Thèse d'Habilitation à diriger des recherches de l'université Paris VI

Spécialité : **Informatique**

Présentée par : **Olivier SIGAUD**

Comportements adaptatifs pour les agents dans des environnements informatiques complexes

Soutenue le **10 décembre 2004** devant le jury composé de :

François	CHARPILLET	RAPPORTEUR
Jean-Paul	HATON	RAPPORTEUR
Jacques	MALENFANT	EXAMINATEUR
Jean-Arcady	MEYER	EXAMINATEUR
Patrice	PERNY	EXAMINATEUR
Marc	SCHOENAUER	EXAMINATEUR
Stewart	WILSON	RAPPORTEUR

À Robin Henri Philippe Gigotin Sigaud,
dit « Monsieur Chou »

Table des matières

1 Introduction générale **17**

1.1 Positionnement scientifique . . . 17
1.1.1 Une approche de l'intelligence . . . 17
1.1.2 Des robots pour expérimenter . . . 18
1.1.3 Limite du cadre robotique . . . 19
1.1.4 Une alternative : les simulateurs réalistes de grande envergure . . . 19
1.1.5 Un domaine d'application : les jeux vidéo . . . 21
1.2 Programme de recherche . . . 22
1.3 Plan du mémoire . . . 24

2 Apprentissage par renforcement **27**

2.1 Origines psychologiques . . . 27
2.1.1 Modèles associationnistes de l'apprentissage . . . 27
2.1.2 Le conditionnement classique . . . 28
2.1.3 Le conditionnement opérant . . . 29
2.1.4 L'apprentissage par essais et erreurs . . . 29
2.1.5 Vers la modélisation informatique . . . 30
2.2 Les modèles de Markov . . . 31
2.2.1 Une famille de modèles . . . 31
2.2.2 Les processus de décision markoviens . . . 32
2.3 Algorithmes d'optimisation du comportement . . . 36
2.3.1 La programmation dynamique . . . 37
2.3.2 Les méthodes de Monte Carlo . . . 38
2.3.3 Les méthodes de différence temporelle . . . 39
2.4 Principaux algorithmes d'apprentissage par renforcement . . . 41
2.4.1 L'algorithme TD(0) . . . 41
2.4.2 L'algorithme SARSA . . . 42
2.4.3 L'algorithme Q-LEARNING . . . 42
2.4.4 Les algorithmes TD(λ), Sarsa(λ) et Q(λ) . . . 43
2.4.5 Les architectures acteur-critique . . . 44
2.5 Architectures biomimétiques et apprentissage par renforcement . . . 44
2.6 Conclusion : limites du cadre de l'apprentissage par renforcement . . . 46

3 Systèmes de classeurs à anticipation **49**
3.1 Les systèmes de classeurs . . . 49
3.1.1 Insuffisance de la notion d'état . . . 49
3.1.2 Historique succinct des systèmes de classeurs . . . 51
3.1.3 Description des systèmes de classeurs . . . 52
3.1.4 Récapitulation . . . 57
3.2 L'apprentissage latent . . . 57
3.2.1 Sources psychologiques . . . 57
3.3 Modèles informatiques de l'apprentissage latent . . . 59
3.3.1 Les architectures DYNA . . . 59
3.3.2 Les systèmes de classeurs à anticipation . . . 61
3.4 Présentation détaillée de MACS . . . 63
3.4.1 Architecture générale . . . 63
3.4.2 Formalisme de MACS . . . 64
3.4.3 Apprentissage du modèle des transitions . . . 65
3.4.4 Apprentissage du modèle des récompenses . . . 69
3.4.5 Algorithme de choix de l'action . . . 71
3.5 Quelques résultats marquants . . . 72
3.6 Comparaison avec les architectures DYNA . . . 73
3.6.1 Comparaison des vitesses d'apprentissage . . . 73
3.6.2 Comparaison de la compacité des modèles . . . 75
3.6.3 Expériences dans des environnements variables . . . 76
3.7 Conclusion . . . 77

4 Processus de Décision Markoviens Factorisés **79**
4.1 Les processus de décision markoviens factorisés . . . 80
4.2 Les réseaux bayésiens dynamiques . . . 81
4.3 Exemple de processus de décision markoviens factorisé . . . 82
4.4 Comparaison des approches . . . 83
4.4.1 Comparaison des représentations . . . 83
4.4.2 Comparaison des mécanismes . . . 84
4.5 Programmation dynamique sur des représentations compactes . . . 85
4.5.1 L'algorithme SPI . . . 85
4.5.2 L'algorithme SPUDD . . . 86
4.6 Résolution de FMDPs par programmation linéaire . . . 87
4.6.1 La programmation linéaire . . . 87
4.6.2 Application directe aux processus de décision markoviens . . . 88
4.6.3 Application compacte pour les FMDPs . . . 88
4.6.4 Approximation linéaire de la fonction de valeur . . . 89
4.6.5 Prise en compte de la structure des dépendances entre variables . . . 90
4.6.6 Réduction du nombre de contraintes . . . 90
4.6.7 Discussion . . . 93
4.7 Apprentissage par renforcement dans les FMDPs . . . 93
4.7.1 Les algorithmes E^3 et Factored-E^3 . . . 94

4.7.2 R_{max}, Factored-R_{max} et la proposition de C. Guestrin . . . 95
4.8 Apprentissage de la structure dans les FMDPs . . . 96
4.9 Travaux futurs et conclusion . . . 99

