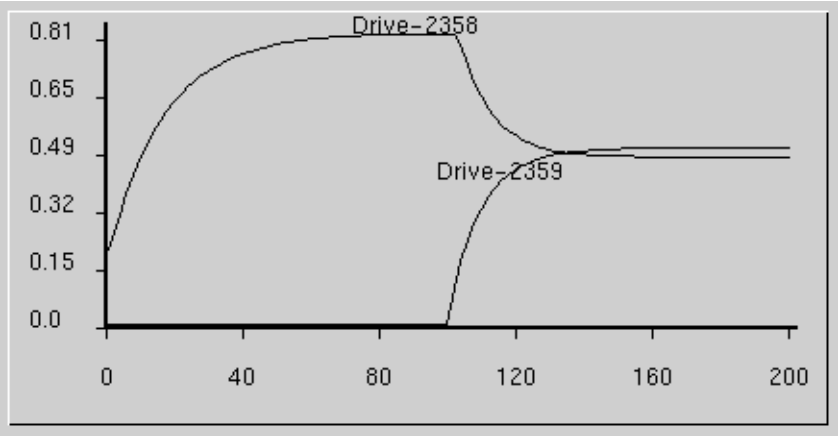

Figure 7.12b Pour l'expérience de la figure 7.12a, re-stabilisation des vitesses de deux pulsions après blessure (t=100) et récupération.

7.3.4 Une grave blessure

Ensuite, nous avons enlevé toutes les pulsions du niveau "diff" qui étaient excitées par une fréquence propre donnée (en effet, toutes les pulsions d'un des deux ensembles de cellules "diff"). Comme avant, le système de navigation est devenu "aveugle" à la fréquence disparue, mais cette fois il a fallu quelques 50 cycles avant qu'un nouveau chemin soit découvert (fig. 7.13), c'est-à-dire avant qu'une des pulsions non utilisées (en effet, la plus instable/changeante!) se stabilise à cette fréquence (fig. 7.14). Cela constitue une conséquence attendue du fait que le *e_taux* est faible, qui nécessite une stimulation persistante du réseau pour assurer la re-convergence vers une configuration stable. À noter également que la vitesse totale de l'agent a baissé après la blessure, parce que les cellules restantes ont dû re-distribuer leur productivité parmi des pulsions différentes.[80]

[80] J'ai voulu trouver un modèle de stabilisation qui rappelerait ce qui se passe dans les organismes vivants, tels que les animaux, où après récupération des blessures du système nerveux les performances

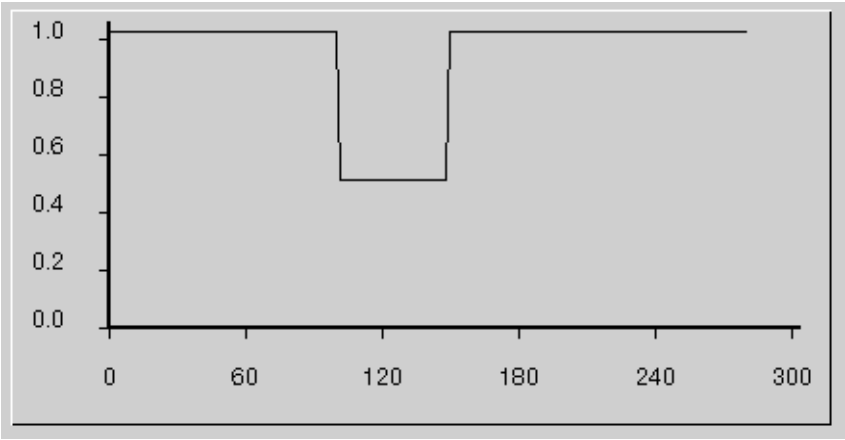

Figure 7.13 La réponse du réseau à une direction avant et après blessure (1 si le réseau répond à la direction, 0.5 sinon). À t=100 la seule cellule qui répondait à la direction de devant (fréquence=0.5) a été tuée. Il a fallu quelque 50 cycles pour que le réseau retrouve cette direction.

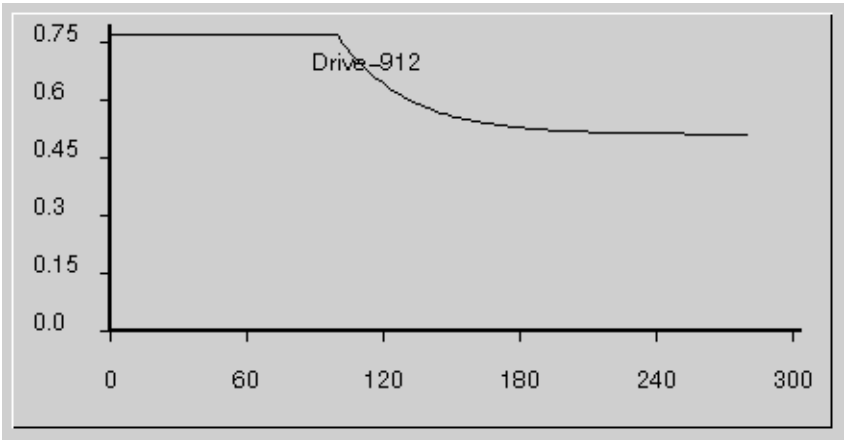

Figure 7.14 Dans l'expérience de la figure 7.13, direction de réponse (ou identificateur de réponse) de la pulsion qui a pris le rôle de devant (avec adaptation développementale continue, cf. paragraphe 7.3.5).

7.3.5 Besoin d'adaptation développementale continue

L'adaptation dite "développementale", c'est-à-dire l'adaptation des fréquences propres d'une pulsion, doit être continue, c'est-à-dire la pulsion doit adapter ses fréquences propres quand elle est stimulée ainsi que quand elle ne l'est pas. La différence est le repère d'adaptation : quand la pulsion est stimulée, elle doit adapter ses fréquences propres par rapport aux fréquences qui l'ont stimulée, tandis que, quand elle n'est pas stimulée, elle doit adapter ses fréquences propres par rapport aux fréquences présentes les plus proches aux siennes. Cela permet aux pulsions de se stabiliser aux fréquences persistantes de manière à permettre une co-stabilisation des pulsions des différents niveaux dans le réseau, c'est-à-dire de manière à permettre *la construction d'un standard social ou d'un langage commun dans le réseau*. Pour montrer le rôle de l'adaptation développementale continue, nous avons mené

sont baissées en termes de vitesse de réponse (mon étude de blessures ressemble celle menée par Beer (1990))

l'expérience suivante : nous avons laissé fonctionner un réseau pour 300 cycles sans adaptation continue (c'est-à-dire avec adaptation développementale seulement en absence de stimulation) et à t=80 nous avons tué une pulsion stimulée du niveau diff. À t=100 environ le réseau s'est stabilisé à une nouvelle configuration. Alors à t=160 nous avons tué une deuxième pulsion stimulée : nous avons choisi la pulsion du niveau de sommation qui correspondait à la fréquence affectée pendant la première blessure (nous avions prévu de ne pas avoir d'autres pulsions avec la même fréquence propre pour forcer l'adaptation développementale). Nous avons laissé re-stabiliser le réseau et nous avons constaté que le réseau n'était plus sensible à la direction qui correspondait à la fréquence affectée par les deux blessures. Que s'est-il passé ? Les pulsions ayant repris les rôles de celles manuellement tuées se sont stabilisées chacune séparément à une fréquence près de celle en entrée (la pulsion diff s'est stabilisée au 0.45 et la pulsion de sommation au 0.55[81]) et n'en ont plus changé. Ainsi, au niveau de sommation la "fusion" des deux fréquences ne pouvait plus avoir lieu, la différence entre les deux étant trop grande par rapport au seuil de comparaison (matching) des identificateurs qui est autour de 0.05 (rappelons que les fréquences d'entrée et de sortie des pulsions diff sont identiques, donc la pulsion diff produit un message avec une fréquence de 0.45, alors que la pulsion de sommation attend des messages avec une fréquence autour de 0.55). La solution était de permettre aux pulsions de s'adapter *continûment* aux fréquences en entrée : à ce point (t=300), nous avons alors modifié le comportement adaptatif des pulsions et nous avons défini l'adaptation développementale comme continue ; le résultat était que, cette fois, les pulsions dont les fréquences étaient dissociées avant, se sont stabilisées de nouveau à des fréquences beaucoup plus près de celles en entrée (la première s'est stabilisée au 0.495 et la deuxième au 0.505) de manière à pouvoir être fusionnées au niveau de la sommation. Les résultats de cette expérience sont donnés dans les figures 7.15 et 7.16.

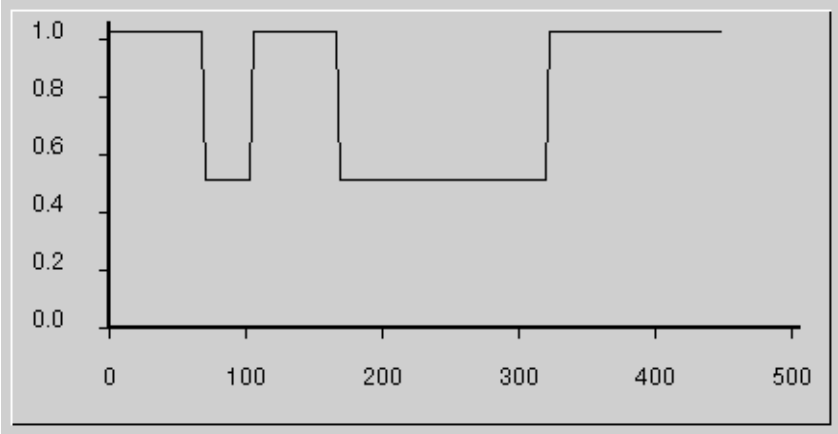

Figure 7.15 Besoin d'adaptation continue : réponse à la direction de devant (0.5). Première partie (jusqu'à t=300) sans adaptation continue. Ainsi, avec une blessure d'un seul niveau (diff), le système se re-stabilise, mais avec une deuxième blessure au niveau de sommation cela se bloque (voir texte). Dans la deuxième partie, nous avons ajouté de l'adaptation continue. Environ 30 cycles plus tard, le réseau retrouve la direction de devant (fréquence de cellule diff = 0.499, fréquence de cellule de sommation = 0.505)

[81] La pulsion diff s'est adaptée par rapport à la fréquence du message envoyé par le capteur correspondant, tandis que la pulsion de sommation s'est adaptée par rapport à la fréquence du message envoyé par le deuxième niveau diff qui est resté intacte tout au long de cette expérience.

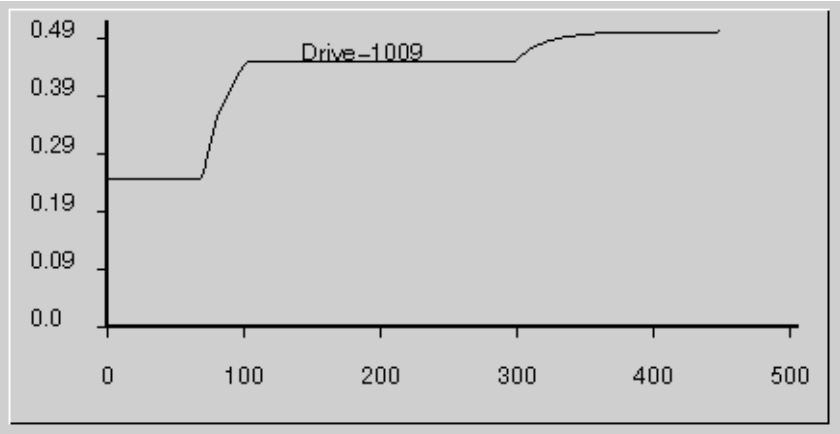

Figure 7.16 L'évolution de la fréquence propre de la pulsion "diff" qui s'est re-stabilisée dans l'expérience de la figure 7.15.

7.3.6 Discussion du modèle et des résultats

Le modèle introduit de cellule motivée constitue une extension du modèle d'unique et uniforme fonction de transfert présenté brièvement dans le paragraphe 2 et diffère des modèles classiques de neurone en deux aspects fondamentaux :

- Il n'y a pas de connexions entre cellules, mais les cellules "flottent" dans un milieu commun et partagent des buffers de messages avec des autres cellules. Cela permet et à la fois nécessite une compétition entre cellules pour la consommation des messages présents en entrée.
- Les cellules répondent aux messages à leur propre rythme et selon des motivations/ préférences locales. Elles ne sont pas de simples "machines à input" et alors des machines manipulables, mais possèdent un degré d'autonomie individuelle qui se manifeste comme une sélectivité envers les messages en entrée. ***La sélectivité est couplée avec un mécanisme de régulation des vitesses de réaction, c'est-à-dire avec un mécanisme de modification de la dynamique d'interaction avec le monde.***

Tout comme les modèles connexionnistes traditionnels cependant, les cellules s'adaptent de manière continue et en parallèle avec l'activité, c'est-à-dire les phases d'apprentissage et d'action ne sont pas séparées (non seulement l'agent s'adapte sans un superviseur, mais il n'a aucune idée qu'il s'adapte). La mémoire du système est à moyen terme, par conséquent le système ne manifeste pas les problèmes habituels d'oubli et de mémoire à court terme. Si jamais il doit oublier, il oubliera! Ce processus d'adaptation repose sur des paramètres endogènes et ne dépend pas d'un maître bénévole qui aurait à préparer les données d'apprentissage, choisir le langage et le niveau d'abstraction des représentations (van de Welde (1991) a insisté sur le besoin de repenser sérieusement ces questions). Si on utilise les critères des performances de Kaelbling (1991), l'organisation présentée est assurée de *converger* à une configuration *correcte* en utilisant un mode d'adaptation d'une *complexité spatio-temporelle constante*. N'oublions pas, pourtant, que l'agent n'apprend rien de nouveau sur son environnement, mais seulement comment mieux utiliser son potentiel

inné ; une étude de dimensionnement aux problèmes d'apprentissage, topologique ou autre, reste à faire.

Ce modèle laisse de la place à l'introduction des *variations génétiques* et à l'émergence d'une division de travail entre cellules qui en résulte. En plus, un tel réseau de cellules motivées reste largement compréhensible et dans le cas de pannes majeures un ingénieur peut intervenir directement au niveau de la pulsion et mettre à jour le système manuellement, puisque le caractère algorithmique de l'activité/traitement de messages est conservé. Dans ce sens, *la variation permet l'auto-organisation qui assure la stabilité qualitative, c'est-à-dire l'opérationalité algorithmique*. En outre, la nature continue tant des fréquences propres que des données permettra de tenter une analyse dynamique de tels réseaux. En principe, les taux d'adaptation et les vitesses minimales/maximales pourraient être variables au cours de la vie de l'agent ; je peux imaginer des taux décroissants d'adaptation qui rendront l'agent de moins en moins robuste aux pannes et des limites de vitesse décroissantes qui rendront l'agent de moins en moins productif (cf. chapitre suivant).

Le comportement temporel du réseau est tel qu'il est assuré de converger à une configuration stable en cas de stimulation continue et de toute façon il "suivra" tout régime "capricieux" de stimulation en accord avec ses motivations internes. Le comportement cellulaire auto-catalytique rend ainsi l'agent paresseux (il travaille à vitesse faible) en l'absence de perturbations importantes, mais permet à sa productivité de monter rapidement quand celui-ci est stimulé. Ce comportement autorégulant de la cellule démontre une fois de plus l'importance du temps comme un paramètre de conception (d'autres observations quant au rôle du temps peuvent se trouver dans la littérature des réseaux de neurones dynamiques, par exemple (Husbands et al. 1993), (Yamauchi & Beer 1994)). Tant que le réseau fonctionne sous des conditions initiales aléatoires ou peu après l'occurrence des blessures, le comportement de l'agent apparaît comme non coordonné et sous-optimal.

7.4 Vers des systèmes de contrôle immunitaire

Nous avons vu que le mécanisme d'adaptation développementale permet au réseau de retrouver une voie de stimulation même dans le cas de pannes majeures. En effet, les cellules s'adaptent aux fréquences des messages en entrée *sans* décider de la légalité de ces messages. Cela peut conduire à une aliénation du réseau dans le cas où ces messages sont plus ou moins faux, par exemple dans le cas ou les fréquences de ces messages ont subi des "mutations". Si ces mutations sont persistantes, les pulsions se stabiliseront à des fréquences détrimentales pour l'intégrité du réseau. Que fait-on ? La solution est d'équiper le réseau d'une possibilité de différencier les messages permis de ceux non permis, c'est-à-dire d'y introduire un système immunitaire[82] :

- Les fréquences étrangères doivent être reconnues par le réseau et les messages porteurs éliminés.

[82] Pour le développement du modèle immunitaire rudimentaire qui suit, j'ai été inspirée d'une description des mécanismes et des théories immunitaires (Golub & Green 1991) et plus particulièrement de la théorie de sélection clonale. Le terme "template d'identification" risquant de faire croire au lecteur qu'on est conforme à la théorie instructive de templates (ibid.), je préfère utiliser par la suite exclusivement le terme "fréquence propre" qui montre qu'il s'agit d'un système sélectif.

Il est facile d'imaginer que cette situation correspond à un *jeu de coopération* : les fréquences "intrus" doivent être reconnues comme différentes de celles légales, et alors détrimentales, pour être éliminées. Cette observation conduit à la vision du comportement social de la cellule comme un comportement fondé sur des mesures de similarité (*genetic kinship*, on dirait en langage de théorie d'évolution). Pour ce faire, un autre type de cellules a été introduit, les cellules défenseurs ayant comme rôle de reconnaître et d'éliminer les messages porteurs de fréquences étrangères. À noter que l'idée initiale de défenseurs qui "connaîtraient" les bonnes fréquences et élimineraient celles ayant une faible affinité avec les premières n'est pas fonctionnelle, car, étant donné la présence de plusieurs fréquences dans le réseau, une fréquence détectée comme étrangère par un défenseur pourrait très bien être valable du point de vue d'un autre et le réseau serait instable. L'alternative est alors d'avoir des ***défenseurs qui reconnaissent directement les mauvaises fréquences*** pour les éliminer : ces mauvaises fréquences sont en réalité les fréquences complémentaires aux fréquences des cellules productrices (c'est-à-dire des cellules qui ont été présentées dans les paragraphes précédents). Cette formulation du problème de la reconnaissance a deux propriétés importantes :

- Le modèle cellulaire reste sélectif. Les cellules défenseurs suivent le même modèle cellulaire que les cellules productrices, à la seule différence fonctionnelle qu'elles "avalent" les messages qui les excitent sans les métaboliser.
- Le réseau comporte alors ***deux forces sociales*** complémentaires, la force de "production", qui est responsable de l'accomplissement de la tâche de l'agent, et la force de "supervision" ou force "policière", qui est responsable de la détection et élimination des éléments étrangers et nocifs pour la même tâche.

Pour modéliser ce comportement social cellulaire, nous avons utilisé le modèle tit-for-tat quantitatif (présenté dans l'annexe A) selon lequel un stimulus social est considéré coopératif (défectif) si sa valeur est supérieure (inférieure) à un certain seuil ($f(stimulus) \geq T \Rightarrow coopération$). Ce modèle permet la description de tout un éventail de relations sociales — des relations de *participation*, ainsi que des situations de jeu — avec des variations dans les paramètres (tels que le seuil T) ainsi que dans les structures (telles que les fonctions de perception et de coopération).

Le comportement social des cellules du système de contrôle peut être décrit en termes de ce modèle comme suit :

- *fonction de perception* (ou stimulus social) $f(x_1, ..., x_n)=min|x_i-e|$, où x_i sont les fréquences des messages en entrée et e est la fréquence propre de la pulsion (en effet, l'adaptation sociale affecte toutes les fréquences propres d'une pulsion),
- *fonction de coopération* : adaptation développementale, c'est-à-dire décalage des fréquences propres, plus exécution de la fonction de transfert en cas de stimulation, comme dans le modèle motivé (une fonction spécifique pour chaque pulsion appartenant à une cellule de production et une fonction d'avalement pour chaque pulsion appartenant à une cellule de défense),
- *fonction de défection* : ne rien faire pour tous les deux types de cellules.

Nous avons également étudié le cas des messages étrangers qui n'apparaissent pas "gratuitement" et aléatoirement dans les buffers, mais sont produits par des cellules capricieuses, maléfiques ou tout simplement défectueuses. Si c'est le cas, l'élimination

de ces messages ne suffit pas ; il faut en plus empêcher les cellules malignes d'accaparer la consommation des messages et de les métaboliser de manière anormale.

- Le réseau doit avoir un *potentiel autocorrectif*, c'est-à-dire les cellules malignes doivent pouvoir re-converger vers le comportement social acceptable.

Pour ce faire, nous avons ajouté la possibilité *d'adaptation interne* à la pulsion (adaptation/convergence de la fréquence de sortie vers la fréquence d'entrée) selon un critère de participation sociale :

$$participation < 0 \Rightarrow adaptation\ interne$$

où $participation = f_{sortie} - f_{entrée}$, $f_{entrée} = 1$ si la pulsion est stimulée et 0 sinon, f_{sortie} est la proportion des fréquences de sortie correspondant à des messages présents dans les buffers de sortie (en effet, la moyenne des proportions qui concernent tous les buffers de sortie). L'inéquation est vraie si un ou plusieurs messages de sortie sont consommés par les cellules de défense.

Le comportement social des cellules et la configuration du réseau social peuvent être résumés comme suit :

- Il y a deux systèmes parallèles de cellules : les cellules productrices et les cellules de défense dont les fréquences d'excitation sont complémentaires.[83] Les fréquences reconnues par les défenseurs sont les fréquences détrimentales pour l'intégrité du réseau social. La fonction de transfert des défenseurs est une fonction d'avalement des messages qui les excitent (pas de sortie de message).
- Le comportement social de toutes les pulsions suit un modèle tit-for-tat quantitatif : le comportement coopératif est l'adaptation développementale, tandis que le comportement défectif est de ne rien faire (pas de participation sociale). Toutes les pulsions détectent l'affinité des messages à leurs fréquences propres selon un critère de distance mono-dimensionnelle des fréquences en question.
- Il y a une mesure supplémentaire de participation sociale qui stimule un processus d'adaptation interne à la pulsion : si la participation est négative (c'est-à-dire si les messages "produits" par la pulsion disparaissent rapidement à cause d'une intervention des défenseurs), l'adaptation affecte les fréquences de sortie plutôt que celles d'entrée.
- Encore une fois, la puissance du modèle de cellule sociale se trouve dans son potentiel de larges variations dans les paramètres (seuils, taux, fréquences etc.) ainsi que dans les structures (fonctions métaboliques de transfert et variantes d'adaptation).

La question de la population. La mesure de participation utilisée est "binaire" : si la pulsion participe normalement au processus, alors cette mesure sera 0, sinon elle sera inférieure à 0, auquel cas une adaptation "interne" aura lieu. Une autre possibilité considérée en perspective est de doter les cellules d'un mécanisme supplémentaire de régulation de vitesse selon une mesure de participation continue qui dépend des proportions des diverses fréquences d'entrée et de sortie dans les buffers correspondants. Dans ce cas, on pourra parler de régulation par rétroaction sociale qui

[83] Notons cependant que, telles qu'elles ont été définies, les cellules de défense ne constituent pas un système au vrai sens du terme (elles n'ont pas de relations entre elles). Dans une extension du modèle, il faudra avoir un vrai réseau de défenseurs parallèle et homomorphe au réseau de la production.

se manifestera comme une sorte de “division de travail” et qui sera nécessaire si l’on dispose de plus grandes populations avec une redondance à tous les niveaux : du point de vue de la stabilité, une solution algorithmique — et alors prétendue minimale — va souffrir de centralisation occasionnelle (comme c’est le cas de la cellule décodeur) et donc ne pourra pas profiter d’une diversité qui est la base de stabilité dans les systèmes distribués. Un générateur de diversité en combinaison avec le jeu de coopération entre les cellules des deux types peut donner naissance à des phénomènes plus complexes et alors à des phénomènes qui pourront être vus comme des comportements appris. Une étude supplémentaire que j’envisage de faire concerne donc les dynamiques d’un réseau producteur-immunitaire de contrôle conçu spécialement pour un problème de navigation avec apprentissage spatial : quel genre de dynamiques émerge-t-il ? et quels sont les types de dynamique nécessaires pour permettre ce type d’apprentissage ?[84]

Les limites. Pour les cellules-capteurs et actionneurs qui n’ont pas de fréquences d’entrée ou de sortie (et qui se situent alors aux frontières de l’agent avec son monde), il n’a pas de sens de parler d’adaptation développementale (dans l’implémentation, les facteurs d’adaptation développementale des pulsions de ces cellules ont été fixés à 0). Les modes d’interaction avec le monde sont fixes et correspondent aux capacités “physiques” de perception et de commande : un mammifère pourrait décider de voler, mais ne pourrait jamais voler. Ces limites sont alors les limites d’adaptation de la cellule et par extension les limites d’apprentissage de l’agent cellulaire. J’imagine qu’un système nerveux artificiel couplé à un système immunitaire artificiel démontrerait cette infériorité en termes des possibilités d’adaptation.[85] J’imagine aussi qu’il serait intéressant d’introduire des parties “nerveuses” complètement internes, c’est-à-dire ni de perception ni de commande, qui détermineraient des modes fixes d’interactions ésotériques.

Le monde des représentations. Il est intéressant d’évoquer de nouveau le problème de l’espace de représentation : un générateur de diversité aurait à provoquer des “mutations” dont le cumul progressif pourrait mener la forme des messages loin de leur point de départ. Cependant, ces mutations ne doivent pas être trop brusques, sinon la sémantique de l’interaction entre cellules risquerait de se briser et le réseau de se désintégrer. Une résistance au “bruit” est alors tout aussi nécessaire. Mais la mutation et la résistance au bruit ne sont pas tout : il serait préférable d’avoir plusieurs alternatives de trajectoire entre deux points dans cet espace, auquel cas le générateur de diversité produirait un éventail de formes mutées qui exploreraient l’espace des possibilités en parallèle. Le gain ne réside pas dans le parallélisme cependant, mais dans l’existence de plusieurs alternatives, donc de redondance : dans l’espace des nombres réels, il n’existe malheureusement qu’une seule possibilité d’aller de 0.3 à 0.5, la métrique de cet espace est linéaire. Encore une fois, la chimie organique me paraît l’espace parfait pour une telle diversité.[86]

---

[84] Stewart et Varela (1991) soulignent et analysent l’importance de deux dynamiques couplées, la dynamique des “réactions” et la méta-dynamique de génération et de sélection de nouvelles réactions. Dans notre modèle, il faudrait inclure une possibilité de “révision” développementale des fréquences propres.

[85] Je ne peux pas arrêter de penser qu’il doit y avoir une bonne raison pourquoi les cellules nerveuses ne prolifèrent pas.

[86] J’aimerais bien savoir quel est le niveau “moléculaire” minimal qui faut. J’incline à penser que les messages seraient plus résistants au bruit, s’ils étaient aussi des systèmes : une chaîne de symboles,

7.5 Illustration : Le robot maladif

7.5.1 Des intrus et des virus dans les buffers

Nous avons effectué une première expérience pour montrer que même avec une ***intrusion aléatoire*** il peut y avoir aliénation du réseau dans le sens où certaines pulsions se stabilisent à des fréquences inutiles et une ou plusieurs directions de navigation (qui sont essentielles pour la tâche de navigation) disparaissent du réseau. Intrusion aléatoire dans un buffer signifie qu'avec une certaine probabilité (le taux d'intrusion) la fréquence d'un message du buffer concerné est mutée de manière aléatoire ; la nouvelle fréquence peut alors être valide ou non pour le réseau, mais sa génération reste aléatoire. Les résultats d'une simulation avec un taux d'intrusion 0.4 dans le buffer d'entrée de l'un des deux niveaux diff sont donnés dans les figures 7.17a et 7.17b. On voit que, 100 cycles après, le réseau ne répond plus à la direction de droite puisque la pulsion qui en était responsable est aliénée. Généralement, comme un certain nombre d'expériences ont montré, un taux élevé d'intrusion a comme conséquence que l'adaptation développementale des pulsions conduit à un réseau où il existe une diversité de fréquences présentes sans qu'il y ait une corrélation entre elles, autrement dit sans qu'il y ait une *signification* de ces fréquences.[87]

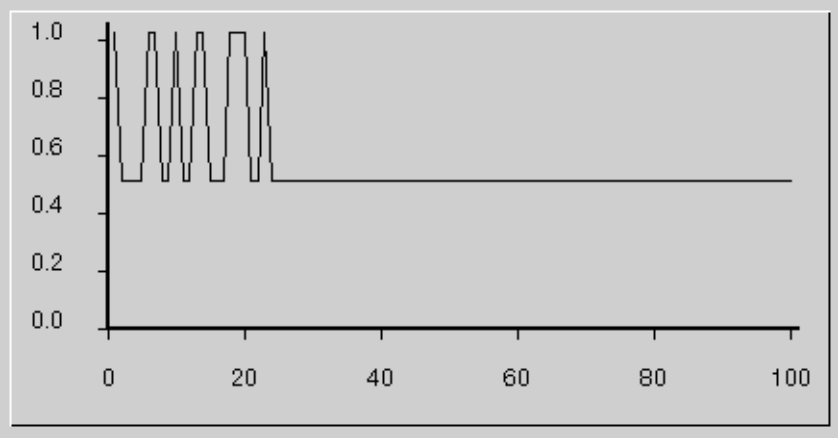

Figure 7.17a Aliénation du réseau par des intrus : 100 cycles avec de l'intrusion. Perte de réponse à la direction de devant avec un taux d'intrusion de 0.4 dans le buffer d'un des deux niveaux diff.

telle que *abbcabxa* ou *10001101101*, n'est pas un système puisqu'il n'existe pas de relation entre les différents symboles et chacun peut muter séparément de ses voisins — mais ce n'est pas le cas des protéines.

[87] Une étude du comportement dynamique de ces réseaux sous divers régimes de perturbation est envisageable afin de voir s'ils se stabilisent dans un état global cohérent et si oui lequel (Kauffman (1993) a démontré l'émergence spontanée "d'ordre" dans des réseaux binaires qui sont soumis à des mutations).

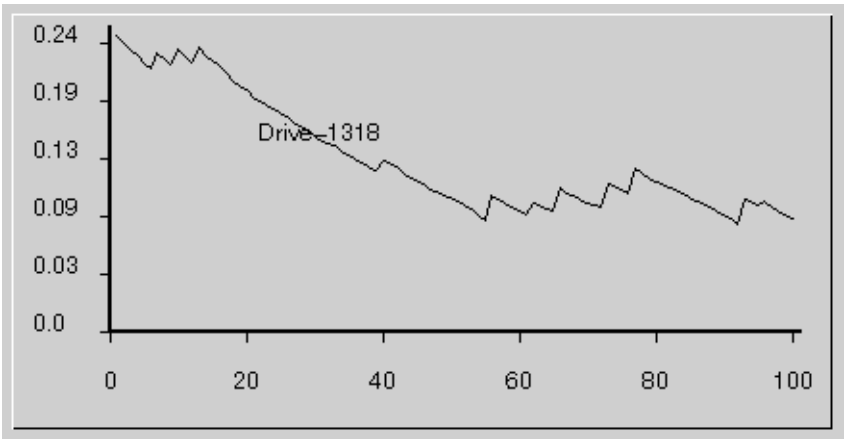

Figure 7.17b Aliénation de la fréquence propre d'une pulsion qui répondait à la direction de droite (0.25) dans l'expérience de la figure 7.17a.

Nous avons répété la même expérience avec une possibilité supplémentaire d'intrusion de ***virus***, qui sont les messages ayant à la fois une donnée et une fréquence étrangère, c'est-à-dire une donnée et une fréquence mutées de manière aléatoire. Comme les simulations ont montré, non seulement cette fois le réseau s'est aliéné et certaines fréquences ont disparu du réseau, mais, comme il pourrait y avoir des messages avec une bonne fréquence et une mauvaise donnée, il y a eu un deuxième type d'aliénation, une *aliénation plus sémantique*, qui a donné naissance à des collisions de l'agent avec les murs et les obstacles (cf. figure 7.18).

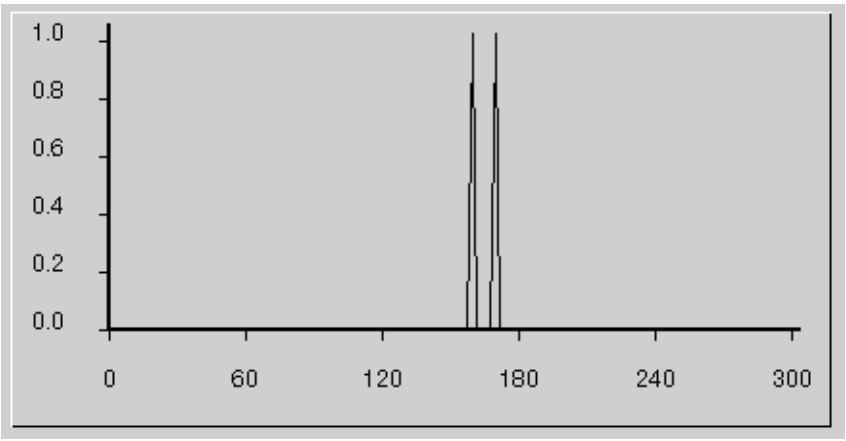

Figure 7.18 Aliénation du réseau par des virus : 300 cycles avec un taux d'intrusion 0.3 avec possibilité de virus (buffer d'un des deux niveaux diff). Il y a eu au total 9 collisions, dont 2 sont visualisées (la valeur de 1 correspond à une collision).

Nous avons alors introduit autour de chacun des buffers deux à quatre cellules défenseurs ayant une à quatre pulsions avec des fréquences propres prenant une des valeurs suivantes : 0.125, 0.375, 0.625 ou 0.875 ; ces fréquences se situent au milieu des intervalles contrôlés, c'est-à-dire au milieu des intervalles entre fréquences opératoires successives. Les seuils de détection de coopération des pulsions des défenseurs ont été définis de la même façon, c'est-à-dire comme à peu près la moitié des intervalles contrôlés, environ 0.06. L'intrusion de messages mutés ou des virus comme précédemment a continué à provoquer une déstabilisation partielle du réseau au sens de perte occasionnelle d'une ou de plusieurs directions de navigation, mais n'a pas conduit à une aliénation du réseau. Dans tous les cas, pour assurer

l'opérationalité du réseau, il faut que les pulsions des défenseurs aient de grandes vitesses minimales et que leur nombre soit suffisamment élevé.[88]

7.5.2 Cellules malignes et réponse auto-immune

Nous avons ensuite expérimenté avec des cellules productrices défectueuses dont les messages de sortie étaient perturbés : leurs fréquences ainsi que leurs données prenaient des valeurs aléatoires (toujours entre 0 et 1). Dans tous les cas et étant donné suffisamment de défenseurs, les messages porteurs de fréquences étrangères ont été correctement éliminés. Cependant, pour les messages qui encapsulent une bonne fréquence avec une mauvaise donnée, les résultats ont été plus subtils : même si une aliénation du réseau n'a pas été possible, il y a eu des zigzag et des détours et cela jusqu'à ce que la cellule défectueuse retrouve le bon chemin (souvent, l'agent ne pouvait plus accéder à sa position-but). Dans les figures 7.19a et 7.19b sont donnés les résultats d'une simulation avec de telles cellules malignes.

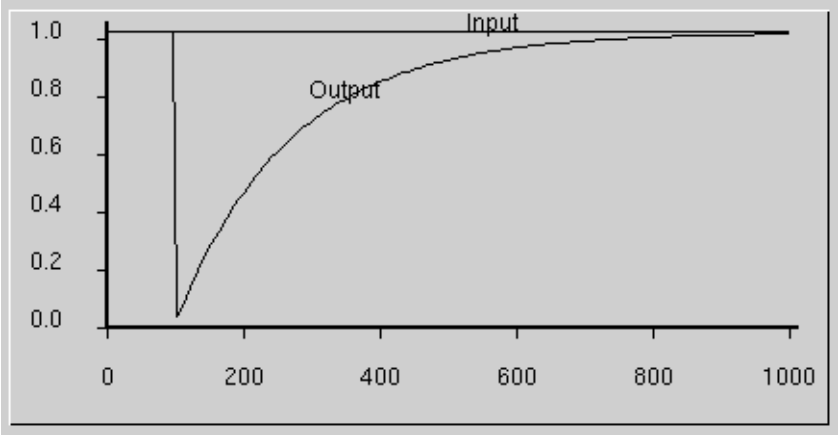

Figure 7.19a Re-stabilisation des cellules malignes. 100 cycles avec un agent normal. À t=100, transformation d'une pulsion "diff" qui correspond à la direction de derrière à un trompeur. Taux de développement très bas (0.0059 à peu près). Les courbes de l'évolution des fréquences d'entrée et de sortie sont visualisées. Le réseau se re-stabilise environ au t=1000, mais le but n'est toujours pas atteint.

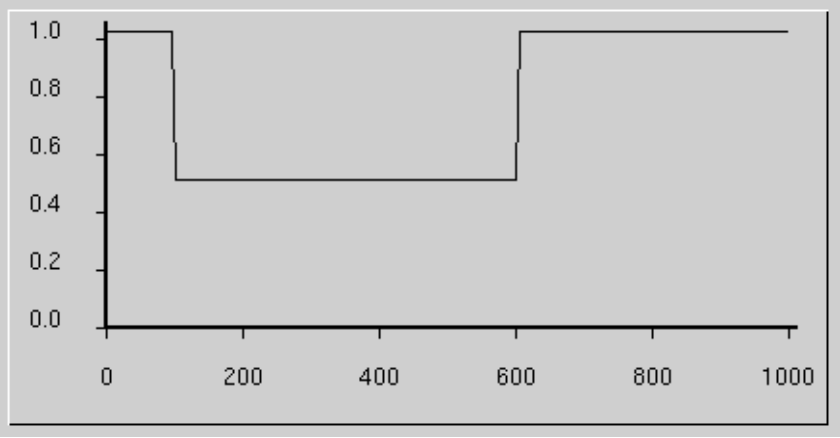

Figure 7.19b Perte de la direction de derrière pour 500 cycles environ dans l'expérience de la figure 7.19a.

[88] Puisque les cellules de défense ne prolifèrent pas, leur nombre doit être suffisamment élevé depuis le début ; une approximation appropriée semble en être la taille du buffer contrôlé.

Nous constatons que la seule existence d'un mécanisme passif de re-stabilisation "sociale" des cellules malignes ne suffit pas, et cela pour deux raisons :

- Il se peut que les cellules malignes n'ont qu'une possibilité limitée de se re-stabiliser, ou même aucune possibilité (c'est le cas des taux développementaux très faibles ou même 0).
- Les conséquences de perte de sensibilité à tel type de message pour une période de temps si longue peuvent être néfastes.

Ce que cette expérience a montré, c'est la nécessité de la reconnaissance non seulement du mauvais message, mais aussi de la *source de provenance* de ce message. Ainsi un message isolé, que ce soit un intrus ou un virus, va être simplement éliminé. Au contraire, si un message vient d'une autre cellule dans le réseau, cette cellule doit être reconnue et "punie" d'une manière ou d'une autre. Notons que le mécanisme de re-stabilisation repose sur une telle "punition" (l'absence de participation à l'activité du réseau social). Cependant, une punition ainsi "distribuée" ne suffit pas : il faut définir dans le réseau social une autorité qui sera responsable de cette punition. Dans le cas des cellules défectueuses, l'autorité, c'est-à-dire le réseau immunitaire doit, ou bien intervenir directement dans le programme des cellules, ou bien tout simplement les tuer, en tout cas démontrer une agressivité directe.[89] La première option nécessiterait un programme assez complexe d'intervention, tandis que la seconde paraît comme une solution simple dans le cas de cellules qui peuvent proliférer.[90]

Les capacités adaptatives et auto-organisationnelles des systèmes immunitaires ont stimulé des recherches en optimisation et en contrôle adaptatif (Bersini & Varela 1991; Bersini 1992). De son côté, l'idée de l'application à la robotique des principes empruntés à l'immunologie n'est pas neuve. Les systèmes d'adaptation biologique en général (y compris le système immunitaire) ont été proposés comme une vaste source d'inspiration lors du développement des systèmes de "process control" (Renders & Hanus 1992). La métaphore du système immunitaire est utilisée dans le mécanisme de sélection d'action de (Ishiguro et al. 1995), qui est directement issu de celui de Maes (1991*b*). Notre mécanisme a été développé dans la problématique de révéler des principes d'organisation cellulaire plutôt que de fournir des solutions à des problèmes pratiques ; le problème de la navigation a servi seulement pour l'illustration de ces principes.

7.5.3 Discussion générale

Nous avons exposé la philosophie derrière la plasticité et l'auto-organisation dans un réseau cellulaire algorithmique. Nous avons montré que pour permettre cette plasticité dans le réseau, il a été nécessaire de faire deux extensions majeures : (a) introduction des motivations/préférences locales dans les cellules qui sont exprimées de manière uniforme comme des fréquences propres d'excitation et qui sont comparées avec les identificateurs des messages en entrée, et (b) remplacement des connexions entre

89 J'ai eu le même résultat, c'est-à-dire le besoin d'agressivité directe dans les relations sociales, lors de mes premières expériences avec le modèle tit-for-tat quantitatif (cf. annexe A).

90 Stahl & Goheen (1963) discutent la forme et la nature des algorithmes moléculaires et postulent qu'il n'existe pas de raison particulière pour que certains algorithmes ne soient pas permis, aucun principe n'est violé! Il s'agit simplement d'une limitation de complexité. Cet argument est donné à l'égard de l'adaptation lamarckienne mais peut s'appliquer également à beaucoup d'autres cas.

niveaux par des buffers partagés de messages. Une autre extension consiste à définir deux types de messages, les messages habituels consommables et les messages catalytiques dont la seule présence suffit pour déclencher chez les cellules excitées la consommation des autres messages ; tous les messages ont deux parties, la partie identificatrice et la partie donnée. Quelques résultats de simulation ont révélé que le réseau cellulaire s'auto-organise en cas de pannes en découvrant des chemins alternatifs de flot de messages et que des pannes multiples ralentissent la réactivité du système aux événements extérieurs, puisque les cellules des niveaux intermédiaires ne sont plus spécialisées à un type de message mais partagent leur temps entre plusieurs motivations. De plus, la présence d'un facteur supplémentaire développemental/ ontogénétique permet l'auto-organisation même dans le cas de pannes majeures dans le réseau, c'est-à-dire lorsqu'une fréquence propre disparaît complètement du réseau.

À noter que le comportement social de la cellule motivée est par définition "coopératif" c'est-à-dire la cellule essaiera de se stabiliser à une nouvelle fréquence même si cette fréquence est détrimentale pour l'intégrité du réseau. Dans ce cas, ce qui se révèle nécessaire, c'est *un système immunitaire, c'est-à-dire un système de cellules parallèle au premier qui surveille et régule l'adaptation "sociale" des autres cellules*. Ces cellules sont également motivées (elles doivent aussi être sociales, mais dans l'exemple étudié leur socialité n'a pas été utile) et leurs fréquences propres d'excitation sont complémentaires à celles des cellules productrices, ce qui a comme conséquence que la reconnaissance et l'élimination des messages-ennemis sont directes. Par défaut, le comportement social de toutes les cellules est coopératif, qu'elles soient de l'un ou de l'autre type : ***l'intégrité du réseau est alors préservée grâce au couplage de deux forces sociales***, la production et la police. Une étude de l'intégrité du réseau se traduit donc par une étude située dans un niveau méta, celui du *réseau des motivations complémentaires et adaptatives*, c'est-à-dire des fréquences propres des différentes cellules. Notons encore qu'il n'y a pas besoin d'un système méta-immunitaire puisque, en principe, le système immunitaire est permis d'agir sur lui-même[91] à condition que les défenseurs soient suffisamment nombreux et divers.

Pour que l'ensemble des deux systèmes (celui de production et le système immunitaire) soit cohérent et non manipulable, la population des fréquences doivent être développés par l'agent lui-même au cours de sa vie comme résultat des interactions dans le réseau. La suite du travail présenté se situe donc dans la problématique de co-développement des deux réseaux *au niveau abstrait de l'espace de représentation des fréquences propres*, c'est-à-dire au niveau de l'information (que ce soit unidirectionnel, comme l'espace des nombres réels, ou non) : je suis convaincue qu'un comportement "appris" n'est en réalité qu'un comportement "développé".

De la sélectivité au sélectionnisme. Si les cellules/pulsions sont sélectives, un comportement appris sera en effet un comportement *généré* par le réseau même et *renforcé* dans un environnement particulier. La sélectivité doit alors être combinée avec un générateur de diversité et un mécanisme d'amplification (ou renforcement) environnementale pour permettre l'apprentissage apparent. Dans le cas du modèle de

[91] Pitrat (1991) insiste que toute la connaissance se trouve au niveau méta et qu'un seul niveau de méta-connaissances avec une possibilité d'agir sur lui-même, c'est-à-dire avec de la réflexivité, suffit pour le développement des capacités cognitives à la hauteur de l'homme et même plus grandes. Inversement, l'absence d'un niveau méta régulateur peut avoir des conséquences désastreuses : le SIDA attaque la partie méta du système immunitaire.

cellule autonome, il faudrait introduire un mécanisme de génération (ou révision) de la fréquence propre dans les cellules productrices ou de défense. Il y aurait alors un jeu de tire-à-la-corde entre les deux populations de cellules dans lequel les unes essaieraient de générer de nouvelles fréquences et les autres essaieraient de les éliminer ; dans ce cas, la reconnaissance et l'élimination des fréquences étrangères ne seraient qu'un effet de bord du jeu social entre les cellules. La question deviendrait donc, dans un environnement stable le réseau se stabilise-t-il à une configuration ? est-ce qu'il "apprend" quelque chose ? Manderick (1992) analyse et compare trois systèmes sélectifs (l'évolution naturelle, le système immunitaire et le cerveau) en soulignant l'importance du sélectionnisme pour la conception de systèmes autonomes cognitifs dans le cadre de la nouvelle IA (ou IA comportementale). (Steels 1994*b*) est le seul exemple de variation et de sélection à l'intérieur du système de contrôle d'un robot et est inspiré des théories sélectionnistes du cerveau développées par Edelman.

7.6 Conclusion

Ce chapitre a présenté le problème d'une organisation cellulaire de topologie dynamique résistante et robuste aux pannes. Le modèle cellulaire entrée-sortie du chapitre 3 doit être étendu en un *modèle sélectif et motivé* de cellule autonome. Ce modèle étendu consiste à voir la cellule comme un système de composants, les pulsions, exécutant en parallèle et étant en compétition permanente entre elles pour la consommation des messages que la cellule reçoit. Si les pulsions sont chargées de la consommation des messages, la physiologie de la cellule est responsable de la coordination des pulsions et de la résolution des conflits ; de même que dans le cas de l'animat (chapitre 3), cette physiologie est indépendante. Pour fonctionner, le mécanisme nécessite en plus des messages composés d'une partie d'identification et d'une partie de donnée. Certains de ces messages doivent être des catalyseurs qui déclencheront des actions métaboliques (des réactions) chez des cellules sans être vraiment consommés. L'opérationalité du modèle et sa robustesse face aux pannes sont assurées d'un mécanisme supplémentaire de modification des dynamiques d'interaction des pulsions avec le monde des messages ; de même que dans le chapitre 4, cette modification concerne les paramètres qui déterminent la dynamique d'interaction (ici les vitesses de consommation des messages) et elle est homéostatique. Contrairement au chapitre 4, cependant, il n'y a pas de représentation explicite du besoin, il n'y a pas de méta-motivation. Quant à la socialité, elle est encore une fois du type "partage du besoin" et se manifeste comme adaptation de l'agent-cellule à la forme des messages rencontrés. Dans le cas des messages intrus et des virus, qui peuvent aliéner le réseau, c'est-à-dire lui faire donner des fausses réponses à ses perturbations, un système cellulaire de défense parallèle au premier, c'est-à-dire un système immunitaire, s'avère nécessaire. Le système de défense repose sur la reconnaissance et l'élimination des messages étrangers. Le couplage entre système immunitaire et système "de production" se manifeste dans la complémentarité des messages traités par les deux systèmes, il s'agit donc d'un couplage entre deux forces sociales au sein du même agent. Le problème de l'espace de représentation a été rediscuté dans ce nouveau contexte sous une perspective future d'apprentissage.

Concernant l'autonomie, qui est vue sous l'angle restreint de l'opérationalité, nous pouvons généraliser sur les résultats des chapitres 4, 5 et 7 pour dire que ***l'essence de l'autonomie des agents est le changement continu, la modification continue de la***

dynamique d'interaction avec le monde (cf. van Gelder & Port 1995). Le changement et l'adaptation sont reliés à la sélectivité de l'agent et son "but" est de rendre l'agent de plus en plus sélectif envers les aspects pertinents de l'environnement et de moins en moins sélectif envers les aspects non pertinents. Cette modification ou adaptation de la dynamique d'interaction est en effet une autorégulation des paramètres qui déterminent cette dynamique, alors une autorégulation des paramètres ayant un caractère temporel, des taux ou des vitesses. Concernant la relation entre autonomie et autorégulation des taux d'adaptation, le même résultat a été obtenu au niveau de l'agent cellulaire (cf. chapitres 4 et 5) et il vient en faveur de notre thèse des principes organisationnels récursifs, c'est-à-dire des principes organisationnels applicables à tous les niveaux d'agents. Bien entendu, les possibilités d'adaptation ne sont en aucun cas limitées à la régulation des paramètres et d'autres alternatives sont considérées en perspective — en particulier l'approche développementale couplée à un système immunitaire de contrôle, comme discuté dans le paragraphe 7.5.3.

Dans le cas du mécanisme d'auto-organisation cellulaire, pour assurer un changement ainsi *continu*, il faudrait trouver un mécanisme induisant des pannes et donc injectant du bruit continûment, et cela a constitué un de mes deux arguments en faveur d'une fonction de sénescence (cf. paragraphe 1.3.6).[92]

Finalement, il est important de souligner une fois de plus l'importance de la diversité et de l'auto-organisation à l'intérieur de l'agent ; à côté des considérations d'opérationalité présentées dans ce chapitre, nous avons dessiné la relation de la diversité avec le sélectionnisme et les perspectives qui s'ouvrent vers cette direction.

[92] Le rôle organisationnel du bruit a été souligné par plusieurs auteurs, par exemple Atlan (1979).

Le problème	Organisation cellulaire de topologie dynamique résistante et robuste aux pannes
L'application	Un problème de navigation
La solution	*Système "motivationnel"* (système de pulsions exécutant en parallèle et étant en compétition permanente entre elles pour la consommation des messages en entrée) Des *buffers partagés* au lieu de connexions *Messages composés* du type (identificateur, donnée) *Messages consommables ou messages catalyseurs* *Mécanisme de compétition* (à l'aide de vitesses de réaction) *Autorégulation de dynamique d'interaction avec le monde* (vitesses de réaction) *Adaptation sociale développementale* : "Socialité" tit-for-tat quant à la reconnaissance *Système immunitaire* de défense parallèle au système de "production" (reconnaissance et élimination des mauvais messages)
Les enseignements	• Auto-organisation continue • Besoin d'adaptation développementale continue • Aliénation du réseau par des intrus et des virus (en l'absence d'un système immunitaire) • Reconnaissance directe des messages étrangers • Couplage entre système de production et système immunitaire : complémentarité des messages reconnus par les deux systèmes • Nécessité d'agressivité directe dans le réseau (le cas des cellules malignes) • L'importance de la sélectivité • Le rôle de la diversité

Tableau 7.1 Tableau récapitulatif du chapitre 7

Chapitre 8 Les agents sénescents

Sum, ergo cogito.
(Miguel de Unamuno, "Del sentimento trágico de la vida", 1912)

8.1 Introduction

8.1.1 À quoi peut ressembler une fonction de sénescence ?

Dans le paragraphe 1.3.6 nous avons discuté du besoin d'une fonction de sénescence comme force motrice de l'émergence de structures de niveau supérieur d'organisation et de l'apprentissage. D'une part, la mortalité des agents d'un niveau d'organisation doit les forcer à bâtir des relations sociales élaborées pour se maintenir en vie, d'où l'émergence des structures de niveau supérieur d'organisation. D'autre part, un agent autonome doit être sénescent quant à ses besoins, pour devoir les réviser continûment et par conséquent d'apprendre (sinon pourquoi le ferait-il ?). Dans le chapitre précédent, les cellules s'auto-organisent spontanément en cas de pannes en essayant de "découvrir" de nouvelles interactions sociales ; une force naturelle de dégradation, telle qu'une fonction de sénescence, pourrait maintenir l'activité de l'auto-organisation à son maximum, et donc maintenir le réseau actif (et l'agent vivant).

Le premier souci lors du développement d'un modèle de sénescence est de maintenir un degré de similitude avec la réalité biologique, mais dans une problématique de *comprendre* cette réalité plutôt que de l'utiliser aveuglément. Le modèle de base doit alors donner naissance à un phénomène de sénescence qui n'apparaît pas à partir d'un certain âge, mais qui est un phénomène continu tout au long de la vie de l'agent et qui conduit progressivement et inévitablement à sa mort : *"... le déclin dans la vieillesse serait le prix à payer pour la vigueur dans la jeunesse. D'une part, la vitesse de sénescence serait accrue par les forces qui tendent à avantager la vigueur dans la jeunesse. D'autre part, elle serait diminuée par d'autres forces qui tendent à retarder les effets nocifs. Ce serait donc l'équilibre entre ces forces opposées qui, en fin de compte, ajusterait le processus du vieillissement et la durée de la vie"* (Jacob 1981, p. 93). Ou, comme Kanungo l'a décrit (1994, p. 267), *"senescence or aging should not be viewed as an isolated and independent phase in the life span of organisms, but should be considered together with development and adulthood phases. These earlier phases may not only influence the organism's longevity but also the rate, duration and mode of its senescence"*.

La force (ou fonction) de sénescence doit être donc modélisée comme une force de développement qui possède les propriétés suivantes :

- La sénescence doit favoriser les relations sociales entre agents, de sorte que les agents sociaux vivent plus longtemps que les agents asociaux.
- La sénescence doit être continue et assurer la mort de l'agent.

- Cependant, la sénescence ne doit pas préspécifier la durée de vie de l'agent ; l'agent doit avoir une possibilité limitée d'agir de manière à prolonger relativement la durée de sa vie.
- La sénescence doit être le résultat d'un couplage de deux forces opposées dont l'une retarde et l'autre accélère la mort de l'agent. Alors l'agent doit chercher les bons compromis entre ces forces qui lui permettront de vivre plus.[93]

Notons également, qu'une force de sénescence qui repose sur le couplage de deux forces opposées et qui ne préspécifie pas la durée de vie des agents, conduit à un système qui n'est pas directement contrôlable et manipulable comme nous le verrons par la suite.

8.1.2 Modèle de sénescence

Pour répondre à ces questions, c'est-à-dire pour trouver un modèle de sénescence conforme aux spécifications données, nous avons imaginé une physiologie différente d'agent (cf. chapitre 1 pour le rôle de la physiologie), dans laquelle le programme et le métabolisme de l'agent sont couplés avec des rétroactions négatives du type : chaque fois qu'un agent métabolise, c'est parce que son programme l'a instruit et il le fait dans une direction qui le rend plus adaptatif à son environnement, mais de son côté chaque action métabolique agit de manière progressivement destructive sur le programme et donc sur sa possibilité future de métaboliser. La fonction de sénescence est alors une rétroaction négative ajoutée à l'instruction du métabolisme par le programme de l'agent (cette situation est illustrée dans les figures 8.1 et 8.2). Le plus simple des modèles conformes à cette idée est celui des programmes unidirectionnels (linéaires) avec des effets de seuil qui déterminent le potentiel des agents à métaboliser. Le rythme de ce métabolisme, qui dépend non seulement du programme mais aussi de l'environnement dans lequel l'agent est situé, est responsable du rythme de sénescence, et ce rythme va être auto-catalytique. Cette problématique d'auto-catalyse négative a été inspirée par une consultation rapide des théories biologiques de sénescence (Warner et al. 1987) et en particulier par l'idée que la sénescence n'est pas un seul mécanisme, mais tout un ensemble de mécanismes agissant à plusieurs niveaux d'organisation et probablement entre-liés, de façon que pratiquement toutes les théories existantes retiennent une part de la vérité (Rusting 1992) : la sénescence doit être à la fois programmée et non programmée, et elle doit être à la fois continue et montrer des effets de catastrophe. L'hypothèse de la membrane de Zs.-Nagy (1994) et la théorie de régulation de gènes de Kanungo (1994) en particulier, tentent une synthèse des données expérimentales et des théories existantes selon un principe commun et indépendant de niveau d'observation, la détérioration progressive de la membrane ou de l'expression des gènes respectivement.

---

[93] François Jacob (ibid., p. 90) cite comme un exemple de sénescence "perturbée", celui de Dorian Gray, l'héros de Oscar Wilde, qui reste mortel mais jeune tandis que son portrait montre les signes progressifs du vieillissement. Ce qui m'avait fascinée à la lecture de cette histoire victorienne, et que je peux maintenant analyser sous mon optique actuelle, c'était non seulement l'idée de la mortalité sans vieillissement apparent, mais aussi que le vrai vieillissement de Dorian, tel qu'il se manifestait sur son portrait, était *beaucoup plus rapide et plus monstrueux* qu'un vieillissement normal. Pourquoi ce vieillisement était-il plus aigu ? Parce que Dorian, ayant coupé la rétroaction de la sénescence apparente, était sorti de ses limites "pour descendre dans les ténèbres les plus sombres", comme l'auteur le décrit. Le besoin de compromis avait disparu.

À part la théorie de Dyson (1985) qui a été citée dans le paragraphe 1.3.5, cette idée de l'importance du métabolisme se retrouve également dans la théorie des radicaux libres (Kirkwood & Holliday 1979) qui ont un effet nocif sur l'ADN, les protéines et d'autres molécules essentielles et dont le cumul dépend du rythme de métabolisme (Rusting (ibid., p. 91) note que la durée de vie des diverses espèces est inversement proportionnelle à leur rythme de métabolisme). ***La sénescence est alors l'usure de la connexion programme-métabolisme! Et cette usure est inévitable : plus l'agent métabolise, plus son programme se détériore.***[94]

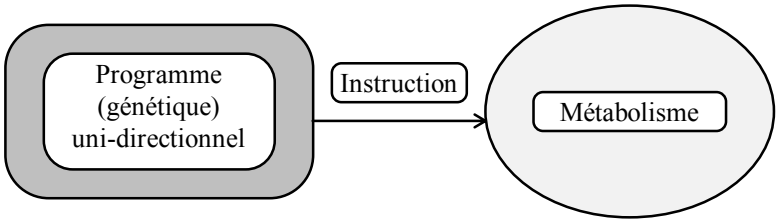

Figure 8.1 Le modèle d'agent sans sénescence.

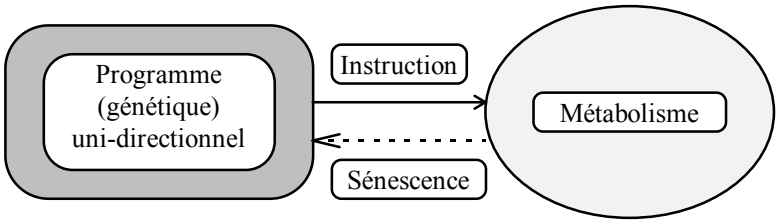

Figure 8.2 Le modèle d'agent sénescent. Désormais les flèches pointillées vont représenter des rétroactions de sénescence.

8.1.3 Domaine d'application

Afin d'implémenter ces idées, nous avons développé un modèle de sénescence pour un problème simple mais suffisamment abstrait pour permettre l'observation de phénomènes assez complexes et servir ainsi de base de validation. Une population d'agents[95] est initialisée dans un espace borné ; chacun des agents a un but individuel et une possibilité de "socialité". Un agent peut en plus rencontrer des obstacles qui, comme leur nom l'indique, vont le retarder face à ses fins. Éventuellement, l'agent atteindra son but et s'arrêtera, c'est-à-dire il mourra ; le système multi-agents sera mort, lorsque tous les agents qui le composent seront morts. Pour simuler ce système, les notions d'espace, de but, d'obstacle et de socialité ont été instanciées comme suit. L'espace borné est une grille bidimensionnelle fermée, le but est un point dans cette grille vers lequel l'agent se dirige, les obstacles sont des objets "inertes" que l'agent détruit, et la socialité repose sur un rayon de perception d'autres agents (par la suite nous allons nous référer à ce rayon comme "socialité" ou "facteur de socialité"). La "socialité" distrait l'agent de son but en lui induisant un "comportement de fou" : il

[94] Notons que, selon la théorie du soma disposable, le vieillissement est dû au cumul des effets nocifs seulement aux gènes des cellules somatiques, le programme des cellules germinatives restant intact (Kirkwood & Holliday, ibid.).

[95] Je préfère parler de populations plutôt que de sociétés puisque les relations sociales n'ont pas une forme hiérarchique.

tourne autour de soi-même sans se déplacer et, éventuellement, lorsqu'un timeout expire, il avance, mais alors dans une direction aléatoire puisqu'il a été perturbé. Du point de vue de l'arbitrage entre ces trois comportements (ou tâches) de base, la destruction des obstacles est prioritaire par rapport à la distraction sociale, qui est à son tour prioritaire par rapport à l'orientation vers le but ; l'arbitrage est donc statique. Ce problème est décrit dans les figures 8.3 et 8.4.

Modèle comportemental de base

1 *S'il existe un obstacle sur la même place que l'agent, alors l'agent le détruit.*

2 *S'il existe d'autres agents dans le rayon de perception (socialité), alors l'agent est perturbé et après un timeout il migre ailleurs.*

3 *Si son but est atteint, alors l'agent s'arrête, sinon il se dirige vers ce but.*

- *Il peut y avoir de 1 à 5 obstacles dans la même place (cela correspond en effet à des obstacles de degrés de difficulté variés).*
- *Le rayon de perception de l'agent (sa socialité) prend une valeur entre 1 et 5.*
- *Le timeout de distraction sociale prend une valeur entre 1 et 10.*[96]
- *Le but est initialisé de façon aléatoire sur la grille.*

Figure 8.3 Modèle comportemental de base.

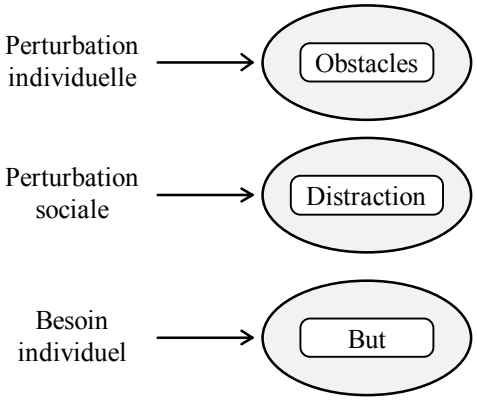

Figure 8.4 Modèle abstrait de comportement : système motivationnel (arbitrage statique entre tâches d'un agent). Le but est un point dans l'espace (dans une grille bidimensionnelle), les obstacles sont des objets dont la présence retarde l'agent et la distraction induit un changement de direction.

La vision de la socialité comme distraction est basée sur l'observation suivante : un agent a un but propre qu'il essaie de mener à terme. Si l'agent rencontre d'autres agents qui ont également leurs propres buts, il est distrait de son but pour participer à une activité sociale, donc il atteint son but plus tard et il vit plus longtemps. Par exemple, dans le cas des cellules du chapitre précédent, un agent-cellule doit se conformer au besoin de sa voisine et traiter le message envoyé par cette dernière, sinon il n'aura rien à faire et il mourra. Dans le cas des agents explorateurs sociaux du chapitre 5, un agent asocial possédant un système d'adaptation rapide terminera sa propre mission très tôt ; une socialité du type "partage du besoin" pourra maintenir

[96] Pour tous les modèles, les paramètres environnementaux et comportementaux ont été reglés dans un monde de taille 40x40 de façon à amplifier relativement les phénomènes observés, par exemple la distraction. Quelques expériences sporadiques avec d'autres paramétrages nous ont permis de voir que les phénomènes ne changaient pas *qualitativement*.

l'activité de l'agent plus longtemps.[97] Dans tous les cas, la socialité-distraction a une signification opérationnelle : le problème global, celui du système multi-agents, est mieux résolu avec de la socialité-distraction que sans. La socialité des cellules peut assurer le voyage des messages des capteurs jusqu'aux actionneurs et donc l'obtention du but de navigation, tandis que la socialité des explorateurs peut assurer la couverture de tout le champ d'exploration et donc la complétion du balayage.

Nous avons implémenté trois modèles comportementaux successifs dont le premier est un modèle "réactif" quant à son besoin, le deuxième un modèle "autonome" (motivé et sénescent) et le troisième un modèle "cognitif" (autorégulant) ; les trois modèles sont présentés dans les paragraphes 8.2, 8.3 et 8.4, respectivement. La sénescence est continue, elle intervient au niveau physiologique comme une boucle de rétroaction négative et elle affecte surtout la socialité. En effet, nous verrons que le système meurt petit à petit si la socialité de ses composants baisse : alors les relations entre les agents se dégradent et le système perd son identité d'entité cohérente, il se décompose. Le paramètre de socialité exprime à la fois égoïsme et tolérance : il exprime l'égoïsme puisque l'agent est distrait de son but et il exprime la tolérance puisqu'il lui permet de vivre plus longtemps que s'il n'était pas social.

Nous montrons que la sénescence doit affecter tous les paramètres de régulation et que le système reste non manipulable et imprévisible quant à sa durée de vie. Nous montrons également que la sénescence favorise les modèles d'agents plus "cognitifs" (ayant des boucles de régulation supplémentaires) et donc qu'elle favorise l'émergence d'organisations de niveau supérieur ayant des relations sociales plus élaborées. Nous avons finalement exploré les effets d'une surpopulation dans l'espace pour montrer qu'elle conduit statistiquement à une baisse de l'espérance de vie.

Finalement, pour tester l'idée que l'observation peut conduire à des conclusions qui ne sont pas fausses mais n'ont pas de vraie relation avec la réalité causale (celle du concepteur), j'ai fait observer le système par des volontaires scientifiques ; leurs observations sont résumées dans le paragraphe 8.5.

8.2 Le modèle réactif

Le premier modèle de sénescence est fondé sur l'observation qu'une socialité constante au cours du temps peut entraîner "l'immortalité" du système multi-agents, c'est-à-dire le système peut rester vivant pour toujours. Cela peut arriver si deux agents ont des buts voisins ou bien le même but *et* la socialité de l'un ou de l'autre est suffisamment élevée (supérieure à la distance entre les buts des deux agents). La solution semble être alors un mécanisme qui fait progressivement baisser la socialité. Une première idée est d'avoir une socialité qui baisse "naturellement", c'est-à-dire pour une raison intrinsèque à l'agent et indépendamment de son activité : cependant, cela conduirait à un phénomène de "mort préprogrammée". Au contraire, comme nous l'avons déjà expliqué, nous nous intéressons à des modèles de sénescence où les

[97] Lors de l'analyse du chapitre 5, nous n'avons fait aucune hypothèse sur la vitesse de l'adaptation des agents individuels : nous avons élaboré un modèle de socialité qui est indépendant de l'opérationalité de chacun des individus et qui améliore les performances du système dans tous les cas. Évidemment, la socialité est beaucoup plus avantageuse dans le cas où un seul agent n'est pas opérationnel (un explorateur asocial avec une adaptation trop rapide n'est pas opérationnel, parce qu'il termine avant de balayer tout le champ).

agents sont mortels *sans* que les conditions et le contexte de cette mort soient préprogrammés.

Nous avons alors introduit un lien de rétroaction entre l'effet de la socialité (la distraction) et la socialité elle-même : à chaque fois que l'agent socialise, c'est-à-dire à chaque fois qu'il est distrait de son but, sa socialité baisse d'un facteur constant qui a été fixé au 0.9. Nous avons également introduit un mécanisme indépendant de dégradation qui utilise un timeout et qui est lui aussi affecté par la rétroaction[98], mais seulement pour amplifier le phénomène de la mort, c'est-à-dire pour assurer que le système meurt assez vite pour qu'un observateur puisse le regarder sans trop s'ennuyer. Ce premier mécanisme de sénescence est décrit dans les figures 8.5 et 8.6.

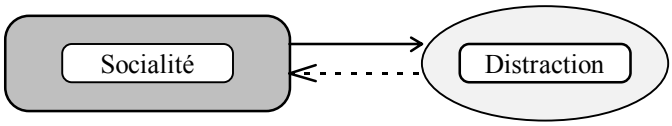

Figure 8.5 Modèle réactif : physiologie. La socialité de l'agent, c'est-à-dire le rayon de perception sociale, se catalyse négativement.

Modèle réactif

Programme : système de dégradation de socialité

Timeout ou distraction sociale

*==> Socialité := socialité*0.9 & timeout := timeout*0.9*

- *timeout ∈ (1,20)*

Figure 8.6 Modèle de base (modèle réactif).

L'observation et l'expérimentation avec ce système ont montré qu'il est manipulable à un certain degré : si l'on déplace l'agent loin de son but, l'agent va toujours essayer d'y retourner, malgré la présence d'obstacles et d'autres agents, mais aussi malgré ces essais de manipulation. Des manipulations successives vont conduire à une baisse de socialité près de 0, de sorte que l'agent ne sera plus sensible à la présence des autres agents. Cependant, comme les manipulations n'affectent pas directement le but de l'agent, il reste manipulable par un agent extérieur qui, après observation, aurait découvert la position-but de l'agent (cela est illustré dans la figure 8.7, où l'on voit la trace de l'agent après plusieurs essais de manipulation).[99] Par exemple, si l'agent en question est un agent balayeur de garage comme celui du chapitre 4, deux cambrioleurs qui veulent — pour une raison ou une autre — le garder loin de sa base pourront y parvenir en lui posant continûment des sacs noirs de l'autre côté du garage (l'agent va rentrer à la base de temps en temps, mais seulement temporairement). En un mot, *le système est réactif quant à sa motivation* ; il paraît donc nécessaire d'avoir une rétroaction au niveau du but qui doit être dynamique.

---

[98] L'idée d'avoir un timeout qui fait dégrader la socialité et qui dégrade de la même manière lui aussi est consistante avec la spécification de la sénescence comme un phénomème auto-catalytique, qui est alors observable mais pas directement contrôlable (cf. la discussion du paragraphe 8.6.2).

[99] J'ai découvert que le système était ainsi manipulable quand certains observateurs m'ont demandé de créer des configurations spécifiques d'agents qu'ils prévoyaient comme intéressantes selon leurs observations antérieures. Une de ces configurations était un seul agent dans le monde (l'observateur a alors dit "ce n'est pas intéressant, cet agent suit un chemin très régulier, alors quand on est seul, on est bête").

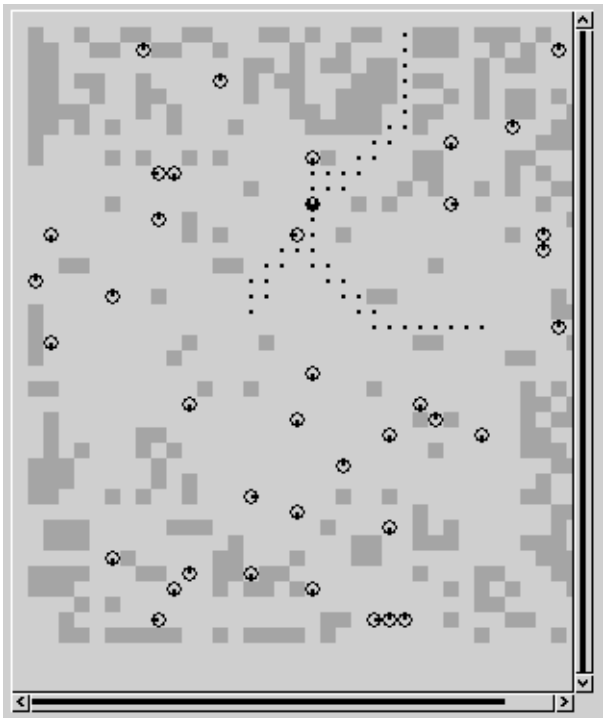

Figure 8.7 Manipulabilité dans le modèle réactif. La trace originale de l'agent noir est celle de gauche, les deux autres étant celles après manipulation (enlèvement et repositionnement dans un autre endroit). L'agent se dirige toujours vers son but initial. Les obstacles sont visualisés en plus foncés.

8.3 Le modèle autonome (motivé et sénescent)

Avoir un but dynamique signifie disposer d'un mécanisme de révision de but. Pour le modèle comportemental de base (fig. 8.3), réviser son but signifie adopter une nouvelle position-but dans l'espace. La question importante qui se pose ensuite est, *quand faut-il réviser son but ?* Suite aux observations du dernier paragraphe, il faut une révision de but s'il y a eu beaucoup d'essais de manipulation. Puisqu'un agent n'a pas de moyen de savoir qu'il a été manipulé (à moins de posséder un gyroscope, un odomètre ou un autre capteur du même type), il ne reste qu'un critère temporel à utiliser : avec un timeout, l'agent révise son but. Il paraît également naturel de dire qu'un agent doit réviser son but, si ce dernier est atteint (dans un effort de survivre). Dans tous les cas, on risque de rencontrer le même problème de manipulabilité à un niveau méta, si l'on n'introduit pas un critère de révision de but qui doit être vrai au départ, puis progressivement devenir faux. Nous avons alors introduit une variable supplémentaire, appelée *adaptativité* pour montrer qu'elle exprime la possibilité de l'agent d'adapter ou réviser son but, et un seuil auquel elle est comparée. Si l'adaptativité est supérieure à ce seuil, l'agent peut réviser son but, sinon il ne le peut pas. La rétroaction affecte ce paramètre d'adaptativité pour assurer la non manipulabilité du système. Toujours pour avoir une durée de vie raisonnable pour un observateur, un facteur de dégradation temporelle de l'adaptativité a été également introduit. Pour que le mécanisme de sénescence soit auto-catalytique, il a aussi fallu introduire deux facteurs de régénération de l'adaptativité lors de l'atteinte du but courant de l'agent ou lors de distraction sociale. Ainsi, *les deux variables d'adaptativité et de socialité sont couplées par une double boucle* : si l'agent est

adaptatif, alors il découvre un nouveau but et il risque de rencontrer d'autres agents et d'en être distrait. Si l'agent est distrait, son adaptativité augmente. Cette double boucle est une boucle de renforcement ou d'auto-catalyse, puisqu'en elle seule elle conduit à une augmentation continue de l'adaptativité et de la socialité. Pourtant, nous avons vu que l'activité même de l'agent conduit à une diminution de la socialité, ce qui à son tour va empêcher l'adaptativité d'augmenter. Si en plus il existe un mécanisme de dégradation temporelle, l'adaptativité va éventuellement diminuer en passant parfois par des maxima locaux. Ce modèle d'agent sénescent est ainsi plus autonome que le précédent (l'utilisateur ne peut pas le maintenir en vie artificiellement par des manipulations persistantes), parce qu'il démontre une autonomie quant à ses motivations, ses buts. Le problème de l'agent devient alors comment socialiser suffisamment pour vivre longtemps, mais pas trop socialiser, sinon il risque de ne pas atteindre ses propres buts à temps et de mourir prématurément. Le modèle est résumé dans les figures 8.8 et 8.9.

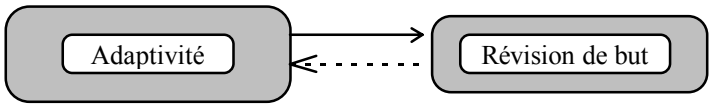

Figure 8.8 Modèle motivé : physiologie. L'adaptativité de l'agent se catalyse négativement. Les liens de récompense positive ne sont pas visualisés.

Comme prévu, les agents dans ce modèle ne sont plus aussi manipulables que ceux du modèle précédent. Dans les figures 8.10 et 8.11 sont visualisés le chemin d'un agent et l'effet de plusieurs tentatives de manipulation dans deux configurations différentes. On constate que, après de telles perturbations persistantes, un agent peut changer de but définitivement ou mourir et que, par ailleurs, plus un agent a "travaillé" pendant sa vie, c'est-à-dire plus il a poursuivi des buts, moins il est manipulable et plus il est susceptible de mourir (comme s'il était épuisé).

Modèle autonome (motivé et sénescent)

Programme
Si son but est atteint ou timeout, alors l'agent essaye de trouver un nouveau but
(si adaptativité≥seuil, alors (il est adaptatif) nouveau but),
sinon mort (la mort est "simulée" comme but=position_courante)

Dégradation temporelle : adaptativité:=adaptativitéf_0*
Régénération lorsqu'un but est atteint : adaptativité:=adaptativité(1+f_1)*
Régénération lors de distraction : adaptativité:=adaptativité(1+f_2)*
*Dégradation de f_1 lors de révision de but : f_1:=f_1*f_0,*
*Dégradation de f_2 lors de révision de but : f_2:=f_2*f_0*
Dégradation de timeout lors de révision de but : timeout:=timeoutf_0*

- *adaptivité (0)∈ (0.5,1), timeout (0)∈ (50,100), seuil ∈ (0.1,0.2)*
  f_0 ∈ (0.05,0.95), f_1(0) ∈ (0.5,0.9), f_2(0) ∈ (0,0.3)

Figure 8.9 Modèle autonome (motivé et sénescent)

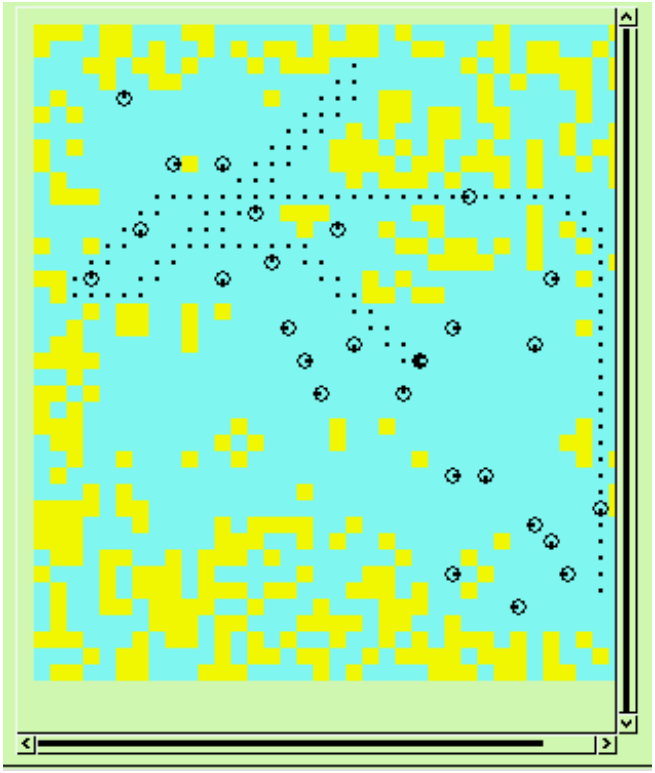

Figure 8.10 La trace d'un agent dans le modèle motivé. La trace de l'agent noir passe par plusieurs buts (on voit les changements de direction dans la trace).

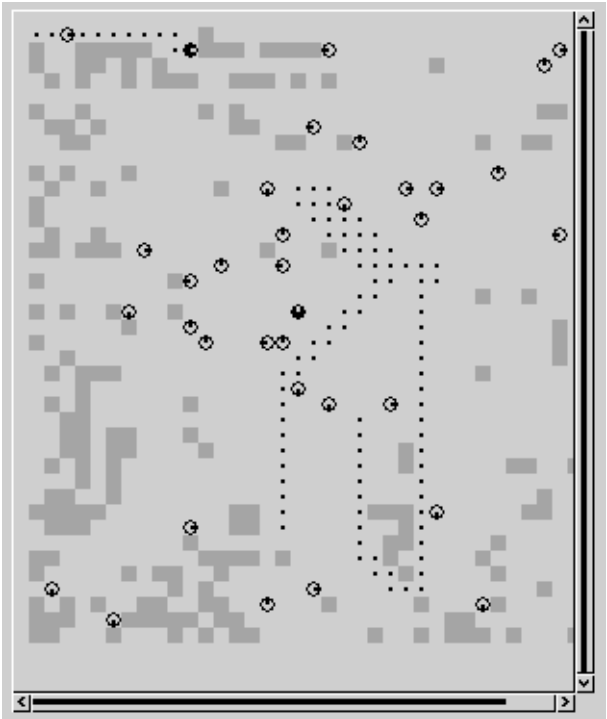

Figure 8.11 Essai de manipulation dans une autre configuration. Après perturbation, un agent essaie de nouveau plusieurs fois d'atteindre son but et à un moment donné il change définitivement de but (il change d'avis, on peut dire) ou il meurt. La trace visualisée est celle de l'agent noir haut à gauche qui est amené par force à ce coin et peu après il meurt.

L'adaptativité constitue le paramètre crucial qui indique si un agent est "vivant" ou pas. Parfois, l'adaptativité ne montre pas de maximum autre que celui à la naissance de l'agent, l'agent reste alors vivant par "inertie", comme s'il était condamné depuis sa naissance (et, il l'est, d'une certaine manière). Dans les autres cas, la courbe de l'adaptativité d'un système multi-agents (qui est calculée comme la moyenne des variables d'adaptativité des individus) va avoir une forme de cloche, tandis que les variables d'adaptativité des individus peuvent montrer un ou plusieurs maxima locaux (cf. figures 8.12, 8.13 et 8.14). L'adaptativité constitue également une mesure de la manipulabilité de l'agent dans un contexte social : si un agent en se déplaçant passe à côté d'un autre agent mort mais dont le rayon de perception est suffisamment élevé et l'adaptativité suffisamment près de son seuil, alors ce deuxième agent peut ressusciter et partir vers un nouveau but. Par conséquent, un système peut être mort tandis que ces composants seraient très vivants dans un autre contexte social.[100] Ceci amène à une autre question : peut-on (et *faut*-il) intervenir sur le système quand il est mourant (vieux) pour augmenter son adaptativité ? Cela semble a priori possible seulement dans les premières étapes de la vie du système (et de l'agent), où on peut par exemple remplacer un agent par un autre plus "jeune" ou injecter de nouveaux agents etc. Mais plus le système vieillit, plus il y a d'agents qui nécessiteraient un tel "remplacement". Qui plus est, si les relations sociales sont "fabriquées" par les agents eux-mêmes, comme ce serait par exemple le cas d'une extension du modèle cellulaire du chapitre précédent, rajeunir un système d'agents serait en principe possible mais pratiquement impossible (à moins d'avoir accès à tout le réseau des relations sociales entre agents ainsi qu'aux détails de leur mécanisme de révision de but, auquel cas on aurait mieux fait d'intervenir de façon plus drastique en brisant une fois pour toutes ces rétroactions nocives).[101]

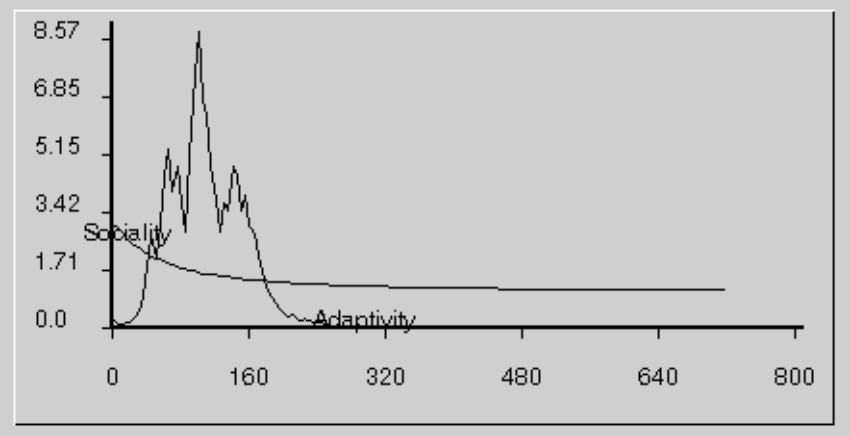

Figure 8.12 La courbe de l'adaptativité (la moyenne des variables d'adaptativité des individus) a une forme de cloche ! En fait, on peut dire que la courbe de l'adaptativité de l'agent montre son évolution développementale (le système développe jusqu'au point de sa

[100] Ces observations ont été stimulées par un chapitre de Robert Rosen (1992) qui décrit les problèmes que la sénescence pose et cite comme un exemple de stupéfaction des biologistes devant le mystère de la vie le problème des cellules qui meurent lors des manipulations (mais on peut également penser aux médecins frustrés devant les organismes malades qui meurent malgré le traitement).

[101] Je crois que le "secret de la vie" doit être observable, mais pas contrôlable.

vigueur maximale et ensuite il décline jusqu'à sa mort). De son côté, la socialité baisse constamment en raison de la dégradation et des boucles de rétroaction négative.

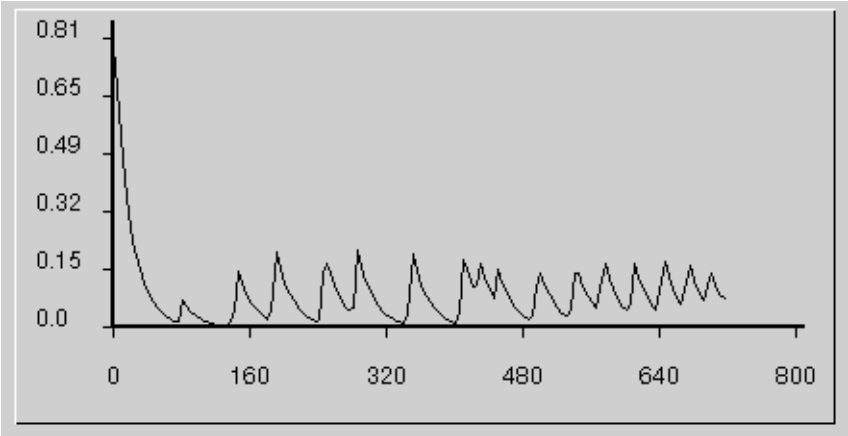

Figure 8.13 L'adaptativité d'un agent fait de petits sauts à chaque "régénération" ; les paramètres de cet agent sont : adaptativité=0.0918, socialité=0.941, timeouts=0, morts=16, regénérations=15.

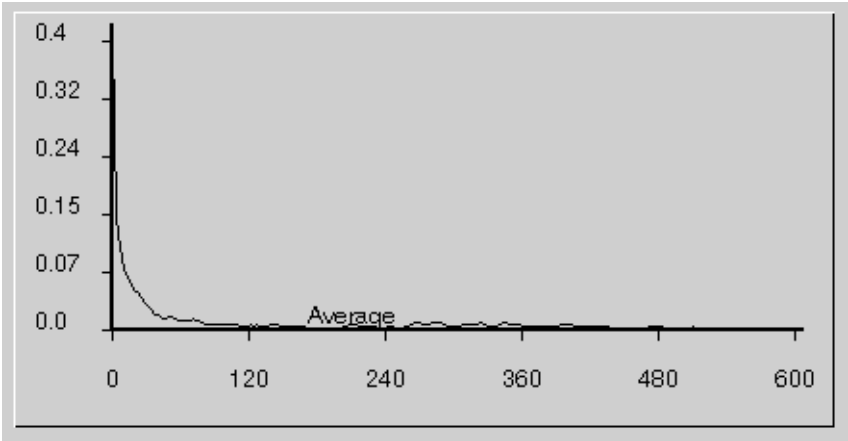

Figure 8.14 Parfois l'adaptativité baisse depuis le début sans présenter des maxima (ici c'est le cas d'un système de 100 agents qui est mort à t=514).

Notons finalement que le mécanisme de révision de but induit une forme de diversité dans les relations spatiales qui déterminent à leur tour les relations sociales (cf. les paragraphes 5.4 et 7.5.3 pour l'importance des générateurs de diversité).