5 Observabilité partielle **101**
5.1 Les processus de décision markoviens partiellement observables . . . 101
5.1.1 État interne et modèle interne . . . 103
5.1.2 Types de problèmes non markoviens . . . 104
5.1.3 Résolution classique . . . 105
5.2 Solutions heuristiques aux problèmes non markoviens . . . 105
5.3 ATNoSFERES . . . 110
5.3.1 Codages génétiques de structures . . . 110
5.3.2 Construction d'une structure de commande par ATNoSFERES . . . 111
5.3.3 Mécanisme de sélection d'un automate performant . . . 115
5.3.4 Mécanisme d'exécution d'un automate . . . 115
5.4 Équivalence avec les systèmes de classeurs . . . 117
5.5 Synthèse des principaux résultats . . . 118
5.6 Conclusion . . . 119

6 Champs de forces pour la coordination spatiale **121**
6.1 Introduction . . . 122
6.2 Champs de potentiel et planification de chemins . . . 123
6.3 Champs de potentiel et coordination spatiale . . . 124
6.4 Présentation de GACS . . . 125
6.4.1 Présentation générale . . . 125
6.4.2 Un formalisme pour la coordination spatiale . . . 126
6.4.3 Sélection automatique d'un contrôleur performant . . . 128
6.5 Validation expérimentale . . . 131
6.5.1 Un premier démonstrateur . . . 131
6.5.2 Un problème plus difficile . . . 132
6.6 Discussion . . . 135
6.7 Travaux futurs . . . 137
6.8 Conclusion . . . 138

7 Champs de forces et dynamique interne **141**
7.1 Introduction . . . 141
7.2 Un environnement de simulation . . . 143
7.3 Extension du formalisme . . . 143
7.4 Étude expérimentale . . . 144
7.4.1 Résultats . . . 144
7.4.2 Fonctionnement de deux contrôleurs dynamiques . . . 144
7.5 Discussion . . . 147
7.6 Conclusion . . . 148

8 Apprentissage par population **151**
8.1 Évolution et apprentissage . . . 151
8.2 Contrôle par population et adaptation . . . 152
8.3 Une famille générique de modèles . . . 154
8.3.1 Une racine commune . . . 154
8.3.2 Evolution des individus . . . 154
8.3.3 Apprentissage des individus . . . 155
8.3.4 Familles d'opérateurs . . . 155
8.3.5 Interactions . . . 155
8.3.6 Travaux apparentés . . . 156
8.4 Discussion des modèles . . . 157
8.4.1 Réflexion sur l'approche Michigan . . . 157
8.4.2 Apprentissage par population et problèmes non markoviens . . . 159
8.5 Présentation de FACS . . . 160
8.5.1 Introduction . . . 160
8.5.2 Prise de contrôle et calcul de l'enchère . . . 161
8.5.3 L'algorithme d'évolution par sélection dans FACS . . . 163
8.5.4 Discussion . . . 164
8.6 Apprentissage de champs de forces . . . 164
8.6.1 Procédé de transformation du modèle . . . 164
8.6.2 Discussion . . . 166
8.7 Conclusion . . . 167

9 Apprentissage de la coordination **169**
9.1 Apprentissage par renforcement continu . . . 169
9.1.1 Difficulté du cadre continu . . . 169
9.1.2 Solution envisagée . . . 171
9.1.3 Un formalisme pour l'acteur . . . 172
9.1.4 Une ébauche de superviseur . . . 172
9.1.5 Adaptation de l'acteur par la correction du superviseur . . . 174
9.1.6 Une validation progressive . . . 175
9.1.7 De nombreux points en suspens . . . 176
9.2 Coordination d'actions collectives . . . 176
9.2.1 Extension du cadre markovien : les jeux de Markov . . . 176
9.2.2 Classes de jeux et notions d'optimalité . . . 178
9.2.3 Processus de décision markoviens multi-agents . . . 180
9.2.4 Les processus de décision markoviens partiellement observables décentralisés . . . 181
9.3 Coordination d'objectifs de déplacement . . . 183
9.3.1 Un cadre formel abstrait . . . 184
9.3.2 Limitations du cadre théorique . . . 184
9.3.3 Eléments de solution pratique . . . 185
9.4 Conclusion . . . 187

10 Convergences **189**
10.1 Une plate-forme expérimentale . . . 189
10.2 Réalisations préliminaires . . . 190
10.3 Un objectif fédérateur . . . 191
10.4 Adéquation des techniques adaptatives . . . 192
10.5 Prise de recul . . . 194
10.6 Bilan et perspectives scientifiques . . . 194
10.7 Retour vers l'intelligence des êtres vivants . . . 196

A Curriculum Vitæ **199**
A.1 État civil . . . 199
A.2 Formation et Diplômes . . . 199
A.2.1 Formation scientifique . . . 199
A.2.2 Formation générale . . . 200
A.2.3 Expérience professionnelle . . . 200
A.3 Enseignement . . . 201
A.3.1 A l'Université PARIS 6 . . . 201
A.3.2 A l'ENSTA . . . 202
A.3.3 Divers . . . 202
A.3.4 Synthèse . . . 202
A.4 Encadrement . . . 203
A.4.1 Thèses . . . 203
A.4.2 DEA . . . 203
A.4.3 Autre . . . 203
A.5 Projet de recherche . . . 204
A.5.1 Problématique scientifique . . . 204
A.5.2 Problématique appliquée . . . 204
A.5.3 Travaux réalisés, apports et visibilité . . . 205
A.5.4 Travaux en cours et perspectives . . . 205
A.6 Collaborations scientifiques, projets contractuels . . . 206
A.7 Animation scientifique . . . 207
A.7.1 Organisation de colloques et séminaires . . . 207
A.7.2 Edition d'ouvrages et numéro spéciaux . . . 207
A.7.3 Jury de thèses . . . 207
A.7.4 Participation à des commissions . . . 207
A.7.5 Membre de comités de programme et relecteur . . . 207
A.8 Publications . . . 208

Résumé

Ce mémoire présente un programme de recherche consacré à la mise au point de comportements adaptatifs pour des agents évoluant dans des simulations réalistes de grande envergure tels que les jeux vidéo. Ce type d'applications concentre toutes les difficultés qui limitent la portée des algorithmes adaptatifs existants, à savoir les problèmes de dimension des environnements, de continuité des variables manipulées, ainsi que le caractère non stationnaire des interactions entre les agents et leur environnement. Notre objectif est de repousser ces limites dans plusieurs directions à la fois.