8.4 Le modèle cognitif (autorégulant)

Le modèle du paragraphe précédent repose sur un réseau de fonctions couplées qui dégradent certains paramètres physiologiques de l'agent. Un agent pourrait vivre davantage s'il avait la possibilité de "retarder" la dégradation de ces paramètres à l'aide d'un mécanisme ayant un effet "régénérant". Mais un mécanisme ainsi régénérant peut être vu comme un mécanisme qui possède de la méta-connaissance, de la connaissance sur l'agent même.[102] Le retardement de la dégradation peut se faire dans le cas du modèle précédent par une régulation de ces mêmes paramètres, c'est-à-

[102] Comme nous l'avons dit dans le chapitre 1 et comme nous venons de le démontrer dans le cas de l'agent explorateur du chapitre 4, le système "cognitif" de l'agent est son système de régulation qui se situe à un niveau méta (en effet, chaque niveau imbriqué de régulation correspond à un niveau "cognitif" supplémentaire).

dire par une autorégulation de l'agent lui-même. Dans un effort de montrer la conservation de la relation entre individualité et socialité lors de la montée des niveaux de représentation/régulation, nous avons implémenté et comparé deux mécanismes d'autorégulation : un mécanisme de régulation de la distance du but nouveau-né et un mécanisme de régulation de la socialité.

Le premier est un mécanisme de régulation de l'individualité de l'agent et est basé sur l'observation suivante : si un agent vise à un but lointain, il risque de trop se distraire par rapport à son but et ainsi subir un timeout prématuré et devoir finalement réviser son but (comme effet de bord, le timeout lui-même diminuerait). Réguler la distance du nouveau but signifie que l'agent va choisir progressivement des buts plus proches ou plus lointains selon que la révision de son but est provoquée par un timeout ou par l'arrivée à son but. Ce mécanisme tend à réguler les motivations des agents de manière à éviter une très grande socialisation.

Le deuxième mécanisme régule directement la socialité et est basé sur l'observation que de toute façon la socialité du système baisse fatalement. Une régulation de cette socialité est alors envisageable et, puisqu'il n'existe pas de critère absolu pour la direction de régulation (hausse ou baisse)[103], nous avons adopté l'alternative d'une régulation selon une valeur endogène de référence : puisque le problème est la baisse de la socialité, on va essayer de la restaurer si sa valeur baisse au-dessous de la valeur de référence. Les deux modèles sont résumés dans la figure 8.15.

Modèle cognitif (autorégulant)

1 Régulation d'individualité
Système de régulation de distance du nouveau but
But atteint ⇒ *distance := distance*(1+r_D)*
Timeout ⇒ *distance := distance*(1- r_D)*
distance du nouveau but := distance(1+bruit)*
2 Régulation de socialité (restauration)
Système de régulation du facteur de socialité
avec une fenêtre w_S de régulation (adaptation)
Si socialité < socialité_propre,
socialité := socialité + r_S(socialité_propre-socialité)*
r_S := r_S(1-dr_S) (dégradation de r_S)*
et w_S := w_S(1-dr_S) (dégradation de w_S)*
r_D ∈ (0.2,0.5), r_S ∈ (0.2,0.5), bruit ∈ (-0.1,0.1), socialité_propre ∈ (2,4), fenêtre d'adaptation ∈ (20,30), dr_S ∈ (0.01,0.05)

Figure 8.15 Modèle cognitif (autorégulant)

8.4.1 Régulation d'individualité

La régulation de l'individualité est une régulation de la distance à laquelle le nouveau but est choisi lors des révisions de but (cf. fig. 8.16). Au début de la simulation, cette distance est choisie aléatoirement entre 0 et la distance maximale permise (la plus grande parmi les quatre distances de l'agent aux bornes de la grille).

On peut observer que, à la différence des autres modèles, ce modèle ne nécessite pas de rétroaction négative affectant les paramètres de régulation — ici le paramètre r_D —

[103] Il n'y a pas de moyen de savoir si on a trop ou trop peu socialisé ni d'estimer la valeur souhaitable de socialité dans une structure/configuration particulière.

parce que la distance n'est pas un paramètre dont la baisse (ou hausse) serait explicitement liée à la fonction de sénescence ; la régulation de la distance est un moyen pour réguler indirectement la socialité.

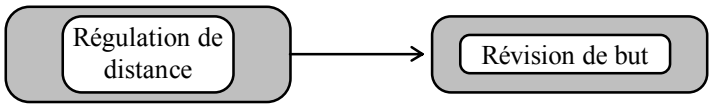

Figure 8.16 Modèle cognitif 1 : physiologie. La régulation de distance est indépendante de l'activité de l'agent et n'est pas affectée par le mécanisme de sénescence.

Ce mécanisme de régulation a été introduit pour montrer que les agents plus adaptatifs dans leur recherche de nouveaux buts (c'est-à-dire les agents qui, d'une certaine manière, "apprennent" de leurs échecs) vivent effectivement plus longtemps (par exemple, parce qu'ils choisissent des buts successifs étant près les uns des autres). Cependant, cette régulation n'a pas conduit à une augmentation considérable de l'espérance de vie par rapport au modèle précédent : en fait, les performances ont fluctué au-dessus et au-dessous de ces dernières. Pourquoi ? Une raison en est que, devant la socialité, l'individualité ne joue qu'un rôle mineur pour la survie de l'agent dans une société. Mais la raison la plus importante en est que cette régulation n'aurait un effet que dans le cas où il y aurait un sens à *choisir* la direction de régulation de distance et donc de socialité, c'est-à-dire à *choisir* entre baisse et hausse de socialité. Le modèle suivant de son côté repose sur une régulation du type hausse de socialité et il sera comparé à celui-ci dans le paragraphe 8.6. Pour l'instant, il suffit de remarquer que la régulation indirecte de la socialité via une régulation de but vers les deux directions a donné des performances instables, car seule la distance du nouveau but n'induit pas un certain degré de socialité plutôt qu'un autre. La raison en est qu'il reste un paramètre incontrôlable, la configuration actuelle des autres agents : choisir un but plus proche pour restreindre les conséquences de la socialité n'a aucun effet si tous les autres agents se trouvent à côté.

8.4.2 Régulation de socialité

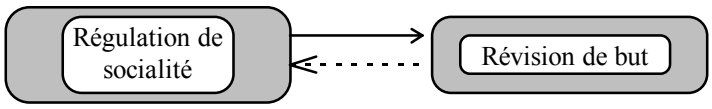

Figure 8.17 Modèle cognitif 2 : physiologie. La régulation de la socialité doit être un mécanisme sénescent aussi.

Nous avons implémenté ce mécanisme de régulation directe de la socialité (cf. fig. 8.18) pour montrer que, par comparaison avec les modèles précédents, il conduit à une plus grande espérance de vie du système multi-agents. Dans le tableau 8.1 sont donnés les résultats comparatifs obtenus avec les quatre modèles pour diverses tailles de populations d'agents. À part l'instabilité du modèle qui régule la distance de but et qui sera rediscutée dans le paragraphe 8.6, les autres résultats sont sans surprise : la régulation conduit à un système qui vit plus longtemps.

	Modèle 1	*Modèle 2*	*Modèle 3.1*	*Modèle 3.2*
30 agents	331.7	365.4	406.7	829.3
50 agents	354.2	409.3	404.5	814.6

Tableau 8.1 Tableau comparatif des durées de vie dans les trois modèles (dans un monde 25x25). Les résultats sont les moyennes de 20 simulations par cas. Modèle 1 = modèle réactif, modèle 2 = modèle autonome, modèle 3 = modèle cognitif (modèle 3.1 = modèle de régulation de distance de but, modèle 3.2 = modèle de régulation de socialité).

Le réseau physiologique de l'agent est représenté schématiquement dans la figure 8.18. Ses éléments essentiels sont les boucles de rétroaction négative et d'instruction (quelques liens secondaires de rétroaction positive et les boucles internes aux systèmes "programmatiques" ne sont pas visualisés).

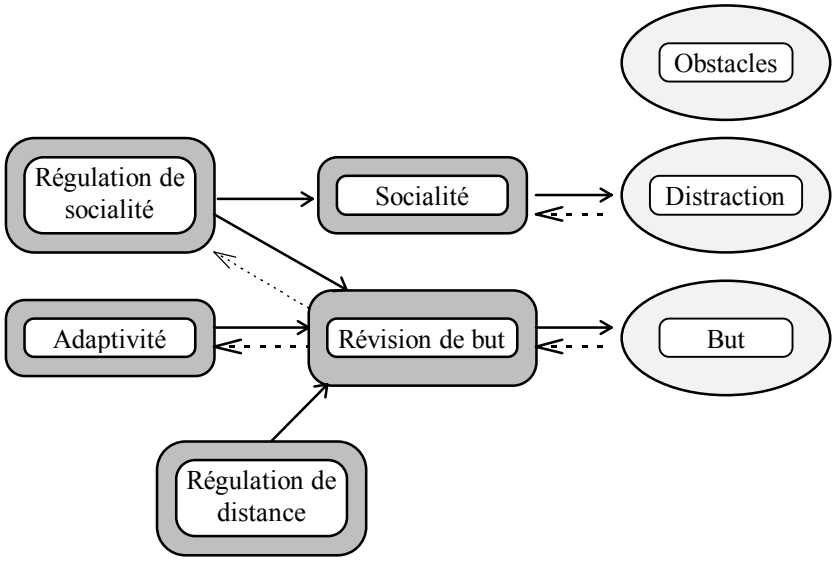

Figure 8.18 Modèle cognitif : réseau physiologique complet. *L'adaptativité globale de l'agent se catalyse négativement*.

8.5 Qu'est qu'un observateur en déduit ?

> *"Nos modèles sont-ils susceptibles de contrôle expérimental ? Peut-on, grâce à eux, faire des prévisions expérimentalement contrôlables ? [...] il me faut répondre à cette question par la négative."*
> *(Réné Thom, "Stabilité structurelle et morphogénèse", 1972)*

Après avoir observé le système moi-même et avoir constaté que le comportement des agents conduisait à la création de formes complexes et à l'émergence de phénomènes dynamiques sans importance par rapport au modèle sous-jacent, j'ai fait observer le système par des volontaires scientifiques pour voir quelles étaient les premières hypothèses qu'ils faisaient quant à la fonctionnalité du système et quelle "tactique" d'observation ils adoptaient ; dans leur majorité c'étaient des informaticiens, dont quelques uns des chercheurs en systèmes multi-agents. Afin de les guider dans leurs observations je leur disais depuis le début que les agents ont une sorte de mission ou de but et que le système est assuré de terminer (mourir). Ma question était alors, "que

font-ils et pourquoi meurent-ils ?". Finalement, et pour leur permettre de comparer, je leur faisais souvent regarder des démonstrations de mes modèles d'explorateurs seuls ou sociaux.

Conclusions des observateurs : Aucun des observateurs ne prenait comme point de départ la fonctionnalité des agents comme individus mais tous arrivaient à des conclusions descendantes du type "Ils cherchent à créer des configurations où il y a beaucoup d'agents concentrés dans des régions précises", "Ils cherchent à relier les ensembles connexes"[104], "Ils se dirigent vers les côtés de la grille, mais s'ils en sont trop nombreux ils s'éparpillent" ou "Ils cherchent à se regrouper, mais pas toujours". Puisque, comme je l'avais prévu, ils faisaient ce genre de conclusions sur la fonctionnalité du système, la deuxième question semblait les confondre davantage et parfois ils arrivaient même à réviser leurs premières hypothèses, comme par exemple "En effet, ils cherchent à consommer le gris, mais pas toujours — sinon ils seraient mieux organisés et plus efficaces —, et lorsqu'ils en ont consommé suffisamment ou lorsqu'ils n'en trouvent pas, ils meurent — sinon pourquoi s'arrêteraient-ils ?". Quant à la socialité des agents, la plupart des observateurs en ont conclu qu'il ne doit pas y avoir de socialité directe au sens d'une perception de l'individualité des autres agents et d'une recherche de la socialisation, mais qu'il y a des effets secondaires d'évitement, de regroupement etc. Seulement très peu d'observateurs ont décrit les phénomènes en termes de perturbations : "Un agent qui ne consomme plus est perturbé quand quelqu'un qui consomme passe à côté" ou "Ceux qui bougent beaucoup excitent les autres". Finalement, après observation des démonstrations des explorateurs, ils avaient tous tendance à former des hypothèses plus élaborées : "Ils doivent avoir une estimation personnelle de quelque chose, par exemple le nombre d'agents ; ils propagent alors cette information — sorte de prosélytisme — et en fonction d'elle ils s'arrêtent" ou encore "Ils doivent échanger des carrés consommés quand ils se rencontrent et quand ceux-là sont suffisamment repartis ils s'arrêtent".

Mode d'observation : La plupart des observateurs observaient le système passivement : seul un nombre limité d'entre eux ont voulu créer et observer leurs propres configurations d'agents qu'ils jugeaient importantes pour une raison ou pour une autre. Dans tous les cas, pratiquement personne n'a manifesté de goût à la manipulation, c'est-à-dire une volonté d'intervenir au système lors de la simulation pour voir par exemple les effets d'une modification locale particulière.[105] C'est-à-dire que pratiquement personne n'a jugé nécessaire de *manipuler* afin de mieux comprendre.

Méta-conclusions (mes conclusions sur les conclusions des autres) : Ma première conclusion était que les observateurs, en essayant de trouver des explications élaborées pour les phénomènes qu'ils trouvaient complexes, sous-estimaient les

---

[104] Sur la fenêtre de la simulation, les obstacles étaient représentés en gris sur un fond plus clair (cf. fig. 8.7), ce que j'expliquais comme "deux propriétés du sol" et que les observateurs interprétaient comme une consommation du gris.

[105] J'imagine que d'autres classes d'observateurs, tels que des biologistes ou des physiciens par exemple, auraient une plus grande tendance à vouloir manipuler le système. Dans un cas isolé, un observateur a déplacé un agent du modèle réactif loin de sa position finale après la mort du système et il a remarqué que cet agent est rentré à sa position précédente. Il a ensuite répété la même expérience plusieurs fois pour m'affirmer à la fin : "Oui, les configurations finales sont stables ; si je perturbe le système, il se re-stabilise toujours à la même configuration globale". Un deuxième observateur à qui j'ai montré cette expérience en a conclu : "L'agent a de la mémoire".

situations qui montraient des régularités (“Cet agent est allé tout droit et après il s’est arrêté, il est bête ; mais ces deux là-bas sont plus intelligents, ils cherchent plus”). Bates (1994) est arrivé à la même conclusion lors de la démonstration de ces agents (*“... it is the oddity, the quirk, that gives personality to a character, and it’s personality that gives life. If so, our architectures must support quirks, and this may mean they need to allow regularities, such as expressed in abstraction barriers, to be broken...”*, p. 124). Un des observateurs a pourtant remarqué, très introspectivement, “le problème est que quand le système s’arrête il n’est plus intéressant, mais quand il bouge, il bouge beaucoup trop et je ne peux rien dire”. Dennett (1987, p. 250-257) discute le problème des pièces de conviction anecdotiques que l’on rencontre en éthologie cognitive et propose une “méthode à la Sherlock Holmes” pour manipuler les animaux sous observation afin d’éviter d’être “trompé” par les animaux eux-mêmes. C’est précisément la constatation que la nature peut nous tromper qui m’a amenée à ma deuxième conclusion : en adoptant une attitude d’observation généralement intentionnelle, les observateurs s’affirment observateurs objectifs, mais seulement par rapport à leurs buts. On m’a par exemple dit : “Cela me suffit de regarder les agents qui courent ; pourquoi cela m’intéresserait-il d’en savoir plus ?”. Un tel observateur paraît alors relativement déçu d’apprendre que ces agents peuvent être complètement mécanistes : “Il peut y avoir simplement des variables initialisées aléatoirement dans chacun des agents et qui dictent quand l’agent termine, auquel cas il n’y a rien dans les agents et moi je me suis trompé dans mes observations.”

La conclusion globale, quant à la valeur d’une fonction de sénescence comme hypothèse de base d’un modèle d’apprentissage, est donc que la nature peut nous tromper (une “nature artificielle” a réussi à tromper facilement beaucoup de scientifiques sûrs d’eux-mêmes) et que les vraies causes de certains phénomènes peuvent être beaucoup plus simples qu’on ne le croit (il suffit de croire que la réalité peut être absurdément simple!).

8.6 Analyse/discussion des modèles et des résultats

8.6.1 Temps et connaissance

Comme pour les modèles des chapitres précédents, le paramètre critique du modèle de sénescence est le temps : le rythme du métabolisme des agents est responsable du rythme de vieillissement du système, *bien que les deux rythmes ne soient pas les mêmes*! Les boucles des rétroactions auto-catalytiques entraînent l’irréversibilité du temps et ainsi le système est assuré de mourir, malgré l’imprévisibilité de son comportement émergent. Ceci dit, la dynamique et la durée de la vie du système ne peuvent pas être déduites du paramétrage seul des agents : en dehors d’un contexte d’interaction, un observateur ne peut rien dire. Il suffit de mettre le même agent dans deux contextes différents pour observer des phénomènes/comportements différents. Dans ce sens, ***le temps comme paramètre de conception n’a pas la même “sémantique” que le temps comme paramètre d’observation*** : le temps comme paramètre d’observation peut avoir un sens ou un autre (en tant qu’observateurs du système nous pouvons *choisir* d’observer aussi bien les durées de vie que certains cycles générés par l’émergence des diverses formes). Cependant, pour l’agent lui-même, le temps comme paramètre de conception (par exemple, les paramètres timeout) a une seule signification : le potentiel de l’agent pour la survie, et ce potentiel n’est dissocié ni des autres paramètres comportementaux de l’agent ni de

son interaction avec son monde et avec les autres agents. Nous attendons d'un agent avec un timeout élevé de socialité de vivre longtemps, mais seulement si son rayon de perception lui permet de socialiser suffisamment et pas trop. Autrement dit, dans un système fondé sur un réseau de rétroactions entre métabolisme et programme, la durée de vie n'est ni préspécifiée ni spécifiable. En plus, la durée de vie *ne doit pas* être spécifiable, sinon un observateur externe pourrait manipuler artificiellement le système et prolonger ou raccourcir sa durée de vie à volonté. Nous pouvons inverser cette conclusion et dire que ***ce qui n'est pas spécifié ne peut être directement contrôlé, par conséquent le comportement d'un système autonome doit présenter un certain degré d'émergence***. Encore une fois, un agent manipulateur externe conserve une liberté limitée d'intervention (je suis arrivée à doubler la vie d'un système en lui injectant de temps en temps des agents tout neufs à des endroits précis) mais cette liberté provient du potentiel propre au système manipulé et n'est en aucun cas liée aux intentions du manipulateur.

Nous avons également constaté que la durée de vie du système augmente si les agents ont la possibilité de modifier leurs dynamiques d'interaction avec le monde et avec les autres agents. Le résultat important reste que, malgré cette possibilité, la durée de vie du système n'est toujours pas spécifiée d'avance. De toute façon, ***il semble que l'ascension des niveaux de régulation peut être vue comme une ascension des niveaux de cognition puisqu'elle fait vivre les agents plus longtemps***.[106]

8.6.2 Régulation et rétroaction

Nous avons montré que la durée de vie du système augmente si les agents possèdent des mécanismes d'autorégulation implicite ou explicite de la socialité qui est le paramètre moteur de sénescence. Les paramètres de ces mécanismes d'autorégulation ont été définis comme des paramètres sénescents puisque cela assure la mortalité du système. Nous avons pu montrer l'immortalité occasionnelle des systèmes dont les paramètres de régulation ne sont pas catalysés négativement par les mêmes rétroactions de sénescence (immortalité occasionnelle signifie que tous les systèmes ayant ce comportement ne sont pas nécessairement immortels). En premier lieu, la figure 8.19 donne les courbes de socialité et d'adaptativité pour un système du type cognitif sans rétroaction négative sur le facteur de restauration de socialité ; ce système est devenu immortel. La figure 8.20 montre ensuite l'évolution d'un système du même type avec une rétroaction négative sur le facteur de restauration, mais sans rétroaction négative sur la fenêtre de restauration. Le système n'est plus immortel, mais la baisse de la socialité est linéaire, donc il est plus "prévisible" au-delà d'un certain point. Un troisième système du même type avec des rétroactions à tous les niveaux est examiné dans la figure 8.21 qui montre une évolution de socialité typique des systèmes auto-catalytiques et notamment un phénomène d'avalanche ou de catastrophe. Nous avons également montré qu'un système du type motivé sans baisse des facteurs f_1, f_2 et *timeout* peut aussi devenir immortel, mais cette fois son immortalité se manifestera non pas comme une stabilisation de la socialité avec une adaptativité qui tend vers 0, mais comme une stabilisation de l'adaptativité avec une

[106] L'absurdité de la relation entre durée de vie et satisfaction individuelle des agents est reflétée dans le monologue d'un de mes observateurs : "Ce n'est pas normal, il y en a qui finissent plus tôt que les autres ! Mais pourquoi alors les autres continuent-ils à chercher ? Cherchent-ils quelque chose que les premiers ont déjà trouvé ?"

socialité qui tend vers 0. La conclusion de ces expériences est qu'***un système est sénescent si les paramètres de son niveau cognitif le plus élevé (c'est-à-dire de son niveau de régulation le plus élevé) sont décroissants***. En plus, l'auto-catalyse telle qu'elle est définie (avec des rétroactions récursives) n'est pas un simple embellissement du mécanisme pour le rendre plus proche à la réalité biologique : l'autocatalyse est le mécanisme de sénescence le plus simple (la décroissance préprogrammée, et alors monotone, étant exclue à cause de prévisibilité et par conséquent de manipulabilité relative). ***L'auto-catalyse est alors une nécessité et en son absence le système risquerait de devenir immortel***.

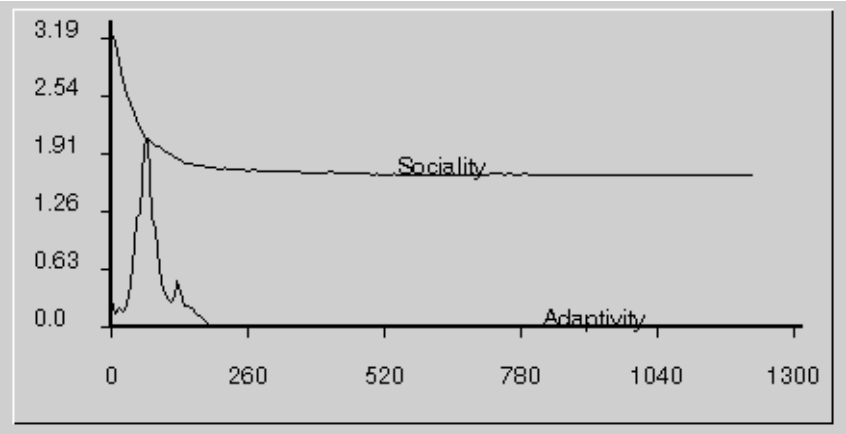

Figure 8.19 Nécessité d'agir sur les paramètres de régulation. La courbe de l'adaptativité et de la socialité d'un système (du type cognitif) "immortel" qui s'est stabilisé à une valeur de socialité (1.65) avec une adaptativité qui tend progressivement vers 0. Ce résultat est dû à l'absence de rétroaction négative qui agit sur le facteur de restauration de la socialité.

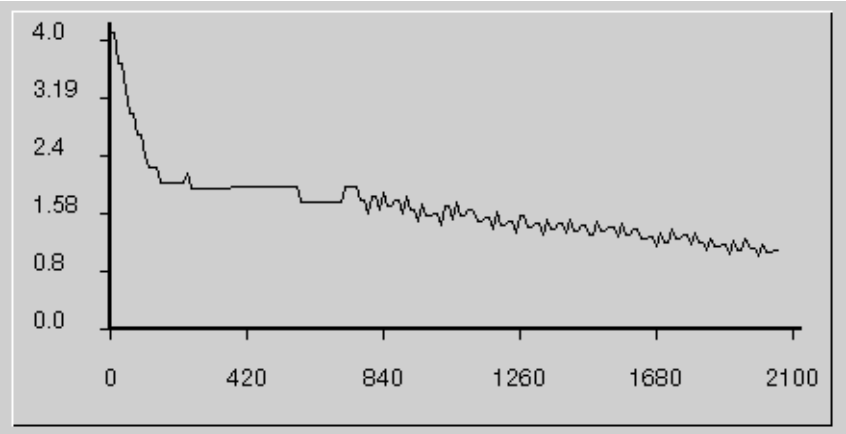

Figure 8.20 Mais *tous* les paramètres de régulation doivent être affectés par le mécanisme de sénescence. La socialité d'un agent dans un système "cognitif" avec rétroaction négative sur le facteur de restauration de socialité mais pas sur le timeout de restauration. Les essais de restauration de socialité sont d'une fréquence uniforme (alors la baisse de la socialité est quasi-linéaire).

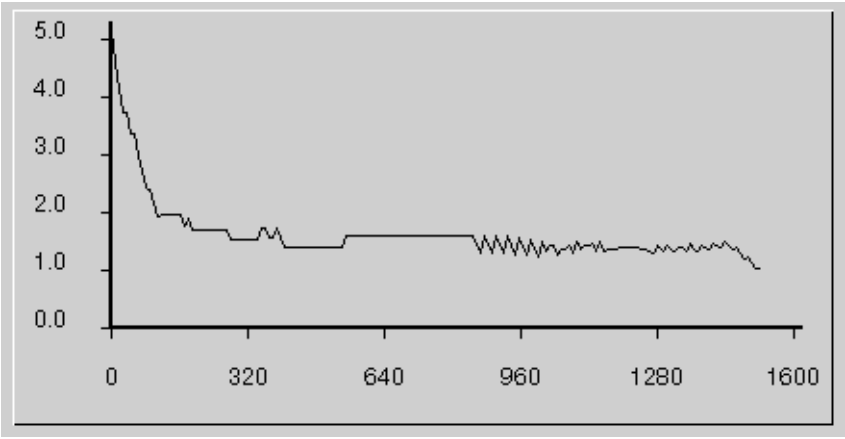

Figure 8.21 La socialité d'un agent dans un système "cognitif" avec des rétroactions à tous les niveaux. Les essais de restauration de socialité ne sont pas de la même fréquence et la socialité montre une dernière "brusque" chute, autrement dit un effet de catastrophe ou d'avalanche (puisque la baisse du facteur de restauration est auto-catalytique par l'intermédiaire de la baisse de la fenêtre de restauration).

8.6.3 Effets de surpopulation

Puisque l'espérance de vie du système augmente quand on régule directement ou indirectement la socialité qui, de son côté, baisse naturellement, j'ai voulu tester une dernière hypothèse : la durée de vie doit être faible pour les petites populations d'agents qui sont dispersés dans l'espace et ne socialisent pas beaucoup, ensuite elle doit monter pour les agents plus nombreux qui socialisent et qui régénèrent plus souvent et finalement elle doit être de nouveau faible pour les agents trop nombreux qui tombent les uns sur les autres tout le temps. Mon hypothèse est donc que ***le système peut donner naissance à un phénomène de "cancer" si son espace vital est petit par rapport au nombre de ses agents. En plus, un mécanisme qui augmente la durée de vie du système décalera nécessairement le point d'émergence de ce phénomène vers des populations plus grandes par rapport à une même taille de l'espace vital.*** Mon objectif durant ces expériences n'est *pas* de donner un modèle de croissance de population expliquant la surpopulation, mais plutôt de montrer que la surpopulation conduit à une mort prématurée. C'est-à-dire, nous n'expliquons pas comment les agents peuvent devenir si nombreux, mais plutôt pourquoi le système meurt une fois il apparaît autant d'agents.

J'ai alors effectué des expériences dans les deux directions : pour une gamme de tailles de population dans un monde de taille constante, ainsi que pour une gamme de tailles de monde avec une population de taille constante. Dans le premier cas, j'ai obtenu des résultats compatibles avec cette hypothèse dans un monde 40x40 et avec une population de 10 à 1000 agents[107], mais la complexité des simulations étant importante et le temps des expérimentations extrêmement long, la quantité des échantillons statistiques n'a pas été suffisante pour généraliser en sécurité. Dans le deuxième cas, les résultats étaient plus intéressants, malgré la même insuffisance des échantillons statistiques : selon l'hypothèse de départ, il devrait exister une taille de population optimale pour une taille de monde donnée et inversement une taille de monde optimale pour une taille de population donnée. Cela est vrai statistiquement

[107] Une simulation avec 500 agents dans un monde 40x40 peut durer de 3 à 5 heures sur une station de travail SUN.

pour les populations, mais pas toujours pour les configurations spécifiques ; une population particulière d'une taille donnée peut vivre progressivement plus longtemps dans des mondes progressivement plus grands, mais une autre population de la même taille peut vivre progressivement plus longtemps dans des mondes progressivement plus petits (la taille des populations allait des 10 aux 80 agents et la taille du monde allait de 15x15 à 40x40). La raison en est que seules les tailles de la population et du monde ne suffisent pas pour "déterminer" la durée de vie du système ; la configuration des agents dans l'espace ainsi que leur paramétrage relatif sont tout aussi importants puisque le degré de socialisation en dépend. De plus, une population d'une certaine taille dans un monde donné peut vivre progressivement plus longtemps si l'on applique les modèles comportementaux motivé, autorégulant selon la distance et autorégulant selon la socialité. Cependant, d'autres populations de la même taille dans le même monde peuvent avoir des relations complètement différentes de durée de vie entre le modèles, toujours à cause des configurations initiales et paramétrages différents. Ces résultats signifient qu'il n'y a *pas de critère absolu d'émergence de cancer*, c'est à dire de critère qui puisse s'appliquer en dehors d'un contexte spécifique de configuration spatiale et de paramétrage et, encore plus important, même si la configuration et le paramétrage sont donnés, on ne peut pas prévoir d'avance l'évolution du système, du moins quantitativement.

Dernière observation, dans le cas d'une population importante située dans un petit monde (80 agents dans un monde 15x15) le modèle de régulation de distance de but conduit souvent à des durées de vie supérieures à celles qu'on trouve si l'on applique les autres modèles. À première vue, cela paraît contradictoire aux analyses précédentes, mais en regardant de plus près, nous pouvons attribuer l'instabilité de ce modèle à sa dépendance de facteurs exogènes à l'agent (tel que la distance des buts qui dépend à son tour de la configuration exacte du système) plutôt que de facteurs endogènes (telle qu'une valeur de référence de la socialité).

8.7 Conclusion : À la recherche du sens de la vie et de la mort

Ce chapitre a présenté un modèle de sénescence par rétroaction entre le métabolisme et le programme de l'agent et trois modèles consécutifs qui augmentent la durée de vie de la population. Il a été montré que chaque niveau de régulation supplémentaire conduit à une augmentation de la durée de vie du système et que cette durée n'est ni préspécifiée ni déduite des phénomènes. Il a été également montré que, dans ce système social, la régulation de la socialité est beaucoup plus importante que la régulation de l'individualité, puisque la socialité est, elle aussi, beaucoup plus importante que l'individualité. Pour assurer la mortalité de l'agent, la rétroaction de sénescence doit affecter tous les paramètres du niveau de régulation le plus élevé. En plus, lorsque le nombre des agents est grand par rapport à la taille de l'espace vital du système, il apparaît un phénomène de "cancer", c'est-à-dire de baisse de l'espérance de vie.

Ce mode de couplage entre programme et métabolisme favorise l'émergence des boucles de régulation imbriquées qui augmentent l'espérance de vie du système. Selon la terminologie du chapitre 1, ces boucles imbriquées de régulation sont des niveaux cognitifs imbriqués. Nous n'avons pas décrit *comment* ces boucles, ces niveaux de régulation pourraient émerger ou évoluer, nous avons seulement montré leur avantage "sélectif".

Il a déjà été souligné plusieurs fois que la fonction de sénescence est vue comme une force motrice de l'apprentissage, par exemple dans un réseau plastique comme ceux décrits dans le chapitre précédent. Il reste deux questions ouvertes :

(a) ***La question de la socialité*** : dans le modèle, la socialité d'un agent doit baisser "brusquement" lors de chaque rencontre avec un autre agent. Pourquoi ? Une première réponse est que la socialisation bâtit une sorte de "standards sociaux" de façon que l'agent devienne de moins en moins sensible à la présence des autres agents (il est de moins en moins perturbé par la présence des autres agents), c'est-à-dire que la socialisation crée une véritable *culture*, un *ordre*.[108] Dans l'exemple précis du chapitre précédent, la culture pourrait alors être la stabilisation sociale des cellules, tandis que la mort de quelques unes injecterait du bruit dans le système, perturberait les cellules encore "vivantes" et stimulerait le processus inhérent d'auto-organisation dans le réseau.

(b) ***La question de l'horloge de la vie*** : dans nos simulations, le séquenceur de la simulation arrête le temps lorsque tous les agents-composants sont morts et cet arrêt constitue la mort du système. Dans un organisme vivant pluricellulaire, quel est ce saut qui fait "mourir" l'organisme et quand est-ce qu'il survient ? Quel est ce système central superviseur qui "arrête" le fonctionnement du système même si beaucoup d'agents restent vivants ? Quel est le coeur du système qui s'arrête de battre ? Quel est le genre "d'information" sur lequel cette supervision repose ? Et d'où vient-elle cette information et comment se propage-t-elle ?

Je ne peux pas deviner ces réponses — s'il est possible de trouver des réponses, même au niveau restreint de la modélisation. Je sais seulement que, si j'étais dieu, je créerais un système sénescent comme celui-ci. Ce serait suffisamment imprévisible pour être intéressant à regarder et je n'aurais pas à m'inquiéter de son destin.

[108] Beaucoup d'auteurs (Schrödinger 1944) (Lévi-Strauss 1962) (Dyson 1985) (Morin 1977) discutent de la complementarité des processus naturels — du biologique au social — générant de l'ordre et du désordre, et qu'il s'agit de maintenir le jeu infiniment.

Le problème	Un modèle de sénescence continue qui assure la mort de l'agent sans préspécifier la durée de sa vie. Ce modèle de sénescence doit favoriser les agents sociaux plutôt que les agents solitaires.
L'application	*Un problème abstrait de poursuite de but avec une socialité du type distraction* *Révision de but* *Régulation de distance de but ou de socialité*
La solution	Sénescence = Rétroaction négative entre métabolisme et programme
Les enseignements	• Modèle réactif ⇒ Manipulabilité • Modèle réactif vs. modèle motivé (autonome) vs. modèle autorégulant (cognitif) • Régulation de distance de but moins performante que la régulation de socialité • Chaque niveau supplémentaire de régulation augmente la durée de vie • Régulation de distance de but ⇒ Performances instables • La rétroaction de sénescence doit affecter tous les paramètres du niveau plus élevé de régulation • Surpopulation ⇒ Baisse de l'espérance de vie

Tableau 8.2 Tableau récapitulatif du chapitre 8

Chapitre 9 Évaluation

"Comprendre, c'est se changer, aller au-delà de soi-même."
(Jean-Paul Sartre, "Questions de méthode", 1960)

9.1 Évaluation quantitative : Conclusions

Nous avons étudié un ensemble de problèmes à deux niveaux d'organisation d'agents autonomes, le niveau de l'animat et le niveau cellulaire, avec un double objectif de résoudre ces problèmes particuliers et d'abstraire de certaines de ces solutions des principes d'organisation d'agents autonomes indépendant de niveau d'organisation.

Problème 1. Au niveau de l'animat, nous avons étudié le problème de description et de conception des systèmes de contrôle d'agents situés. Nous avons explicité notre choix de base, qui était de rester dans la problématique connexionniste algorithmique plutôt que de classification et de considérer l'architecture comme un langage de programmation plutôt que comme une structure panacée, passe-partout. La cellularité est définie comme le mode d'organisation d'un réseau de composants ayant une syntaxe de connexions homogène, mais des fonctions de transfert hétérogènes ; cela doit permettre à la fois une efficacité face aux problèmes spécifiques et une incrémentalité de conception. La partie cellulaire de l'organisation est couplée avec une partie physiologique qui contient des variables binaires ou continues commandées par plusieurs endroits dans le réseau et des structures réflexes. La forme hiérarchique de l'organisation a été ensuite décrite : cellule, agrégat de cellules, tâche (agrégat d'agrégats). La tâche homéostatique a été définie comme le quantum de tâche dans l'organisation et deux systèmes de contrôle d'agent autonome ont été implémentés informatiquement et testés en simulation, celui d'un agent explorateur et celui d'un agent gestionnaire d'un atelier industriel. Nous sommes alors passés à la considération du problème de l'action et de la sélection, pour parler d'unification structure-action, de sélection d'action qui est reflétée dans la structure, de composition d'action et de séparation des différentes dynamiques dans l'organisation. Le besoin de définir la plupart des actionneurs comme des réflexes non commandés à l'intérieur de l'agent a été analysé. Le problème du dimensionnement a été discuté sous l'angle de la compositionalité de l'action, de même que le besoin d'avoir des structures physiologiques d'adaptation dans l'organisation. Finalement, le problème de l'espace de représentation a été présenté et une brève analyse a été tentée à base des exemples précis concernant les problèmes de la concaténation et de la séparation des messages, le problème de l'identification et de la sémantique dans le réseau (cf. tableau 3.3).

Problème 2. Nous avons ensuite étudié le problème de balayage, une instance du problème d'exploration, dans le cas d'un seul agent : comment un agent fait-il pour épuiser toutes les sources d'intérêt dans un espace délimité et comment reconnaît-il

l'accomplissement de sa mission ? Il a été montré qu'un agent qui possède deux tâches exprimant des motivations couplées et une motivation récursive, c'est-à-dire une représentation de son besoin, constitue une solution simple à ce problème. L'état du monde dans lequel l'agent se trouve constitue une source de perturbation persistante pour l'agent, qui doit modifier la dynamique de son interaction avec le monde afin de mieux répondre à son besoin, et donc mieux résoudre notre problème. La modification de la dynamique d'interaction concerne les taux d'adaptation de l'agent et est homéostatique ; le couplage agent-monde est alors opérationnel. La version opérationnelle de l'agent-balayeur a été comparée avec d'autres versions non opérationnelles : l'adaptation exogène (selon les perceptions de l'agent au lieu de son signal de rétribution interne) et la méta-adaptation dépendante (dans laquelle le taux d'adaptation est proportionnel à la perturbation). Finalement, nous avons mené une étude du problème dans des mondes plus grands pour aboutir à la conclusion que, puisque la taille du monde n'est pas un paramètre libre du problème du balayage, aucune autorégulation d'aucun paramètre, ne pourra donner un système de contrôle capable de résoudre le problème du balayage dans un monde de taille quiconque (cf. tableau 4.1).

Problème 3. Le problème du balayage a été également étudié dans le cas de plusieurs agents au lieu d'un seul. Premièrement a été étudié l'évolution des performances en fonction de la taille de la population et il a été montré que ces performances arrivent à un point de saturation. Ensuite a été étudié l'effet de l'introduction d'une possibilité supplémentaire de dispersion comme une action instrumentale de fourragement/chasse. Cette fois il y a eu une baisse des performances, qui est due à l'instabilité des formes spatiales dynamiques. L'implémentation et la comparaison de plusieurs modèles de socialité réactive ont conduit à la conclusion que la socialité nécessaire est du type "partage global du besoin". Une première version non opérationnelle de socialité tit-for-tat a été analysée : le manque d'opérationalité semble être dû à la stabilisation de la dynamique de l'interaction agent-monde. Nous avons présenté également d'autres observations assorties, le phénomène du saut social, le rôle de la diversité et le comportement du système en présence d'agents-trompeurs. Finalement, le manque de besoin d'autorégulation a été attribué à l'absence de paramètre libre lié directement à la socialité et à la présence d'une seule force sociale dans le système multi-agents (cf. tableau 5.1).

Problème 4. Toujours au même niveau de l'animat, nous avons étudié le problème d'un agent chargé de la régulation d'un atelier industriel. Il a été premièrement montré que le système motivationnel de l'agent nécessite un mécanisme de persistance. L'étude d'un ensemble de modèles de persistance a révélé que le modèle opérationnel est celui d'une persistance dissipative dont les paramètres sont couplés avec ceux du processus industriel ou de la tâche de la régulation. Le séquencement de l'activité de l'agent a été transcrit dans une physiologie élaborée qui consiste en un ensemble de variables binaires ou continues régulant le système cellulaire de contrôle globalement. Une dernière étude au niveau d'une population d'agents managers a montré que la spécialisation n'est pas avantageuse à cause de l'indépendance des unités de production et de l'uniformité de "l'information" accessible aux agents. L'importance de la présence de bruit, au sens de l'asynchronisme entre événement et perception de l'agent, a finalement été soulignée (cf. tableau 6.3).

Problème 5. Ensuite, nous avons étudié le problème de l'opérationalité et de l'intégrité d'un réseau cellulaire auto-organisant ayant une topologie dynamique

résistante et robuste aux pannes de matériel, c'est-à-dire à une perte de cellules. Le modèle cellulaire entrée-sortie du problème 1 doit être étendu en un *modèle sélectif et motivé* de cellule autonome. Ce modèle étendu consiste à voir la cellule comme un système de composants, les pulsions, exécutant en parallèle et étant en compétition permanente entre elles pour la consommation des messages que la cellule reçoit. Si les pulsions sont chargées de la consommation des messages, la physiologie de la cellule est responsable de la coordination des pulsions et de la résolution des conflits ; de même que dans le cas de l'animat (problème 1), cette physiologie est indépendante. Pour fonctionner, le mécanisme nécessite des messages composés d'une partie d'identification et d'une partie de donnée. Certains de ces messages doivent être des catalyseurs qui déclencheront des actions métaboliques (des réactions) chez des cellules sans être vraiment consommés. L'opérationalité du modèle et sa robustesse face aux pannes sont assurées d'un mécanisme supplémentaire de modification des dynamiques d'interaction des pulsions avec le monde des messages ; de même que dans le problème 2, cette modification concerne les paramètres qui déterminent la dynamique d'interaction (ici les vitesses de consommation des messages) et est homéostatique. Contrairement au problème 2, cependant, il n'y a pas de représentation explicite du besoin, il n'y a pas de méta-motivation. Quant à la socialité, elle est encore une fois du type "partage du besoin" et se manifeste comme adaptation de l'agent-cellule à la forme des messages rencontrés. Dans le cas des messages intrus et des virus, qui peuvent aliéner le réseau, c'est-à-dire lui faire donner des fausses réponses à ses perturbations, un système cellulaire de défense parallèle au premier, c'est-à-dire un système immunitaire, s'avère nécessaire. Le système de défense repose sur la reconnaissance et l'élimination des messages étrangers. Le couplage entre système immunitaire et système "de production" se manifeste dans la complémentarité des messages traités par les deux systèmes, il s'agit donc d'un couplage entre deux forces sociales au sein du même agent. Le problème de l'espace de représentation a été rediscuté dans ce nouveau contexte sous une perspective future d'apprentissage (cf. tableau 7.1).

Problème 6. Une argumentation sur la nature de l'autonomie (présentée dans le paragraphe 1.3.6), nous a ensuite conduits à la constatation qu'un élément essentiel (ou même l'élément primaire) de l'autonomie doit être la sénescence. La sénescence est vue à la fois comme la force motrice de l'apprentissage et comme la base de l'émergence récursive de structures organisationnelles. Par exemple, dans le cas du mécanisme d'auto-organisation cellulaire présenté, une force induisant des pannes continûment pourrait servir de source permanente de perturbation et donc de moteur de l'auto-organisation et d'apprentissage. Un modèle de sénescence est alors élaboré, qui repose sur une rétroaction entre métabolisme et programme de l'agent, ainsi que trois modèles consécutifs qui augmentent l'espérance de vie du système multi-agents. Il a été montré que chaque niveau de régulation supplémentaire conduit à une augmentation de la durée de vie du système et que cette durée n'est ni préspécifiée ni déduite des phénomènes. Il a été également montré que, dans ce système social, la régulation de la socialité est beaucoup plus importante que la régulation de l'individualité, puisque la socialité est, elle aussi, beaucoup plus importante que l'individualité. Pour assurer la mortalité de l'agent, la rétroaction de sénescence doit affecter tous les paramètres du niveau plus élevé de régulation. En plus, lorsque le nombre des agents est grand par rapport à la taille de l'espace vital du système, apparaît un phénomène de "cancer", c'est-à-dire de baisse de l'espérance de vie (cf. tableau 8.2).

Principes d'organisation d'agents autonomes. Après une "méta-étude" des diverses études menées et après la comparaison des solutions aux divers problèmes posés, nous avons abouti que les agents autonomes des deux niveaux d'organisation, les agents-animats et les agents-cellules, avaient un nombre de propriétés en commun d'où nous avons déduit un nombre de principes d'organisation d'agents autonomes qui abstraient les deux types d'agents. Nous avons alors identifié les agents autonomes comme des entités ayant un système motivationnel et étant opérationnellement couplés avec leur environnement, c'est-à-dire il existe un couplage, une correspondance entre agent et monde, telle que l'agent doit s'autoréguler continûment pour accomplir sa mission et pour être opérationnel. La mission de l'agent est "décrite" dans son organisation comme un besoin que l'agent veut ramener à 0 et l'autorégulation concerne non pas ce besoin premier, mais un ou plusieurs paramètres architecturaux (organisationnels) qui déterminent la réaction de l'agent à ses perturbations. Nous avons plus précisément constaté que, dans les deux niveaux d'agents étudiés, ces paramètres organisationnels autorégulés sont en effet de paramètres ayant une signification temporelle, tels que de taux ou de vitesses d'adaptation, qui déterminent donc la dynamique (temporelle) de l'interaction de l'agent avec son monde ou la dynamique de la perturbation (cf. tableau 9.1).

	Agent explorateur cellulaire	***Agent cellule***
Motivations	*Chasse, retour*	*Différentes "pulsions"*
Méta-motivation	*Oui (en raison du problème de terminaison)*	*Non*
Socialité	*Adaptation sociale*	*Adaptation sociale (développementale)*
Autorégulation (de dynamique d'interaction)	*Paramètres d'adaptation*	*Vitesses de réaction (consommation des messages)*
Apprentissage	*Sénescence (en perspective)*	

Tableau 9.1 Tableau comparatif des propriétés des agents des deux niveaux d'organisation (cf. fig. 1.1, tableau 1.1)

Dynamiques. L'importance de la dynamique de l'interaction agent-monde est précisément l'idée qui relie les différentes solutions, les différents modèles. À propos de l'organisation cellulaire du chapitre 3, nous avons parlé de séparation des dynamiques et de besoin d'une physiologie indépendante régulant la dynamique globale du réseau. L'agent balayeur du chapitre 4 a eu besoin d'un mécanisme d'autorégulation de dynamique pour rester opérationnel. Les balayeurs sociaux du chapitre 5 ont dû avoir une dynamique sociale synchrone avec la dynamique individuelle, plutôt que parallèle et indépendante, et cette dynamique a dû définir un changement continu. L'agent manager du chapitre 6 a vu son problème de planification se transcrire dans sa physiologie comme un couplage de variables qui suivent des dynamiques différentes. Les cellules autonomes du chapitre 7 sont autonomes parce que, tout comme le balayeur du chapitre 4, elles possèdent un mécanisme d'autorégulation de leurs dynamiques d'interaction. Finalement, dans le modèle de sénescence du chapitre 8, c'est la dynamique des interactions (du métabolisme) qui est responsable de la durée de vie et une régulation de cette dynamique conduit à des systèmes qui vivent plus, qui paraissent donc plus cognitifs (cf. tableau 9.2).

Niveaux d'organisation. Nos objectifs sont-ils atteints ? L'objectif du premier ordre, qui est de trouver des solutions spécifiques aux problèmes posés, est atteint : nous

avons élaboré des mécanismes ou des modèles d'agents qui répondent à chacun des problèmes étudiés. L'objectif du deuxième ordre, qui est d'élaborer des principes d'organisation d'agent autonome indépendants de son niveau d'organisation, est également atteint mais il retient un caractère restreint : si les principes abstraient les propriétés des deux types d'agents, il n'est néanmoins *pas* prouvé que ces mêmes principes peuvent s'appliquer à d'autres problèmes analogues. C'est-à-dire, nous ne fournissons pas de preuves formelles en faveur de la généralité absolue de ces principes, mais seulement des indications empiriques de généralité.[109] Ce qui paraît intéressant en soi dans la généralisation effectuée est que les principes qui ont été révélés utiles ou pratiques dans une optique de conception sont similaires à ceux que la nature a "inventés" lors de toute l'histoire du vivant sur notre planète ; le meilleur exemple est l'importance des paramètres temporels qui déterminent les dynamiques des interactions d'un agent avec son monde.[110]

	Intégration de différentes dynamiques	***Régulation de dynamiques***
Agent cellulaire	*oui*	
Agent explorateur solitaire	*oui*	*oui (homéostatique)*
Agent explorateur social	*oui*	
Agent manager	*oui*	
Agent cellule	*oui*	*oui (homéostatique)*
Agent sénescent	*oui*	*oui (selon une valeur de référence)*

Tableau 9.2 Tableau des enseignements concernant les dynamiques d'interaction.

9.2 Évaluation qualitative : Perspectives

Niveau	***Perspectives***
Cellule	*Apprentissage (cf. agent cellulaire)* *Extensions des modèles cellulaires*
Agent cellulaire	*Autres applications* *Robotique* *Apprentissage (cf. cellule)*
Agents sociaux	*Autonomie sociale*

Tableau 9.3 Tableau des perspectives

Puisque notre objectif a été double et notre démarche est systémique, l'ouverture de l'approche est multiple (cf. tableau 9.3). D'une part, on peut aborder de nouveaux problèmes d'agents autonomes et surtout dans des domaines ou des contextes différents. D'autre part, plus nous disposons des solutions particulières à des problèmes spécifiques, plus nous pourrons nous poser des questions de deuxième ordre. Finalement, il reste à voir si les principes d'organisation d'agents autonomes sont complets et sinon ce qui leur manque et pourquoi.

Autres applications. J'envisage d'aborder d'autres problèmes impliquant des agents autonomes et inspirés de la robotique comportementale, tels que le "box pushing problem" (Parker 1994; Kube & Zhang 1994; Mataric et al. 1995) et le problème de

[109] Dennett (1987) serait tenté d'appeler ces indications des pièces de conviction.

[110] *" ... dans tous les cas, la régulation des systèmes biologiques opère sur des équilibres et des vitesses de réaction."* (Jacob 1970, p. 323)

coopération en présence d'une ressource partagée (Steels 1994*c*; Numaoka 1995). Ces deux problèmes sont intéressants du fait d'inclure des aspects "sociaux" non abordés jusqu'à présent, à savoir la coordination et la compétition respectivement.

Robotique. La tradition de l'IA comportementale veut que toute architecture soit validée sur un robot réel. Dans cette perspective, une implémentation robotique présente quelques particularités intéressantes : comment gérer une multitude d'objets informatiques (en l'occurrence, les cellules) se mettant en activité ou en sommeil de façon asynchrone et comment respecter les contraintes de temps réel ? La définition des systèmes physiologiques indépendants complique les besoins de gestion. À côté de ces considérations portant sur l'aspect système informatique de support, est-il possible d'élaborer une véritable méthodologie de conception qui traduirait un système de contrôle cellulaire en un programme compilé informatique ? La question la plus importante demeure pour autant de savoir s'il est possible d'implémenter les systèmes de contrôle cellulaire de façon totalement distribuée. Les recherches en génie neuromorphe (Sejnowski & Koch 1994) sont précisément centrées autour de l'élaboration de matériel analogique VLSI pour l'implémentation de systèmes neuromorphes.

Apprentissage. Comme on l'a déjà souligné plusieurs fois jusqu'à maintenant, une grande question ouverte est celle de l'apprentissage : je dois pouvoir montrer que la combinaison du modèle de cellule autonome du chapitre 7 avec une instanciation de la fonction de sénescence du chapitre 8 et avec un générateur de diversité peut donner naissance à des "comportements appris", par exemple à un apprentissage spatial. S'il est possible de trouver un tel mécanisme d'apprentissage, il sera ensuite intéressant d'explorer les types d'apprentissage que cette approche permet. Il faudrait également trouver au moins un cas où cet apprentissage dû à la mortalité/sénescence permettrait l'émergence au sens de la "création" de nouvelles organisations d'ordre supérieur.

Extensions des modèles cellulaires. J'envisage deux extensions immédiates des deux modèles cellulaires introduits. La première concerne la régulation sociale de la cellule autonome du chapitre 7 : comme je l'ai expliqué dans ce chapitre, la régulation sociale doit reposer, elle aussi, sur une régulation de vitesses. Pour ce permettre, il faut cependant introduire des populations des cellules ayant la même fonction de transfert, auquel cas la régulation dépendra d'une mesure de densité de messages. Cette extension est envisageable dans un contexte d'apprentissage développemental, dans lequel il y aura une population dynamique. La deuxième extension concerne la discrimination entrées-sorties, dont j'ai parlé dans le chapitre 3 : dans ce modèle une cellule consultera son environnement *sans* distinguer des connexions (ou des buffers) d'entrée et de sortie. Son "but" sera donc de maintenir une certaine densité de messages dans son environnement, ce qui équivaut à maintenir un équilibre entre ses pulsions internes (dans le modèle à connexions du chapitre 1, une connexion "suffoquera" et se noiera si les messages qui y voyagent ne sont pas consommés, donc la topologie du réseau sera également dynamique). La motivation derrière ces extensions est de comprendre dans quel cas c'est avantageux d'avoir des connexions et pourquoi.[111] Une troisième étude envisagée concerne un réseau immunitaire couplé au "réseau de production", toujours dans un contexte d'apprentissage : quel est la forme des dynamiques qui "émergent" ? Est-ce que l'organisme "apprend" quelque

[111] C'est un mystère que les cellules nerveuses possèdent des connexions, tandis que les autres cellules n'en possèdent pas (ou que ces connexions sont implicites et dynamiques).

chose ? Cette extension doit reposer, comme nous l'avons déjà dit dans le chapitre 7, sur la complémentarité des messages traités par les deux systèmes et sur le "jeu" entre les deux.

Autonomie sociale. Nous avons vu dans le chapitre 5, qu'un problème "social" n'est intéressant que quand il comporte deux forces sociales couplées entre elles : un tel système pourra être considéré comme autonome si la relation entre les deux forces est dynamique, donc s'il est auto-organisant. Un exemple de système social autonome est une micro-économie, par exemple une bourse : les deux forces sociales sont l'achat et la vente et leur couplage est dynamique parce qu'un agent change continûment de rôles d'achat et de vente. Ce système devient intéressant dès qu'on introduit de la diversité. Les "conflits" entre agents, autrement dit les perturbations locales, stimulent l'auto-organisation du système. Un générateur de diversité interne au système peut donc perpétuer cette activité. Dans cette perspective, nous avons commencé à étudier certains modèles micro-économiques (Benos & Tzafestas 1995). L'hypothèse et la démarche de la systémique est précisément que les enseignements de l'étude de ce système social sous l'angle de l'autonomie seront indépendants du contexte économique de départ.

Systémique. Les études présentées concernent deux niveaux de la hiérarchie du vivant artificiel ou alternatif, le niveau de l'animat cellulaire et le niveau de la cellule. L'extension des modèles cellulaires et l'étude de l'autonomie sociale se situent dans une optique d'élargissement des niveaux d'application des considérations systémiques. D'une part, j'espère que l'étude d'autres modèles cellulaires conduira à un approfondissement du problème de l'espace de représentation : comme je l'ai déjà dit, je crois que, finalement, les messages doivent être des systèmes eux aussi, ce qui signifie que les mêmes principes ou des principes similaires pourront s'appliquer à un niveau plus bas de la hiérarchie, le niveau "moléculaire". D'autre part, l'étude d'une société autonome se situe à un niveau encore plus élevé dans la hiérarchie. Dans tous les cas, il s'agit de montrer que les mêmes principes s'appliquent à tous les niveaux.

Méthode. Si les résultats obtenus jusqu'à maintenant ont été quantitativement intéressants, dans le sens d'avoir répondu aux objectifs de départ, ce n'est que le futur qui peut juger de leur intérêt qualitatif. Dans tous les cas, un travail par principe ascendant et synthétique ne peut qu'être conforme aux "9 lois de Dieu" (Kelly 1994) :

"Distribute being". La généralisation et l'abstraction ont été intéressantes parce que les problèmes étudiés ont été de nature différente. Pour élargir le domaine d'application des principes présentés, je dois continuer à aborder des problèmes à première vue disparates.

"Control from the bottom up". Les problèmes étudiés suivent un ordre de *complexité ascendante qui ne concerne pas les solutions, mais les principes* : par exemple, j'ai étudié le problème de description et de spécification du système de contrôle d'un agent-animat *avant* de m'intéresser à la relation entre son système motivationnel et sa mission, et si j'ai ensuite retourné au niveau de la cellule, ce n'était que pour appliquer les mêmes principes et identifier ce qui manquait. Je peux alors prévoir un va-et-vient continu entre niveaux d'organisation différents qui servira à raffiner et étendre les principes de base.

"Cultivate increasing returns". Lors de l'élaboration des divers modèles, j'ai essayé de les définir et les décrire indépendamment des détails d'implémentation (ici, indépendamment des détails matériels de niveau d'organisation), précisément pour

pouvoir ensuite les instancier sélectivement dans des cas particuliers. Un exemple en est le modèle tit-for-tat quantitatif (annexe A) : même si toute sa puissance est loin d'être explorée et utilisée, il a été initialement défini hors contexte d'autonomie, mais ensuite instancié dans le cadre des deux niveaux d'agents autonomes étudiés et dans le cas des cellules autonomes il a fait apparaître le besoin d'un système immunitaire. Je peux prévoir que d'autres modèles particuliers pourront jouer un tel rôle de "moteur d'évolution" des principes (par exemple, j'espère beaucoup au modèle de sénescence).

"Grow by chunking". Problèmes disparates, principes ascendants, modèles spécifiques abstraits : grâce à ces propriétés de la démarche, l'évolution des principes lors de ces années de thèse a été une évolution à sauts brusques. Une fois un problème spécifique résolu, les principes abstraits correspondants et le différentiel des principes de base étaient immédiatement disponibles pour intégration dans le corps des principes d'organisation et alors pour instanciation dans d'autres problèmes. Je ne sais pas si l'ensemble des briques de base que j'ai défini est complet : je crois que non.

"Maximize the fringes". Pour que cette démarche évolutive et comparative apporte des fruits, il faut assurer que les problèmes sont disparates mais avec des propriétés communes, c'est-à-dire il faut *choisir* des problèmes voisins. Une généralisation et une synthèse des résultats ne sera alors possible que si l'on trace les bonnes analogies entre problèmes à première vue non corrélés. Je ne sais pas s'il existe une véritable méthode d'analogie, et j'avoue que jusqu'à maintenant j'ai été purement intuitive en ce qui concerne le choix des problèmes abordés. Tant que l'analogie reste une démarche mal définie, on continuera à avoir à prendre le risque de l'intuition.

"Honor your errors". Comme toute activité humaine, la recherche scientifique est susceptible aux erreurs. L'important, c'est de reconnaître que *les erreurs aussi font avancer les idées* et que, si on ne fait pas d'erreurs, c'est peut-être parce qu'on n'avance pas. Je répète deux des erreurs que j'ai faites et que j'ai trouvées importantes pour l'évolution des principes et de ma compréhension de leur signification. La première est celle du paragraphe 4.5, où j'avais prévu comme direction de méta-adaptation la direction fausse : ce que j'en ai appris est que la direction d'adaptation doit être celle qui correspond au feedback négatif stabilisant, plus le monde est hostile plus lentement l'agent réagit. Une deuxième erreur que j'ai trouvée tout aussi importante est ma fausse prédiction de cancer dans les grandes populations sans considérer les autres facteurs intervenants (paragraphe 8.6.3) : ce que j'en ai appris est qu'il ne faut jamais sous-estimer le pouvoir de l'interaction et que c'est là où se trouve toute notre liberté. Dans les deux cas, les erreurs et leur correction m'ont conduite à mieux comprendre les phénomènes étudiés. Je dois souhaiter continuer à faire des erreurs.

"Pursue no optima ; have multiple goals". Aborder simultanément ou progressivement plusieurs problèmes différents a l'avantage de minimiser les conséquences des désastres partiels. Mais, qui plus important est, la démarche ascendante et comparative a l'avantage d'être suffisamment ouverte pour ne pas avoir un but précis qui, une fois atteint, marquerait la fin de la recherche. Résoudre certains problèmes particuliers d'agents autonomes peut être ou ne pas être intéressant. Mais essayer de comprendre à partir de ces solutions partielles n'a pas de fin. La volonté de comprendre : c'est ce qui génère de nouveaux problèmes et de nouveaux buts partiels.

"Seek persistent disequilibrium". Une façon de maintenir notre intérêt scientifique est "d'injecter du bruit" là où on ne l'attend pas : par exemple, douter et réviser les hypothèses de départ ou de travail, une fois le problème résolu. Ainsi, chaque réponse que nous donnons va fermer définitivement une question, mais seulement pour en ouvrir beaucoup d'autres. Je trouve ces principes abstraits intéressants, pas tellement pour avoir généralisé les solutions particulières, mais surtout pour avoir donné naissance à des nouvelles questions (qu'est-ce qui émerge en présence d'un moteur de sénescence ? quel est le rôle de la diversité pour l'autonomie sociale ? combien de systèmes d'autorégulation faut-il au cas de plusieurs paramètres libres et quels sont les relations entre eux ? etc.).

"Change changes itself". J'ai commencé ce travail dans une problématique de conception très utilitariste qui a ensuite évolué vers l'étude de l'autonomie (des agents) et qui s'éloigne progressivement de mes considérations initiales d'ingénieur. Ce changement d'optique et de perspective, je ne l'avais pas prévu et il est, me semble-t-il, dans le bon sens. J'ai alors appris à apprécier l'imprévisibilité relative de l'évolution de la pensée et à croire en sa puissance. On dit que la gourmandise intellectuelle de l'homme est sans fond. Je n'ai certainement pas fini.

9.3 Interlude : Petite pause devant le futur

Je viens de dire que les principes qui ont été révélés utiles dans une optique de conception ressemblent à ceux que la nature a inventés. La question qui naît naturellement à partir de cette observation est si ces principes sont alors *nécessaires* pour les systèmes autonomes et, par extension, pour le vivant. C'est-à-dire, l'organisation du vivant comme nous le connaissons, est-elle la seule alternative possible et pourquoi ?

La discipline de la vie artificielle part de l'hypothèse des multiples alternatives et se donne comme sujet "l'étude du vivant tel qu'il pourrait être au lieu du vivant tel que nous le connaissons" (Langton 1988*b*).[112] Pour certains (par exemple, Heppenheimer 1986), le chemin, de l'automate de Vausancon à Frankenstein de Mary Shelley et des logiciels vivants (Ferber 1989*a*, p. 423-424) à Robotsville (Speckley 1986), ne paraît plus si long. Hans Moravec (1988) pousse cette idée aux extrêmes : il se fait à haute voix le champion d'une vision futuriste, en parlant d'un monde futur post-biologique (*"The post-biological world"*, p. 5) et d'une entreprise génétique en cours qui entraîne le monde vers un univers où les machines évoluent en une espèce supérieure qui remplacera petit à petit l'homme, son créateur (*"a genetic takeover"*, ibid., p. 3). Farmer et d'A. Belin (1991) vont plus loin encore en affirmant que c'est l'ordre naturel qui dirige cette entreprise génétique : *"The natural order of evolution is change. No species has persisted for ever. [...] There is no reason to believe that we are immune to this."* (p. 836).

Pour Langton, l'essentiel est de comprendre que la nature est dynamique, que notre technologie fait partie de la nature au même droit que les êtres vivants[113]. Dans cette

[112] Cette hypothèse me paraît comme le sous-produit de l'avancement de la technologie d'un homme qui aspire éternellement vers la création d'êtres vivants et qui recherche désespérément son frère dans l'univers.

[113] Il a exposé cette idée lors de son discours inaugural du quatrième workshop en vie artificielle (juillet 1994 à Cambridge, Massachusetts).

optique, notre avancement scientifique et technologique se situe dans le cadre plus large de l'évolution naturelle. Peut-être que nous sommes le moyen que la vie a inventé pour se reproduire. Ce que j'ai appris, de mes succès et de mes échecs de modélisation, ce en quoi j'ai changé lors de ces années de thèse, c'est que, si en concevant nous retombons sur les mêmes principes que la nature, c'est peut-être parce que le vivant tel que nous le connaissons est le seul vivant qu'il pourrait y avoir. Avec la physique de notre univers, la chimie telle que nous la connaissons est peut-être la seule chimie possible, et avec cette chimie, la biologie telle que nous la connaissons est peut-être la seule biologie possible (et ainsi de suite)[114]. Je suis alors amenée à croire qu'une vie artificielle n'est pas possible, s'il existe un autre vivant quelque part ailleurs dans l'univers et si jamais on parvient à créer un autre vivant, cet autre vivant doit être le même vivant que nous connaissons. Quoi qu'il en soit, ce que nous ferons et ce que nous deviendrons, c'est ce que "l'ordre naturel" nous a chargés de faire et de devenir : nous ne pouvons pas reculer.

Ordre des temps si les machines
Se prenaient enfin à penser
Sur les plages de pierreries
Des vagues d'or se briseraient
L'écume serait mère encore

(Guillaume Apollinaire,
"Calligrammes", 1925)

[114] N'oublions pas que pour Varela (1979), l'étude du vivant et de son organisation doit se faire au niveau abstrait et immatériel de l'information et que, *"de ce point de vue, il n'y a aucune différence entre la physique et la cybernétique"* (ibid., p. 43), et *"ces concepts peuvent s'appliquer sans solution de continuité aux capacités cognitives de l'homme et même à la dynamique sociale"* (ibid., p. 13).

Bibliographie

Ackley, D., M. Littman (1991). Interactions between learning and evolution, pp. 487-509, in *(Langton et al. 1991)*.

Ackley, D., M. Littman (1994). A case for lamarckian evolution, pp. 3-10, in *(Langton 1994)*.

Agre, P.E. (1985). Routines, *M.I.T. Artificial Intelligence Memo 828*.

Agre, P.E. (1988). The dynamic structure of everyday life, *M.I.T. Artificial Intelligence Laboratory Technical Report TR-1085*, October.

Agre, P., D. Chapman (1987). Pengi : An Implementation of a Theory of Activity, *Proceedings AAAI-87*, Morgan Kaufmann, San Mateo, California, pp. 268-272.

Agre, P., D. Chapman (1991). What are plans for?, in *(Maes 1991a)*.

Agre, P. (1993). The symbolic worldview: Reply to Vera and Simon, *Cognitive Science, Special Issue on Situated Action,* 17(1993):61-69.

Agre, P.E. (1995). Computational research on interaction and agency, *Artificial Intelligence, Special Volume on Computational Research on Interaction and Agency,* 72(1995):1-52.

Ali Cherif, A. (1993). A constitution and an economic model for the organisation and emergence of collective behaviour in a colony of robots, *Rapport de recherche, Université Paris-Nord*.

Almássy, N., P. Verschure (1992). Optimizing self-organizing control architectures with genetic algorithms: The interaction between natural selection and ontogenesis, *Technical Report No. 92.10, Department of Computer Science, University Zurich-Irchel*, 10p, also *Proceedings of the Second Conference on Parallel Problem Solving from Nature (PPSN '92)*, R. Männer & B. Manderick (Eds.), North-Holland, September 1992.

Almgren, R. (1989). *On knowledge-based planning and programming systems for Flexible Automatic Assembly*, University of Linköping, Studies in Science and Technology, Thesis no. 176, LiU-Tek-Lic-1989:16.

Anderson, J.A., E. Rosenfeld (1988). *Neurocomputing, Foundations of research*, MIT Press.

Anderson, J.A., A. Pellionisz, E. Rosenfeld, Eds. (1991). *Neurocomputing 2, Directions for research*, MIT Press.

Anderson, T.L., M. Donath (1991). Animal behavior as a paradigm for developing robot autonomy, in *(Maes 1991a)*.

Angle, C.M., R.A. Brooks (1990). Small Planetary Rovers, *Proceedings IEEE International Workshop on Intelligent Robots and Systems (IROS) '90*, Japan.

Arkin, R. (1987). Motor schema based navigation for a mobile robot : An approach to programming by behaviour, *Proceedings 1987 International Conference on Robotics and Automation*, Raleigh, NC, March.

Arkin, R.C. (1989). Motor-schema based mobile robot navigation, *International Journal of Robotics Research*, 8(4):92-112.

Arkin, R.C., R.R. Murphy (1990). Autonomous navigation in a manufacturing environment, *IEEE Transactions on Robotics and Automation*, Vol. RA-6(4):445-454, August.

Arkin, R.C. (1991). Integrating behavioral, perceptual and world knowledge in reactive navigation, in *(Maes 1991a)*.

Arkin, R.C., T. Balch, E. Nitz (1993). Communication of behavioral state in multi-agent retrieval tasks, *IEEE International Conference on Robotics and Automation*, Atlanta, GA, May.

Asama, H., M.K. Habib, I. Endo, K. Ozaki, A. Matsumoto, Y. Ishida (1991). Functional distribution among multiple mobile robots in an autonomous and decentralized robot system, *Proceedings 1991 IEEE International Conf. on Robotics and Automation*, Sacramento, CA, April.

Ashby, W.R. (1960). *Design for a brain - The origin of adaptive behaviour*, Chapman & Hall, London, 1952, 2nd edition revised, 1960.

Atlan, H. (1972). *L'organisation biologique et la théorie de l'information*, Hermann - Éditeurs des Sciences et des Arts, Paris.

Atlan, H. (1979). *Entre le cristal et la fumée - Essai sur l'organisation du vivant*, Éditions du Seuil, Paris.

Axelrod, R., W.D. Hamilton (1981). The evolution of cooperation, *Science*, 211:1390-1396.

Axelrod, R. (1984). *The evolution of co-operation*, Basic Books, 1984, also Penguin Books, 1990.

Axelrod, R., D. Dion (1988). The further evolution of cooperation, *Science*, 242:1385-1390.

Bak, P., H. Flyvbjerg, B. Lathrup (1994). Evolution and coevolution in a rugged fitness landscape, pp. 11-42, in *(Langton 1994)*.

Baldwin, K.E. (1989). Autonomous manufacturing systems, *Proceedings of the 1989 IEEE International Symposium on Intelligent Control*, Albany, NY, September, pp. 214-220.

Balkenius, C. (1995). *Natural intelligence in artificial creatures*, Ph.D. Thesis, Lund University Cognitive Studies 37, May.

Bates, J. (1994). The role of emotion in believable agents, *Communications of the ACM*, 37(7):122-125, July.

Becker, G.S. (1976). Altruism, egoism and genetic fitness: Economics and sociobiology, *Journal of Economic Literature*, 14:817-826.

Beckers, R., O.E. Holland, J.-L. Deneubourg (1994). From local actions to global tasks: Stigmergy and collective robotics, pp. 181-189, in *(Brooks & Maes 1994)*.

Beer, R.D., H.J. Chiel, L.S. Sterling (1989). Heterogeneous neural networks for adaptive behaviour in dynamic environments, in D.S. Touretzky (Ed.) : *Advances in Neural Information Processing Systems 1*, Morgan Kaufmann, pp. 577-585.

Beer, R. (1990). *Intelligence as adaptive behavior - An experiment in computational neuroethology*, Academic Press, San Diego, CA.

Beer, R.D., H.J. Chiel (1990). Neural implementation of motivated behaviour : feeding in an artificial insect, in D. Touretzky (Ed.), *Advances in Neural Information Processing Systems 2*, Morgan Kaufmann, pp. 44-51.

Beer, R.D., H.J. Chiel (1991). The Neural Basis of Behavioral Choice in an Artificial Insect, in *(Meyer & Wilson 1991)*.

Beer, R.D. (1995). A dynamical systems perspective on agent-environment interaction, *Artificial Intelligence, Special Volume on Computational Research on Interaction and Agency,* 72(1995):173-215.

Belew, R.K., J. McInerney, N.N.Schraudolph (1991). Evolving networks :Using the genetic algorithm with connectionist learning, pp. 511-547, in *(Langton et al. 1991)*.

Benos, A., E. Tzafestas (1995). Alternative distributed models for the comparative study of stock market phenomena, *Proceedings 1995 Joint Conference on Information Systems (JCIS'95)*, Durham, NC, October, also *Cahier de Recherche du Groupe HEC 551/1995*, 20p.

Bersini, H., F.J. Varela (1991). The immune recruitment mechanism: A selective evolutionary strategy, *Proceedings ICGA 1991*.

Bersini, H. (1992). Immune network and adaptive control, in *(Varela & Bourgine 1992)*.

Bersini, H., V. Detours (1994). Asynchrony induces stability in cellular automata based models, pp. 382-387, in *(Brooks & Maes 1994)*.

Bjorke, O. (1979). Computer-aided part manufacturing, *Computers in Industry*, 1:3-9.

Blumberg, B. (1994). Action selection in hamsterdam : Lessons from ethology, pp. 108-117, in *(Cliff et al. 1994)*.

Bond, A.H., L. Gasser, Eds. (1988). *Readings in Distributed Artificial Intelligence*, Morgan Kaufmann, San Mateo, CA.

Booker, L.B. (1991). Instinct as an inductive bias for learning behavioural sequences, in *(Meyer & Wilson 1991)*.

Braitenberg, V. (1984). *Vehicles - Experiments in synthetic psychology*, Bradford Books/MIT Press.

Brooks, D.R., E.O. Wiley (1988). *Evolution as entropy - Toward a unified theory of biology*, University of Chicago Press, 2nd edition.

Brooks, R.A. (1986*a*). A robust layered control system for a mobile robot, *IEEE Journal of Robotics and Automation*, 2(1):14-23.

Brooks, R.A. (1986*b*) Achieving Artificial Intelligence through building robots, M.I.T. AI Memo No. 899, May.

Brooks, R.A. (1989). A robot that walks : Emergent behaviors from a carefully evolved network, M.*I.T. AI Memo No. 1091*, also *Neural Computation*, 1, pp. 253-262.

Brooks, R.A. (1990). The Behavior Language - User's Guide, *M.I.T. AI Memo No. 1227*, 35 p.

Brooks, R.A. (1991*a*). Intelligence without Representation, *Artificial Intelligence, Special Issue on the Foundations of Artificial Intelligence*, 47(1-3):139-159.

Brooks, R.A. (1991*b*). Intelligence without Reason, *M.I.T. AI Memo No. 1293*, April, 27 p., also *Computers and Thought Workshop, IJCAI-91*.

Brooks, R.A. (1991*c*). Elephants don't play chess, in *(Maes 1991a)*.

Brooks, R.A. (1994). Coherent behavior from many adaptive processes, in *(Cliff et al. 1994)*, pp. 22-29.

Brooks, R.A., A.M. Flynn (1989). A robot being, in *(Dario et al. 1989)*, pp. 679-701.

Brooks, R.A., P. Maes, M. Mataric, G. More (1990). Lunar base construction robots, *Proceedings IEEE International Workshop on Intelligent Robots and Systems (IROS) '90*, Japan.

Brooks, R., P. Maes, Eds. (1994). *Artificial Life IV, Proceedings of the Fourth Interdisciplinary Workshop on the Synthesis and Simulation of Living Systems*, MIT Press, Cambridge, MA.

Castelfranchi, C. (1990). Social power: A point missed in multi-agent, DAI and HCI, pp. 49-62, *in (Demazeau & Müller 1990)*.

Castelfranchi, C. (1993). Reactivity at the social level : Some basic issues, *Proceedings of 1993 Italian Workshop on DAI*, D. D'Aloisi & M. Miceli (Eds.).

Castelfranchi, C. (1995). Commitments : From individual intentions to groups and organizations, pp. 41-48, *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS95)*, San Francisco, CA, June.

Cecconi, F., D. Parisi (1992). Neural networks with motivational units, in *(Meyer et al. 1992)*, pp. 346-355.

Chandrasekaran, B., A. Goel, D. Allemang (1988). Connectionism and information-processing abstractions (The message still counts more than the medium), *AI Magazine*, 9(4):24-34, Winter.

Chapman, D., P. Agre (1986). Abstract reasoning as emergent from concrete activity, in M.P. Georgeff & A.L. Lansky (Eds.), *Reasoning About Actions and Plans, Proceedings of the 1986 Workshop*, Timberline, Oregon, Morgan Kaufmann, Los Altos, CA, pp. 411-424.

Chapman, D. (1990). Vision, Instruction and Action, *M.I.T. Artificial Intelligence Laboratory Technical Report TR-1204*, April.

Clancey, W.J. (1989). The frame-of-reference problem in cognitive modeling, *Proc. Annual Conf. Cognitive Science Society*, Lawrence Erlbaum, Hillsdale, NJ, pp. 107-114.s

Clark, A. (1993) Minimal rationalism, *Cognitive Science Research Paper 270*, Univ. of Sussex, 19p.

Cliff, D., I. Harvey, P. Husbands (1993). Explorations in evolutionary robotics, *Adaptive Behavior*, 2(1):73-110.

Cliff, D., P. Husbands, J.-A. Meyer, S.W. Wilson, Eds. (1994). *From animals to animats 3, Proceedings of the 3rd International Conference on Simulation of Adaptive Behavior*, Bradford/MIT Press, Cambridge, MA.

Coiffet, P. (1986). *La Robotique - Principes et Applications*, Éditions Hermès, Paris.

Coiffet, P. (1993). *Robot habilis, robot sapiens - histoire, développements et futur de la robotique*, Éditions Hermès, Paris.

Colombetti, M., M. Dorigo (1992). Learning to control an autonomous robot by distributed genetic algorithms, in *(Meyer et al. 1992)*.

Colombetti, M., M. Dorigo (1994). Training agents to perform sequential behavior, *Adaptive Behavior*, 2(3):247-275, Winter.

Connell, J. (1990). *Minimalist mobile robotics : a colony-style architecture for an artificial creature*, MIT Press, Cambridge, MA.

Conte, R., M. Miceli, C. Castelfranchi (1991). Limits and levels of cooperation: Disentangling various types of prosocial interaction, pp. 153-165, in *(Demazeau & Müller 1991).*

Covrigaru, A.A., R.K. Lindsay (1991). Deterministic autonomous systems, *AI Magazine*, 12(3):110-117, Fall.

Darche, P. (1994). Le paradigme acteur appliqué aux systèmes embarqués communicants : ActNet, un réseau d'acteurs robotiques, *Thèse de Doctorat de l'Université Paris VI*, Mars.

Dario, P., G. Sandini, P. Aebischer (1989). *Robots and biological systems: Towards a New Bionics?*, NATO ASI Series, Vol. F/102.

Dawkins, R. (1976). *The selfish gene*, Oxford Univ. Press.

Dawkins, R. (1988). The evolution of evolvability, in *(Langton 1988a)*.

de Boer, M.J.M., F.D. Fracchia, P. Prusinkiewicz (1991). Analysis and simulation of the development of cellular layers, pp. 465-483, in *(Langton et al. 1991)*.

Delaye, C., J. Ferber (1992). Morphological and behavioural adaptation of robots using genetic algorithms, *Presentation Artificial Life III*, Santa Fe.

Delaye, C. (1993). *Structures et organisations des systèmes multi-agents autonomes et adaptatifs*, Thèse de Doctorat de l'Université Pierre et Marie Curie, Décembre.

Demazeau, Y., J.-P. Müller, Eds. (1990). *Decentralized A.I., Proceedings of the First European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW '89)*, Elsevier/North-Holland.

Demazeau, Y., J.-P. Müller, Eds. (1991). *Decentralized A.I. 2*, *Proceedings of the Second European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW '90)*, North-Holland.

Deneubourg, J.-L., S. Goss, J.M. Pasteels, D. Fresneau, J.-P. Lachaud (1987). Self-organization mechanisms in ant societies (II): Learning in foraging and division of labor, *"From individual to collective behavior in social insects"*, J.-M. Pasteels & J.-L. Deneubourg (Eds.), Birkhäuser, Basel, pp. 177-196.

Deneubourg, J.-L., S. Goss, N. Franks, A. Sendova-Franks, C. Detrain, L. Chrétien (1991). The Dynamics of Collective Sorting : Robot-Like Ants and Ant-Like Robots, in *(Meyer & Wilson 1991)*.

Dennett, D.C. (1987). *The intentional stance*, MIT Press.

De Sousa, R. (1990). The sociology of sociobiology, *International Studies in the Philosophy of Science*, 4(3):271-283.

Desrochers, A.A., M. Silva, Eds. (1994). *IEEE Transactions on Robotics and Automation, Special Issue on Computer Integrated Manufacturing*, 10(2), April.

Donnart, J.-Y., Meyer, J.-A. (1994). A hierarchical classifier system implementing a motivationally autonomous animat, pp. 144-154, in *(Cliff et al. 1994)*.

Doty, K.L., R.E. Van Aken (1993). Swarm robot materials handling paradigm for a manufacturing workcell, *Proceedings of the 1993 IEEE International Conference on Robotics and Automation*, Atlanta, GA, May, pp. 778-782.

Drogoul, A., J. Ferber, B. Corbara, D. Fresneau (1992). A behavioral simulation model for the study of emergent social structures, in *(Varela & Bourgine 1992)*.

Drogoul, A., J. Ferber (1992). From Tom Thumb to the Dockers : Some experiments with foraging robots, in *(Meyer et al. 1992)*.

Drogoul, A. (1993). De la simulation multi-agents à la résolution collective de problèmes - Une étude de l'émergence de structures d'organisation dans les systèmes multi-agents, *Thèse de Doctorat d'Université Paris VI*, November.

Durfee, E.H., V.R. Lesser, D.C. Corkill (1989). Trends in cooperative distributed problem solving, *IEEE Transactions on Knowledge and Data Engineering*, 1(1):63-83, March.

Dyson, F. (1985). *Origins of Life*, Cambridge University Press.

Edelman, G.M. (1992). *Bright air, brilliant fire: On the matter of mind*, Basic Books, 1992, traduit en français sous le titre *"Biologie de la conscience"*, Éditions Odile Jacob.

Engelmore, R., T. Morgan, Eds. (1988). *Blackboard Systems*, Addison-Wesley, Reading, MA.

Farmer, J.D., A.d'A. Belin (1991). Artificial Life : The Coming Evolution, in *(Langton et al. 1991)*.

Ferber, J. (1989*a*). *Des objets aux agents : Une étude des structures de représentation et des communications en Intelligence Artificielle*, Thèse d'État, Université Pierre et Marie Curie, Paris, June.

Ferber, J. (1989*b*). Eco-Problem-Solving : How to solve problems by interactions, *Proceedings of the 9th Workshop on Distributed Artificial Intelligence*, pp. 113-128.

Ferber, J., E. Jacopin (1990). A multi-agent satisfaction planner for building plans as side effects, *Rapport LAFORIA 07/90*, January, 11p.

Ferber, J., E. Jacopin (1991). The Framework of Eco-Problem-Solving, pp. 103-114, in *(Demazeau & Müller 1991).*

Ferber, J., A. Drogoul (1992). Using reactive multi-agent systems in simulation and problem-solving, pp. 53-80, *Distributed Artificial Intelligence: Theory and Praxis*, by N.M. Avouris & L. Gasser (Eds.).

Ferber, J. (1994*a*). Reactive Distributed Artificial Intelligence: principles and applications, *Foundations of Distributed Artificial Intelligence*, G.M.P. O'Hara & N.R. Jennings (Eds.).

Ferber, J. (1994*b*). La kénétique : Des systèmes multi-agents à une science de l'interaction, *Revue Internationale de Systémique*, 8(1):13-27.

Ferber, J. (1994*c*). Coopération réactive et émergence, *Intellectica*, 19(2):19-52.

Ferber, J. (1995). *Les systèmes multi-agents : Vers une intelligence collective*, InterEditions, Paris.

Ferrell, C. (1994). Failure recognition and fault tolerance of an autonomous robot, *Adaptive Behavior*, 2(4):375-398.

Findler, N.V., R. Malyankar (1993). Alliances and social norms in societies of non-homogeneous, interacting agents, *Proceedings 2nd International Symposium on Simulation Societies '93 (Approaches to simulating social phenomena and social processes)*, July.

Firby, R.J. (1987). An investigation into reactive planning in complex domains, *Proceedings AAAI-87*.

Fischer, K. (1993). Modelling autonomous systems in a flexible manufacturing system, *Proceedings of the IJCAI '93 Workshop on Dynamically Interacting Robots*, Chambéry, France, August, pp. 109-116.

Flynn, A.M. (1987). Gnat Robots (And How They Will Change Robotics), *M.I.T. AI Working Paper 295*, June, also *Proceedings IEEE Micro Robots and Teleoperators Workshop*, November 1989.

Flynn, A.M., R.A. Brooks (1989*a*). Building Robots : Expectations and Experiences, *IEEE/RSJ International Workshop on Intelligent Robots and Systems (IROS '89)*, September, Tsukuba, Japan, pp. 236-243.

Flynn, A.M., R.A. Brooks (1989*b*). Battling reality, *MIT AIM-1148*, October.

Fontana, W. (1991). Algorithmic chemistry, pp. 159-209, in *(Langton et al. 1991)*.

Fukuda, T., Y. Kawauchi (1990). Cellular robotic system (CEBOT) as one of the realization of self-organizing intelligent universal manipulator, *Proceedings 1990 IEEE International Conference on Robotics and Automation*, Cincinatti, Ohio, May.

Fukuda, T., H. Hosokai, Y. Kawauchi, M. Buss (1990*a*). Dynamically reconfigurable robotic system (DRRS) - System configuration and implementation as CEBOT, *Proceedings of the 5th International Symposium on Robotics Research*.

Fukuda, T., Y. Kawauchi, H. Asama (1990*b*). Analysis and evaluation of Cellular Robotics (CEBOT) as a Distributed Intelligent System by Communication Information Amount, *Proceedings IEEE International Workshop on Intelligent Robots and Systems (IROS) '90*, Japan.

Fukuda, T., T. Ueyama, F. Arai (1991). Control strategy for a network of cellular robots, *Proceedings 1991 IEEE International Conference on Robotics and Automation*, Sacramento, CA, April.

Galliers, J.R. (1990). The positive role of conflict in cooperative multi-agent systems, pp. 33-46, in *(Demazeau & Müller 1990)*.

Gasser, L., M.N. Huhns, Eds. (1989). *Distributed Artificial Intelligence, Vol. 2*, Pitman Publishing/Morgan Kaufmann, London/San Mateo, CA.

Gat, E. (1992). Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots, *Proceedings AAAI-92*.

Gat, E. (1993). On the role of stored internal state in the control of autonomous mobile robots, *AI Magazine*, Spring.

Georgeff, M.P., A.L. Lansky (1987). Reactive reasoning and planning, *Proceedings AAAI-87*.

Goel, N.S., R.L. Thompson (1988). Movable Finite Automata (MFA) : A new tool for computer modeling of living systems, in *(Langton 1988a)*.

Goldberg, D.E. (1989). *Genetic algorithms in search, optimization and machine learning*, Addison-Wesley, Reading, MA.

Golub, E.S., D.R. Green (1991). *Immunology - A synthesis*, 2nd Edition, Sinauer Associates, Sunderland, MA.

Goss, S., J.-L. Deneubourg (1992). Harvesting by a group of robots, in *(Varela & Bourgine 1992)*.

Gould, S.J. (1977). *Ever since Darwin - Reflections in natural history*, Penguin Books, Harmondsworth, Middlesex.

Guillot, A., J.-A. Meyer (1994). Synthetic animals in synthetic worlds, *Groupe de BioInformatique, École Normale Supérieure*, Paris.

Gullahorn, J.T., J.E. Gullahorn (1963). A computer model of elementary social behavior, pp. 375-386, in *Computers and Thought*, by E.A. Feigenbaum & J.E. Feldman (Eds.), McGraw-Hill, NY.

Gutowitz, H. (1993). Complexity-seeking ants, *Proceedings of the 2nd European Conference on Artificial Life (ECAL'93)*, Brussels, May.