Nous commençons notre exposé par une présentation du cadre général de l'apprentissage par renforcement et de ses limites, avant de décrire dans les chapitres suivants les différentes extensions ou alternatives qui permettent de s'affranchir progressivement de ces limites.

Ainsi, au chapitre 3, nous montrons comment les systèmes de classeurs permettent de donner une représentation compacte de l'espace d'états, puis nous détaillons nos travaux dans le domaine des systèmes de classeurs à anticipation, dont le principal atout est d'accélérer la convergence de l'apprentissage. Cette voie de recherche est prolongée au chapitre 4 par une présentation des réseaux bayésiens dynamiques, qui nous permettent de formaliser efficacement les travaux empiriques menés dans le cadre des systèmes de classeurs à anticipation. La poursuite des travaux sur les réseaux bayésiens dynamiques est au cœur de nos efforts pour élaborer des mécanismes adaptatifs capables de traiter des problèmes de plus grande taille.

Un deuxième axe développé dans le cadre de ce mémoire concerne le traitement des problèmes dans lesquels l'hypothèse de Markov n'est pas vérifiée et, en particulier, les problèmes non stationnaires. Nous présentons au chapitre 5 une architecture fondée sur les algorithmes génétiques et destinée à résoudre certains problèmes non markoviens, puis nous exposons au chapitre 8 nos efforts pour repenser cette architecture dans un cadre intégrant des mécanismes d'apprentissage par renforcement.

Cette voie de recherche nous conduit à généraliser les mécanismes fondamentaux des systèmes de classeurs pour abstraire une approche générique que nous appelons « apprentissage par population ».

Notre troisième axe de recherche concerne le traitement des problèmes continus et des problèmes multi-agents. Les deux sources de difficulté se trouvent réunies dans les problèmes de coordination spatiale des agents. Au chapitre 6, nous exposons une première approche de cette problématique, qui consiste à ajuster automatiquement grâce à un algorithme génétique multi-critères les paramètres d'un ensemble de champs de force qui permettent de déplacer des agents relativement les uns aux autres. Nous montrons au chapitre 7 comment cette approche peut être étendue à la prise en compte de variables internes pour résoudre des problèmes de sélection de l'action, puis nous montrons au chapitre 8 comment l'approche peut elle aussi être enrichie par des mécanismes d'apprentissage par renforcement dans le cadre de notre modèle générique d'apprentissage par population. Enfin, au chapitre 9, nous présentons nos efforts les plus récents visant à résoudre les problèmes de navigation dans un espace continu et les problèmes de coordination multi-agents dans le cadre général des recherches sur l'apprentissage par renforcement.

Après avoir exposé ces différentes techniques qui visent toutes à étendre le domaine d'application des méthodes

d'apprentissage par renforcement à des problèmes vastes et complexes, nous tirons un bilan de la possibilité de les appliquer directement aux simulations de type jeux vidéo qui motivent nos travaux. Il apparaît clairement que d'importants efforts théoriques restent à fournir, si bien que nous concluons sur la nécessité de nous tourner à nouveau vers le fonctionnement des systèmes vivants pour y rechercher d'autres mécanismes susceptibles de répondre à nos attentes.

Summary

The thesis describes a research program devoted to the design of adaptive behaviors for agents immerged in large scale realistic simulation environments such as video games. In this application domain, all the obstacles faced by current adaptive algorithms are present : large scale problems, continuous variables, non stationarity of the agents-environment interactions. Our goal is to overcome several of the corresponding limitations at once.

We start by presenting the general reinforcement learning framework and its limitations, before describing in subsequent chapters the different extensions or alternatives that result in progressively overcoming these limitations.

Thus, in chapter 3, we show how Learning Classifier Systems give a compact representation of the state space, then we present our work on Anticipatory Learning Classifier Systems, resulting in a faster learning convergence. This approach is extended in chapter 4 with a presentation of Dynamic Bayes Networks, giving rise to an efficient formalization of our empirical work with Anticipatory Learning Classifier Systems. This research trend is central to our attempt to deal with large scale problems.

Our second research trend is devoted to the resolution of non Markov and non stationary problems. In chapter 5, we present a framework based on Genetic Algorithms designed to solve non Markov problems and we show in chapter 8 how we strive to include into this framework some Reinforcement Learning mechanisms. This trend results in an attempt to abstract out of the fundamental mechanisms of Learning Classifier Systems a generic approach that we call “population learning”.

Our third research trend deals with continuous and multiagent problems, whose intersection relies in spatial coordination problems. In chapter 6, we describe our first approach to this problem, which consists in tuning with a multicriteria Genetic Algorithm the parameters of force fields acting on agents. In chapter 7, we indicate how this approach is extended to deal with internal variables and, in chapter 8, we show how Reinforcement Learning mechanisms can be included in the framework in the context of our “population learning” approach. Finally, in chapter 9, we present our most recent efforts to tackle spatial navigation problems with continuous Reinforcement Learning and to learn how to coordinate a team of agents thanks to Reinforcement Learning mechanisms.

After having presented these different techniques designed to extend the applicability of Reinforcement Learning mechanisms to complex problems, we discuss their current applicability to simulation environments such as up-to-date video games. We show that a serious research effort is still necessary and we conclude to the necessity of investigating further the adaptive mechanisms of living animals to find some new sources of inspiration.

Remerciements

Devenir directeur de thèse pose un problème logique insoluble : pour diriger une thèse, il faut une habilitation. Pour obtenir une habilitation, il faut avoir démontré en pratique sa capacité à diriger des thèses. Il se trouve fort heureusement des individus suffisamment altruistes pour accepter le rôle de directeur officiel de thèses dont ils confient l'encadrement effectif à de futurs candidats à l'habilitation. En ce qui me concerne, c'est Jean-Arcady Meyer qui a joué ce rôle. Il a su m'octroyer l'autonomie nécessaire pour que l'exercice prenne tout son sens, tout en veillant à m'épargner quelques inévitables erreurs de jeunesse.

J'ai considérablement appris au contact de Jean-Arcady-Meyer-et-Agnès-Guillot, duo scientifique inséparable qui illustre à merveille le principe selon lequel le tout est plus que la somme des parties. En l'occurrence, ce tout est parvenu à combiner ces vertus fondamentales que sont, d'une part, l'opiniâtreté nécessaire pour mener à bien des projets d'une grande originalité scientifique avant qu'il y ait une communauté de chercheurs pour y croire et, d'autre part, l'exigence et la rigueur indispensables pour faire reconnaître la valeur académique de ces recherches au plan international. Je tiens à les remercier pour tout ce qu'impliquent ces trois années passées ensemble, mais en particulier pour l'ambiance de travail fructueuse et cordiale qu'ils font régner au sein de l'AnimatLab.

Ce mémoire n'aurait pas de raison d'être s'il ne s'était pas trouvé quelques jeunes inconscients pour accepter d'affronter le défi de la thèse en ajoutant à la difficulté intrinsèque de l'exercice le lourd handicap de mon encadrement.

Le premier, et non le moindre, est Pierre Gérard, qui non seulement a survécu à l'épreuve, mais en plus est parvenu à décrocher un poste académique avec tous les honneurs. J'ai déjà eu l'occasion de remercier Pierre, mais je tiens à répéter ici que je mesure pleinement la chance que j'ai eue de débuter ma carrière d'encadrant avec un doctorant de sa valeur, à la fois sur les plans scientifiques et humains.

Le second est Fabien Flacher, qui est bien parti pour survivre à son tour. Fabien a fait preuve d'une grande autonomie et d'une grande exigence sur la qualité de son travail, traits de caractère qui m'ont permis de me reposer sur lui en bien des occasions où je n'avais pas la disponibilité nécessaire pour m'acquitter pleinement de mes devoirs envers lui.

Les deux suivants sont Thomas Degris et Thierry Gourdin, par ordre alphabétique, qui se sont jetés avec la naïveté de la jeunesse dans la grande aventure dont il est question dans ce document. Les aptitudes techniques stupéfiantes du premier et le recul scientifique remarquable du second sont mes principales raisons d'espérer mener à bien le projet déraisonnable que je vais exposer dans les pages qui suivent.

Samuel Landau occupe une place à part dans cette liste, puisque notre collaboration résulte simplement d'intérêts scientifiques communs, hors de toute contrainte académique. Merci à Samuel pour l'énergie qu'il a dépensée sans compter pour faire avancer quelques-unes des idées exposées dans ce mémoire.

Je ne remercierai pas nommément tous les stagiaires que j'ai encadrés mais, d'une façon générale, je tiens à dire à tous mes collègues qui doutent parfois, devant la difficulté de bien faire notre métier dans des conditions trop rarement idéales, qu'il faut mesurer la chance que nous avons d'être au contact de ces jeunes qui ne demandent qu'à

donner de leur temps et de leur énergie pour participer à nos projets, pourvu qu'on leur confie une tâche motivante et propice à leur apprentissage.

On a souvent besoin d'un plus matheux que soi, mais les matheux patients à l'égard de ceux qui le sont moins ne courent pas les rues. Merci à Pierre-Henri Wuillemin pour ne jamais s'être lassé de mes divagations lorsqu'il a accepté de relire les nombreuses versions successives de mon chapitre 4 et, plus généralement, pour la générosité dont il fait preuve dans nos travaux communs. Je ne remercierai pas ici tous mes collègues enseignants ou chercheurs du LIP6 qui le méritent, parce que la litanie serait trop longue. Je pense en particulier à ceux qui se dévouent pour accepter des tâches collectives en ces temps troublés où chaque responsabilité implique d'affronter des difficultés à peine surmontables. J'espère que chacun se reconnaîtra et que personne ne m'en voudra de ne pas avoir été nommé explicitement.

Par contre, je tiens à remercier nommément Ghislaine, Jacqueline et Christophe, qui assurent le fonctionnement du laboratoire dans des conditions parfois surréalistes. Je n'ose imaginer ce que serait la vie ici sans eux.

Je tiens à remercier tous les examinateurs de mon jury d'habilitation, Marc Schoenauer, Patrice Perny et Jacques Malenfant pour avoir bien voulu perdre quelques soirées en compagnie de mon manuscrit. Je m'incline encore plus profondément devant mes rapporteurs, François Charpillet, Jean-Paul Haton et Stewart Wilson qui ont accepté d'évaluer ce travail. Le dernier, en particulier, est d'autant plus méritant que le français n'est pas sa langue maternelle, ce qui ne lui simplifie pas l'exercice.