Hackwood, S., G. Beni (1992). Self-organization of sensors for swarm intelligence, *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, Nice, France, May.

Halperin, J.R.P. (1991). Machine Motivation, in *(Meyer & Wilson 1991)*.

Halperin, J.R.P., D.W. Dunham (1992). Postponed conditioning : Testing a hypothesis about synaptic strengthening, *Adaptive Behavior*, 1(1):39-63.

Hamilton, W.D. (1964). The genetical evolution of social behaviour - I & II*, Journal of Theoretical Biology*, 7(1964):1-16 and 17-52.

Hartley, R., F. Pipitone (1991). Experiments with the subsumption architecture, *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, Sacramento, CA, April.

Harvey, I. (1994). Evolutionary robotics and SAGA : The case for hill-crawling and tournament selection, pp. 299-326, in *(Langton 1994)*.

Havel, I.M. (1993). Artificial thought and emergent mind, *Proceedings IJCAI-93*, pp. 758-766.

Hayes, P. (1990). The frame problem and related problems in artificial intelligence, in J.F. Allen, J. Hendler, A. Tate (Eds.), *Readings in Planning*, Morgan Kaufmann, San Mateo, California.

Heppenheimer, T.A. (1986). Man makes man, in *(Minsky 1986b)*.

Hogg, T., B.A. Huberman (1993). Better than the best: The power of cooperation, in L. Nadel and D. Stein (Eds.), *SFI 1992 Lecture Notes in Complex Systems*, pp. 163-184, Addison-Wesley.

Huberman, B.A. (1988). The Ecology of Computation, in B.A. Huberman (Ed.), *The Ecology of Computation*, North-Holland , Amsterdam.

Huberman, B.A., N.S. Glance (1993). Evolutionary games and computer simulations, *Proceedings National Academy of Sciences (USA)*, Vol. 90, pp. 7716-7718.

Huhns, M.N., Ed. (1987). *Distributed Artificial Intelligence*, Pitman Publishing, Morgan Kaufmann, London/San Mateo, CA.

Husbands, P., I. Harvey, D. Cliff (1993). Analysing recurrent dynamical networks evolved for robot control, *Cognitive Science Research Paper 265*, Univ. of Sussex, January.

Iba, H., H. de Garis, T. Higuchi (1992). Evolutionary learning of predatory behaviors based on structured classifiers, in *(Meyer et al. 1992)*.

Ishiguro, A., Y. Watanabe, Y. Uchikawa (1995). An immunological approach to dynamic behavior control for autonomous mobile robots, *Proceedings IROS'95*, Pittsburgh, PA.

Jacob, F. (1970). *La logique du vivant - Une histoire de l'hérédité*, Éditions Gallimard, Paris.

Jacob, F. (1981). *Le jeu des possibles - Essai sur la diversité du vivant*, Éditions Fayard, Paris.

Jacopin, E. (1993). *Algorithmique de l'interaction: Le cas de la planification*, Thèse de Doctorat de l'Université Paris, LAFORIA Thèse 93/11, Septembre.

Jones, P.F. (1992). *CAD/CAM: Features, applications and management*, Macmillan Press Ltd., London.

Kaelbling, L.P. (1986). An Architecture for Intelligent Reactive Systems, in M.P. Georgeff, A.L. Lansky (Eds.), *Reasoning About Actions and Plans, Proceedings*

of the 1986 Workshop, Timberline, Oregon, Morgan Kaufmann, Los Altos, California.

Kaelbling, L.P. (1991). Foundations of learning in autonomous agents*, Robotics and Autonomous Systems*, 8(1991):131-144.

Kaelbling, L.P., S.J. Rosenschein (1991). Action and planning in embedded agents, in *(Maes 1991a)*.

Kagawa, Y. (1992). A Behavior-Based Approach to Coordinate Multiple Mobile Robots, *submitted to the 1992 IEEE Conference on Robotics and Automation*.

Kanungo, M.S. (1994). *Genes and aging*, Cambridge University Press.

Kauffman, S.A. (1993). *The origins of order - Self-organization and selection in evolution*, Oxford University Press.

Kawauchi, Y., M. Inaba, T. Fukuda (1992). Self-organizing intelligence for cellular robotic system "CEBOT" with genetic knowledge production algorithm, *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, Nice, France, May.

Kelly, K. (1994). *Out of control*, Addison-Wesley.

Kephart, J.O. (1994). How topology affects population dynamics, pp. 447-463, in *(Langton 1994)*.

Kirkwood, T.B.L., R. Holliday (1979). The evolution of ageing and longevity, *Proceedings of the Royal Society of London, Series B: Biological Sciences*, 205(1979):531-546.

Klopf, A.H., J.S. Morgan, S.E. Weaver (1993). A hierarchical network of control systems that learn: Modeling nervous system function during classical and instrumental conditioning, *Adaptive Behavior*, 1(3):263-319.

Koza, J.R. (1991*a*). Genetic Evolution and the Co-Evolution of Computer Programs, in *(Langton et al. 1991)*.

Koza, J.R. (1991*b*). Evolution and Co-Evolution of Computer Programs to Control Independently-Acting Agents, in *(Meyer & Wilson 1991)*.

Koza, J.R. (1992). Evolution of subsumption using genetic programming, in *(Varela & Bourgine 1992)*.

Kube, C.R., H. Zhang (1994). Collective robotics : From social insects to robots, *Adaptive Behavior*, 2(2):189-218.

Kuipers, B., Y. Byun (1991). A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations, *Robotics and Autonomous Systems*, 8:47-63.

Kusiak, A. (1990). Manufacturing systems : A knowledge- and optimization-based approach, *Journal of Intelligent and Robotic Systems*, 3:27-50.

Langton, C.G., Ed. (1988*a*). *Artificial Life, The Proceedings of an Interdisciplinary Workshop on the Synthesis and Simulation of Living Systems*, September 1987, Los Alamos, New Mexico, Addison-Wesley, Redwood City, CA.

Langton, C.G. (1988*b*). Artificial Life, in *(Langton 1988a)*.

Langton, C.G., C. Taylor, J.D. Farmer, S. Rasmussen, Eds. (1991). *Artificial Life II, Proceedings of the Workshop on Artificial Life*, February 1990, Santa Fe, New Mexico, Addison-Wesley.

Langton,C.G., Ed. (1994) *Artificial Life III*, Addison-Wesley, Reading, MA.

Levi, P., T. Will (1994). A study of the dynamical behaviour of a self-organised production system, *Poster Presentation, Artificial Life IV Workshop*, Cambridge, MA, July, 6p.

Lévi-Strauss, C. (1962). *La pensée sauvage*, Librairie Plon, Paris.

Lindenmayer, A., P. Prusinkiewicz (1988). Developmental models of multicellular organisms : A computer graphics perspective, in *(Langton 1988a)*.

Lindgren, K. (1991). Evolutionary Phenomena in Simple Dynamics, pp. 295-312, in *(Langton et al. 1991)*.

Lumer, E.D., B. Faieta (1994). Diversity and adaptation in populations of clustering ants, in *(Cliff et al. 1994)*, pp. 501-508.

Maes, P. (1989). The Dynamics of Action Selection, *Proceedings of the 1989 International Joint Conference on Artificial Intelligence*, Detroit, pp. 991-997.

Maes, P., Ed. (1991*a*). *Designing Autonomous Agents : Theory and Practice from Biology to Engineering and Back*, MIT Bradford Press, reprinted from Special Issue of the Journal on Robotics and Autonomous Systems, 6(1&2), June 1990.

Maes, P. (1991*b*). A Bottom-Up Mechanism for Action Selection in an Artificial Creature, in *(Meyer & Wilson 1991)*.

Maes, P. (1991*c*). Situated Agents Can Have Goals, in *(Maes 1991a).*

Maes, P. (1991*d*). Adaptive Action Selection, *Proceedings Cognitive Science Conference*.

Maes, P. (1991*e*). The Agent Network Architecture (ANA), *Proceedings AAAI Spring Symposium on Integrated Intelligent Architectures*, March.

Maes, P. (1991*f*). Learning Behavior Networks from Experience, in *(Varela & Bourgine 1992)*.

Maes, P. (1994*a*). Modeling adaptive autonomous agents, *Artificial Life*, 1(1&2):135-162.

Maes, P. (1994*b*). Agents that reduce work and information overload, *Communications of the ACM*, 37(7):31-40, July.

Malcolm, C.A., T. Smithers (1988). Programming assembly robots in terms of task achieving behavioural modules : First experimental results, *Proceedings International Advanced Robotics Programme, 2nd Workshop on Manipulators, Sensors and Steps Towards Mobility*, Manchester, also *DAI Research Paper 410*, Edinburgh University.

Malcolm, C., T. Smithers (1991). Symbol grounding via a hybrid architecture in an autonomous assembly system, in *(Maes 1991a)*.

Manderick, B. (1992). Selectionist systems as cognitive systems, in *(Varela & Bourgine 1992)*, pp. 441-447.

Mataric, M.J. (1990). Environment learning using a distributed representation, *Proceedings 1990 IEEE International Conf. on Robotics and Automation*, Cincinatti, Ohio, May.

Mataric, M.J. (1991). Navigating with a rat brain : A neurobiologically-inspired model for robot spatial representation, in *(Meyer & Wilson 1991)*.

Mataric, M.J. (1992*a*). Integration of representation into goal-driven behavior-based robots, *IEEE Transactions on Robotics and Automation*, 8(3):304-312.

Mataric, M.J. (1992*b*). Minimizing complexity in controlling a mobile robot population, *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, Nice, France, May.

Mataric, M.J. (1992*c*). Designing emergent behaviors : From local interactions to collective intelligence, in *(Meyer et al. 1992)*.

Mataric, M.J. (1994). *Interaction and intelligent behavior*, Ph.D. Thesis, MIT Artificial Intelligence Laboratory Technical Report AITR-1495, May.

Mataric, M.J., M. Nilsson, K.T. Simsarian (1995). Cooperative multi-robot box-pushing, *Proceedings IROS'95*, Pittsburgh, PA.

Matsumoto, A., H. Asama, Y. Ishida, K. Ozaki, I. Endo (1990). Communication in the autonomous and decentralized robot system ACTRESS, *Proceedings IEEE International Workshop on Intelligent Robots and Systems (IROS) '90*, Japan.

Maturana, H.R., F.J. Varela (1980). *Autopoiesis and cognition - The realization of the living*, Boston Studies in the Philosophy of Science, Vol. 42. D. Reidel Publishing, Dordrecht/Boston.

Maturana, H.R., F.J. Varela (1987). *The tree of knowledge : The biological roots of human understanding*, New Science Library/Shambhala, Boston.

McFarland, D. (1992). Animals as cost-based robots, *International Studies in the Philosophy of Science*, 6(2):133-153.

McFarland, D. (1994). Towards robot cooperation, pp. 440-444, in *(Cliff et al. 1994)*.

McFarland, D., T. Bösser (1993). *Intelligent behavior in animals and robots*, MIT Press, Cambridge, MA.

Meyer, J.-A., S.W. Wilson (1991). *From Animals to Animats, Proceedings of the First International Conference on Simulation of Adaptive Behavior*, Paris, 1990, Bradford MIT Press.

Meyer, J.-A., H.L. Roitblat, S.W. Wilson (1992). *From Animals to Animats 2, Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, Hawaii, 1992, Bradford MIT Press.

Meyer, J.-A., A. Guillot (1991). Simulation of Adaptive Behavior in Animats : Review and Prospect, in *(Meyer & Wilson 1991)*.

Meyer, J.-A., A. Guillot (1994). From SAB90 to SAB94 : Four years of animat research, in *(Cliff et al. 1994)*, pp. 2-11.

Meyer, J.-A. (1995). The animat approach to cognitive science, in *Comparative Approaches to Cognitive Science*, H.L. Roitblat, J.-A. Meyer (Eds.), MIT Press.

Meystel, A. (1985). Intelligent control : Issues and highlights, *Proceedings of the IEEE Symposium on Intelligent Control*.

Meystel, A. (1988). Intelligent control in robotics, *Journal of Robotic Systems*, 5:269-308.

Meystel, A. (1989). Intelligent control : A sketch of the theory, *Journal of Intelligent and Robotic Systems*, 2:97-107.

Meystel, A. (1991). *Autonomous mobile robots - Vehicles with cognitive control*, World Scientific.

Millán, J. del R. (1994). Learning efficient reactive behavioral sequences from basic reflexes in a goal-directed autonomous robot, pp. 266-274, in *(Cliff et al. 1994)*.

Minsky, M., Ed. (1968). *Semantic Information Processing*, MIT Press, Cambridge, MA.

Minsky, M. (1986*a*). *The Society of Mind*, Simon and Schuster, New York.

Minsky, M., Ed. (1986*b*). *Robotics*, Omni Press, Garden City, New York.

Minsky, M. (1991). Logical versus analogical or symbolic versus connectionist or neat versus scruffy, *AI Magazine*, 12(2):34-51.

Monod, J. (1970). *Le hasard et la nécessité - Essai sur la philosophie naturelle de la biologie moderne*, Éditions du Seuil, Paris.

Moravec, H. (1988). *Mind children - The future of human and robot intelligence*, Harvard University Press, Cambridge, MA.

Morin, E. (1977). *La méthode 1. La nature de la nature*, Éditions du Seuil, Paris.

Morin, E. (1980). *La méthode 2. La vie de la vie*, Éditions du Seuil, Paris.

Morowitz, H.J. (1994). Artificial biochemistry, life before enzymes, pp. 381-388, in *(Langton 1994)*.

Nolfi, S., D. Floreano, O. Miglino, F. Mondada (1994). How to evolve autonomous robots : Different approaches in evolutionary robotics, pp. 190-197, in *(Brooks & Maes 1994)*.

Noreils, F. (1993). Toward a robot architecture integrating cooperation between mobile robots : Application to indoor environment, *International Journal of Robotics Research*, 12(1):79-98.

Nowak, M.A., R.M. May (1992). Evolutionary games and spatial chaos, *Nature*, 359:826-829.

Nowak, M., K. Sigmund (1992). Tit-for-tat in heterogeneous populations, *Nature*, 355:250-253.

Numaoka, C (1995). Introducing the blind hunger dilemma: Agents' properties and performance, pp. 290-296, *Proceedings ICMAS'95*, San Francisco, CA.

Parker, L.E. (1992). Adaptive action selection for cooperative agent teams, in *(Meyer et al. 1992)*.

Parker, L.E. (1994). *Heterogeneous multi-robot cooperation*, Ph.D. Thesis, MIT AI Lab TR-1465.

Payton, D.W., J.K. Rosenblatt, D.M. Keirsey (1990). Plan guided reaction, *IEEE Transactions on Systems, Man and Cybernetics*, 20(6):1370-1382.

Payton, D.W., T.E. Bihari (1991). Intelligent real-time control of robotic vehicles, *Communications of the ACM*, 34(8).

Payton, D.W., D. Keirsey, J. Krozel, K. Rosenblatt (1992). Do whatever works: A robust approach to fault-tolerant autonomous control, *Journal of Applied Intelligence*, Vol. 2, pp. 225-250.

Peng, J., R.J. Williams (1993). Efficient learning and planning within the Dyna frameowrk, *Adaptive Behavior*, 1(4):437-454.

Pitrat, J. (1991). *Métaconnaissance - Futur de l'Intelligence Artificielle*, Hermès, Paris.

Pollock, J.L. (1993). The phylogeny of rationality, *Cognitive Science*, 17:563-588.

Premvuti, S., S. Yuta (1990). Consideration on the cooperation of multiple autonomous mobile robots, *Proceedings 1990 IEEE International Workshop on Intelligent Robots and Systems (IROS '90)*, Japan.

Premvuti, S., S. Yuta (1993). An experiment in realizing autonomous and cooperative behaviours between multiple mobile robots, *IEEE International Workshop on Emerging Technologies and Factory Automation (Technology for the Intelligent Factory)*, pp. 301-304, R. Zurawski and T. S. Dillon (Eds.), Melbourne, Australia.

Prusinkiewicz, P. (1994). Visual models of morphogenesis, *Artificial Life*, 1(1&2):61-74.

Ram, A., R. Arkin, G. Boone, M. Pearce (1994). Using genetic algorithms to learn reactive control parameters for autonomous robotic navigation, *Adaptive Behavior*, 2(3):277-305.

Ranky, P.G. (1986). *Computer integrated manufacturing - An introduction with case studies*, Prentice-Hall, Englewood Cliffs, NJ.

Rasmussen, S., C. Knudsen, R. Feldberg (1991). Dynamics of programmable matter, pp. 211-254, in *(Langton et al. 1991)*.

Ray, T.S. (1991). An approach to the synthesis of life, pp. 371-408, in *(Langton et al. 1991)*.

Ray, T.S. (1994). An evolutionary approach to synthetic biology: Zen and the art of creating life, *Artificial Life*, 1(1&2):179-209.

Renders, J.-M., R. Hanus (1992). Biological learning metaphors for adaptive process control : A general strategy, *Proceedings of the 1992 International Symposium on Intelligent Control*, Scotland, August.

Resnick, M., F. Martin (1990). Children and Artificial Life, *M.I.T. E&L Memo No. 10*, November.

Reynolds, C. (1987). Flocks, herds and schools : A distributed behavioural model, *SIGGRAPH-87*, 21(4).

Ribeiro, F., J.-P. Barthès, E. Oliveira (1992). Dynamic selection of action sequences, pp. 189-195, in *(Meyer et al. 1992)*.

Roitblat, H.L. (1991). Cognitive action theory as a control architecture, in *(Meyer & Wilson 1991)*.

Rolstadås, A., Ed. (1988). *Computer-aided production management*, Springer Verlag, Berlin/Heidelberg.

Rose, M.R., C.E. Finch, Eds. (1994). *Genetics and evolution of aging*, Kluwer Academic Publishers, Dordrecht/Boston.

Rosen, R. (1992). Cells and senescence, in E. E. Bittar (Ed.), *Developmental biology*, JAI Press, Greenwich, CT, pp. 191-203.

Rosenschein, S.R., L.P. Kaelbling (1986). The synthesis of digital machines with provable epistemic properties, in *"Theoretical Aspects of Reasoning About Knowledge"*, by J.Y. Halpern (Ed.), Morgan-Kaufmann, pp. 83-98.

Rusting, R.L. (1992). Why do we age?, *Scientific American*, December, pp. 86-95.

Saridis, G.N. (1985). Foundations of the theory of intelligent control, *Proceedings 1st International Symposium on IEEE Symposium on Intelligent Control*.

Saridis, G.N. (1987). Knowledge implementation : Structures of intelligent control systems, *Proceedings 1987 IEEE International Symposium on Intelligent Control*, Philadelphia, PA.

Schöner, G., C. Engels (1994). Dynamic field architecture for autonomous systems, pp. 242-253, *Proceedings 1994 Perception to Action Conference (PerAc'94)*, Lausanne.

Schmajuk, N.A. (1994). Behavioral dynamics of escape and avoidance : A neural network approach, pp. 118-127, in *(Cliff et al. 1994)*.

Schoppers, M.J. (1987). Universal plans for reactive robots in unpredictable environments, *Proceedings 1987 International Joint Conference on Artificial Intelligence*, pp. 1039-1046.

Schraft, R.D., E. Degenhart, M. Hägele (1993). Service robots: The appropriate level of automation and the role of operators in the task execution, *Proceedings of the 1993 IEEE Systems, Man and Cybernetics Conference*, pp. 163-169.

Schrödinger, E. (1944). *What is life ? The physical aspect of the living cell*, Cambridge University Press.

Sejnowski, T., C. Koch (1994). Report to the National Science Foundation : Workshops on Neuromorphic Engineering, July.

Shin, K.G., M.E. Epstein (1985). Communication primitives for a distributed multi-robot system, *Proceedings 1985 International Conference on Robotics and Automation*, St. Louis, MO.

Situated Action (1993). Special Issue on Situated Action, *Cognitive Science*, 17(1993).

Slotine, J.-J.E. (1994). Stability in adaptation and learning, in *(Cliff et al. 1994)*, pp. 30-34.

Smithers, T., C.A. Malcolm (1987). A behavioural approach to robot task-planning and off-line programming, *Proceedings International Advanced Robotics*

Programme, 1st Workshop on Manipulators, Sensors and Steps Towards Mobility, Karlsruhe, also *DAI Research Paper 306*, Edinburgh University.

Smithers, T., C.A. Malcolm (1989). Programming robotic assembly in terms of task achieving behavioural modules, *Journal of Structural Learning*, 2(10), also DAI RP 417, Edinburgh University.

Smithers, T. (1994). On why better robots make it harder, pp. 64-72, in *(Cliff et al. 1994)*.

Speckley, R. (1986). Scenes from the twenty-first century, in *(Minsky 1986b)*.

Stahl, W.R., H.E. Goheen (1963). Molecular algorithms, *Journal of Theoretical Biology*, 5(1963):266-287.

Steels, L. (1990). Cooperation Between Distributed Agents through Self-Organisation, in *(Demazeau & Müller 1990a)* and in *Proceedings IEEE International Workshop on Intelligent Robots and Systems (IROS) '90*, Japan.

Steels, L. (1991*a*). Exploiting analogical representations, in *(Maes 1991a)*.

Steels, L. (1991*b*). Towards a Theory of Emergent Functionality, in *(Meyer & Wilson 1991)*.

Steels, L. (1994*a*). The artificial life roots of artificial intelligence, *Artificial Life*, 1(1&2):75-110, Fall 1993/Winter 1994.

Steels, L. (1994*b*). Emergent functionality of robot behavior through on-line evolution, pp. 8-14, in *(Brooks & Maes 1994)*.

Steels, L. (1994*c*). A case-study in the behavior-oriented design of autonomous agents, pp. 445-452, in *(Cliff et al. 1994)*.

Stewart, J., F.J. Varela (1991). Morphogenesis in shape-space, elementary meta-dynamics in a model of the immune metwork, *Journal of Theoretical Biology*, 153(1991):477-498.

Suchman, L. (1987). *Plans and situated actions - The problem of human-machine communication*, Cambridge University Press, Cambridge, England.

Sugihara, K., I. Suzuki (1990). Distributed motion coordination of multiple mobile robots, *Proceedings 5th IEEE International Symposium on Intelligent Control 1990*, September.

Sutton, R.S. (1991). Reinforcement learning architectures for animats, in *(Meyer & Wilson 1991)*.

Suzuki, T., T. Kanehara, A. Inaba, S. Okuma (1993). On algebraic and graph structural properties of assembly petri nets, *Proceedings of the 1993 IEEE International Conference on Robotics and Automation*, Atlanta, GA, pp. 507-514.

Tamayo, P., H. Hartman (1988). Cellular automata, reaction-diffusion systems and the origin of life, in *(Langton 1988a)*.

Taylor, C.E. (1991). "Fleshing out" Artificial Life II, in *(Langton et al. 1991)*.

Teitelbaum, P., V.C. Pellis, S.M. Pellis (1991). Can Allied Reflexes Promote the Integration of a Robot's Behavior?, in *(Meyer & Wilson 1991)*.

Thearling, K., T.S. Ray (1994). Evolving multi-cellular artificial life, pp. 283-288, in *(Brooks & Maes 1994)*.

Theraulaz, G., S. Goss, J. Gervet, J.-L. Deneubourg (1991). Task Differentiation in Polistes Wasp Colonies : A Model for Self-Organizing Groups of Robots, in *(Meyer & Wilson 1991)*.

Toates, F. (1986). *Motivational systems*, Cambridge University Press, Cambridge, UK.

Toates, F. (1994). What is cognitive and what is *not* cognitive?, in *(Cliff et al. 1994)*, pp. 12-21.

Travers, M. (1988). Animal Construction Kits, in *(Langton 1988a)*.

Trivers, R. (1971). The evolution of reciprocal altruism, *Quarterly Review of Biology*, 46(4):35-57.

Tsotsos, J.K. (1995). Behaviorist intelligence and the scaling problem, *Artificial Intelligence*, 75(1995):135-160.

Turing, A.M. (1952). The chemical basis of morphogenesis, *Philosophical Transactions of the Royal Society of London, Series B: Biological Sciences*, 237:37-72.

Tyrrell, T. (1993*a*). *Computational mechanisms for action selection*, Ph.D. Thesis, Centre for Cognitive Science, University of Edinburgh.

Tyrrell, T. (1993*b*). The use of hierarchies for action selection, *Adaptive Behavior*, 1(4):387-420.

Tyrrell, T. (1994). An evaluation of Maes's bottom-up mechanism for behavior selection, *Adaptive Behavior*, 2(4):307-348.

Tzafestas, E. (1995). Introduction to RAGS Release 1.0, *LAFORIA*, January 1995. Revised version : January 1996.

Ueyama, T., T. Fukuda, F. Arai (1992). Configuration of communication structure for distributed intelligent robotic system, *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, Nice, France, May.

van de Welde, W. (1991). Toward learning robots, *Robotics and Autonomous Systems*, 8(1991):1-6.

van Gelder, T., R. Port (1995). It's about time : An overview of the dynamical approach to cognition, *Research Report 116*, Indiana University at Bloomington, Dept. of Cognitive Science, October 1994, in *"Mind as Motion: Explorations in the dynamics of cognition"*, R. Port & T. van Gelder (Eds.), Bradford/MIT Press, 39p.

Varela, F. (1979). *Principles of biological autonomy*, Elsevier/North-Holland, New York, French edition *"Autonomie et connaissance : Essai sur le vivant"*, Editions du Seuil, Paris.

Varela, F. (1989). *Connaître - Les sciences cognitives, tendances et perspectives*, Éditions du Seuil, Paris.

Varela, F.J., E. Thompson, E. Rosch (1991) *The embodied mind - Cognitive science and human experience*, MIT Press.

Varela, F.J., P. Bourgine (1992). *Toward a practice of autonomous systems, Proceedings of the First European Conference on Artificial Life*, Paris, December, MIT Press/Bradford Books.

Verschure, P.M.F.J., B.J.A. Kröse, R. Pfeifer (1992). Distributed adaptive control : The self-organization of structured behavior, *Robotics and Autonomous Agents*, 9(1992):181-196.

Verschure, P.M.F.J., R. Pfeifer (1992). Categorization, representations, and the dynamics of system-environment interaction : A case study in autonomous systems, pp. 210-217, in *(Meyer et al. 1992)*.

von Bertalanffy, L. (1968). *General system theory*, George Braziller, Inc., New York, traduit en français sous le titre *"Théorie générale des systèmes"*, Dunod, Paris, 1973.

von Neumann, J. (1966). *Theory of self-reproducing automata*, Edited and completed by A. W. Burks, University of Illinois Press, Urbana and London.

Wang, J., G. Beni (1988). Pattern generation in cellular robotic systems, *Proceedings 1988 IEEE International Symposium on Intelligent Control*, Arlington, VA, August.

Wang, J., G. Beni (1989). Cellular robotic system with stationary robots and its application to manufacturing lattices, *Proceedings 1989 IEEE International Symposium on Intelligent Control*, Albany, NY, September.

Wang, J., G. Beni (1990). Distributed computing problems in Cellular Robotic Systems, *Proceedings IEEE International Workshop on Intelligent Robots and Systems (IROS) '90*, Japan.

Wang, J. (1990). *The theory and engineering of cellular robotic systems*, Ph.D. Thesis, University of California, Santa Barbara, June.

Wang, P.K.C. (1991). Navigation strategies for multiple autonomous mobile robots moving in formation, *Journal of Robotic Systems*, 8(2):177-195.

Warner, H.R., R.N. Butler, R.L. Sprott, E.L. Schneider, Eds. (1987). *Modern biological theories of aging*, Raven Press, Aging Series Vol. 31, New York, NY.

Weinberg, G.M. (1975). *An Introduction to General Systems Thinking*, John Wiley & Sons.

Werner, E. (1989). Cooperating agents: A unified theory of communication and social structure, pp. 3-36, in *(Gasser & Huhns 1989)*.

Werner, E. (1990). Distributed cooperation algorithms, pp. 17-31, in *(Demazeau & Müller 1990)*.

Werner, G.M. (1994). Using second order neural connections for motivation of behavioral choice, pp. 154-161, in *(Cliff et al. 1994)*.

Wiener, N. (1961). *Cybernetics - or control and communication in the animal and the machine*, 2nd Edition, MIT Press, Cambridge, MA.

Wilson, E.O. (1975). *Sociobiology - The new synthesis*, Belknap Press/Harvard University Press, Cambridge, MA.

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

Wright, P.K. D.A. Bourne, Eds. (1988). *Manufacturing intelligence*, Addison-Wesley, Reading, MA.

Yamauchi, B.M., R.D. Beer (1994). Sequential behavior and learning in evolved dynamical neural networks, *Adaptive Behavior*, 2(3):219-246.

Zeghal, K. (1994). Vers une théorie de la coordination d'actions : Application à la navigation aérienne, *Thèse de Doctorat de l'Université Paris VI,* Décembre.

Zs.-Nagy, I. (1994). *The membrane hypothesis of aging*, CRC Press, London.

Annexe A Le modèle tit-for-tat quantitatif

A.1 Introduction

Ma motivation derrière les expérimentations en modélisation sociale a été le besoin de trouver un modèle de socialité qui pourrait ensuite être instancié et intégré avec des comportements asociaux d'agents autonomes. Si on recherche une similitude biologique, ce modèle de socialité généralisée doit ressembler la socialité vertébrée plutôt que l'eusocialité des insectes, afin de permettre l'émergence de possibilités cognitives de plus haut niveau à partir de possibilités réactives et pro-sociales (ce point a été soulevé par McFarland (1994)). On a déjà dit dans le chapitre 1 que cela se traduit en une réciprocité motivée des agents stimulée par la reconnaissance des agents-frères et que cela renvoie aux conclusions correspondantes de la biologie de l'évolution (Hamilton 1964; Trivers 1971) et de la théorie de coopération de Axelrod (1984). Ce modèle de socialité doit encore être réactif, adaptatif et dimensionnable, afin de permettre l'intégration avec les autres comportements réactifs et adaptatifs et afin de permettre l'exploration de son potentiel pour des fonctionnalités cognitives plus élaborées. Comme on l'a aussi souligné, les comportements sociaux considérés ne doivent pas se limiter aux situations de jeu, où les agents sont par définition compétitifs, mais ils doivent inclure plus généralement des relations de participation, où les agents prennent des mesures individuelles de satisfaction sociale.[115] L'idée est que les phénomènes sociaux sont dus principalement aux évaluations individuelles différentes de la même situation sociale ; ces "évaluations" dépendent en effet des besoins personnalisés des différents agents ainsi que de leurs structures de perception et d'action et la diversité est modélisée comme une variation génétique de départ (bien entendu, un processus d'adaptation au cours de la vie des agents aura tendance à amplifier ou à restreindre cette diversité).

À part les travaux en coopération et en jeux d'évolution, ce modèle a été partiellement stimulé du paradigme de Eco-Problem-Solving (EPS) et les modèles de sélection d'action d'agents autonomes (Maes 1989) (Ribeiro et al. 1992). EPS (Ferber 1989*b*) (Ferber & Jacopin 1991) repose sur un modèle d'agent ayant trois variables d'état binaire (satisfaction, fuite, liberté), qui définissent un total de six états (il existe trois états impossibles/inaccessibles) et un ensemble d'actions primitives qui affectent ces variables (Ferber & Jacopin 1990). Ce modèle a été utilisé avec succès pour résoudre

[115] Dans ce sens, la participation correspond à ce que Axelrod et Dion (1988) appellent "a behavior-dependent context of play". C'est encore consistant avec l'analyse opérationnelle de Becker (1976) qui a suggéré que l'altruisme (la coopération) a un avantage sélectif dans des contextes qui impliquent des interactions sociales ou physiques généralisées. De Sousa (1990) soutient que les modèles sociobiologiques sont des candidats valables pour l'étude des phénomènes sociaux.

des problèmes classiques en intelligence artificielle (Ferber & Jacopin 1991) (Drogoul 1993) qui sont généralement abordés à l'aide de méthodes de planification. Cependant, ce paradigme ne définit pas de mesure de coopération qui pourrait être utilisé pour diriger un système multi-agents vers un consensus collectif, puisque la satisfaction est binaire ; il est alors appliqué avec succès seulement dans des cas qui impliquent non une coopération mais une coordination spatiale d'actions. Le besoin de fonctions continues de satisfaction a été identifié dans le contexte des agents autonomes (par exemple, Steels 1994*b*) et est directement issu des réflexions concernant la sélection d'action.

A.2 Le tit-for-tat et la quantification

Comme tout comportement, les comportements sociaux dépendent d'une motivation interne à l'agent (qui est inversement proportionnelle au degré de satisfaction sociale que l'agent éprouve) et d'un ou de plusieurs stimuli externes ; ces stimuli externes expriment des mesures de l'activité sociale correspondante qui peuvent être interprétées comme amicales (coopératives) ou hostiles (défectives). On attend d'un agent d'être coopératif dans des mondes coopératifs et défectif dans des mondes hostiles. La perception de l'amicalité ou hostilité du monde se fait à l'aide d'un critère de seuil : si l'activité sociale dépasse le seuil, le monde est considéré comme coopératif, sinon il est considéré comme hostile. L'agent va ainsi modifier son degré individuel de participation à l'activité sociale, selon sa propre évaluation. Le stimulus perçu est une sorte "d'objet social" (ou "hint", selon Hogg & Huberman (1993)), abstrait or matériel, directement visible par et accessible aux agents. Les actions coopératives ou défectives des agents affectent la valeur de cet objet, de manière que l'interaction sociale prenne la forme d'un processus continu et auto-catalysant qui repose sur la présence de motivations communes des agents (cf. fig. A.1 pour une récapitulation du modèle)

Comportement social
Si perception(stimulus_ social) $\geq T$ *,*
alors "environnement coopératif ou amical, alors coopération"
sortie a(perception)
sinon "environnement non coopératif ou hostile, alors défection"
sortie b(perception)
où :
stimulus_social est l'objet social impliqué,
T est le seuil de coopération,
a() et b() sont les fonctions d'action de l'agent dans un environnement amical ou hostile, respectivement.

Figure A.1 Le modèle tit-for-tat quantitatif [116]

Ce modèle est en réalité un modèle tit-for-tat généralisé et quantifié, dans lequel les perceptions, les actions et la satisfaction de l'agent sont continues plutôt que binaires ou discrètes ; on a utilisé comme modèle de base le modèle tit-for-tat original, parce qu'il a été prouvé optimal et "evolutionary stable" (Dawkins 1976, Axelrod & Hamilton 1981). La quantification du modèle permet la variation au niveau de la

[116] Dans un contexte de jeu, tel que le dilemme du prisonnier, un agent tit-for-tat est un agent qui a la stratégie simple de coopérer au début du jeu et de répéter ensuite le dernier mouvement de son adversaire (Axelrod & Hamilton 1981).

population, ainsi que l'introduction des composantes adaptatives capables de modifier ces actions afin d'avoir une meilleure satisfaction de l'agent. Les fonctions *a()* and *b()* sont essentiellement des fonctions métaboliques, tandis que la fonction de perception (dénotée désormais *f()*) peut inclure des mécanismes de filtrage qui distordent la "réalité objective" ; toutes les trois fonctions peuvent inclure des effets de bord, tels que la mobilité, l'agression et la fuite, et on peut les complexifier à volonté. Le filtrage peut encore être considérée comme une interprétation de la réalité en termes de ce que l'agent comprend. Malgré l'absence de représentations explicites des croyances, intentions etc., ces trois fonctions sont en effet des fonctions implicites prédictives du comportement des autres agents. La mesure de coopération donnée par la fonction de perception exprime une satisfaction individuelle immédiate plutôt que la fitness ultime de l'agent. Cette mesure est la moyenne des participations de tous les agents en interaction et elle peut être vue comme une *rétribution inclusive*, par analogie avec la fitness inclusive de Hamilton. L'hétérogénéité inhérente dans le modèle permet de constater que la coopération, la défection et un ensemble d'autres phénomènes sociaux n'existent pas réellement, mais ils sont dans l'oeil de l'observateur, tandis que les agents, de leur côté, essayent simplement de maximiser leurs satisfactions individuelles : notons, cependant, que contrairement à la situation dans un jeu, les agents dans notre contexte n'ont aucune volonté particulière de coopérer ou pas coopérer, il n'existe pas d'agent manipulateur qui profite du jeu.

A.3 Applications

J'ai programmé trois applications du modèle tit-for-tat quantitatif pour explorer les effets des différentes formes de variation ou diversité au niveau d'une population fixe sans étudier les populations/sociétés dans une perspective évolutive.

A.3.1 Adaptation et dépendances environnementales : Variations de paramètres

La première application est un problème d'affection où les agents cherchent à maximiser l'affection qu'ils reçoivent des autres agents (qui est calculée comme la moyenne sur l'ensemble des agents). La question est, ***quel est le mécanisme qui maximise la satisfaction individuelle ?*** Les fonctions *a()*, *b()* et *f()* des agents sont $a(x)=a*x$, $b(x)=b*x$ et $f(x)=x$ (pas de filtrage de la réalité) et toutes les valeurs sont normalisées entre 0 et 1. L'agent tit-for-tat classique (modèle non quantifié) correspond à $a=b=1$. Un agent est "rationnel", s'il est plus affectif quand il coopère que quand il ne coopère pas : $a(T) \geq b(T)$ d'où $a \geq b$. Ce problème devient intéressant en présence de diversité des agents, parce que si les agents sont identiques ils coopèrent depuis le début et le système reste stable (dans ce cas les agents n'ont pas de problème à résoudre).

Variations et perturbations. J'ai simulé le système avec une diversité des paramètres *a*, *b* et *T*, ainsi qu'avec une perturbation sociale, dans laquelle avec une certaine probabilité *p* ($0<p<1$) la valeur de l'affection globale est multipliée d'un facteur de *f* ($0<f<1$). Dans tous les cas, le système s'est stabilisé rapidement (dans moins de 10 cycles) à une configuration où tous les agents sont défectifs ; en réalité, *la perturbation ne joue aucun rôle en présence de diversité*. Notons que la variation de paramètres peut être considérée comme la cause génétique de "misimplementation"et "misperception" (Axelrod & Dion 1988), tandis que les perturbations correspondent à des modifications dans les "règles du jeu".

Agents adaptatifs. J'ai ensuite introduit des structures d'adaptation qui affectent *a* et *b* ou *T*. La première alternative était ***l'adaptation active*** qui affecte *a* et *b* (si on n'est pas content, il faut apprendre à donner plus d'affection soi-même) et la deuxième ***l'adaptation passive*** qui affecte *T* (si on n'est pas content, il faut apprendre à se contenter de moins d'affection). Dans les deux cas, la formule d'adaptation est une formule proportionnelle *a=a+taux*diff* (ou *T=T-taux*diff*) où *diff=T-affection_moyenne_perçue_durant_w* et *taux*, *w* sont le taux et la fenêtre d'adaptation respectivement. L'adaptation a lieu seulement en cas de défection (*diff>0*). L'introduction de ce mécanisme d'adaptation n'a pas changé qualitativement la dynamique du système. Pourquoi ? Il semble que les agents doivent calculer leur satisfaction de manière "introspective", c'est-à-dire ils doivent se sentir satisfaits lorsqu'ils coopèrent eux-mêmes et non pas lorsqu'ils perçoivent la coopération (*diff=T-affection_moyenne_offerte durant_w*). Ceci profite de l'introduction d'un paramètre supplémentaire : ***générosité***, qui est la probabilité *g* ($0 \leq g \leq 1$) de coopérer dans un monde hostile. Les expérimentations ont montré que les performances des agents passifs sont nettement supérieures à celles des agents actifs qui sont à leur tour légèrement meilleures que celles des agents non adaptatifs. Dans tous les cas, les variations conduisent à une augmentation des performances par rapport au cas des agents uniformes. Le résultat de la comparaison des agents actifs et passifs est que *le paramètre critique pour la coopération est le seuil* (la perception) et pas le *a* (l'action).

Adaptation et environnement. J'ai ensuite voulu explorer la relation entre activité/passivité et degré de perturbation environnementale : ma prévision était que les agents actifs seraient plus performants dans les mondes plus prévisibles — peu perturbés — tandis que les agents passifs seraient plus performants dans les mondes plus perturbés et moins prévisibles. Un ensemble d'expérimentations avec des populations homogènes et hétérogènes dans des mondes de perturbation variée ont montré que les performances des agents ne dépendent pas du degré de perturbation (qui paraît contre-intuitif, puisque la perturbation est une invitation à l'apprentissage). J'ai alors conclu que *la variation interne est plus importante que la perturbation externe* et que *le mode passif est toujours plus performant indépendamment des spécificités de l'environnement*.

Agents multi-modaux. Ensuite, j'ai testé des agents qui possèdent les deux modes d'adaptation et qui essaient les deux pour converger au mode le plus performant (j'ai programmé ce mécanisme en introduisant deux probabilités de sélection de mode actif ou passif qui dépendent du succès cumulé du mode — initialement, *p(actif)=p(passif)=0.5*). À ma surprise initiale, les agents multi-modaux n'ont pas pu se stabiliser au mode le plus performant (le mode passif) et la raison en est que le mode actif a "exploité" la supériorité du mode passif qui modifiait le seuil dans le bon sens.

Adaptation de générosité. J'ai expérimenté avec des agents qui adaptent également les paramètres de générosité, et j'ai trouvé comme prévu que les performances augmentaient de façon drastique, mais aussi que les agents actifs devenaient alors beaucoup plus performants que les passifs (parce qu'ils étaient par défaut moins opérationnels, donc la mesure *diff* était plus importante).

A.3.2 Une économie artificielle : Variations de structures

La deuxième application est une économie artificielle, c'est-à-dire une population d'agents qui reçoivent un salaire et qui font des dépenses, qui consomment. L'objet social est la consommation/dépense des agents. La question est : ***est-ce que les agents irrationnels peuvent survivre dans l'économie ?*** Les agents sont satisfaits s'ils peuvent dépenser autant qu'ils décident : le critère de coopération est $consommation\_dé cidé e \leq capital * seuil$, où le *seuil* est un facteur de sécurité et le *capital* est la richesse cumulée de l'agent. Si les décisions des agents ne sont pas "irrationnelles", ils auront des degrés de satisfaction élevés.

Chaque agent reçoit à chaque cycle un salaire *s* et décide de consommer *c*facteur_aléatoire*, où *c* est le coût associé et *facteur_aléatoire* est un facteur qui prend une valeur entre 0.9 et 1.1 (le coût n'est pas exactement le même à chaque cycle). À part cette consommation régulière, les agents font des investissements périodiques (chaque T_i) qui sont calculés comme $(s-c)*T_i*seuil*un\_facteur\_aléatoire$, où *un_facteur_aléatoire* est comme avant (cela correspond à un investissement "préplanifié" qui est consistant avec la rationalité de l'agent). J'ai défini trois types d'agents consommateurs :

(a) Les consommateurs rationnels. La régulation de leurs dépenses est conservatrice, ce qui leur permet de faire des investissements. Ils montrent un degré d'adaptation sociale par mimétisme en décalant un peu leurs dépenses vers la moyenne sociale. $a(x)=x$, $b(x)=capital*seuil$ (=la consommation maximale permise) et $f(x)=consommation\_décidée+ 0.1*(x-consommation\_décidée)$.

(b) Les consommateurs agressifs. Ils ne font pas d'investissements, mais ils veulent de temps en temps se montrer riches en dépensant plus que tous les autres. Ils ont un $seuil=1$ (c'est-à-dire ils peuvent dépenser d'un coup toute leur richesse). Leur agressivité est exprimée périodiquement (chaque T_i) et nécessite un paramètre supplémentaire : $f(x) = max (x*(1+agressivité), consommation\_décidée)$.

(c) Les consommateurs "scrooge".[117] Ils ont un "sentiment d'insécurité" qui se traduit en un filtre de perception un peu particulier et un paramètre supplémentaire : si les autres agents sont contents quand leurs dépenses propres ne dépassent pas un seuil de sécurité, les agents scrooge sont contents quand la moyenne sociale ne dépasse cette limite ($f(x)=x$). Chaque fois que cela est faux, ces agents réduisent alors absurdément leur seuil de sécurité en utilisant une formule proportionnelle $seuil=seuil-(x*insécurité/capital)$. Ils ne font pas d'investissements et n'ont pas de mécanisme d'adaptation sociale.