Au passage, je voudrais remercier ma soeur, Dominique Sigaud, parce qu'elle écrit des romans admirables dans une société où il faut être fort pour tenir la position de sincérité sans concession qu'elle a choisi d'adopter.

Enfin, je remercie Marie-Laure, qui mérite amplement son habilitation à taper des bouts de thèse, et les 6 ECTS de son UE de mère de famille de niveau 100.

Chapitre 1

Introduction générale

« Si nous savions ce que nous sommes en train de faire, nous n'appellerions pas cela de la recherche, n'est-ce pas ? »
A. Einstein

1.1 Positionnement scientifique

1.1.1 Une approche de l'intelligence

L'approche animat [1] est un courant scientifique né au milieu des années quatre-vingt visant à la fois à comprendre l'intelligence des êtres vivants et à mettre au point des systèmes capables d'interagir en permanence de façon adaptée avec leur environnement. Elle repose sur le présupposé selon lequel le meilleur moyen de comprendre l'intelligence consiste à commencer par étudier le comportement. Sous le terme de « comportement », on désigne la façon dont des agents, qu'ils soient vivants ou artificiels, interagissent avec leur environnement.

Cette approche se présente comme une alternative à l'intelligence artificielle dite « classique », qui focalise ses recherches sur l'examen des mécanismes de l'esprit humain, étudié essentiellement au travers de la capacité à manipuler des symboles [2]. Les critiques de cette approche précédente ont été nombreuses [3]. Elles reposent toutes sur le rejet de l'idée d'une autonomie du niveau des connaissances symboliques, idée sur laquelle l'intelligence artificielle classique a bâti son programme de recherche. En effet, les difficultés à appliquer les problèmes d'intelligence artificielle aux problèmes concrets de la vie quotidienne et aux problèmes de robotique ont montré qu'il était difficile de mettre au point des systèmes intelligents sans résoudre le problème de l'ancrage des symboles [4] dans les aspects sub-symboliques du comportement.

L'approche animat se concentre donc sur l'étude du comportement. La difficulté que pose l'étude de l'intelligence au travers de cette nouvelle approche est que les problèmes d'interaction que résolvent les êtres vivants sont souvent trop complexes pour être étudiés sereinement dans le cadre de travaux de laboratoire. En effet, en cherchant trop à simplifier les situations d'interaction pour pouvoir les étudier en détail, on prend le risque de passer à côté des problèmes fondamentaux auxquels l'intelligence répond effectivement.

[1] [Wilson, 1985, Wilson, 1987, Wilson, 1991, Meyer, 1996b, Meyer, 1996a, Meyer, 1998, Guillot & Meyer, 2001]
[2] [Newell & Simon, 1972, Newell, 1980]
[3] [McCarthy & Hayes, 1969, Dreyfus, 1972, Dennett, 1990, Brooks, 1991, Varela et al., 1991, Dreyfus, 1996, Clark, 1997]
[4] [Harnad, 1990]

La réponse à cette difficulté consiste à commencer par identifier les mécanismes fondamentaux du comportement au travers de l'interaction d'animaux simples avec leur environnement, puis à se tourner peu à peu vers des animaux dotés de facultés de plus en plus complexes, jusqu'à atteindre le degré de complexité qui permet de comprendre l'intelligence humaine.

Une telle démarche conduit naturellement à étudier le moteur de l'évolution qui a permis l'émergence d'organismes dotés d'une intelligence symbolique à partir d'organismes plus simples, donc s'intéresser à l'ensemble des mécanismes adaptatifs qui opèrent au sein du vivant, conformément au paradigme central des sciences du vivant. C'est ce qu'exprime très bien J. Jozefowiez01 dans sa thèse :

> « L'adaptation est la caractéristique fondamentale du comportement, celle qu'il nous faut étudier si nous voulons réellement comprendre celui-ci. Les animaux sont des machines à s'adapter, c'est ainsi qu'ils ont évolué, c'est ainsi qu'ils doivent être étudiés. » in [Jozefowiez, 2001] p. 21

Dans cette nouvelle perspective, l'évolution des espèces, l'adaptation au milieu et la mise en œuvre de comportements adaptatifs apparaissent donc comme les racines qu'il faut étudier en priorité si l'on veut comprendre l'intelligence en général et celle de l'homme en particulier. C'est dans cette perspective que se situe la dimension la plus fondamentale de notre projet de recherche.

1.1.2 Des robots pour expérimenter

Une fois posée cette perspective épistémologique qui fait reposer l'étude de l'intelligence sur l'examen des mécanismes et comportements adaptatifs des animaux, il faut se donner les moyens d'étudier ces mécanismes et comportements dans un cadre scientifique. La démarche générale de l'approche animat consiste à élaborer des systèmes capables d'exhiber des comportements adaptatifs et à les confronter à des environnements dans lesquels la plus-value apportée par cette capacité peut être mise en évidence, voire mesurée rigoureusement.

Pour ce qui est du choix d'un environnement d'évaluation, les chercheurs qui ont adopté cette démarche avancent deux arguments pour préférer la réalisation de systèmes physiques plongés dans le monde réel plutôt que de travailler sur des systèmes logiciels intégrés dans des environnements de simulation. D'une part, le monde réel pose des problèmes d'une richesse et d'une complexité qu'aucune simulation n'est susceptible d'égaler, que ce soit pour son caractère dynamique et imprévisible, mais aussi par les sources de difficultés matérielles innombrables qu'il pose : bruits divers, frottements et glissements, variations optiques, acoustiques et de température, etc. La faculté d'adaptation du vivant est taillée pour faire face à ce niveau de difficulté et l'on risque de manquer des traits caractéristiques de cette faculté en se donnant une version simplifiée des problèmes auxquels elle est confrontée [5]. D'autre part, lorsqu'on s'appuie sur des mécanismes adaptatifs, cette richesse extrême du monde réel, au lieu d'être une source de difficultés, peut aussi offrir des opportunités d'adaptation que des mécanismes appropriés peuvent exploiter pour trouver une solution efficace à un problème comportemental donné [6].