Rationalité versus variation versus agressivité. J'ai simulé un éventail de populations homogènes et hétérogènes avec des variations des coûts et des salaires et j'ai trouvé que *la satisfaction des agents ne dépend pas de la variation tant que les agents restent rationnels*, mais les agents agressifs avec un ratio *c/s* près de 1 ou un facteur élevé d'agressivité et les agents scrooge avec un facteur élevé d'insécurité ont tendance à devenir de plus en plus mécontents. Cependant, il existe des agents agressifs et scrooge qui réussissent à survivre et à augmenter leurs richesses au cours des simulations aux côtés des agents rationnels qui sont toujours les plus performants. Généralement, on peut constater que les agents scrooge sont des agents marginaux dans le sens où ils ne participent pas trop à l'évolution de la société.

---

[117] En leur donnant ce nom, je fais allusion au héros avare de Dickens.

Une économie inégale. J'ai ensuite simulé le système avec des sous-populations d'agents riches et pauvres et j'ai trouvé que les agents les moins contents étaient les agents pauvres et trop agressifs ou trop avares. Pour les agents agressifs, j'ai encore trouvé que, plus ils sont agressifs, moins le ratio *c/s* compte.

Perturbations exogènes. J'ai ensuite voulu tester la stabilité de l'économie face à deux perturbations exogènes "brusques", l'augmentation des coûts ou l'augmentation des salaires. Dans les deux cas, les agents les plus touchés de la perturbation étaient les agents les moins opérationnels : en cas de hausse des coûts, les agents très agressifs ont été conduits au désespoir, tandis que les agents très avares se sont relativement détendus. En cas de hausse des salaires, on obtient les résultats inverses (les agents agressifs se détendent et les agents avares détériorent).

Manipulabilité de l'économie. Après avoir prouvé la robustesse des agents rationnels, j'ai effectué un dernier jeu de simulations pour identifier les conditions sous lesquelles les agents irrationnels deviendraient très malheureux. J'ai alors trouvé que l'introduction d'agents agressifs mais très riches pourraient conduire les agents pauvres très irrationnels à la mort. La conclusion de ces simulations est que les agents rationnels sont robustes, tandis que les agents irrationnels sont relativement manipulables, puisqu'ils dépendent plus radicalement du contexte social dans lequel ils se trouvent. Plus particulièrement, les agents agressifs sont irrationnels quant à leurs actions et alors ils peuvent s'avérer très dangereux pour toute la société puisqu'ils modifient les standards sociaux, tandis que les agents scrooge sont irrationnels quant à leurs perceptions et c'est pourquoi ils peuvent devenir dangereux seulement pour eux-mêmes. Notons finalement qu'en dehors d'un contexte social particulier on ne peut pas juger des performances d'un agent à partir de son paramétrage seul.

A.3.3 Productivité, mobilité et agressivité : Variations d'interactions

Dans la dernière application développée, j'ai cherché à comprendre la relation entre spatialité et socialité. J'ai alors défini un système dans lequel l'objet social en question exprime la "productivité" et les agents sont situés dans une grille bidimensionnelle ; un agent cherche à maximiser sa participation sociale et il est content si la productivité moyenne dans sa place est proche de la sienne. Si ce critère n'est pas satisfait, il fuit dans une place voisine. Les fonctions du modèle tit-for-tat quantitatif sont instanciées comme suit : $f(x)=x$, $a(x)=$*productivité* et $b(x)=0$ (avec fuite comme effet de bord), *seuil=productivité*tolérance* et *productivité* et *tolérance* sont deux paramètres supplémentaires normalisés.

Variations et besoin d'agressivité directe. J'ai simulé le système avec des variations aux paramètres des agents et j'ai trouvé que les agents les plus heureux étaient les moins productifs indépendamment de la valeur de leur *tolérance*, parce que, malgré le besoin de fuites fréquentes, ils se retrouvaient contents dans presque tout contexte social. Inversement, les agents les plus malheureux étaient les agents très productifs et peu tolérants, parce qu'ils ne pouvaient pas explicitement chercher à se satisfaire en chassant les agents beaucoup moins productifs et devaient faire confiance à la possibilité de ces derniers agents de détecter la baisse de productivité sociale et de fuir. Or, souvent les agents peu productifs ne peuvent pas s'en rendre compte pour fuir. *Ce qui paraît alors nécessaire est un mécanisme d'agressivité directe*.

Implémentation d'agressivité. Le nouveau modèle modifié pour l'agent agressif est présenté dans fig. A.2. Alors la satisfaction de l'agent est définie comme détection de coopération et absence d'agression. J'ai simulé le système des agents agressifs et j'ai trouvé que leur comportement est devenu beaucoup plus naturel et plus "équilibré" qu'avant : les agents ont tendance à chasser les agents moins productifs et à fuir des agents plus productifs. Apparemment, les agents très productifs fuient beaucoup moins souvent que les agents peu productifs. Ce mécanisme d'agressivité directe rend alors le système beaucoup plus "performant" puisque les agents sont en moyenne beaucoup plus heureux qu'en son absence (il permet par exemple aux agents très productifs et peu tolérants de maximiser leur satisfaction). Notons que telle qu'elle a été définie, l'agressivité de l'agent n'est pas une motivation originale de chasser les autres mais simplement un mécanisme de résolution de conflits (Galliers (1990) a fait des observations similaires). Finalement, il est intéressant de remarquer encore une fois que les agents les plus malheureux sont les agents plus productifs, parce que c'est eux qui peuvent être plus facilement manipulés par le contexte social.

Comportement agressif
Si perception(social_stimulus) ≥ T,
alors
s'il existe une agression dans l'environnement
si je suis en colère (depuis le cycle d'exécution précédent)
"précédemment en colère, je suppose que tout va bien maintenant,
je retourne dans mon état normal et je donne une nouvelle chance"
sortie a(perception)
sinon "environnement hostile,
puisque je ne suis pas en colère, je suis probablement la cause de l'agression des autres, alors je fuis"
sortie b(perception) et fuite comme effet de bord
sinon "tout va bien"
sortie a(perception)
sinon "Je suis plus productif/fort que les autres, alors j'agresse,
mais je continue à produire normalement"
sortie a(perception) et agression comme effet de bord

Figure A. 2 L'agent agressif (avec une mémoire d'un pas)

Agressivité et isolationnisme. J'ai ensuite voulu voir si le modèle de l'agent agressif favorise l'isolation des agents. J'ai alors simulé la même population d'agents dans des environnements de taille variée et j'ai trouvé qu'un environnement suffisamment grand permet aux agents de s'isoler, tandis qu'un environnement plus petit les force à chercher des compromis avec les autres agents. [118] En fait ces compromis peuvent être perçus par un observateur extérieur comme une "tricherie subtile" ("subtle cheating", Trivers 1971), où les agents "trompés" persistent dans leur choix ainsi "sous-optimal", parce qu'il est moins coûteux que l'absence totale d'interaction. Les expériences ont encore montré qu'un agent socialement inadapté (tel qu'un agent peu productif dans une société de productifs ou un agent très productif dans une société de paresseux) a tendance à s'isoler et ceci non à cause de ses possibilités propres mais à cause de ses interactions avec les autres agents.

J'ai aussi exploré le potentiel auto-organisationnel du système face à des perturbations (par exemple, une re-initialisation des vitesses) et j'ai trouvé que le système tend

[118] On a ignoré des facteurs tels que les coûts de communication (qui rendent les petits groupes préférables) et les mésures absolues de produit (qui rendent les grands groupes souhaitables).

toujours à se stabiliser à une nouvelle configuration et que si les perturbations sont fréquentes et importantes le système va paraître toujours en état de transition. Finalement, j'ai effectué un nombre d'expériences avec des motivations variées des agents et j'ai montré que les agents ont tendance à se rassembler dans des groupes sociaux selon ces motivations. Comme dans l'application précédente, on a montré que le même agent peut avoir un destin différent dans des environnements différents et aussi que la variation est la base de l'auto-organisation et de la stabilité face aux perturbations ; Dans la plupart des cas étudiés, *les contraintes supplémentaires sont des mécanismes qui induisent encore plus de variation*.

A.4 Discussion

Le modèle tit-for-tat quantitatif est un modèle comportemental sociobiologique[119] qui repose sur les principes de réciprocité et de reconnaissance sociale et qui peut s'instancier dans toute une gamme de problèmes de modélisation sociale qui impliquent une participation sociale de l'agent sans se restreindre dans un contexte de jeu. Le modèle permet en plus une hétérogénéité de paramètres et de structures : les agents essaient de résoudre leur instanciation propre de l'unique problème qui existe, la maximisation individuelle de satisfaction, autrement dit le prolongement de leur vie. *La stabilité et la robustesse aux perturbations sont précisément la conséquence des variations et des interactions entre les agents hétérogènes*.

Pour prendre une idée de la complexité potentielle d'un système d'agents tit-for-tat quantitatifs, on peut considérer deux agents entre qui les interactions peuvent être classifiées dans un des quatre types du tableau A.1 (Hamilton 1964) :

Émetteur	*Destinataire*	
	+	-
+	mutualité	fraude
-	espionnage	hostilité

Tableau A.1 Les quatre types d'interaction entre deux agents selon Hamilton (1964) (+ et - dénotent la direction de changement du fitness de l'agent après l'interaction)

L'agent i (i=1,2) peut être complètement spécifié par le quadruple (T_i, $a_i()$, $b_i()$, $f_i()$). La *mutualité* émerge lorsque $y(1)=(a_1(f_1(1))+a_2(f_2(1)))/2 \geq max(T_1,T_2)$, où $y(t)$ est la valeur de l'objet social après t cycles. L'hostilité émerge lorsque $y(1)<min(T_1,T_2)$. Entre les deux extrêmes, on rencontre tout un éventail de cas d'espionnage/fraude (les deux phénomènes sont complémentaires : quand un agent qui joue en premier est espionné c'est comme si l'autre agent jouait en premier et le trompait). L'espionnage/la fraude va être continu(e), c'est-à-dire l'agent trompé ne va jamais comprendre qu'il est trompé, si :

$$f_1(y(1)) \geq T_1, f_2(y(1))<T_2, \text{ avec } y(1) = (a_1(f_1(1))+a_2(f_2(1)))/2,$$
$$f_1(((a_1(y(t)) + b_2(y(t))) / 2) \geq T_1, \ \forall t=1,2...,$$

[119] La diversité et l'adaptativité font que le comportement actuel d'un agent particulier ne peut pas être prédit dans un vide environnemental, c'est-à-dire en dehors d'un contexte social spécifique, et peut varier largément de contexte à contexte. Dans ce sens, ce modèle sociobiologique renvoie à une potentialité génétique plutôt qu'à un déterminisme génétique (cf. (Gould 1977) et (De Sousa 1990) pour des discussions à ce sujet).

$$f_2(((a_1(y(t)) + b_2(y(t))) / 2) < T_2, \forall t=1,2...,$$
avec $y(t)=a_1(f_1(y(t-1))) + b_2(f_2(y(t-1)))/2$.

S'il existe un *t* tel qu'au moins une de ces inégalités devient fausse, c'est le moment où le système se résout à l'hostilité. On peut voir que s'il existe plus de deux agents spécifiés par des quadruples différents, la complexité des expressions analytiques qui décrivent le système monte rapidement. On peut encore dire que l'agent-i est "rationnel" si

$$f_i(x) \geq T_i \Rightarrow f_i(a_i(f_i(x))) \geq T_i, \text{ et}$$
$$f_i(x) < T_i \Rightarrow f_i(b_i(f_i(x))) < T_i$$ [120].

Ces inégalités signifient que l'agent est rationnel s'il traite ses propres actions de façon rationnelle, c'est-à-dire s'il détecte le résultat d'une action coopérative (non coopérative) de lui-même comme coopérative (non coopérative), sinon et selon le contexte de l'interaction il va paraître comme idiot ou méchant (*"sucker"* ou *"con man"*, respectivement (Axelrod 1984)). Bien entendu, seulement peu d'agents sont conformes à cette règle : la plupart du temps, on rencontre des agents qui possèdent des filtres sophistiqués de perception et qui inclinent par exemple à croire qu'ils sont plus intelligents ou mieux informés que leurs voisins ou qu'ils ont un autre avantage relatif. Dans tous les cas, un agent "irrationnel" peut sous certaines conditions sociales survivre et prospérer, donc la rationalité ne peut qu'avoir un sens ainsi individualiste : Pollock (1993) a souligné que la rationalité, qu'il a définie comme "l'outil pour bien vivre" (*"the tool for living the good life"*), est dépendante du contexte et est contrainte des possibilités, de la connaissance et de l'environnement de l'agent. Alors les agents agressifs et scrooge de l'économie artificielle peuvent dans certains contextes être considérés comme des agents rationnels ! Cependant, il faut remarquer que les agents rationnels ont été les agents les moins manipulables et ceci *sans* posséder des capacités "cognitives" plus élaborées. La raison de cette robustesse est que leurs décisions et leurs actions reposent sur des facteurs principalement endogènes et leur réactivité aux événements extérieurs est restreinte : ***ce qui se manifeste alors comme rationalité est en effet un conservatisme et une résistance aux changements, à l'innovation***.[121] Notons que l'évolution des standards sociaux dans l'économie artificielle est due exclusivement à la présence d'agents agressifs, c'est-à-dire à la présence d'agents *manipulables*! La proportion des agents manipulables dans la population doit cependant rester faible pour assurer sa stabilité face aux perturbations.

[120] Si *a()*, *b()* et *f()* sont des fonctions linéaires, $a(x)=k_a x$, $b(x)=k_b x$ et $f(x)=k_f x$, ces inégalités se traduisent en $k_b<(T_i/k_f^2)<k_a$ — la présence de la fonction *f()* de filtrage change la forme de l'espace de perception.

[121] Cela ne doit pas être une simple coincidence qu'en vieillissant on devient à la fois plus rationnel et plus conservateur.

Annexe B Simulations et simulateurs

B.1 Simulation

Une simulation d'un système réel est nécessaire et utile si l'expérimentation avec le système n'est pas pratique. Une simulation est encore utile s'il n'existe pas de méthode d'analyse du système réel ou si elle a un coût prohibitif. Finalement, une simulation est utile si les propriétés ou les résultats recherchés ne peuvent pas être décrits analytiquement (comme c'est par exemple le cas des formations spatiales du "game of life" de Conway). Dans tous les cas, la simulation d'un système repose sur un modèle du système tel qu'il existe une correspondance 1:1 avec le système, de façon que toutes les opérations possibles du système réel aient un équivalent dans le modèle et que l'état du système réel puisse être déduit sans ambiguïté de l'état de son modèle (et vice versa). Un modèle est alors une entité plus facile à manipuler et expérimenter avec que le système réel et les résultats dans l'espace d'états du modèle peuvent être transposés dans l'espace d'états du système réel. Les simulations effectuées dans le cadre de cette thèse présentent la même particularité que les simulations rencontrées dans les travaux en vie artificielle : on présume un modèle comportemental qui peut ne pas avoir un équivalent réel (ou pour lequel on cherche à voir si les phénomènes auxquels il donne naissance ressemblent à ceux retrouvés avec un système réel)[122] et ***on simule sa dynamique faute de moyen d'analyse*** — c'est le cas des systèmes complexes, hétérogènes et adaptatifs. Dans ce sens, la simulation du modèle doit être contrastée à son analyse et pas à une implémentation éventuelle robotique ou autre. Tant qu'on reste à un niveau abstrait comportemental, les résultats obtenus ont une valeur qualitative et indépendante des détails d'implémentation, mais alors — comme Smithers (1994) l'a souligné — on ne peut pas prétendre pouvoir évaluer en simulation et prédire certains aspects d'une implémentation du modèle.

Puisque les modèles en question contiennent un grand nombre de paramètres, leur étude ne peut qu'avoir un caractère itératif : définition et exécution d'un ensemble d'expériences (simulations) avec plusieurs paramétrages et évaluation des résultats obtenus qui conduira au raffinement du modèle et ainsi de nouveau. Faute d'outil convenable, j'ai programmé un simulateur qui devait être à la fois assez générique pour servir de plate-forme d'expérimentation pour les différents modèles que j'ai envisagés depuis le début et assez performant dans les expérimentations particulières. Mon souci principal pendant la phase de la programmation a été donc d'automatiser le processus de définition et d'exécution de simulations, afin d'accélérer l'étude et de me concentrer au développement et la comparaison des modèles.

J'ai alors défini et développé un mécanisme hiérarchique de description/spécification des simulations à base d'un système standard et uniforme d'initialisation et d'exécution. Ainsi, les résultats obtenus avec un modèle particulier sont directement associés à sa spécification, ce qui permet de "structurer" le processus d'expérimentation. Le système développé possède en plus un ensemble d'outils qui facilitent la définition de nouveaux modèles et de nouvelles simulations ; il est ainsi à la fois un simulateur et un générateur de simulations.

---

[122] Les modèles étudiés sont alors des réalités alternatives plutôt que des représentations de la réalité.

B.2 RAGS : Un simulateur d'agents réactifs

RAGS (Reactive AGents Simulator) a été développé pendant une période initiale de six mois et il abstrait actuellement les caractéristiques communes aux simulations effectuées avec l'ensemble des modèles présentés dans cette thèse. J'ai aussi programmé une variante qui répond aux spécificités de la simulation des agents cellulaires. Le simulateur offre un ensemble d'outils pour faciliter la définition d'une simulation et rendre le système accessible à des utilisateurs qui ne connaissent pas ses détails d'implémentation et de fonctionnement, mais qui veulent définir et expérimenter avec leurs propres modèles dans un temps raisonnable. Le simulateur a été également utilisé par un stagiaire de DEA dans le cadre de l'étude de (Benos & Tzafestas 1995), ce qui m'a permis de valider et de raffiner ses fonctionnalités.

La taille du code source de la version actuelle du simulateur est environ 830K et le langage de base est ObjectWorks/Smalltalk-80, version 4.1, avec le générateur d'interfaces VisualWorks, version 2.0. La taille du code total, qui comprend en plus le code de toutes les applications programmées, est environ 1.9M. Le simulateur RAGS est décrit plus en détail dans (Tzafestas 1995).

Composantes de simulation. Une simulation multi-agents (une expérience) est une instance de la classe-mère *RagsProblem*, ou d'une de ses sous-classes, et contient les composantes suivantes (les entités de simulation) :

(a) *Agents* : Toutes les entités modélisées comme des agents, qui sont alors engagées dans une activité et qui répondent aux événements qui ont lieu dans leur environnement. Les agents sont différents des objets (passifs) à cause de leur nature dynamique ou de leur mobilité. Ils peuvent être simples ou composés d'autres agents autonomes et situés ou pas (l'explorateur du chapitre 4 est situé dans un monde physique, tandis que ses cellules-composantes sont généralement considérées comme des entités non situées). Ils sont typiquement hétérogènes quant à leurs possibilités.

(b) *Monde (ou environnement)* : C'est le monde dans lequel les agents sont situés et qu'ils peuvent percevoir et modifier à volonté. En général, il s'agit d'un repositoire géométrique (bidimensionnel) d'agents, d'objets et de signaux immatériels hétérogènes. Toutes les "communications" entre agents se font par l'intermédiaire de cet espace, qui constitue alors un cadre unifié de perception et de communication.

(c) *Observateurs* : C'est un ensemble d'objets qui fonctionnent comme des "parasites" de la simulation. Ils sont attachés à toutes les entités (les agents ou le monde) qui doivent être observées pendant la simulation pour obtenir et enregistrer les résultats, directs ou indirects, désirés : ces résultats sont ensuite traduits automatiquement en statistiques qui peuvent être sauvegardées ou visualisées sur une fenêtre. Le mécanisme d'observation est développé à base du mécanisme de *dépendance* de Smalltalk-80. Il y a un éventail de classes d'observateurs simples ou composés.

(d) *Terminaison* : C'est un objet représentant une expression booléenne (c'est-à-dire une condition logique) simple ou composée et paramétrable qui sert de critère de terminaison de la simulation (complétion de l'expérience). Elle peut dépendre des autres entités de la simulation, par exemple de l'état d'un agent particulier ou du monde.

Initialisation et exécution. L'initialisation de la simulation à partir de sa spécification implique une initialisation récursive de ses différentes composantes (monde, agents, observateurs, terminaison). Le pas de la simulation (ou le cycle d'exécution) est une boucle à travers la liste des agents. Chacun des agents exécute à son propre rythme (qui dépend de son horloge interne personnelle) et selon ses propres critères.

Spécification. Le mécanisme de spécification de toutes les entités de simulation est indépendant de leur nature, de leur structure et de leur sémantique. Une spécification est un dictionnaire d'associations <clé ou attribut>-<valeur>, où <valeur> peut être une autre spécification. Cela permet d'avoir une méthode d'initialisation récursive uniforme pour toutes les entités : il suffit pour chaque classe d'entités de connaître la liste des attributs de spécification et de pouvoir leur donner des valeurs par défaut, en cas d'absence de valeur spécifiée. Les spécifications ont alors la même structure que les entités correspondantes.

Interfaces. RAGS possède d'interfaces de deux types : interfaces de visualisation et de programmation graphique et interfaces textuelles et fichiers de description. Les mêmes principes de modularité ont été mis en oeuvre pour la définition des interfaces graphiques avec le développement d'une classe abstraite d'interfaces qui inclut toute la fonctionnalité de base (ouverture, sauvegarde, mise à jour, gestion d'interfaces imbriquées etc.). J'ai ensuite programmé un ensemble d'interfaces et d'éditeurs spécifiques pour les diverses classes d'entités, afin de permettre la programmation graphique et l'inspection/édition de la simulation en ligne.

B.3 Illustration : Implémentation des agents sénescents

B.3.1 La hiérarchie des classes

La classe des simulations des agents sénescents
- *SenescenceProblem (sous-classe de RagsProblem)*

La classe des agents sénescents
- *SenescentAgent (sous-classe de SituatedCompositeAgent)*

Les tâches des agents sénescents
- *SocialDistraction (sous-classe de IndivisibleTask)*
- *RegulatedSocialDistraction (sous-classe de SocialDistraction)*
- *GoalAttraction (sous-classe de IndivisibleTask)*
- *RevisableGoalAttraction (sous-classe de GoalAttraction)*
- *DistanceRegulatedGoalAttraction (sous-classe de RevisableGoalAttraction)*
- *ObstacleRemoval (sous-classe de IndivisibleTask)*

Les systèmes de dégradation de socialité, de révision de but et de régulation de distance ou de socialité
- *LinearSocialityRegulationSystem (sous-classe de SimulationObject)*
- *AutocatalyticSocialityRegulationSystem (sous-classe de LinearSocialityRegulationSystem)*
- *GoalRevisionSystem (sous-classe de SimulationObject)*
- *DistanceRegulationSystem (sous-classe de GoalRevisionSystem)*
- *SocialityDegradationSystem (sous-classe de SimulationObject)*

Par la suite est présenté le code comportemental des agents (la plupart du code concernant l'accès, l'initialisation, la visualisation, les interfaces et les protocoles d'expérimentation a été omis).

B.3.2 La simulation de la sénescence

RagsProblem subclass: #SenescenceProblem

```
	instanceVariableNames: ''

!SenescenceProblem methodsFor: 'accessing'!

getAdaptivity
	|n|
	n := self agents size.
	^((self agents inject: 0 into: [:x :y| x+(y adaptivity)]) / n) asFloat!

getSociality
	|n|
	n := self agents size.
	^((self agents inject: 0 into: [:x :y| x+(y sociality)]) / n) asFloat!

maxDistance
	|dim|
	dim := self world dimension.
	^(dim x) + (dim y)! !

!SenescenceProblem methodsFor: 'simulation'!

done
	"Terminaison de la simulation."
	self agents do: [:agt | agt done ifFalse: [^false]].
	^true!

step
	super step.
	"Mise-à-jour des observateurs."
	self changed: #sociality with: self getSociality.
	self changed: #adaptivity with: self getAdaptivity!

!SenescenceProblem class methodsFor: 'specifications'!

defaultAgentsSpec
	"Spécification des agents suivant le premier modèle (modèle réactif)."
	^#(#SenescentAgent
			#(1
				#(	#population 10
					#place #randomPlace
					#spec #default)))!

defaultEnvironmentSpec
	^#(#RagsWorld
		#(	#dimension #(#Point 50 50)
			#worldPlaceSpec #(#TracedPlace #(#TracedPlace #defaultSamplesSpec))))!

defaultObserversSpec
^#(#CompoundObserver
	#(	#Adaptivity #(#HistoryRecorder #(#aspect #adaptivity))
		#Sociality #(#HistoryRecorder #(#aspect #sociality))
		#Agents #(#CollectionObserver
```

```
                        #(      #aspect #agentsAtCaste:
                                #parameter #SenescentAgent
                                #singleSpec #(#CompoundObserver
                                        #(      #Sociality #(#HistoryRecorder #(#aspect #sociality))
                                                #Encounters #(#Enumerator #(#aspect #encounter))
                                                #Adaptivity #(#HistoryRecorder #(#aspect
#adaptivity))
                                                #Deaths #(#Enumerator #(#aspect #death))
                                                #Timeouts #(#Enumerator #(#aspect #timeout))
                                                #Regenerations #(#Enumerator #(#aspect
#regeneration))
                                                #Distance #(#HistoryRecorder #(#aspect
#distance))))))))!

defaultTerminationSpec
        ^#(#Termination #(      #aspect #done
                                #value true))!

fifthAgentsSpecWith: n dimension: dim
        "Spécification des agents suivant le modèle cognitif avec régulation de socialité (modèle 3b)."
        |spec agtSpec|
        agtSpec := Specification new.
        agtSpec at: #place put: #randomPlace; at: #population put: n; at: #spec put:
                SenescentAgent fifthSpec.
        spec := PopulationSpecification new.
        spec
                at: #SenescentAgent
                put: (Specification newWithSpecs: (Array with: agtSpec)).
        spec := self specWithAgentsSpec: spec.
        spec at: #EnvironmentSpec put: (self environmentSpecWithDim: dim).
        ^spec!

fourthAgentsSpecWith: n dimension: dim
        "Spécification des agents suivant le modèle cognitif avec régulation de socialité, mais SANS
        décroissance du timeout de la régulation (modèle 3b sans régulation du timeout)."
        |spec agtSpec|
        agtSpec := Specification new.
        agtSpec at: #place put: #randomPlace; at: #population put: n; at: #spec put:
                SenescentAgent fourthSpec.
        spec := PopulationSpecification new.
        spec
                at: #SenescentAgent
                put: (Specification newWithSpecs: (Array with: agtSpec)).
        spec := self specWithAgentsSpec: spec.
        spec at: #EnvironmentSpec put: (self environmentSpecWithDim: dim).
        ^spec!

secondAgentsSpecWith: n dimension: dim
        "Spécification des agents suivant le modèle motivé sans régulation (modèle 2)."
        |spec agtSpec|
        agtSpec := Specification new.
        agtSpec at: #place put: #randomPlace; at: #population put: n; at: #spec put:
                SenescentAgent secondSpec.
        spec := PopulationSpecification new.
        spec
                at: #SenescentAgent
                put: (Specification newWithSpecs: (Array with: agtSpec)).
        spec := self specWithAgentsSpec: spec.
        spec at: #EnvironmentSpec put: (self environmentSpecWithDim: dim).
        ^spec!

thirdAgentsSpecWith: n dimension: dim
        "Spécification des agents suivant le modèle cognitif avec régulation de distance (modèle 3a)."
        |spec agtSpec|
        agtSpec := Specification new.
        agtSpec at: #place put: #randomPlace; at: #population put: n; at: #spec put:
```

```
            SenescentAgent thirdSpec.
        spec := PopulationSpecification new.
        spec
            at: #SenescentAgent
            put: (Specification newWithSpecs: (Array with: agtSpec)).
        spec := self specWithAgentsSpec: spec.
        spec at: #EnvironmentSpec put: (self environmentSpecWithDim: dim).
        ^spec! !
```

B.3.3 L'agent sénescent

SituatedCompositeAgent subclass: #SenescentAgent

```
        instanceVariableNames: 'heading color '

!SenescentAgent methodsFor: 'initialize'!

initializeFrom: spec
        super initializeFrom: spec.
        self heading: (self class headings at:
                        (((self problem nextRandomFor: #headings at: #one) * 4) truncated + 1))! !

!SenescentAgent methodsFor: 'accessing'!

adaptivity
        | x |
        x := self tasks detect: [:aTask | (aTask isKindOf: RevisableGoalAttraction)]
                                ifNone: [].
        x isNil
            ifTrue: [^0]
            ifFalse: [^x goalRevisor adaptivity]!

distance
        "Distance de régulation du nouveau but."
        | x |
        x := self tasks detect: [:aTask | (aTask isKindOf: DistanceRegulatedGoalAttraction)]
                                ifNone: [].
        x isNil
            ifTrue: [^0]
            ifFalse: [^x goalRevisor distance]!

done
        ^(self tasks select: [:aTask| aTask done = false]) isEmpty!

goal
        ^(self tasks detect: [:aTask | (aTask isKindOf: GoalAttraction)
            | (aTask isKindOf: RevisableGoalAttraction)
            | (aTask isKindOf: DistanceRegulatedGoalAttraction)]) goal!

isAdaptive
        ^(self adaptivity) >= (self threshold)!

sociality
        ^(self tasks detect: [:aTask| aTask isKindOf: SocialDistraction]) range!

threshold
        | x |
        x := self tasks detect: [:aTask | aTask isKindOf: RevisableGoalAttraction]
                                ifNone: [].
        x isNil
            ifTrue: [^1 negated]
            ifFalse: [^x goalRevisor threshold]! !
```

```
!SenescentAgent methodsFor: 'actions'!

doDecaySocialityRestorationFactor
	|x|
	x := self tasks detect: [:aTask| (aTask isKindOf: RegulatedSocialDistraction)] ifNone: [nil].
	x isNil ifFalse: [x doDecaySocialityRestorationFactor]!

doRewardFromSociality
	|x|
	x := self tasks detect: [:aTask| (aTask isKindOf: RevisableGoalAttraction)] ifNone: [nil].
	x isNil ifFalse: [x doRewardFromSociality]!

elapsed: dt
	|ct cm|
	ct := nil. cm := 0.
	self tasks do: [:aTask|		|x|
		x := aTask sense.
		(x > cm) ifTrue: [ct := aTask. cm := x]].
	"Résolution des conflits entre tâches à l'aide des priorités statiques."
	ct notNil ifTrue: [ct act].
	"Mise-à-jour des observateurs."
	self
		changed: #adaptivity with: self adaptivity;
		changed: #sociality with: self sociality;
		changed: #distance with: ((self distance) / (self problem maxDistance))!

!SenescentAgent class methodsFor: 'defaults'!

defaultTasksSpec
	"Spécification des agents suivant le premier modèle (modèle réactif)."
	^Specification
		newWithKeys: (Array
				with: #SocialDistraction
				with: #ObstacleRemoval
				with: #GoalAttraction)
		withSpecs: (Array
				with: SocialDistraction defaultSpec
				with: ObstacleRemoval defaultSpec
				with: GoalAttraction defaultSpec)!

fifthSpec
	"Spécification des agents suivant le modèle cognitif avec régulation de socialité (modèle 3b)."
	|spec|
	spec := self defaultSpec.
	spec at: #tasks put: self fifthTasksSpec.
	^spec!

fifthTasksSpec
	^Specification
		newWithKeys: (Array
				with: #RegulatedSocialDistraction
				with: #ObstacleRemoval
				with: #RevisableGoalAttraction)
		withSpecs: (Array
				with: RegulatedSocialDistraction secondSpec
				with: ObstacleRemoval defaultSpec
				with: RevisableGoalAttraction defaultSpec)!

fourthSpec
	"Spécification des agents suivant le modèle cognitif avec régulation de socialité, mais SANS
	décroissance du timeout de la régulation (modèle 3b sans régulation du timeout)."
	|spec|
	spec := self defaultSpec.
	spec at: #tasks put: self fourthTasksSpec.
	^spec!
```

```
fourthTasksSpec
        ^Specification
                newWithKeys: (Array
                                with: #RegulatedSocialDistraction
                                with: #ObstacleRemoval
                                with: #RevisableGoalAttraction)
                withSpecs: (Array
                                with: RegulatedSocialDistraction defaultSpec
                                with: ObstacleRemoval defaultSpec
                                with: RevisableGoalAttraction defaultSpec)!

headings
        |ar|
        ar := Array new: 4.
        ar      at: 1 put: (0@1);
                at: 2 put: (0@(-1));
                at: 3 put: (1@0);
                at: 4 put: ((-1)@0).
        ^ar!

secondSpec
        "Spécification des agents suivant le modèle motivé sans régulation (modèle 2)."
        |spec|
        spec := self defaultSpec.
        spec at: #tasks put: self secondTasksSpec.
        ^spec!

secondTasksSpec
        ^Specification
                newWithKeys: (Array
                                with: #SocialDistraction
                                with: #ObstacleRemoval
                                with: #RevisableGoalAttraction)
                withSpecs: (Array
                                with: SocialDistraction defaultSpec
                                with: ObstacleRemoval defaultSpec
                                with: RevisableGoalAttraction defaultSpec)!

thirdSpec
        "Spécification des agents suivant le modèle cognitif avec régulation de distance (modèle 3a)."
        |spec|
        spec := self defaultSpec.
        spec at: #tasks put: self thirdTasksSpec.
        ^spec!

thirdTasksSpec
        ^Specification
                newWithKeys: (Array
                                with: #SocialDistraction
                                with: #ObstacleRemoval
                                with: #DistanceRegulatedGoalAttraction)
                withSpecs: (Array
                                with: SocialDistraction defaultSpec
                                with: ObstacleRemoval defaultSpec
                                with: DistanceRegulatedGoalAttraction defaultSpec)! !
```

B.3.4 Les tâches de l'agent sénescent

IndivisibleTask subclass: #GoalAttraction

```
        instanceVariableNames: 'goal priority '
```

```
!GoalAttraction methodsFor: 'initialize'!

initializeFrom: spec
	|pb x y|
	super initializeFrom: spec.
	priority := 1.
	pb := self problem.
	x := ((pb nextRandomFor: #goal at: #x) * (pb world dimension x)) truncated + 1.
	y := ((pb nextRandomFor: #goal at: #y) * (pb world dimension y)) truncated + 1.
	goal := (x@y)! !

!GoalAttraction methodsFor: 'accessing'!

done
	^(self agent place point = goal)!

!GoalAttraction methodsFor: 'actions'!

act
	|agt aPoint dir|
	agt := self agent.
	aPoint := goal - (agt place point).
	"Se diriger en escalier vers le but."
	(aPoint x abs) < (aPoint y abs)
		ifTrue: [dir := (0@(aPoint y sign))]
		ifFalse: [dir := ((aPoint x sign)@0)].
	(dir = (0@0)) ifFalse:
			[agt heading: dir; advanceBy: dir; changed: #move.
			((goal - (agt place point)) = (0@0)) ifTrue: [agt changed: #death]]!

sense
	self done
		ifTrue: [^0]
		ifFalse: [^priority]! !
```

IndivisibleTask subclass: #SocialDistraction

```
	instanceVariableNames: 'range timeout time priority sociality Degradator '

!SocialDistraction methodsFor: 'initialize'!

initializeFrom: spec
	super initializeFrom: spec.
	priority := 2.
	range := (1 to: 5) at:
	(((self agent problem nextRandomFor: #sociality at: #range) * 5) truncated + 1).
	timeout := (1 to: 10) at:
	(((self agent problem nextRandomFor: #sociality at: #timeout) * 10) truncated + 1).
	time := timeout! !

!SocialDistraction methodsFor: 'accessing'!

done
	|agt pointList|
	agt := self agent.
	pointList := (agt findExclPointsInRange: range truncated) asSet.
	pointList do: [:point |
		(self agent place world selectAllObjectsOfCaste: agt caste at: point)
			isEmpty ifFalse: [^false]].
	^true!

!SocialDistraction methodsFor: 'actions'!
```

```
act
	|agt|
	agt := self agent.
	time := time - 1.
	(time > 0) ifTrue:
		[agt heading: (agt class headings at:
			(((self problem nextRandomFor: #sociality at: #heading) * 4) truncated + 1)).
		agt changed: #encounter.
		agt doRewardFromSociality]
	ifFalse:
		[time := timeout truncated.
		[agt place world validPoint: ((agt place point) + (agt heading))] whileFalse:
			[agt heading: (agt class headings at:
			(((self problem nextRandomFor: #sociality at: #heading) * 4) truncated + 1))].
		agt advanceBy: agt heading; changed: #move].
	socialityDegradator act!

sense
	self done
		ifTrue: [self agent changed: #sense with: range. ^0]
		ifFalse: [^priority]! !

!SocialDistraction class methodsFor: 'defaults'!

defaultSocialityDegradatorSpec
	^Specification
		newWithKeys: #(#SocialityDegradationSystem)
		withSpecs: (Array with: SocialityDegradationSystem defaultSpec)!
```

IndivisibleTask subclass: #ObstacleRemoval

```
	instanceVariableNames: 'priority '

!ObstacleRemoval methodsFor: 'initialize'!

initializeFrom: spec
	super initializeFrom: spec.
	priority := 3! !

!ObstacleRemoval methodsFor: 'accessing'!

done
	^((self agent place detectObjectOfCaste: #Sample) = false)!

!ObstacleRemoval methodsFor: 'actions'!

act
	|agt obj|
	agt := self agent.
	obj := agt place detectObjectOfCaste: #Sample.
	(obj = false) ifFalse:
		[agt place removeObject: obj]!

sense
	self done
		ifTrue: [^0]
		ifFalse: [^priority]! !
```

GoalAttraction subclass: #RevisableGoalAttraction

```
	instanceVariableNames: 'goalRevisor '

!RevisableGoalAttraction methodsFor: 'actions'!
```

```
doRewardFromSociality
	goalRevisor doRewardFromSociality!

sense
	self done
		ifTrue: [goalRevisor doRewardFromGoalCompletion].
	goalRevisor act.
	self done
		ifTrue: [^0]
		ifFalse: [^priority]! !

!RevisableGoalAttraction class methodsFor: 'defaults'!

defaultGoalRevisorSpec
	^Specification
		newWithKeys: #(#GoalRevisionSystem)
		withSpecs: (Array with: GoalRevisionSystem defaultSpec)!
```

SocialDistraction subclass: #RegulatedSocialDistraction

```
	instanceVariableNames: 'sociality Regulator '

!RegulatedSocialDistraction methodsFor: 'actions'!

act
	super act.
	sociality Regulator act!

doDecaySocialityRestorationFactor
	sociality Regulator doDecayRestorationFactor! !

!RegulatedSocialDistraction class methodsFor: 'defaults'!

defaultSocialityRegulatorSpec
	^Specification
		newWithKeys: #(#LinearSocialityRegulationSystem)
		withSpecs: (Array with: LinearSocialityRegulationSystem defaultSpec)!

secondSocialityRegulatorSpec
	^Specification
		newWithKeys: #(#AutocatalyticSocialityRegulationSystem)
		withSpecs: (Array with: AutocatalyticSocialityRegulationSystem defaultSpec)!

secondSpec
	|spec|
	spec := self defaultSpec.
	spec at: #socialityRegulator put: self secondSocialityRegulatorSpec.
	^spec! !
```

RevisableGoalAttraction subclass: #DistanceRegulatedGoalAttraction

```
	instanceVariableNames: ''

!DistanceRegulatedGoalAttraction class methodsFor: 'defaults'!

defaultGoalRevisorSpec
	^Specification
		newWithKeys: #(#DistanceRegulationSystem)
		withSpecs: (Array with: DistanceRegulationSystem defaultSpec)! !
```

SimulationObject subclass: #GoalRevisionSystem

```
        instanceVariableNames: 'adaptivity decayRate completionReward sociality Reward timeout time
threshold task '

!GoalRevisionSystem methodsFor: 'initialize'!

initializeFrom: spec
        |pb|
        super initializeFrom: spec.
        pb := self task agent problem.
        adaptivity := (pb nextRandomFor: #goalRevision at: #adaptivity) * 0.5 + 0.5.
        decayRate := (pb nextRandomFor: #goalRevision at: #decayRate) * 0.9 + 0.05.
        completionReward := (pb nextRandomFor: #goalRevision at: #completionReward) * 0.4 + 0.5.
        socialityReward := (pb nextRandomFor: #goalRevision at: #socialityReward) * 0.3.
        timeout := ((pb nextRandomFor: #goalRevision at: #timeout) * 50) truncated + 50.
        threshold := (pb nextRandomFor: #goalRevision at: #threshold) * 0.1 + 0.1.
        time := timeout!



initializeIn: x from: spec
        self initialize; task: x.
        task goalRevisor: self.
        self initializeFrom: spec!

!GoalRevisionSystem methodsFor: 'actions'!

act
        |agt|
        agt := task agent.
        time := time - 1.
        (time <= 0) ifTrue: [time := timeout].
        adaptivity := adaptivity * decayRate.
        (task agent place point = task goal)
        ifTrue:
                [(adaptivity >= threshold) ifTrue:
                        [agt changed: #regeneration.
                        socialityReward := socialityReward * decayRate.
                        completionReward := completionReward * decayRate.
                        timeout := (timeout * decayRate) truncated.
                        time := timeout.
                        self doNewGoal]]
        ifFalse: [(time=timeout) ifTrue:
                [timeout := (timeout * decayRate) truncated.
                time := timeout.
                socialityReward := socialityReward * decayRate.
                completionReward := completionReward * decayRate.
                (adaptivity >= threshold)
                ifTrue:  [agt changed: #timeout.
                        self doTimeout.
                        self doDie; doNewGoal]
                ifFalse: [task goal: task agent place point.
                        agt changed: #death.
                        self doDie]]]!

doDie
        "Do nothing!!"!

doNewGoal
        |pb x y|
        pb := task agent problem.
        x := ((pb nextRandomFor: #goal at: #x)
                        * (pb world dimension x)) truncated + 1.
        y := ((pb nextRandomFor: #goal at: #y)
                        * (pb world dimension y)) truncated + 1.
```

```
        task goal: (x@y)!

doRewardFromGoalCompletion
        adaptivity := (adaptivity * (1 + completionReward)) min: self class maxAdaptivity.

doRewardFromSociality
        adaptivity := (adaptivity * (1 + socialityReward)) min: self class maxAdaptivity.

doTimeout
        "Do nothing!!"! !
```

SimulationObject subclass: #SocialityDegradationSystem

```
        instanceVariableNames: 'timeout time task '

!SocialityDegradationSystem methodsFor: 'initialize'!

initializeFrom: spec
        super initializeFrom: spec.
        timeout := (1 to: 20) at:
                (((self task agent problem nextRandomFor: #socialityDegradation at: #timeout) * 20)
                                truncated + 1).
        time := timeout!

initializeIn: x from: spec
        self initialize; task: x.
        task socialityDegradator: self.
        self initializeFrom: spec!

!SocialityDegradationSystem methodsFor: 'actions'!

act
        time := time - 1.
        (time > 0)
        ifFalse:
                [               |x|
                time := timeout.
                x := task range.
                x := 0.9 * x.
                task range: x.
                x := task timeout.
                x := 0.9 * x.
                task timeout: x.
                (task time > x) ifTrue: [task time: x]]! !
```

GoalRevisionSystem subclass: #DistanceRegulationSystem

```
        instanceVariableNames: 'distance rate '

!DistanceRegulationSystem methodsFor: 'initialize'!

initializeFrom: spec
        |pb|
        super initializeFrom: spec.
        pb := self task agent problem.
        distance := ((pb nextRandomFor: #distanceRegulation at: #distance) * (self maxAllowedDistance))
truncated + 1.
        rate := (pb nextRandomFor: #distanceRegulation at: #rate) * 0.3 + 0.2! !

!DistanceRegulationSystem methodsFor: 'actions'!

doDie
        super doDie.
```

```
        distance := (distance * (1+rate)) rounded min: self maxAllowedDistance!

doNewGoal
        "Rather complicated geometrical function."
        |pb aPoint d dict aSet key n point1 dim x1 x2 y1 y2 flag|
        pb := task agent problem.
        aPoint := task agent place point.
        dim := task agent problem world dimension.
        x1 := aPoint x.
        x2 := (dim x) - (aPoint x).
        y1 := aPoint y.
        y2 := (dim y) - (aPoint y).
        d := (((((pb nextRandomFor: #distanceRegulation at: #noise)
                        * 0.2) + 0.9) * distance) rounded min: self maxAllowedDistance) max: 1.
        dict := Dictionary new.
        dict
                at: #nw put: (x1 + y1 - 2);
                at: #ne put: (x2 + y1 - 1);
                at: #sw put: (x1 + y2 - 1);
                at: #se put: (x2 + y2).
        aSet := dict keys select: [:aKey| d<=(dict at: aKey)].
        key := aSet asArray
                        at: (((pb nextRandomFor: #distanceRegulation at: #direction) * aSet size) truncated
+ 1).
        flag := false.
        aSet := Set new.
        (key = #nw) ifTrue:
                [(d < (x1 min: y1))
                        ifTrue: [n := d. flag := true]
                        ifFalse: [n := (dict at: key) - d].
                flag
                ifTrue:
                        [point1 := aPoint.
                        0 to: n do: [:i|        |yp|
                                yp := (point1 y) - i.
                                aSet add: ((x1 + y1 - yp - d)@yp)]]
                ifFalse:
                        [point1 := 1@1.
                        0 to: n do: [:i|        |xp|
                                xp := (point1 x) + i.
                                aSet add: (xp@(x1 + y1 - xp - d))]]].
        (key = #ne) ifTrue:
                [(d < (x2 min: y1))
                        ifTrue: [n := d. flag := true]
                        ifFalse: [n := (dict at: key) - d].
                flag
                ifTrue:
                        [point1 := aPoint.
                        0 to: n do: [:i|        |yp|
                                yp := (point1 y) - i.
                                aSet add: ((x1 - y1 + yp + d)@yp)]]
                ifFalse:
                        [point1 := (dim x)@1.
                        0 to: n do: [:i|        |xp|
                                xp := (point1 x) - i.
                                aSet add: (xp@(xp - x1  + y1 - d))]]].
        (key = #sw) ifTrue:
                [(d < (x1 min: y2))
                        ifTrue: [n := d. flag := true]
                        ifFalse: [n := (dict at: key) - d].
                flag
                ifTrue:
                        [point1 := aPoint.
                        0 to: n do: [:i|        |yp|
                                yp := (point1 y) + i.
                                aSet add: ((x1 - y1 + yp - d)@yp)]]
```

```
                ifFalse:
                        [point1 := 1@(dim y).
                        0 to: n do: [:i         |xp|
                                xp := (point1 x) + i.
                                aSet add: (xp@(xp - x1 + y1 + d))]]].
        (key = #se) ifTrue:
                [(d < (x2 min: y2))
                        ifTrue: [n := d. flag := true]
                        ifFalse: [n := (dict at: key) - d].
                flag
                ifTrue:
                        [point1 := aPoint.
                        0 to: n do: [:i         |yp|
                                yp := (point1 y) + i.
                                aSet add: ((x1 - yp + y1 + d)@yp)]]
                ifFalse:
                        [point1 := dim.
                        0 to: n do: [:i         |xp|
                                xp := (point1 x) - i.
                                aSet add: (xp@(x1 - xp + y1 + d))]]].
        aPoint := aSet asArray
                        at: (((pb nextRandomFor: #distanceRegulation at: #point) * aSet size) truncated + 1).
        task goal: aPoint!

doTimeout
        super doTimeout.
        distance := (distance * (1-rate)) rounded max: 1!


maxAllowedDistance
        |dim aPoint x1 x2 y1 y2|
        aPoint := task agent place point.
        dim := task agent problem world dimension.
        x1 := aPoint x.
        x2 := (dim x) - (aPoint x).
        y1 := aPoint y.
        y2 := (dim y) - (aPoint y).
        ^(((x1 + y1 - 2)
                max: (x1 + y2 - 1))
                max: (x2 + y1 - 1))
                max: (x2 + y2)! !
```

SimulationObject subclass: #LinearSocialityRegulationSystem

```
        instanceVariableNames: 'timeout time eigenSociality restorationFactor decayRate task '

!LinearSocialityRegulationSystem methodsFor: 'initialize'!

initializeFrom: spec
        |pb|
        super initializeFrom: spec.
        pb := self task agent problem.
        timeout := ((pb nextRandomFor: #socialityRegulation at: #timeout) * 10) + 20.
        time := timeout.
        eigenSociality := ((pb nextRandomFor: #socialityRegulation at: #eigenSociality) * 2) + 2.
        restorationFactor := ((pb nextRandomFor: #socialityRegulation at: #restorationFactor) * 0.3) + 0.2.
        decayRate := ((pb nextRandomFor: #socialityRegulation at: #decayRate) * 0.04) + 0.01!

initializeIn: x from: spec
        self initialize; task: x.
        task socialityRegulator: self.
        self initializeFrom: spec!

!LinearSocialityRegulationSystem methodsFor: 'actions'!
```

```
act
        time := time - 1.
        (time > 0)
        ifFalse:
        [(task agent sociality < eigenSociality) ifTrue:
                [               |x|
                time := timeout.
                x := task range.
                task range: (x + (restorationFactor * (eigenSociality - x))).
                self doDecayRestorationFactor ]]!

doDecayRestorationFactor
        restorationFactor := restorationFactor * (1 - decayRate)! !
```

LinearSocialityRegulationSystem subclass: #AutocatalyticSocialityRegulationSystem

```
        instanceVariableNames: ''

!AutocatalyticSocialityRegulationSystem methodsFor: 'actions'!

act
        time := time - 1.
        (time > 0)
        ifFalse:
        [(task agent sociality < eigenSociality) ifTrue:
                [               |x|
                time := timeout truncated.
                x := task range.
                task range: (x + (restorationFactor * (eigenSociality - x))).
                self doDecayRestorationFactor; doDecayRestorationTimeout]]!

doDecayRestorationTimeout
        timeout := timeout * (1 - decayRate)!
```

C Simulation d'agents cellulaires

C.1 Considérations de programmation

Si la simulation des autres modèles d'agents étudiés ne présente aucune difficulté particulière, il n'est pas de même pour la simulation des agents cellulaires. La particularité de ce modèle concerne la gestion du temps : pour assurer un fonctionnement propre des réseaux cellulaires algorithmiques, il faut trouver un mécanisme de "synchronisation", de façon que les messages soient traités dans l'ordre qui faut. J'ai alors modifié la méthode d'exécution de l'agent cellulaire : elle doit parcourir la liste des cellules autant de fois que la profondeur du réseau cellulaire (la profondeur du réseau est la longueur maximale des chemins de flot de messages, depuis les cellules d'entrée, généralement les cellules capteurs, jusqu'aux cellules de sortie, généralement les cellules actionneurs) en exécutant à chaque parcours seulement les cellules du niveau correspondant. Par exemple, le réseau de la figure 3.6 a une profondeur de 7 et la cellule de composition de la tâche de chasse exécute pendant le quatrième parcours. Ce mécanisme d'exécution présente un seul désavantage, une complexité au moins O(n) où n est le nombre de cellules ; cependant, les fonctions de transfert des cellules étant simples, le temps réel de réponse reste faible. De plus, une décomposition d'un algorithme conformément aux principes du chapitre 3 minimise le nombre des cellules et réduit alors indirectement cette complexité. Les mêmes considérations de synchronisation s'appliquent également au modèle cellulaire du chapitre 7 ; cette fois on parcourt les buffers des différents niveaux en distribuant à chaque parcours les messages des buffers du niveau correspondant. Dans ce cas, la gestion des vitesses diversifiées d'exécution des différents drives est plus délicate aussi et se fait de manière centrale dans le programme informatique.[123]

C'est le problème de la gestion du temps qui a fait émerger les questions de l'espace de représentation et de la physiologie inerte (non cellulaire) : le problème de l'espace de représentation est directement lié à la profondeur du réseau et au nombre des cellules nécessaires pour l'implémentation d'un algorithme, tandis que le problème de la synchronisation est lié au problème de la physiologie et des variables globales (une variable globale représentée comme une cellule induirait un problème de synchronisation si elle devait être affectée par des cellules de niveaux différents - cela s'est d'ailleurs passé dans une version antérieure du programme cellulaire du manager du chapitre 6 où les variables motivationnelles étaient localisées dans des cellules spécialisées).

[123] Cela me fait croire qu'il ne suffit pas de trouver un bon modèle de cellule autonome, il faudrait également penser à une implémentation "naturelle". Est-il possible d'inventer une chimie alternative artificielle dans laquelle les vitesses de réaction seraient aussi directement implémentables que dans notre chimie naturelle ?

J'ai programmé trois versions de réseau cellulaire : dans la première, il y avait autant de classes de cellules que des fonctionnalités de cellules, avec des mécanismes différents de traitement des messages d'entrée (exécution continue, exécution seulement après stimulation etc.) et avec des variables spécialisées pour chaque fonctionnalité. Cette version avait le désavantage pratique de définir un très grand nombre de classes avec peu de fonctionnalité. Le désavantage s'est avéré conceptuel aussi : on supposait que les cellules étaient hétérogènes non seulement selon les fonctionnalités, mais aussi selon les modes d'exécution et d'accès à leurs variables internes. Cette première version non opérationnelle a stimulé la définition du principe de la cellularité.

Une deuxième version beaucoup plus concise en termes de taille de code, et qui a été stimulée d'un essai d'écrire un réseau cellulaire en C++ dans le cadre de l'étude du chapitre 6, définissait une seule classe de cellule avec une fonction de transfert et avec un vecteur de 8 variables internes, ainsi que deux vecteurs de 8 lignes d'entrée et 8 lignes de sortie. Les cellules des différents types étaient alors implémentées comme des cellules ayant des fonctions de transfert différentes qui utilisaient un sous-ensemble des variables internes. En plus, et conformément au principe de la syntaxe homogène, le traitement des entrées et des sorties était uniforme : tous les messages étaient traités comme des nombres réels et toutes les lignes d'entrée et de sortie étaient traitées sans considérations d'ordre, c'est-à-dire comme des ensembles plutôt que des tableaux de valeurs.

Les divers problèmes identifiés m'ont ensuite conduite à modifier cette version quant à la nature des messages qui voyagent dans les connexions : il me semble que seuls les messages réels ne suffisent pas et que si au contraire on permet des messages plus "discrets" (qui pourraient être vus dans certains cas comme des symboles) on gagne en complexité de réseau. Dans la troisième version, les messages pourraient être des objets Smalltalk sans restriction de type ; en effet, j'ai utilisé seulement des valeurs numériques, des symboles et des tableaux ou des dictionnaires de nombres et/ou de symboles (les tableaux et les dictionnaires sont utiles pour représenter les messages concaténés qui se composent et se décomposent). Ces considérations sont discutées du point de vue conceptuel dans le paragraphe 3.6.

Par la suite sont présentés des extraits du code comportemental des agents cellulaires (la plupart du code concernant l'accès, l'initialisation, la visualisation, les interfaces et les protocoles d'expérimentation a été omis).

C.2 Agents cellulaires algorithmiques

Nous présentons des extraits du code comportemental des agents cellulaires algorithmiques et en particulier des extraits provenant de trois classes :

La classe de la cellule algorithmique
ContinuousCell (sous-classe de SimpleAgent)

La classe des robots cellulaires algorithmiques
CellularRobot (sous-classe de SituatedCompositeAgent)

Le robot cellulaire qui correspond à l'application du chapitre 6 (régulation d'atelier).
Pedro (sous-classe de CellularRobot)

SimpleAgent subclass: #ContinuousCell

```
        instanceVariableNames: 'transferFn inputLines outputLines agent internalTime timer marker markValue
variables '

!ContinuousCell methodsFor: 'initialize-release'!

initialize
        super initialize.
        inputLines := Set new.
        outputLines := Set new.
        marker := 0.
        variables := Array new: 8 withAll: 0!

initializeFrom: spec
        super initializeFrom: spec.
        variables := (spec at: #variables) copy.
        self resetInternalTime!

!ContinuousCell methodsFor: 'accessing'!

mark
        marker := marker - 1!

marked
        ^(marker=0)!

resetInternalTime
        internalTime := timer!

unmark
        marker := markValue!

!ContinuousCell methodsFor: 'transfer functions'!

aFn: list
        |col x|
        col := list asArray.
        x := ((col first) - (col at: 2)) abs.
        (x > 0.5) ifTrue: [x := x - 0.5].
        (x = 0.5) ifTrue: [x := 0.5 + (self class noTracePoint)].
        (x < (0.8 * (self class saturationPoint))) ifTrue: [^x].
        ^self class saturationPoint!

bFn: list
        |col x a z|
        col := list asArray.
        x := ((col first) - (col at: 2)) abs.
        a := self class noTracePoint.
        (x>(a-1)) & (x<a) ifTrue: [x := x + 1].
        (x>(a+0.5))
                ifTrue: [x := x - 0.5]
                ifFalse:
                        [(x = 0.75) ifTrue: [x := 0.25].
                        (x = 0.5) ifTrue: [x := a + 0.5]].
        z := x.
        ^z!

cFn: list
        (list size < 2) ifTrue: [^nil].
        ^self
```

```
                encode: ((list inject: 0 into: [:sum :x| sum + x]) * 10)
                with: (self variablesAt: 1)
                base: self class decoderBase!

composeMotivation: list
        |m dir dict r local|
        dir := self class saturationPoint.
        m := 0.
        r := self agent releaser.
        local := self variablesAt: 1.
        (r isNil) | (r = local) ifTrue:
                [list do: [:rec|        |x|
                        x := rec at: #motivation ifAbsent: nil.
                        x notNil ifTrue: [m := x].
                        x := rec at: #direction ifAbsent: nil.
                        x notNil ifTrue: [dir := x]]].
        dict := Dictionary new.
        dict at: #motivation put: m; at: #direction put: dir; at: #releaser put: local.
        ^dict!

computeMotivation: list
        |agt x dict dem|
        agt := self agent.
        x := agt varAt: (self variablesAt: 1).
        dem := (agt problem findUnitCalled: (self variablesAt: 1)) demand.
        (x = 0.5)
                ifTrue: [x := x * dem]
                ifFalse: [x := x * dem * (self variablesAt: 2)].      "Persistence factor"
        dict := Dictionary new.
        dict at: #motivation put: x.
        ^dict!

decode: list
        |x dir|
        list isEmpty ifTrue: [^nil].
        x := (self class saturationPoint) * 0.8.
        (GenericUtility shuffleArray: list asArray) do:
                [:elm|      |data dir1 val|
                        val := self decode: elm base: self class decoderBase.
                        data := val at: #first.
                        dir1 := val at: #second.
                        (data < x) ifTrue: [x := data. dir := dir1]].
        dir notNil ifTrue:
                [(self equal: dir with: 0.25 threshold: 0.1) ifTrue: [^#left].
                (self equal: dir with: 0.5 threshold: 0.1) ifTrue: [^#front].
                (self equal: dir with: 0.75 threshold: 0.1) ifTrue: [^#right].
                (self equal: dir with: 1 threshold: 0.1) ifTrue: [^#back]].
        ^nil!

dirToCenter: list
        |agt dict dir aPoint aSymbol|
        agt := self agent.
        aPoint := (agt problem center) - (agt place point).
        ((aPoint x abs) < (aPoint y abs))
                ifTrue: [dir := (0@(aPoint y sign))]
                ifFalse: [dir := ((aPoint x sign)@0)].
        dir = (0@0)
        ifTrue: [dir := self class saturationPoint]
        ifFalse:
                [aSymbol := agt computeRelativeDirectionFrom: dir.
                aSymbol = #front ifTrue: [dir := 0.5].
                aSymbol = #back ifTrue: [dir := 1].
                aSymbol = #right ifTrue: [dir := 0.75].
                aSymbol = #left ifTrue: [dir := 0.25]].
        dict := Dictionary new.
        dict at: #priority put: (self variablesAt: 1); at: #direction put: dir.
```

```
        ^dict!

dirToUnit: list
        |agt aPoint dir aSymbol|
        agt := self agent.
        aPoint := (agt problem findUnitCalled: (self variablesAt: 1)) place point.
        aPoint := aPoint - (agt place point).
        ((aPoint x abs) < (aPoint y abs))
        & (agt rnd next < 0.95)
                ifTrue: [dir := (0@(aPoint y sign))]
                ifFalse: [dir := ((aPoint x sign)@0)].
        dir = (0@0)
        ifTrue: [dir := self class saturationPoint]
        ifFalse:
                [aSymbol := agt computeRelativeDirectionFrom: dir.
                aSymbol = #front ifTrue: [dir := 0.5].
                aSymbol = #back ifTrue: [dir := 1].
                aSymbol = #right ifTrue: [dir := 0.75].
                aSymbol = #left ifTrue: [dir := 0.25]].
        ^dir!

dummy: list
        ^0!

motor: list
        |dir agt|
        list isEmpty ifTrue: [^nil].
        dir := list asArray first.
        dir isNil ifTrue: [^nil].
        agt := self agent.
        agt
                turnTo: dir;
                advanceBy: (agt computeAbsoluteDirectionFrom: #front);
                changed: #move!

moveToUnit: list
        |agt dict dir|
        agt := self agent.
        dir := list asArray first.
        dict := Dictionary new.
        dict at: #priority put: (self variablesAt: 1); at: #direction put: dir.
        (dir = self class saturationPoint) ifTrue: [^dict].
        agt center ifTrue: [^dict]
        ifFalse: [^nil]!

obstacleBack: list
        ^self bumper: #back!

obstacleFront: list
        ^self bumper: #front!

obstacleLeft: list
        ^self bumper: #left!

obstacleRight: list
        ^self bumper: #right!

prio: list
        |m dir dict|
        dir := self class saturationPoint.
        m := 0.
        list do: [:rec|         |x|
                x := rec at: #priority.
                (x > m) ifTrue:
                        [m := x. dir := rec at: #direction]].
        dict := Dictionary new.
```

```
        dict at: #direction put: dir.
        ^dict!

prioAll: list
        |m dir rel|
        dir := self class saturationPoint.
        rel := nil.
        (list size = 0) ifTrue:
                [(self agent) timer: 1; resetReleaser. ^dir].
        m := 0.
        (GenericUtility shuffleArray: list) do: [:rec|         |x|
                x := rec at: #motivation.
                (x > m) ifTrue:
                        [m := x. dir := rec at: #direction. rel := rec at: #releaser]].
        self agent setReleaserTo: rel.
        (self agent timer = 0) ifTrue:
                [(dir = self class saturationPoint)
                        ifTrue: [self agent timer: (self variablesAt: 1)]                "Ts"
                        ifFalse: [self agent timer: (self variablesAt: 2)]].             "Tm"
        ^dir!

traceBack: list
        ^self trace: #back!

traceFront: list
        ^self trace: #front!

traceLeft: list
        ^self trace: #left!

traceRight: list
        ^self trace: #right! !

!ContinuousCell methodsFor: 'utilities'!

connectTo: aCell
        |line|
        line := CellConnection input: self output: aCell data: 0.
        self outputLines add: line.
        aCell inputLines add: line!

decode: x base: b
        |dict val|
        dict := Dictionary new.
        val := (x/b) rounded.
        dict at: #first put: val; at: #second put: (x-(b*val)).
        ^dict!

encode: x with: y base: b
        ^(x*b) + y!

equal: x with: y threshold: z
        ^(x - y) abs <= z!

execute
        |result|
        result := self perform: self transferFn withArguments: (Array with: self scanInputs).
        self output: result!
greater: x than: y threshold: z
        ^(x - y) > z!

output: x
        self outputLines do: [:connection| connection data: x]!

scanInputs
        ^(self inputLines asArray collect: [:connection| connection data]) select: [:x| x notNil]! !
```

```
!ContinuousCell methodsFor: 'private'!

bumper: dir
	|pb absoluteDirection flag agt aPoint|
	agt := self agent.
	pb := agt problem.
	absoluteDirection := agt computeAbsoluteDirectionFrom: dir.
	aPoint := agt place point + absoluteDirection.
	flag := (pb isValidPlaceAt: aPoint) not.
	flag ifTrue: [^self class saturationPoint].
	dir = #front ifTrue: [^0.5].
	dir = #back ifTrue: [^1].
	dir = #right ifTrue: [^0.75].
	^0.25!

trace: dir
	|env absoluteDirection flag agt aPoint x|
	agt := self agent.
	env := agt place world.
	absoluteDirection := agt computeAbsoluteDirectionFrom: dir.
	aPoint := agt place point + absoluteDirection.
	flag := true.
	(env validPoint: aPoint) ifTrue:
		[flag := ((env at: aPoint) selectObjectsOfCaste: #Sample) isEmpty].
	(self agent rnd next < 0.1) ifTrue: [flag := flag not].
	flag ifTrue: [ x:= self class noTracePoint] ifFalse: [x:= 0].
	dir = #front ifTrue: [^(x+0.5)].
	dir = #back ifTrue: [^(x+1)].
	dir = #right ifTrue: [^(x+0.75)].
	^(x+0.25)! !

!ContinuousCell class methodsFor: 'defaults'!

defaultMarkValue
	^1!

defaultTransferFn
	^#dummy:!

!ContinuousCell class methodsFor: 'constants'!

decoderBase
	^5!

noTracePoint
	^5!

saturationPoint
	^1000! !
```

SituatedCompositeAgent subclass: #CellularRobot

```
	instanceVariableNames: ''

!CellularRobot methodsFor: 'accessing'!

cells
	^self components! !

!CellularRobot methodsFor: 'actions'!

elapsed: dt
	self execute!
```

```
execute
	|done|
	self executeReflexes.
	done := false.
	self cells do: [:cell| cell resetInternalTime; unmark].
	[done] whileFalse:
		[done := true.
		self cells do: [:cell|
			cell decrementInternalTime.
			((cell marked) not) & (cell internalTime <= 0)
				ifTrue: [cell mark; execute.
					done := false]]]!

executeReflexes
	"By default, do nothing. To implement in subclasses."!
```

CellularRobot subclass: #Pedro

```
	instanceVariableNames: 'center vars timer heading rnd releaser '

!Pedro methodsFor: 'initialize-release'!

initialize
	super initialize.
	timer := 1.
	rnd := Random new.
	releaser := nil!

initializeFrom: aSpec
	|bumper1 bumper2 bumper3 bumper4 trace1 trace2 trace3 trace4 a1 a2 a3 a4 b1 b2 b3 b4 c1 c2 c3 c4
decoder motor spec mtu1 mtu2 mtu3 mtu4 dtu1 dtu2 dtu3 dtu4 p1 p2 p3 p4 m1 m2 m3 m4 cm1 cm2 cm3 cm4 dtc
pa|
	super initializeFrom: aSpec.

	"Initialize the cells."

	"First, initialize the navigation system."

	"4 obstacle sensors, 4 trace sensors, 4 A-cells, 4 B-cells, 4 C-cells,
		1 decoder, 1 actuator."

	"4 obstacle sensors."

	spec := ContinuousCell defaultSpec.
	spec at: #transferFn put: #obstacleFront:.
	bumper1 := ContinuousCell newFrom: spec in: self.

	spec := ContinuousCell defaultSpec.
	spec at: #transferFn put: #obstacleRight:.
	bumper2 := ContinuousCell newFrom: spec in: self.

	spec := ContinuousCell defaultSpec.
	spec at: #transferFn put: #obstacleBack:.
	bumper3 := ContinuousCell newFrom: spec in: self.

	spec := ContinuousCell defaultSpec.
	spec at: #transferFn put: #obstacleLeft:.
	bumper4 := ContinuousCell newFrom: spec in: self.

	"4 trace sensors."

	spec := ContinuousCell defaultSpec.
	spec at: #transferFn put: #traceFront:.
```

```
trace1 := ContinuousCell newFrom: spec in: self.

spec := ContinuousCell defaultSpec.
spec at: #transferFn put: #traceRight:.
trace2 := ContinuousCell newFrom: spec in: self.

spec := ContinuousCell defaultSpec.
spec at: #transferFn put: #traceBack:.
trace3 := ContinuousCell newFrom: spec in: self.

spec := ContinuousCell defaultSpec.
spec at: #transferFn put: #traceLeft:.
trace4 := ContinuousCell newFrom: spec in: self.

"Decoder + actuator"

spec := ContinuousCell defaultSpec.
spec at: #transferFn put: #decode:; at: #timer put: 8.
decoder := ContinuousCell newFrom: spec in: self.

spec := ContinuousCell defaultSpec.
spec at: #transferFn put: #motor:; at: #timer put: 9.
motor := ContinuousCell newFrom: spec in: self.

"4 A-cells."

spec := ContinuousCell defaultSpec.
spec at: #transferFn put: #aFn:; at: #timer put: 6.
a1 := ContinuousCell newFrom: spec in: self.
a2 := ContinuousCell newFrom: spec in: self.
a3 := ContinuousCell newFrom: spec in: self.
a4 := ContinuousCell newFrom: spec in: self.

"4 B-cells."

spec := ContinuousCell defaultSpec.
spec at: #transferFn put: #bFn:; at: #timer put: 6.
b1 := ContinuousCell newFrom: spec in: self.
b2 := ContinuousCell newFrom: spec in: self.
b3 := ContinuousCell newFrom: spec in: self.
b4 := ContinuousCell newFrom: spec in: self.

"4 C-cells."

spec := ContinuousCell defaultSpec.
spec at: #transferFn put: #cFn:; at: #timer put: 7.
c1 := ContinuousCell newFrom: spec in: self. c1 variablesAt: 1 put: 0.5.
c2 := ContinuousCell newFrom: spec in: self. c2 variablesAt: 1 put: 0.75.
c3 := ContinuousCell newFrom: spec in: self. c3 variablesAt: 1 put: 1.
c4 := ContinuousCell newFrom: spec in: self. c4 variablesAt: 1 put: 0.25.

"Next, initialize the tasks."

"Per task: 1 move-to-unit cell, 1 dirToUnit cell, 1 prio cell, 1 compute-motivation cell,
                    1 compose-motivation cell"

spec := ContinuousCell defaultSpec.
spec at: #transferFn put: #moveToUnit:; at: #timer put: 2.
mtu1 := ContinuousCell newFrom: spec in: self. mtu1 variablesAt: 1 put: 2.
mtu2 := ContinuousCell newFrom: spec in: self. mtu2 variablesAt: 1 put: 2.
mtu3 := ContinuousCell newFrom: spec in: self. mtu3 variablesAt: 1 put: 2.
mtu4 := ContinuousCell newFrom: spec in: self. mtu4 variablesAt: 1 put: 2.

spec := ContinuousCell defaultSpec.
spec at: #transferFn put: #dirToUnit:.
dtu1 := ContinuousCell newFrom: spec in: self. dtu1 variablesAt: 1 put: #'ServiceUnit-1'.
```

```
dtu2 := ContinuousCell newFrom: spec in: self. dtu2 variablesAt: 1 put: #'ServiceUnit-2'.
dtu3 := ContinuousCell newFrom: spec in: self. dtu3 variablesAt: 1 put: #'ServiceUnit-3'.
dtu4 := ContinuousCell newFrom: spec in: self. dtu4 variablesAt: 1 put: #'ServiceUnit-4'.

spec := ContinuousCell defaultSpec.
spec at: #transferFn put: #prio:; at: #timer put: 3.
p1 := ContinuousCell newFrom: spec in: self.
p2 := ContinuousCell newFrom: spec in: self.
p3 := ContinuousCell newFrom: spec in: self.
p4 := ContinuousCell newFrom: spec in: self.

spec := ContinuousCell defaultSpec.
spec at: #transferFn put: #computeMotivation:; at: #timer put: 2.
m1 := ContinuousCell newFrom: spec in: self. m1 variablesAt: 1 put: #'ServiceUnit-1';
        variablesAt: 2 put: 41.
m2 := ContinuousCell newFrom: spec in: self. m2 variablesAt: 1 put: #'ServiceUnit-2';
        variablesAt: 2 put: 41.
m3 := ContinuousCell newFrom: spec in: self. m3 variablesAt: 1 put: #'ServiceUnit-3';
        variablesAt: 2 put: 41.
m4 := ContinuousCell newFrom: spec in: self. m4 variablesAt: 1 put: #'ServiceUnit-4';
        variablesAt: 2 put: 41.

spec := ContinuousCell defaultSpec.
spec at: #transferFn put: #composeMotivation:; at: #timer put: 4.
cm1 := ContinuousCell newFrom: spec in: self. cm1 variablesAt: 1 put: #Substance1.
cm2 := ContinuousCell newFrom: spec in: self. cm2 variablesAt: 1 put: #Substance2.
cm3 := ContinuousCell newFrom: spec in: self. cm3 variablesAt: 1 put: #Substance3.
cm4 := ContinuousCell newFrom: spec in: self. cm4 variablesAt: 1 put: #Substance4.

"Global: 1 dir-to-center cell, 1 prio-all cell."

spec := ContinuousCell defaultSpec.
spec at: #transferFn put: #dirToCenter:; at: #timer put: 1.
dtc := ContinuousCell newFrom: spec in: self. dtc variablesAt: 1 put: 1.
spec := ContinuousCell defaultSpec.
spec at: #transferFn put: #prioAll:; at: #timer put: 5.
pa := ContinuousCell newFrom: spec in: self.
pa variablesAt: 1 put: 3; variablesAt: 2 put: 10.

"Now, initialize the connections."

bumper1 connectTo: a1. pa connectTo: a1.
bumper2 connectTo: a2. pa connectTo: a2.
bumper3 connectTo: a3. pa connectTo: a3.
bumper4 connectTo: a4. pa connectTo: a4.

trace1 connectTo: b1. pa connectTo: b1.
trace2 connectTo: b2. pa connectTo: b2.
trace3 connectTo: b3. pa connectTo: b3.
trace4 connectTo: b4. pa connectTo: b4.

a1 connectTo: c1. b1 connectTo: c1.
a2 connectTo: c2. b2 connectTo: c2.
a3 connectTo: c3. b3 connectTo: c3.
a4 connectTo: c4. b4 connectTo: c4.

c1 connectTo: decoder.
c2 connectTo: decoder.
c3 connectTo: decoder.
c4 connectTo: decoder.

decoder connectTo: motor.
dtu1 connectTo: mtu1.
mtu1 connectTo: p1.
dtc connectTo: p1.
m1 connectTo: cm1.
```

```
		p1 connectTo: cm1.
		cm1 connectTo: pa.
		dtu2 connectTo: mtu2.
		mtu2 connectTo: p2.
		dtc connectTo: p2.
		m2 connectTo: cm2.
		p2 connectTo: cm2.
		cm2 connectTo: pa.
		dtu3 connectTo: mtu3.
		mtu3 connectTo: p3.
		dtc connectTo: p3.
		m3 connectTo: cm3.
		p3 connectTo: cm3.
		cm3 connectTo: pa.
		dtu4 connectTo: mtu4.
		mtu4 connectTo: p4.
		dtc connectTo: p4.
		m4 connectTo: cm4.
		p4 connectTo: cm4.
		cm4 connectTo: pa.

		#(#'ServiceUnit-1' #'ServiceUnit-2' #'ServiceUnit-3' #'ServiceUnit-4')
			do: [:aKey| self vars at: aKey put: 0.5]!

!Pedro methodsFor: 'accessing'!

catalyze: unitName
		"If not catalyzed, catalyze it!!"
		|x|
		x := vars at: unitName.
		(x = 0.5) ifTrue: [x := 100].
		x := x - 1.
		vars at: unitName put: x!

discatalyze: unitName
		vars at: unitName put: 0.5!

resetReleaser
		releaser := nil!

resetTimer
		timer := 0!

!Pedro methodsFor: 'actions'!

executeReflexes
		|x flag col|
		"If at a unit, catalyze the unit and center = false.
		Discatalyze all the other units.
		If at center, center = true.
		Decay timer. Check reset timer (reset it, if at a unit
					whose demand dropped to 0, or releaser is nil.
		If timer is zero, reset releaser."
		self decrementTimer.
		flag := false.
		x := self place agents detect: [:agt| agt isKindOf: ServiceUnit] ifNone: nil.
		x notNil ifTrue:
			[self catalyze: x name; center: false.
			x decrementDemandBy: 1.
			flag := (x demand = 0)].
		col := self problem units asSet.
		col remove: x ifAbsent: [].
		col do: [:u| self discatalyze: u name].
		(self place point = (self problem middlePoint))		"Self at center"
			ifTrue: [self center: true. flag := true].
		flag | (releaser isNil) ifTrue: [self resetTimer].
```

```
(timer = 0) ifTrue: [self resetReleaser]! !
```

C.3 Agents cellulaires plastiques

Nous présentons finalement des extraits du code comportemental des agents cellulaires plastiques et en particulier des extraits provenant des classes suivantes :

La classe de la cellule plastique
PlasticCell (sous-classe de SimpleAgent)

La classe des agents cellulaires plastiques
CellularAgent (sous-classe de SituatedCompositeAgent)

L'agent cellulaire plastique qui correspond à l'application du chapitre 7 (navigation).
ObstacleAvoidanceAgent (sous-classe de CellularAgent)

La classe de la pulsion
Drive (sous-classe de SimpleAgent)

Classes de support
Buffer (sous-classe de Object)
CellMessage (sous-classe de Object)

SimpleAgent subclass: #PlasticCell

```
        instanceVariableNames: 'drives agent internalTime timer inputBuffers outputBuffers name type '

!PlasticCell methodsFor: 'initialize-release'!

initialize
        super initialize.
        self initializeName.
        drives := Set new.
        inputBuffers := Set new.
        outputBuffers := Set new!

!PlasticCell methodsFor: 'accessing'!

speed
        ^self drives inject: 0 into: [:sum :dr| sum + (dr speed)]!

!PlasticCell methodsFor: 'actions'!

act
        ^self act2!

act2
        "Without internal timer.
        Distribute speed itself and not its residual."
        |dict aSet listn listp xn xp xp1 uselessSet|

        "1.     Update drives' speeds."
        dict := Dictionary new.
        self drives do: [:d|   dict at: d put: d computeDSpeed].
        "The listp is the list of drives that want to increase their speed, while
        the listn is the list of drives that want to decrease their speed."
        listn := Set new. listp := Set new.
        dict associationsDo: [:assoc|
                assoc value <= 0 ifTrue: [listn add: assoc key] ifFalse: [listp add: assoc key]].
```

```
        dict associationsDo: [:assoc| assoc value: ((assoc value) + (assoc key speed))].
        xn := listn inject: 0 into: [:sum :y| sum + (dict at: y)].
        xp := listp inject: 0 into: [:sum :y| sum + (dict at: y)].
        (xp+xn) > 1 ifTrue: [
                "If the sum of the intended new speeds is higher than 1,
                then prune a little the 'ambitious' drives, i.e. those in the listp
                list."
                xp1 := 1-xn.
                listp do: [:d|          dict at: d put: ((dict at: d)*xp1/xp)]].
        listp do: [:d|          d speed: (dict at: d)].
        listn do: [:d|          d speed: (dict at: d)].

        "2.     *<Consume>* messages or *<Adapt>*."
        aSet := Set new.
        uselessSet := Set new.
        inputBuffers do: [:buff| uselessSet addAll: buff contents].
        self drives do: [:d|   (d catalysed) & (d stimulated)
                                        ifTrue: [aSet addAll: d act]
                                        ifFalse: [d adaptTo: uselessSet; messages: Set new]].
        ^aSet!

clearMessages
        self drives do: [:d|   d messages: Set new]!

clobberAdaptation
        self drives do: [:dr| dr clobberAdaptation]!

doDeceiver
        self drives do: [:dr| dr doDeceiver]!

normalizeSpeeds
        |total|
        total := self drives inject: 0 into: [:sum :dr| sum + (dr speed)].
        (total > 1) ifTrue:
                [self drives do: [:dr| dr speed: ((dr speed) / total)]]!

resetTime
        internalTime := timer!

updateInputParticipation
        self drives do: [:dr | dr updateInputParticipation]!

updateOutputParticipation
        self drives do: [:dr | dr updateOutputParticipation]! !

!PlasticCell methodsFor: 'connections'!

connectAsDefensorToInput: buffer
        inputBuffers add: buffer.
        buffer defensors add: self!

connectToInput: buffer
        inputBuffers add: buffer.
        buffer destinations add: self!

connectToOutput: buffer
        outputBuffers add: buffer.
        buffer inputs add: self!
```

SituatedCompositeAgent subclass: #CellularAgent

```
        instanceVariableNames: 'noise mesgList rnd buffers '

!CellularAgent methodsFor: 'initialize-release'!
```

```
flushMessages
	self buffers do: [:buffer |
		buffer contents release.
		buffer contents: Set new]!

initialize
	super initialize.
	rnd := Random new.
	buffers := SortedCollection new sortBlock: [:x :y| (x level) <= (y level)]!

reinitializeMessages
	self flushMessages.
	self cells do: [:cell|
		cell resetTime; clearMessages]!

!CellularAgent methodsFor: 'actions'!

distributeMessages: buff
	" Unused messages remain into the buffer for drive adaptation to take place."
	|mCol driveList|
	mCol := buff contents.
	driveList := Set new.
	buff destinations do: [:cell| driveList addAll: cell drives].
	buff defensors do: [:cell| driveList addAll: cell drives].
	driveList := SortedCollection		"Sort all cells' drives according to speed."
			withAll: driveList
			sortBlock: [:d1 :d2|
					"Add some noise in the evaluation of drives."
				((d1 speed) + (noise negated + (2 * (rnd next) * (noise)))) >
				((d2 speed) + (noise negated + (2 * (rnd next) * (noise))))].
	[(mCol isEmpty) | (driveList isEmpty)] whileFalse:
		[		|drive miniList flag|
		miniList := Set new.
		drive := driveList removeFirst.
		drive inputTemplates do: [:pattern|	|mesg|
			mesg := mCol detect: [:x| drive match: x template to: pattern] ifNone: [].
			mesg notNil ifTrue: [mCol remove: mesg. miniList add: mesg]].
		flag := true.
		drive catalysts do: [:pattern|	|mesg|
			mesg := mCol detect: [:x| drive match: x template to: pattern] ifNone: [].
			mesg notNil ifTrue: [miniList add: mesg copy] ifFalse: [flag := false]].
		drive catalysed: flag.
		miniList do: [:mesg| drive attach: mesg]].
	buff contents: mCol!

elapsed: dt
	"Since the buffers are sorted in increasing order of level,
	the messages are processed in the right order."
	self flushMessages.
	(self cells select: [:cell| cell drives asArray first inputTemplates isEmpty])
		do: [:cell| |out|
			out := cell act.
			cell outputBuffers do: [:b| b addAll: out copy]].
	self buffers do: [:buff|
		self distributeMessages: buff.
		buff defensors do: [:cell| cell act].
		buff inputs do: [:cell| cell updateOutputParticipation].
		buff destinations do: [:cell|	|out|
			cell updateInputParticipation.
			out := cell act.
			cell outputBuffers do: [:b| b addAll: out copy]]].
	self getAllDrives do: [:dr|
		(dr participation < 0) ifTrue: [dr adaptOToI].
		dr participation: 0]!
```

CellularAgent subclass: #ObstacleAvoidanceAgent

instanceVariableNames: 'heading goal openACells openBCells openCCells openADefensors openBDefensors openCDefensors updateFlag clashFlag speedFlag '

!ObstacleAvoidanceAgent methodsFor: 'initialize'!