S'ajoute à cela le fait que développer un simulateur dans le seul but de tester les capacités adaptatives d'un mécanisme particulier peut être long, difficile et coûteux. Si l'on s'intéresse au mécanisme du vol à ailes battantes, par exemple, il est sans doute plus rentable d'acquérir un robot capable de battre des ailes que de se doter d'un simulateur réaliste de mécanique des fluides capable de rendre compte de façon détaillée de l'ensemble des phénomènes pertinents pour la sustentation d'un oiseau simulé. Ou encore, dès lors que l'on s'intéresse aux mécanismes de la vision, il est souvent plus judicieux d'utiliser les informations vidéos fournies par la caméra d'un robot plutôt que de calculer à une cadence élevée des images artificielles au sein d'un simulateur.

[5] [Brooks, 1991, Clark, 1997]
[6] [Nehmzow, 1992]

De nombreux chercheurs qui se réclament de l'approche animat ont donc recours à des robots mobiles plongés dans des environnements plus ou moins étroitement contrôlés pour démontrer les capacités adaptatives de leurs systèmes [7].

1.1.3 Limite du cadre robotique

Si elle est parfaitement justifiée sur le plan épistémologique, cette approche pose de nombreux problèmes sur le plan pratique, car les robots disponibles aujourd'hui sont loin d'atteindre le degré de fiabilité, de maturité et de disponibilité que l'on pourrait souhaiter pour réaliser dans de bonnes conditions des expériences centrées sur l'adaptation.

En effet, la plupart des robots disposent d'une autonomie énergétique très faible et exhibent rarement des capacités d'action proches de celles dont disposent les animaux pour résoudre les problèmes auxquels ils sont confrontés. En outre, la relative pauvreté de leurs capteurs fait que la plupart des robots sont indifférents à la richesse des signaux provenant du monde extérieur. Enfin, ces mêmes robots sont souvent fragiles et tombent facilement en panne, ce qui rend extrêmement fastidieux et coûteux en temps la réalisation de séries d'expériences dotées d'une portée statistiquement significative. Toutes ces contraintes conduisent les chercheurs à contrôler très strictement les cadres expérimentaux dans lesquels ils font évoluer leurs robots, afin de ne pas leur poser des problèmes qu'ils sont incapables de résoudre à cause de limitations strictement matérielles, ce qui revient à se prémunir contre la complexité intrinsèque du monde réel. Or cette complexité était pourtant l'argument fondamental pour s'astreindre à n'étudier le comportement que dans un cadre robotique.

La relative immaturité de la robotique mobile, discipline très récente, rend donc caduque pour l'instant une partie des prises de position qui voudraient faire des robots le seul moyen d'investiguer efficacement les interactions entre un système intelligent et son environnement.

Par ailleurs, la plupart de ces robots sont coûteux et impliquent donc la mobilisation de moyens financiers qu'il n'est pas toujours facile de réunir.

Cette situation est sans doute temporaire, dans la mesure où de nombreuses recherches visent à lever la plupart de ces contraintes, parfois en s'inspirant dans la mesure du possible des propriétés spécifiques des êtres vivants.

1.1.4 Une alternative : les simulateurs réalistes de grande envergure

La tendance naturelle des chercheurs qui choisissent d'utiliser des simulations pour étudier le fonctionnement de mécanismes adaptatifs consiste à développer un petit simulateur extrêmement simplifié qui suffit tout juste à mettre en évidence la capacité spécifique à laquelle ils s'intéressent. Le danger sous-jacent, comme nous l'avons déjà souligné, est de passer à côté du problème réel auquel sont confrontés les êtres vivants et de développer des mécanismes adaptatifs *ad hoc* qui ne valent que pour l'environnement simulé auquel ces mécanismes sont confrontés.

Cependant, il existe une autre voie, que nous allons adopter dans le cadre de ce mémoire. Cette voie consiste à immerger des agents logiciels dotés de capacités adaptatives dans des environnements logiciels complexes, souvent développés dans un cadre industriel, avec des objectifs qui ne se limitent généralement pas à la réalisation d'expériences pour la recherche sur les comportements adaptatifs. Face à l'état actuel de la robotique, en effet, il apparaît raisonnable, en attendant de disposer d'une génération de robots mieux équipés en capteurs, plus agiles, plus fiables et moins onéreux, de se tourner vers des simulations de grande envergure pour y trouver des environnements certes

[7] [Arkin, 1989, Hasegawa et al., 1999, Jakobi, 1997, Nolfi & Floreano, 2000]

un peu moins riches, mais tout de même au moins aussi dynamiques et imprévisibles que la plupart des cadres expérimentaux sévèrement contrôlés de la robotique.