initializeFrom2: spec withA: n B: m C: p

```
|r cell dSpec aBuffer bBuffer cBuffer decoderBuffer dirs sumsBuffer|

"4 obstacle sensors, 4 trace sensors, n A-cells, m B-cells, p C-cells,
        1 catalyst (direction), 1 decoder, 1 actuator."

super initializeFrom: spec.
r := Random new.
dirs := #(0.5 1 0.75 0.25).

"Buffer initialization."
aBuffer := (Buffer new initialize) name: #ABuffer; level: 1.
bBuffer := (Buffer new initialize) name: #BBuffer; level: 1.
cBuffer := (Buffer new initialize) name: #CBuffer; level: 2.
sumsBuffer := (Buffer new initialize) name: #SumsBuffer; level: 3.
decoderBuffer := (Buffer new initialize) name: #DecoderBuffer; level: 4.
buffers add: aBuffer.
buffers add: bBuffer.
buffers add: cBuffer.
buffers add: sumsBuffer.
buffers add: decoderBuffer.

"Now, initialize the cells and the connections."

"4 obstacle sensors."

dSpec := Drive defaultSpec.
dSpec     at: #minSpeed put: 1; at: #rRate put: 0; at: #transferFn put: #obstacleFront:;
                at: #minRate put: 0; at: #maxRate put: 0;
                at: #outputTemplates put: (Array with: 0.5).
cell := PlasticCell newFrom: (PlasticCell specWithDrives: (Array with: dSpec)) in: self.
cell connectToOutput: aBuffer; clobberAdaptation.

dSpec := Drive defaultSpec.
dSpec     at: #minSpeed put: 1; at: #rRate put: 0; at: #transferFn put: #obstacleBack:;
                at: #minRate put: 0; at: #maxRate put: 0;
                at: #outputTemplates put: (Array with: 1).
cell := PlasticCell newFrom: (PlasticCell specWithDrives: (Array with: dSpec)) in: self.
cell connectToOutput: aBuffer; clobberAdaptation.

dSpec := Drive defaultSpec.
dSpec     at: #minSpeed put: 1; at: #rRate put: 0; at: #transferFn put: #obstacleLeft:;
                at: #minRate put: 0; at: #maxRate put: 0;
                at: #outputTemplates put: (Array with: 0.25).
cell := PlasticCell newFrom: (PlasticCell specWithDrives: (Array with: dSpec)) in: self.
cell connectToOutput: aBuffer; clobberAdaptation.

dSpec := Drive defaultSpec.
dSpec     at: #minSpeed put: 1; at: #rRate put: 0; at: #transferFn put: #obstacleRight:;
                at: #minRate put: 0; at: #maxRate put: 0;
                at: #outputTemplates put: (Array with: 0.75).
cell := PlasticCell newFrom: (PlasticCell specWithDrives: (Array with: dSpec)) in: self.
cell connectToOutput: aBuffer; clobberAdaptation.

"4 trace sensors."

dSpec := Drive defaultSpec.
```

```
dSpec     at: #minSpeed put: 1; at: #rRate put: 0; at: #transferFn put: #traceFront:;
                    at: #minRate put: 0; at: #maxRate put: 0;
                    at: #outputTemplates put: (Array with: 0.5).
cell := PlasticCell newFrom: (PlasticCell specWithDrives: (Array with: dSpec)) in: self.
cell connectToOutput: bBuffer; clobberAdaptation.

dSpec := Drive defaultSpec.
dSpec     at: #minSpeed put: 1; at: #rRate put: 0; at: #transferFn put: #traceRight:;
                    at: #minRate put: 0; at: #maxRate put: 0;
                    at: #outputTemplates put: (Array with: 0.75).
cell := PlasticCell newFrom: (PlasticCell specWithDrives: (Array with: dSpec)) in: self.
cell connectToOutput: bBuffer; clobberAdaptation.

dSpec := Drive defaultSpec.
dSpec     at: #minSpeed put: 1; at: #rRate put: 0; at: #transferFn put: #traceLeft:;
                    at: #minRate put: 0; at: #maxRate put: 0;
                    at: #outputTemplates put: (Array with: 0.25).
cell := PlasticCell newFrom: (PlasticCell specWithDrives: (Array with: dSpec)) in: self.
cell connectToOutput: bBuffer; clobberAdaptation.

dSpec := Drive defaultSpec.
dSpec     at: #minSpeed put: 1; at: #rRate put: 0; at: #transferFn put: #traceBack:;
                    at: #minRate put: 0; at: #maxRate put: 0;
                    at: #outputTemplates put: (Array with: 1).
cell := PlasticCell newFrom: (PlasticCell specWithDrives: (Array with: dSpec)) in: self.
cell connectToOutput:bBuffer; clobberAdaptation.

"Direction-toward goal"

dSpec := Drive defaultSpec.
dSpec     at: #minSpeed put: 1; at: #rRate put: 0; at: #transferFn put: #generateDir:;
                    at: #minRate put: 0; at: #maxRate put: 0;
                    at: #outputTemplates put: #(#direction).
cell := PlasticCell newFrom: (PlasticCell specWithDrives: (Array with: dSpec)) in: self.
cell connectToOutput: aBuffer; connectToOutput: bBuffer; clobberAdaptation.

"Decoder + actuator"

dSpec := Drive defaultSpec.
dSpec     at: #minSpeed put: 1; at: #rRate put: 0; at: #transferFn put: #decode:;
                    at: #minRate put: 0; at: #maxRate put: 0;
                    at: #inputTemplates put: (Array with: 0.5 with: 1 with: 0.75 with: 0.25);
                    at: #variables put: ((Array new: 8 withAll: 0) at: 1 put: ((r next * 0.2) + 0.3);
                                        at: 8 put: Drive defaultMatchingThreshold; yourself);
                    at: #outputTemplates put: #(#motor).
cell := PlasticCell newFrom: (PlasticCell specWithDrives: (Array with: dSpec)) in: self.

cell connectToInput: sumsBuffer; connectToOutput: decoderBuffer; clobberAdaptation.

dSpec := Drive defaultSpec.
dSpec     at: #minSpeed put: 1; at: #rRate put: 0; at: #transferFn put: #motor:;
                    at: #minRate put: 0; at: #maxRate put: 0;
                    at: #inputTemplates put: #(#motor).
cell := PlasticCell
                    newFrom: (PlasticCell specWithDrives: (Array with: dSpec) timer: 5)
                    in: self.
cell connectToInput: decoderBuffer; clobberAdaptation.

"n A-cells."

9 to: (8+n) do: [:i|   |k dSet newDirs|
          newDirs := GenericUtility shuffleArray: dirs.
          k := ((r next) * 3) truncated + 1.
          dSet := Set new.
          1 to: k do: [:j|         |inDir outDir|
                    inDir := newDirs at: j.
```

```
                    outDir := newDirs at: j.
                    dSpec := Drive defaultSpec.
                    dSpec at: #minSpeed put: (r next) * 0.2; at: #rRate put: ((r next) * 0.4) + 0.1;
                            at: #maxSpeed put: ((r next) * 0.2) + 0.8;
                            at: #minRate put: ((r next) * 0.05) + 0.05;
                            at: #maxRate put: ((r next) * 0.05) + 0.15;
                            at: #dFactor put: (r next) * 0.05;
                            at: #transferFn put: #aFn:; at: #catalysts put: #(#direction);
                            at: #inputTemplates put: (Array with: inDir);
                            at: #outputTemplates put: (Array with: outDir);
                            at: #variables put: ((Array new: 8 withAll: 0) at: 1 put: 5;
                                    at: 8 put: Drive defaultMatchingThreshold; yourself).
                    dSet add: dSpec].
            cell := PlasticCell newFrom: (PlasticCell specWithDrives: dSet asArray) in: self.
            cell type: #A.
            cell connectToInput: aBuffer; connectToOutput: cBuffer].

"m B-cells."

(9+n) to: (8+n+m) do: [:i      |k dSet newDirs|
            newDirs := GenericUtility shuffleArray: dirs.
            k := ((r next) * 3) truncated + 1.
            dSet := Set new.
            1 to: k do: [:j         |inDir outDir|
                    inDir := newDirs at: j.
                    outDir := newDirs at: j.
                    dSpec := Drive defaultSpec.
                    dSpec at: #minSpeed put: (r next) * 0.2; at: #rRate put: ((r next) * 0.4) + 0.1;
                            at: #maxSpeed put: ((r next) * 0.2) + 0.8;
                            at: #minRate put: ((r next) * 0.05) + 0.05;
                            at: #maxRate put: ((r next) * 0.05) + 0.15;
                            at: #dFactor put: (r next) * 0.05;
                            at: #transferFn put: #bFn:; at: #catalysts put: #(#direction);
                            at: #inputTemplates put: (Array with: inDir);
                            at: #outputTemplates put: (Array with: outDir);
                            at: #variables put: ((Array new: 8 withAll: 0) at: 1 put: 1;
                                    at: 8 put: Drive defaultMatchingThreshold; yourself).
                    dSet add: dSpec].
            cell := PlasticCell newFrom: (PlasticCell specWithDrives: dSet asArray) in: self.
            cell type: #B.
            cell connectToInput: bBuffer; connectToOutput: cBuffer].

"p C-cells."

(9+n+m) to: (8+n+m+p) do: [:i           |k dSet newDirs|
            newDirs := GenericUtility shuffleArray: dirs.
            k := ((r next) * 3) truncated + 1.
            dSet := Set new.
            1 to: k do: [:j         |inDir outDir|
                    inDir := newDirs at: j.
                    outDir := newDirs at: j.
                    dSpec := Drive defaultSpec.
                    dSpec at: #minSpeed put: (r next) * 0.2; at: #rRate put: ((r next) * 0.4) + 0.1;
                            at: #maxSpeed put: ((r next) * 0.2) + 0.8;
                            at: #minRate put: ((r next) * 0.05) + 0.05;
                            at: #maxRate put: ((r next) * 0.05) + 0.15;
                            at: #dFactor put: (r next) * 0.05;
                            at: #transferFn put: #cFn:;
                            at: #variables put: ((Array new: 8 withAll: 0)
                                    at: 8 put: Drive defaultMatchingThreshold; yourself);
                            at: #inputTemplates put: (Array with: inDir with: inDir);
                            at: #outputTemplates put: (Array with: outDir).
                    dSet add: dSpec].
            cell := PlasticCell newFrom: (PlasticCell specWithDrives: dSet asArray) in: self.
            cell type: #C.
            cell connectToInput: cBuffer; connectToOutput: sumsBuffer]!
```

```
initializeFrom4: spec withA: n B: m C: p
	|r cell dSpec dirs|

	"The usual plus 4 defensor cells in each one of the A, B and C buffers."
	self initializeFrom2: spec withA: n B: m C: p.
	r := Random new.

	"Directions complementary to the legal ones (0.5, 0.75, 0.25, 1)."
	dirs := #(0.125 0.375 0.625 0.875).

	"4 defensor cells for each buffer with all 4 forbidden directions."

	(Array with: self getABuffer with: self getBBuffer with: self getCBuffer)

	do: [:buff|			|s|
		s := 4.
		1 to: s do: [:i|		|k dSet newDirs|
			newDirs := GenericUtility shuffleArray: dirs.
			k := 4.
			dSet := Set new.
			1 to: k do: [:j|		|inDir|
				inDir := newDirs at: j.
				dSpec := Drive defaultSpec.
				dSpec at: #minSpeed put: 1; at: #rRate put: 0; at: #maxSpeed put: 1;
					at: #minRate put: 0; at: #maxRate put: 0;
					at: #dFactor put: 0; at: #transferFn put: #swallow:;
					at: #variables put: ((Array new: 8 withAll: 0)
						at: 8 put: Drive defaultMatchingThreshold; yourself);
					at: #inputTemplates put: (Array with: inDir);
					at: #outputTemplates put: Array new.
				dSet add: dSpec].
			cell := PlasticCell newFrom: (PlasticCell specWithDrives: dSet asArray) in: self.
			cell clobberAdaptation; connectAsDefensorToInput: buff]]!

initializeFrom: spec
	self initializeFrom4: spec withA: 4 B: 4 C: 3!

!ObstacleAvoidanceAgent methodsFor: 'accessing'!

done
	^(self place point = goal)!

getABuffer
	^self buffers detect: [:buffer | buffer name = #ABuffer]!

getBBuffer
	^self buffers detect: [:buffer | buffer name = #BBuffer]!

getCBuffer
	^self buffers detect: [:buffer | buffer name = #CBuffer]!

getSumsBuffer
	^self buffers detect: [:buffer | buffer name = #SumsBuffer]!

!ObstacleAvoidanceAgent methodsFor: 'actions'!

computeSpeed
	^3
		+ (self collectADrives inject: 2 into: [:y :dr| dr speed min: y])
		+ (self collectBDrives inject: 2 into: [:y :dr| dr speed min: y])
		+ (self collectCDrives inject: 2 into: [:y :dr| dr speed min: y])!

elapsed: dt
	"Those three flags are used for pure observational purposes."
	updateFlag := false.
```

```
        clashFlag := false.
        speedFlag := false.

        super elapsed: dt.
        updateFlag ifFalse:
                [self updateDirsWith: Set new].
        speedFlag ifFalse: [self changed: #totalSpeed with: self computeSpeed].
        clashFlag ifFalse: [self changed: #clash with: 0]!

updateDirsWith: aSet
        #(#front #back #left #right) do: [:dir|     |x|
                (aSet includes: dir) ifTrue: [x := 1] ifFalse: [x := 0.5].
                self changed: dir with: x]! !
```

Object subclass: #Buffer

```
        instanceVariableNames: 'contents inputs destinations name level defensors perturbation rnd virus '

!Buffer methodsFor: 'initialize-release'!

initialize
        self contents: Set new.
        destinations := Set new.
        inputs := Set new.
        defensors := Set new.
        perturbation := self class defaultPerturbation.
        rnd := Random new.
        virus := false.
        level := 1!

!Buffer methodsFor: 'accessing'!

add: mesg
        (rnd next < perturbation)
                ifTrue: [contents add: (self mutateMessage: mesg)]
                ifFalse: [contents add: mesg]!

addAll: mesgList
        mesgList do: [:x| self add: x]!

mutateMessage: mesg
        virus
                ifTrue: [^CellMessage template: rnd next data: rnd next time: mesg time]
                ifFalse: [              |dirs templ|
                        dirs := #(0.125 0.375 0.625 0.875).
                        templ := (dirs at: (((rnd next) * 4) truncated + 1)) +
                                        (((rnd next) * 0.12) - 0.06).
                        ^CellMessage template: templ data: rnd next time: mesg time]!

!Buffer class methodsFor: 'defaults'!

defaultPerturbation
        ^0! !
```

SimpleAgent subclass: #Drive

```
        instanceVariableNames: 'inputTemplates outputTemplates catalysts transferFn speed maxSpeed
minSpeed rate templateGenerationFn messages catalysed cell variables name dFactor dThreshold minRate
maxRate rRate participation '

!Drive methodsFor: 'initialize-release'!

initialize
```

```
		super initialize.
		self initializeName.
		participation := 1.
		variables at: 8 put: self class defaultMatchingThreshold!

initializeFrom: spec
		super initializeFrom: spec.
		variables := (spec at: #variables) copy.
		self speed: self minSpeed.
		self rate: self minRate!

!Drive methodsFor: 'accessing'!

attach: mesg
		messages add: mesg!

inputDirection
		inputTemplates isEmpty ifTrue: [^2].
		^inputTemplates asArray first!

isDeceiver
		^transferFn = #deceive:!

noise
		^self cell agent rnd next * 0.2!

outputDirection
		outputTemplates isEmpty ifTrue: [^2].
		^outputTemplates asArray first!

stimulated
		inputTemplates isEmpty ifTrue: [^true].
		^((messages size) - (catalysts size) > 0)!

!Drive methodsFor: 'actions'!

act
		^self actAdapt!

actAdapt
		|out col t|
		col := self messages.
		t := self speed.
		col isEmpty ifFalse:
			[t := t + (col inject: 100 into: [:x :y| x min: (y time)]).
			self adaptTo: col].
		out := self perform: self transferFn withArguments: (Array with: col).
		messages := Set new.
		out isNil
			ifTrue: [^Set new]
			ifFalse: [		|tSet|
				tSet := outputTemplates.
				tSet isEmpty
					ifTrue: [tSet := self perform: templateGenerationFn with: out].
				^tSet collect: [:templ|		CellMessage template: templ data: out time: t]]!

actNoAdapt
		|out col t|
		col := self messages.
		t := self speed.
		col isEmpty ifFalse: [t := t + (col inject: 100 into: [:x :y| x min: (y time)])].
		out := self perform: self transferFn withArguments: (Array with: col).
		messages := Set new.
		out isNil
			ifTrue: [^Set new]
			ifFalse: [		|tSet|
```

```
                tSet := outputTemplates.
                tSet isEmpty
                        ifTrue: [tSet := self perform: templateGenerationFn with: out].
                ^tSet collect: [:templ|         CellMessage template: templ data: out time: t]]!

adaptOToI
        |templ|
        (dFactor = 0) ifTrue: [^self].
        inputTemplates isEmpty ifFalse: [^self].
        templ := inputTemplates asArray first.
        self outputTemplates: (self outputTemplates asArray collect: [:t|
                        t isSymbol
                                ifTrue: [t]
                                ifFalse: [t+ ((templ - t)*dFactor)]])!

adaptTo: mesgList
        |mesg affinity|
        mesg := nil.
        affinity := -1.
        (dFactor = 0) ifTrue: [^self].
        mesgList do: [:m|   |x|
                x := self computeAffinityTo: m template.
                (x > affinity) ifTrue: [mesg := m. affinity := x]].
        mesg isNil ifFalse:
                [self inputTemplates: (self inputTemplates asArray collect: [:t|
                        t isSymbol
                                ifTrue: [t]
                                ifFalse: [t+ ((mesg template - t)*dFactor)]]).
                self outputTemplates: (self outputTemplates asArray collect: [:t|
                        t isSymbol
                                ifTrue: [t]
                                ifFalse: [t+ ((mesg template - t)*dFactor)]])]!

clobberAdaptation
        dFactor := 0!

computeDSpeed
        ^self computeDSpeedOK!

computeDSpeedOK
        (self catalysed) & (self stimulated)
                ifTrue: [^(rate * (maxSpeed - speed))]
                ifFalse: [^(rate * (minSpeed - speed))]!

computeInputParticipation
        (self catalysed) & (self stimulated)
                ifTrue: [^1]
                ifFalse: [^0]!

computeOutputParticipation
        |s col x|
        col := self outputTemplates.
        col isEmpty ifTrue: [^1].
        s := self cell outputBuffers size.
        x := self cell outputBuffers inject: 0 into: [:sum :buff|       |q cont|
                cont := buff contents copy.
                q := 0.
                col do: [:templ|    |y|
                        y := cont detect: [:m| m template = templ] ifNone: nil.
                        y isNil ifFalse: [q := q + 1. cont remove: y]].
                q := q / (col size).
                sum + q].
        (x=s) ifTrue: [^1].
        ^0!

computeSpeed
```

```
		(self catalysed) & (self stimulated)
			ifTrue: [^speed + (rate * (maxSpeed - speed))]
			ifFalse: [^speed + (rate * (minSpeed - speed))]!

doDeceiver
		|r|
		r := Random new.
		self
			outputTemplates: (self outputTemplates asArray collect: [:t|		r next]);
			transferFn: #deceive:;
			minSpeed: ((r next) * 0.5) + 0.5;
			maxSpeed: 1;
			variablesAt: 3 put: r next!

updateInputParticipation
		participation := participation - (self computeInputParticipation)!

updateOutputParticipation
		participation := participation + (self computeOutputParticipation)! !

!Drive methodsFor: 'transfer functions'!

aFn: list
		|col x|
		col := list asArray.
		x := ((col first data) - ((col at: 2) data)) abs.
		(x > 0.5) ifTrue: [x := x - 0.5].
		(x = 0.5) ifTrue: [x := 0.5 + (self class noTracePoint)].
		(x < (0.8 * (self class saturationPoint))) ifTrue: [^x].
		^self class saturationPoint!

avg: list
		list isEmpty ifTrue: [^0].
		^(list inject: 0 into: [:sum :x| sum + (x data)]) / (list size)!

bFn: list
		|col x a z|
		col := list asArray.
		x := ((col first data) - ((col at: 2) data)) abs.
		a := self class noTracePoint.
		(x>(a-1)) & (x<a) ifTrue: [x := x + 1].
		(x>(a+0.5))
			ifTrue: [x := x - 0.5]
			ifFalse:
				[(x = 0.75) ifTrue: [x := 0.25].
				(x = 0.5) ifTrue: [x := a + 0.5]].
		z := x.
		^z!

cFn: list
		(list size < 2) ifTrue: [^nil].
		^(list inject: 0 into: [:sum :x| sum + (x data)])!

deceive: list
		^self variablesAt: 3!

decode: list
		|x dir list1 t done s|
		list isEmpty ifTrue: [^nil].
		list1 := (SortedCollection withAll: list sortBlock: [:z :y| (z time) > (y time)]) asArray.
		t := (list1 first time) + (self variablesAt: 1).
		x := 1.
		done := false.
		s := list1 size.
		[done | (x > s)] whileFalse:
			[((list1 at: x) time > t) ifTrue: [done := true] ifFalse: [x := x+1]].
```

```
        list1 := list1 copyFrom: 1 to: (x min: s).
        x := (self class saturationPoint) * 0.8.
        (GenericUtility shuffleArray: list1 asArray) do:
                [:elm| ((elm data) < x) ifTrue: [x := elm data. dir := elm template]].
        list1 := list1 collect: [:elm| self findSymbolicDirectionFrom: elm template].
        (self cell agent) updateFlag: true; updateDirsWith: list1.
        ^self findSymbolicDirectionFrom: dir!

generateDir: list
        |aSymbol agt aPoint dir|
        agt := self cell agent.
        aPoint := (agt goal) - (agt place point).
        (aPoint = (0@0)) ifTrue: [^nil].
        ((aPoint x abs) < (aPoint y abs))
        & (agt rnd next < 0.9)
                ifTrue: [dir := (0@(aPoint y sign))]
                ifFalse:
                        [dir := ((aPoint x sign)@0).
                        dir = (0@0) ifTrue:
                                [(agt rnd next < 0.5)
                                        ifTrue: [dir := (-1)@0]
                                        ifFalse: [dir := 1@0]]].
        aSymbol := agt computeRelativeDirectionFrom: dir.
        aSymbol = #front ifTrue: [^0.5].
        aSymbol = #back ifTrue: [^1].
        aSymbol = #right ifTrue: [^0.75].
        aSymbol = #left ifTrue: [^0.75].
        ^nil!

max: list
        |x|
        list isEmpty ifTrue: [^0].
        x := 0.
        list do: [:elm| ((elm data) > x) ifTrue: [x := elm data]].
        ^x!

min: list
        |x|
        list isEmpty ifTrue: [^1000].
        x := 1000.
        list do: [:elm| ((elm data) < x) ifTrue: [x := elm data]].
        ^x!

motor: list
        |dir agt flag aPoint|
        agt := self cell agent.
        list isEmpty ifTrue:
                [agt speedFlag: true; changed: #totalSpeed with: agt computeSpeed;
                clashFlag: true; changed: #clash with: 0. ^nil].
        dir := list asArray first data.
        agt speedFlag: true; changed: #totalSpeed with: ((list asArray first time) + (self speed)).
        aPoint := agt place point + (agt computeAbsoluteDirectionFrom: dir).
        flag := agt problem isValidPlaceAt: aPoint.
        flag ifFalse: [agt clashFlag: true; changed: #clash with: 1. ^nil].
        agt clashFlag: true; changed: #clash with: 0.
        agt
                turnTo: dir;
                advanceBy: (agt computeAbsoluteDirectionFrom: #front);
                changed: #move!

obstacleBack: list
        ^self bumper: #back!

obstacleFront: list
        ^self bumper: #front!
```

```
obstacleLeft: list
	^self bumper: #left!

obstacleRight: list
	^self bumper: #right!

sum: list
	list isEmpty ifTrue: [^0].
	^(list inject: 0 into: [:sum :x| sum + (x data)])!

swallow: list
	"Transfer function for defensor cells' drives: 'swallows' the recognized messages."
	^nil!

traceBack2: list
	^self trace: #back!

traceBack: list
	^self traceBack2: list!

traceFront: list
	^self trace: #front!

traceLeft: list
	^self trace: #left!

traceRight: list
	^self trace: #right! !

!Drive methodsFor: 'template generation function'!

defaultGenerateTemplate: x
	^Set new! !

!Drive methodsFor: 'private'!

bumper: dir
	|env absoluteDirection flag agt aPoint|
	agt := self cell agent.
	env := agt place world.
	absoluteDirection := agt computeAbsoluteDirectionFrom: dir.
	aPoint := agt place point + absoluteDirection.
	(env validPoint: aPoint) ifTrue: [flag := (env at: aPoint) hasObstacle] ifFalse: [flag := true].
	flag ifTrue: [^self class saturationPoint].
	dir = #front ifTrue: [^0.5].
	dir = #back ifTrue: [^1].
	dir = #right ifTrue: [^0.75].
	^0.25!

trace: dir
	|env absoluteDirection flag agt aPoint x|
	agt := self cell agent.
	env := agt place world.
	absoluteDirection := agt computeAbsoluteDirectionFrom: dir.
	aPoint := agt place point + absoluteDirection.
	flag := true.
	(env validPoint: aPoint) ifTrue:
		[flag := ((env at: aPoint) selectObjectsOfCaste: #Sample) isEmpty].
	(agt rnd next < 0.1) ifTrue: [flag := flag not].
	flag ifTrue: [ x:= self class noTracePoint] ifFalse: [x:= 0].
	dir = #front ifTrue: [^(x+0.5)].
	dir = #back ifTrue: [^(x+1)].
	dir = #right ifTrue: [^(x+0.75)].
	^(x+0.25)! !

!Drive methodsFor: 'utilities'!
```

```
computeAffinityBetween: x and: y
        |s|
        (x isSymbol) | (y isSymbol) ifTrue: [^(-1)].
        s := (x-y) abs.
        (s <= dThreshold) ifTrue: [^s].
        ^(-1)!

computeAffinityTo: x
        (self inputTemplates isEmpty) | (x isSymbol) ifTrue: [^(-1)].
        ^((self inputTemplates inject: 0 into: [:t :tot|
                        tot + (self computeAffinityBetween: x and: t)])
                / (self inputTemplates size))!

decode: x base: b
        |dict val|
        dict := Dictionary new.
        val := (x/b) rounded.
        dict at: #first put: val; at: #second put: (x-(b*val)).
        ^dict!

encode: x with: y base: b
        ^(x*b) + y!

equal: x with: y threshold: z
        ^(x - y) abs <= z!

greater: x than: y threshold: z
        ^(x - y) > z!

match: x to: y
        (x isSymbol) | (y isSymbol)
                ifTrue: [^(x=y)].
        ^self equal: x with: y threshold: (self variablesAt: 8)! !

!Drive class methodsFor: 'constants'!

defaultMatchingThreshold
        ^0.05!

noTracePoint
        ^5!

saturationPoint
        ^1000! !
```

Object subclass: #CellMessage

```
        instanceVariableNames: 'template data time '
CellMessage comment:
'Important:
The "time" instance variable corresponds to an actual speed!! I didn"t have time to change this.
As a consequence, the drives" actions should compute in terms of minimum time (which corresponds
to minimum speed), to account for the last message arrival time.'!

!CellMessage class methodsFor: 'instance creation'!

template: x data: y
        ^self template: x data: y time: 0!

template: x data: y time: t
        |inst|
        inst := self new.
        inst template: x; data: y; time: t.
```

^inst! !

Glossaire

Les différents termes sont définis dans les paragraphes dont les numéros sont donnés entre parenthèses. Parfois, certains termes sont utilisés avec leur signification habituelle sans nuance scientifique, par exemple le terme structure.

Action — Ce qui fait changer une entité (§3.4.2, selon Jacopin 1993). Dans la discussion sur la cellularité (chapitre 3), chaque structure (cellule ou ensemble connecté de cellules) correspond à une action, par conséquent les deux termes sont utilisés sans discrimination. L'action est située, donc métabolique (§3.4.2), c'est-à-dire l'action est une modification interne à l'agent ayant un impact sur le monde de l'agent, ce n'est pas une action à distance à partir des représentations non matériellement fondées.

Adaptation — L'harmonisation d'un comportement aux exigences d'un environnement imprévisible (§1.2). Au niveau de la structure, elle correspond à la modification de certains paramètres (voir, par exemple, les modèles d'adaptation du chapitre 4).

Agent — Une entité réelle ou abstraite qui est capable d'agir sur elle-même et sur son environnement, qui dispose d'une représentation partielle de cet environnement, qui, dans un univers multi-agents, peut communiquer avec d'autres agents, et dont le comportement est la conséquence de ses observations, de sa connaissance et des interactions avec les autres agents (§1.1, selon Ferber 1989*a*). Dans notre contexte de conception, un agent (artificiel) est un artefact créé par l'homme et destiné à accomplir une mission, résoudre un problème ou réaliser une tâche (§1.1).

Agrégat — Un ensemble de cellules qui accomplissent collectivement une "fonction" particulière de perception, de commande ou de traitement intermédiaire (§3.2.3).

Apprentissage — L'acquisition de nouvelles structures d'interaction avec le monde, autrement dit l'apparition de nouveaux comportements (§1.2).

Arbitrage — Si plusieurs "comportements" deviennent actifs en même temps, comment font-ils pour s'arbitrer entre eux ? (§3.4.1).

Architecture — L'ensemble de relations et de principes auxquels doivent se conformer les systèmes de contrôle des agents constituant les solutions à des problèmes spécifiques (§3.1.1, cf. "organisation"). L'architecture constitue donc le langage de programmation des systèmes de contrôle.

Autonomie — Loi propre. Elle représente la génération, l'affirmation de sa propre identité, la régulation interne, la définition de l'intérieur (§1.3, selon Varela 1979).

Autorégulation — Régulation d'un agent par lui-même : réglage homéostatique d'un ou de plusieurs paramètres organisationnels, c'est-à-dire régulation continue du paramètre entre deux limites, une limite inférieure et une limite supérieure (§1.2, §1.3.3). L'autorégulation peut avoir lieu dans l'espace sensori-pulsion de l'agent ou dans l'espace sensori-moteur.

Auto-organisation — Fait d'être organisé par soi-même. Dans le chapitre 7, un réseau cellulaire auto-organisant est un réseau cellulaire capable de se stabiliser par lui-même à une configuration particulière de relations entre cellules : l'auto-organisation est alors vue comme une forme primitive d'apprentissage.

Besoin (premier) — La tâche ou mission de l'agent : la réalisation graduelle de la tâche se manifeste comme baisse graduelle et disparition du besoin. Toute l'activité de l'agent a donc comme "but" d'amener le(s) besoin(s) premier (s) de l'agent à 0 (§1.3.2). Le besoin correspond à ce qui est appelé "pulsion" en éthologie (cf. motivation).

Cancer — La diminution de l'espérance de vie d'un système dont l'espace vital est petit par rapport au nombre de ses agents, c'est-à-dire d'un système surpeuplé (§8.6.3).

Catalyseur — Dans les réseaux cellulaires plastiques du chapitre 7, un message catalyseur est un message qui n'est pas consommé par les cellules, mais dont la seule présence suffit pour stimuler une réaction chez une pulsion (§7.2.1). Les messages catalyseurs sont en effet les messages qui doivent être partagés par plusieurs cellules simultanément (dans l'implémentation, les cellules font des copies locales des messages catalyseurs sans les consommer, cf. annexe C).

Cellularité — Le mode d'organisation d'un réseau de composants ayant des fonctions de transfert hétérogènes mais une syntaxe de connexions homogène. Ces composants sont appelés cellules (§3.1.3). La plasticité se manifestera comme la possibilité de découvrir de nouvelles sémantiques des connexions.

Cognitif — Le comportement d'un agent est appelé plus cognitif que celui d'un autre, si le premier agent est plus opérationnel que le deuxième, c'est-à-dire s'il accomplit sa mission

mieux que le deuxième. Un observateur externe sera alors tenté de dire que le premier agent possède plus de "connaissance" que le deuxième et cette possibilité d'utiliser de connaissances constitue son avantage opérationnel. Un agent couplé à son environnement peut démontrer un comportement cognitif sans posséder des représentations (§1.2, cf. connaissance, représentation).

Comportement — La façon dont un agent agit. Le comportement est une notion relative selon le contexte dans lequel l'agent se situe et selon le point de vue de l'observateur externe qui le décrit (§1.2). Nous utilisons également le terme dans le même sens que la robotique comportementale, pour indiquer la structure qui donne naissance au comportement en question ; par exemple le comportement de dispersion (§5.3) est la structure qui donne à l'agent sa possibilité de se disperser.

Connaissance — Choses connues, le savoir. En termes plus techniques, la connaissance nécessite la présence des variables dans l'organisation —en l'absence de variables, il s'agit d'un automate (§1.2, cf. représentation, cognitif). La connaissance est donc dynamique.

Connexionnisme — Un mode d'organisation distribuée (qui repose sur un réseau de composants ayant des connexions entre eux) dans lequel l'action a véritablement lieu au niveau de la connexion. Si les composants sont des neurones artificiels, il s'agit d'un réseau neuronal. Selon cette définition, la cellularité définit une organisation connexionniste non neuronale (§3.1.2, 3.1.3). Cette définition a un sens exclusivement structurel et ne restreint guère la classe des problèmes abordés ; dans le paragraphe 3.1.2 nous distinguons les problèmes algorithmiques de ceux de classification.

Coopération — Le type de socialité qui conduit à une amélioration des performances individuelles d'un agent *selon ses propres critères* (§1.3.4).

Couplage — Relation d'influence mutuelle entre deux entités différentes. Pour Varela (1979) une entité et son milieu son couplés en certains points et l'activité endogène de l'un influence l'autre et vice versa. Ces points de couplage peuvent être perçus par un observateur externe comme une sorte de "connaissance" que chacune des entités a de l'autre sans impliquer des représentations au sens classique du terme. Il existe donc de la connaissance *sans* représentation (§1.2, cf. connaissance, représentation, cognitif).

Couplage opérationnel La loi de la "réaction" d'un agent à une perturbation est telle que, si on connaissait d'avance la perturbation exacte, la réaction serait optimale, c'est-à-dire les performances de l'agent seraient statistiquement meilleures que celles de toutes les autres réactions possibles avec la même organisation de base. Cette loi de réaction doit alors inclure des paramètres opérationnels, c'est-à-dire des paramètres de tâche (§1.3.3).

Dynamique Modification continue dans le temps. C'est l'aspect le plus important des systèmes autonomes ; deux propriétés sont particulièrement importantes, l'intégration de diverses dynamiques dans le même système et l'autorégulation des paramètres des dynamiques (§1.2).

Émergence L'apparition d'une structure nouvelle ou d'un comportement nouveau (§1.2). Qualification d'une structure ou d'un comportement comme émergent : *"[A behavior] is emergent if it can only be defined using descriptive categories that are not necessary to describe the behavior of the constituent components"* (Steels 1994*a*, p. 78). Le comportement d'un système autonome doit démontrer un degré d'émergence (§8.6.1).

Espace de représentation

L'espace abstrait des actions (cellulaires), c'est-à-dire l'espace dans lequel le concepteur définit et conçoit les actions. Il s'agit d'un espace mathématique et sa propriété la plus importante doit être la possibilité de permettre la définition (et même l'émergence) de sémantiques hétérogènes dans le réseau (§3.6).

Fréquence propre Une pulsion (composant de la cellule du chapitre 7, cf. pulsion) reconnaît parmi les messages en entrée ceux qui portent comme identificateur une valeur particulière, la template d'identification de la pulsion. Cette template d'identification est appelée fréquence propre (cet analogue physique a été adopté parce que les messages sont des nombres réels). Une pulsion peut avoir plusieurs fréquences propres, si sa fonction de transfert nécessite la "fusion" des messages (§7.2.1).

Intentionnel Un agent est appelé intentionnel si son comportement *peut* être prédit en lui attribuant des croyances, des désirs et des facultés rationnelles (§1.3.1, selon Dennett 1987).

Interaction Une relation entre deux éléments A et B, telle qu'il existe une double action de A sur B et de B sur A ($A \leftrightarrow B$ et non $A \rightarrow B$).

Intrus Un message composé du type (*fréquence, donnée*) est un intrus si la *fréquence* est étrangère, c'est-à-dire si elle n'est reconnue d'aucune des cellules du réseau (§7.4, §7.5.1).

Métaboliser — "Consommer" des messages (§3.3.2). Dans le chapitre 7, métaboliser signifie consommer des messages sélectivement, c'est-à-dire seulement les messages de certains types.

Métabolisme — La partie comportementale de l'agent qui est responsable de la "consommation" des messages provenant de l'extérieur et alors responsable de la réponse de l'agent à ses perturbations (§1.3.5). Puisque les agents métabolisent sélectivement, le métabolisme est plus qu'un simple processus d'entrée/sortie, c'est en effet une activité interne à l'agent qui doit répondre aux besoins propres de l'agent et répondre selon les spécificités de l'environnement.

Motivation — Une variable qui exprime la "volonté" individuelle de l'agent pour agir et résoudre le problème posé ; inversement, elle constitue "l'image" ou "l'idée" que l'agent a du monde et donc elle représente une mesure de l'état d'avancement de la résolution (§1.2). La motivation de l'agent dépend à la fois de sa pulsion interne (son besoin ou sa "faim") et du stimulus externe rencontré (cf. §4.3).

Opérationalité — Un agent est plus opérationnel qu'un autre s'il accomplit sa mission mieux que le deuxième. La notion de l'opérationalité, du point de vue du concepteur de l'agent, est une notion principalement relative et correspond à la notion de viabilité du point de vue de l'observateur (§1.2).

Organisation — L'ensemble des relations qui définissent une machine comme une unité (§1.2, selon Varela 1979). Dans la discussion sur la cellularité, au lieu du terme organisation nous utilisons souvent le terme "architecture" (§3.1.1).

Paramètre libre — Un paramètre dont la valeur n'est pas connue d'avance et elle varie en cours de l'exécution (§1.2).

Perturbation — La différence entre la motivation de l'agent et l'état du monde tel qu'il est perçu par l'intermédiaire des systèmes de perception (§1.2). L'agent s'engage dans une activité "d'ajustement" de ses motivations à ses perturbations, qui a comme but de fond l'atteinte de son nirvana (§1.3.2).

Physiologie — Le mode de couplage entre programme et métabolisme (§1.3.5). Dans un réseau cellulaire, on appelle système physiologique un système de structures qui régulent le réseau globalement (§3.2.1) ; ce système ne peut pas apprendre. La physiologie constitue donc un milieu d'interactions interne à l'agent qui simplifie la conception (grâce à sa globalité) mais qui contraint les formes possibles de métabolisme et par conséquent les possibilités d'apprentissage.

Plasticité

***(a) Point de vue fonctionnel* :** La propriété d'un réseau de s'auto-organiser (apprendre) face à des situations imprévues ou sa propriété de s'adapter à des conditions d'entrée imprévues (§7.1.1).
***(b) Point de vue causal* :** La propriété d'un réseau de s'auto-organiser afin de se maintenir en tant qu'unité cohérente d'activité algorithmique innée, c'est-à-dire la propriété du réseau de maintenir les relations sociales (§7.1.3).

Programme

La partie comportementale d'un agent qui détermine ce que son métabolisme fait (§1.3.5). Le terme est encore utilisé dans le sens plus large de système de contrôle ou de solution à un problème donné, inspiré de la terminologie d'informatique (par exemple, §3.1.1).

Pulsion

Le composant de la cellule-agent du chapitre 7. Elle est capable de métaboliser des messages *sélectivement* par l'intermédiaire d'un mécanisme de reconnaissance(§7.2.1). Ce terme est adopté pour montrer que chacun de ces composants est une entité indépendante et réactive et est emprunté à la littérature du comportement animal (le terme est d'ailleurs utilisé dans son sens habituel — celui d'un variable motivationnelle interne — dans le paragraphe 4.3).

Réactif

Le comportement d'un agent est appelé réactif s'il répond aux aspects dynamiques de son monde (§1.2). À ne pas confondre avec la notion du réflexe. La réactivité est une propriété relative à un observateur : un observateur aura tendance à qualifier un agent qui répond rapidement aux aspects dynamiques de son environnement de réactif, et il aura même tendance de lui attribuer un système de contrôle du type S-R (stimulus-réflexe).

Réflexe

Une structure est appelée une structure réflexe si elle n'est pas commandée à l'intérieur de l'agent, c'est-à-dire si elle n'a pas de connexions avec la partie cellulaire (la partie métabolique). De son côté, un capteur réflexe (§3.2.3) est un capteur au signal duquel l'agent doit répondre aussi vite que possible, par exemple un capteur d'obstacle. La solution idéale serait de connecter ce capteur directement aux actionneurs correspondants, d'où l'absence de connexions avec la partie cellulaire. Au niveau de la description finale, toute action est un réflexe déterministe et mécaniste (§3.4.2).

Représentation

Image d'un objet (§1.2). Le terme est utilisé dans deux contextes : *(1)* représentation qu'un concepteur a de son objet de conception (par exemple, représentation d'action, paragraphe 3.6) et *(2)* représentation qu'un agent a de son monde (par exemple, la variable représentationnelle, §3.3.1, §4.3).

(État de) satisfaction	État dans lequel besoin=0 (§1.3.2).
Sélection d'action	Si plusieurs actions sont possibles simultanément, comment l'agent fait-il pour choisir la plus pertinente (celle qui le mène plus près de son but ou ses buts) ? (§3.4.1).
Sélectivité	Un agent ne considère que les aspects de son monde qui sont relatifs à ses motivations et demeure indifférent au reste. Dans ce sens, l'agent *choisit* (ou fait croire qu'il choisit) ce qui importe pour la réalisation de sa mission, de son projet personnel (§1.3.2). La différence entre un modèle instructif et un modèle sélectif d'agent est analysée dans le paragraphe 7.1.3.
Sénescence	Le vieillissement de l'agent. Le rôle de la sénescence est de forcer l'agent à apprendre. L'autonomie devient ainsi le moyen de prolongation de la durée de vie de l'individu (§1.3.6). À une autre échelle, la sénescence doit être la force motrice de l'émergence des structures de niveau supérieur d'organisation.
Socialité	Une forme d'interaction entre agents qui induit une perturbation de provenance sociale (§1.2).
Structure	Une entité matérielle ou logicielle qui détermine le comportement d'un agent. Cette entité est généralement un système de composants, c'est-à-dire un ensemble de composants ayant un certain nombre de relations entre eux (§1.2).
Système	Ensemble d'éléments en interaction, ensemble d'éléments liés par un ensemble de relations. Un système ne peut pas s'appréhender par l'étude de ses parties prises isolément. La théorie du système général est une science de la "totalité" qui cherche à formuler des principes valables pour les "systèmes" en général indépendamment de la nature des éléments qui les composent et des relations, des "forces", qui les relient (von Bertalanffy 1968, p. 36).
Système d'éveil	Un agrégat de cellules ayant le rôle d'activer la tâche quand il faut et de jouer éventuellement le rôle d'interface avec les autres tâches (§3.2.3).
Système immunitaire	Dans un réseau cellulaire, le système immunitaire est le système de cellules capable de détecter les messages étrangers et les éliminer (§7.4). D'un point de vue plus causal, c'est le système de cellules qui surveille et régule l'adaptation sociale des autres cellules (§7.5.3).
Système motivationnel	Un ensemble de motivations avec un mécanisme central de régulation (§1.3.2). La régulation centrale équivaut à une prise de décision.
Système périphérique	Dans un réseau cellulaire, on appelle système périphérique un système de perception ou de commande qui est

irrépressible, c'est-à-dire dont on ne peut pas empêcher le fonctionnement, et dont le fonctionnement est indépendant des autres parties du réseau cellulaire (§3.2.3).

Systémique, démarche — Résoudre un ensemble de problèmes particuliers, ensuite généraliser à partir de ces solutions et élaborer un modèle ou une solution ou des principes suffisamment abstraits pour s'appliquer à tous les problèmes étudiés (Introduction, selon von Bertalanffy 1968 et Weinberg 1975). Dans ce travail, il a été question de généraliser quant aux principes d'organisation d'agents autonomes de niveaux différents : agent-animat et agent-cellule.

Tâche — (a) Le problème que le concepteur de l'agent (ou des agents) veut résoudre (§1.1).
(b) La mission ou le projet personnel de l'agent (§1.1).
(c) Dans l'organisation cellulaire du chapitre 3, une structure du troisième ordre, après celui de la cellule et de l'agrégat. Une tâche correspond au comportement (behavior) rencontré dans la littérature de la robotique comportementale et comprend plusieurs agrégats (§3.2.3).

Variable essentielle — Variable dont la survie de l'agent dépend (§1.3.3, selon Ashby 1960). Dans notre modèle d'agent autonome, les variables essentielles sont les besoins (premiers) de l'agent.

Virus — Un message composé du type (*fréquence, donnée*) est un virus si la *donnée* est étrangère, c'est-à-dire si la donnée est mutée de manière aléatoire. Un agent attaqué des virus peut démontrer une "aliénation sémantique" en donnant de fausses réactions à ses perturbations, par exemple entrer en collision avec les murs dans le cas de la navigation (§7.5.1).

Index des figures

Chapitre 1

Chapitre 3

Chapitre 4

Chapitre 5

Chapitre 6

Chapitre 7

Chapitre 8

Annexe A

Index des tableaux

Chapitre 1

Chapitre 2

Chapitre 3

Chapitre 4

Chapitre 5

Chapitre 6

Chapitre 7

Chapitre 8

Chapitre 9

Annexe A

Articles liés à la thèse

Ces articles n'ont pas été cités dans le texte, car ce dernier est plus récent et donc les mêmes idées s'y retrouvent plus élaborées.

Tzafestas, E.S. (1993). De la machine à vapeur à Beethoven en passant par les robots autonomes, *LAFORIA Research Report 93/16*, May, 42 p.

Tzafestas, E.S. (1994*a*). A cellular control architecture for autonomous robots, pp. 70-79, *Proceedings of the 1994 International Workshop on Intelligent Robotic Systems*, Grenoble, July.

Tzafestas, E.S. (1994*b*). A design framework for societies of autonomous robots, *Poster Presentation Artificial Life IV Workshop*, Cambridge, MA, July 1994, also *LAFORIA Research Report 94/10*, June, 25p.

Tzafestas, E.S. (1994*c*). Implementing reactive algorithms on a cellular control architecture, pp. 191-201, *Proceedings of the 10th International Workshop on Autonomous Mobile Systems (AMS '94)*, Stuttgart, October, Springer-Verlag, Informatik Aktuell Series.

Tzafestas, E.S. (1994*d*). Agentifying the process : Task-based or robot-based decomposition ?, pp. 582-587, *Proceedings of the 1994 IEEE International Conference on Systems, Man and Cybernetics*, San Antonio, TX, October.

Tzafestas, E.S. (1994*e*). Operational coupling : A design principle and an intermediate representation level, *"The Role of Dynamics and Representation in Adaptive Behaviour and Cognition" (DRABC'94) Workshop Notes*, San Sebastián, December.

Tzafestas, E.S. (1995*a*). Reactive robots in the service of production management, pp. 449-456, *Proceedings of the 4th International Conference on Intelligent Autonomous Systems (IAS-4)*, Karlsruhe, March.

Tzafestas, E.S. (1995*b*). Variation et auto-organisation à l'intérieur du robot : Le cas d'un modèle de fréquences propres, *Actes des Journées de Rochebrune "Évolution et organisation"*, pp. 181-192, Rochebrune, France, March.

Tzafestas, E.S. (1995*c*). Compromising algorithmicity and plasticity in autonomous agent control systems : The motivated, social cell, *Accepted for poster presentation 1995 European Conference on Artificial Life (ECAL'95)*, Granada, June, extended and revised version appeared as *LAFORIA Research Report 95/33* under the title "Compromising algorithmicity and plasticity in autonomous agent control systems : The autonomous cell", December, 36 p.

Tzafestas, E.S. (1995*d*). Beyond cooperation-ism and competition-ism (Exploring social phenomena with a generalized tit-for-tat model), p. 464, *Proceedings First International Conference on Multi-Agent Systems (ICMAS'95)*, San Francisco, CA, June, extended and revised version appeared as *LAFORIA Research Report 95/34* under the title "Beyond cooperation and competition : Explorations with a quantitative tit-for-tat model", December, 24 p.

Tzafestas, E.S. (1995*e*). Peripheral cellular control : An eigen-frequency model and a case-study in self-organization, pp. 483-488, Vol. I, *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'95)*, Pittsburgh, PA, August.

Tzafestas, E.S. (1996). Aging agents, *LAFORIA Research Report 96/01*, January, 22 p.