Ces environnements logiciels peuvent être répartis sur une échelle allant d'un extrême très abstrait, lorsqu'il s'agit par exemple de la représentation informatique de processus de calcul travaillant conjointement sur un ensemble de machines, à un extrême très concret, très proche du monde réel, lorsqu'il s'agit de simulateurs d'environnements tridimensionnels incluant par exemple des calculs de mécanique du solide pour l'évolution dynamique des objets. Plus l'environnement simulé est abstrait, plus les mécanismes adaptatifs que l'on prétend y appliquer doivent eux-mêmes être abstraits. Par exemple, nous verrons que l'on dispose de représentations mathématiques des mécanismes d'apprentissage par renforcement qui peuvent s'appliquer à des problèmes très éloignés de ceux auxquels sont confrontés les êtres vivants. Au contraire, plus l'environnement simulé est proche du monde réel, plus les mécanismes adaptatifs concrets tels qu'ils sont incarnés dans la nature peuvent s'appliquer directement. Par exemple, les organismes vivants ont inventé des solutions spécifiques pour résoudre des problèmes de localisation, d'orientation, ou de navigation. Les capacités correspondantes peuvent être cruciales pour diriger des personnages simulés dans un univers tridimensionnel du type des jeux vidéo, alors qu'elles n'ont pas de sens lorsqu'on s'intéresse au dialogue entre des processus logiciels distribuant des calculs au travers d'un réseau.

C'est donc plutôt du côté des simulateurs les plus réalistes que l'approche animat trouve les environnements les plus susceptibles de contribuer directement à la compréhension des capacités d'adaptation spécifiques des êtres vivants. Si l'on accepte de prendre en compte des environnements logiciels de degré de réalisme variable, alors le nombre de simulateurs disponibles est énorme. Les simulateurs de conduite ou de vol, les écosystèmes simulés, les simulateurs militaires, les jeux vidéo, voire les simulateurs génériques destinés à la réalisation rapide d'autres simulateurs sont légions. Tous ne sont pas immédiatement accessibles, tous ne se prêtent pas directement à l'introduction d'agents logiciels pour réaliser des expérimentations, mais ceux qui ont ces deux propriétés sont tout de même nombreux et leur nombre va croissant grâce à l'expansion d'internet.

Choisir de travailler dans de tels environnements procure donc de nombreux avantages, surtout lorsque l'effort de développement à fournir pour y intégrer ses propres agents est réduit. D'une part, le coût en temps et en argent pour réaliser des expérimentations dans de tels environnements est en général considérablement plus faible qu'en robotique. D'autre part, ces simulateurs sont conçus la plupart du temps pour répondre à un besoin applicatif précis, si bien que développer des agents dotés de capacités d'adaptation pour ces environnements permet de contribuer directement à répondre à ce besoin, ce qui n'est en général pas le cas des simulations *ad hoc* développées à des fins de recherche, ni des nombreuses expériences de robotique mobile menées dans les laboratoires.

Si notre projet de recherche consiste donc au plan fondamental à contribuer à la compréhension des mécanismes et comportements adaptatifs qui opèrent dans le vivant, pour nous donner les moyens de cette contribution, nous nous focalisons au plan applicatif sur la mise au point de systèmes adaptatifs capables d'évoluer dans des environnements simulés de grande envergure. Si nous aurons parfois recours à des environnements *ad hoc* à des fins de mise au point et s'il ne nous paraît pas judicieux d'écarter des domaines applicatifs abstraits tels que l'« *Autonomic Computing* [8] » ou moins accessibles tels que les simulations militaires, c'est sans doute dans le cadre des jeux vidéo que nous trouvons le meilleur compromis entre réalisme, accessibilité, complexité des environnements et facilité de mise en œuvre. Ce domaine nous fournira donc nos applications privilégiées pour la conduite de notre projet de recherche.

[8] [IBM Corporation, 2003]

1.1.5 Un domaine d'application : les jeux vidéo

L'industrie du jeu vidéo a connu ces dernières années une croissance très rapide, pour atteindre un budget qui dépasse au niveau mondial celui de l'industrie du cinéma. Dans un premier temps, l'augmentation de la puissance de calcul des machines a permis de produire des jeux qui immergent les joueurs dans des univers tridimensionnels au graphisme de plus en plus époustouflant. Dans un second temps, elle permet l'intégration de techniques d'intelligence artificielle au sens large pour définir le comportement des adversaires des joueurs, ou celui des créatures qui peuplent les mondes dans lesquels ils évoluent. Pour un état de l'art de l'intégration des techniques d'intelligence artificielle dans les jeux vidéo, nous renvoyons aux travaux de thèse de G. Robert [9]. Dans le cadre de ce document, nous allons nous contenter d'évoquer trois catégories de jeux qui nous intéressent particulièrement en raison des problèmes scientifiques qu'ils posent.

Tout d'abord, les jeux de stratégie en temps réel ou RTS [10] opposent généralement plusieurs armées dans un univers bidimensionnel. Comme dans les *wargames*, les joueurs doivent attribuer des actions à leurs diverses unités, mais le caractère temps réel se traduit par le fait qu'il n'y a pas de notion de « tour de jeu » pendant lequel le joueur déplace toutes les unités qu'il veut. Il faut agir vite pour ne pas se trouver débordé par l'ennemi. Les jeux de stratégie en temps réel offrent un cadre particulièrement adapté pour des travaux ayant trait à la coordination entre plusieurs agents. Par contre, l'environnement de jeu ne possède pas le degré de réalisme des jeux impliquant un univers tridimensionnel. Le jeu « *Stratagus* », version libre de droit et gratuite du jeu *Warcraft* [11], est un jeu de stratégie en temps réel qui nous sert de base d'expérimentation dans le cadre du projet SMAARTS financé par le Laboratoire d'Informatique de Paris 6 (LIP6).

Une seconde catégorie de jeux, les jeux de tir en première personne ou FPS [12], fournit des environnements de simulation beaucoup plus proches du monde réel. Dans ces jeux, un joueur ou une équipe de joueurs doivent affronter une équipe adverse vis-à-vis d'un objectif qui peut être simplement de survivre ou bien de remplir une mission clairement délimitée dans l'espace et le temps (se rendre maître d'un bâtiment ou du drapeau adverse, libérer des otages, empêcher le passage d'un convoi, etc.). L'univers de jeu est restreint, les objectifs et actions possibles sont limités, si bien que cette catégorie de jeux constitue, pour tester nos techniques, un excellent compromis entre la catégorie précédente, trop abstraite et la catégorie suivante, trop ouverte. En outre, bien qu'appartenant tous à des sociétés privées, ces jeux offrent un moyen d'accès naturel pour immerger des agents adaptatifs dans les univers qu'ils proposent. Dans la plupart de ces jeux, en effet, il est possible d'ajouter à l'un ou l'autre camp des personnages pilotés par l'ordinateur appelés « bots » et il existe des solutions techniques fournies sur internet pour programmer soi-même le comportement de ces bots.

La plupart des sociétés qui développent ces jeux commencent par construire un moteur générique de fonctionnement du jeu, puis greffent sur ce moteur générique différentes extension [13] qui constituent autant de jeux différents. Les trois principaux moteurs sont *Unreal Tournament* [14], *Half-Life* [15], et *Quake* [16]. À *Unreal Tournament*, il existe différents modes de jeu : un simple combat individuel pour la survie [17], un jeu en équipe où il s'agit de capturer le

[9] [Robert, 2002]
[10] *Real Time Strategies*
[11] ©*Blizzard North*
[12] *First Person Shooters*
[13] Appelées « mods » par les spécialistes
[14] ©*Epic Games*
[15] ©*Valve Software*
[16] ©*ID Software*
[17] *Death Match*

drapeau adverse [18], un jeu où il s'agit de prendre le contrôle de zones occupées par l'ennemi [19], un jeu dans lequel il faut réaliser une série d'objectifs [20] ou bien un combat pour la survie en équipe [21].

Sur le moteur de *Half-Life*, de très nombreuses extensions ont été développées, la plus populaire étant sans conteste *Counter Strike*, dans lequel une équipe de terroristes et une équipe de contre-terroristes s'affrontent, soit pour poser une bombe à un endroit prédéfini ou en empêcher l'équipe adverse, soit pour libérer des otages tenus par l'équipe adverse. Ces parties s'effectuent en équipe au travers du réseau internet. Il est fréquent de rajouter parmi les participants à une partie des « bots » écrits par des joueurs passionnés qui mettent leur code sur internet à disposition de toute la communauté des joueurs. Certains bots atteignent un degré de sophistication tout à fait remarquable qui permet aux joueurs de faire des parties en leur compagnie sans se lasser trop rapidement du caractère répétitif de leur comportement.

Nous verrons que G. Robert utilise une autre extension de *Half-Life* dénommée « *Team Fortress* », qui n'est rien d'autre que le *Capture the Flag* de *Unreal Tournament* transposé sur *Half-Life*. À noter aussi le succès grandissant d'un jeu un peu plus complexe, *Enemy Territory* [22], qui repose sur le moteur de *Quake3*. La particularité de ce jeu est que les personnages incarnés par les joueurs ont des spécialités et des rôles différents les uns des autres, ce qui ouvre la porte à des problématiques de coordination plus riches. Nous disposons donc avec les jeux de tir en première personne d'une grande variété de cadres expérimentaux pour tester nos techniques adaptatives. Il faut noter toutefois que, comme le signale avec raison D. Aha [23], les jeux de tir en première personne présentent le défaut vis-à-vis des jeux de stratégie en temps réel de demander avant tout aux agents de disposer de bonnes capacités de réaction en temps réel, plutôt que des capacités à élaborer des stratégies complexes pour obtenir un avantage significatif sur le camp ennemi.

Enfin, les jeux de rôle en ligne, persistants et massivement multi-joueurs constituent une troisième catégorie. Ces jeux se distinguent des précédents par le fait qu'ils fournissent des univers très vastes, dans lesquels les actions possibles sont nettement plus variées et les objectifs beaucoup plus diversifiés. En outre, la personnalité des personnages est prise en compte de façon moins sommaire, chaque personnage étant souvent doté de caractéristiques physiques, psychologiques et intellectuelles qui lui sont propres. Par rapport aux jeux de tir en première personne, ces jeux posent donc des problèmes sensiblement plus proches de ce qu'un être vivant est susceptible de rencontrer dans le monde réel. En conséquence, ils devraient fournir à terme un cadre expérimental plus adapté que les catégories précédentes à notre objectif fondamental de modélisation de l'intelligence des êtres vivants.

Cependant, l'immaturité actuelle des techniques adaptatives auxquelles nous nous intéressons font que tous les problèmes supplémentaires que posent ces jeux sont des obstacles difficiles à surmonter pour l'instant. En outre, ces jeux sont encore peu nombreux et, à notre connaissance, il n'en existe pas au moment où nous écrivons ce document qui fournisse un cadre technique simple pour programmer soi-même le comportement des créatures qui le peuplent.

1.2 Programme de recherche

Comme la plupart des domaines mettant en jeu des environnements simulés, le domaine des jeux vidéo est vaste et propose au chercheur intéressé par les comportements adaptatifs une gamme de problèmes très différents les uns

[18] *Capture the Flag*
[19] *Domination*
[20] *Assault*
[21] *Team Death Match*
[22] ©*ID Software*
[23] [Aha & Molineaux, 2004]

des autres. Certains jeux se jouent dans des mondes découpés en case, d'autres sont dotés d'une représentation continue de l'espace. Certains mettent en présence deux adversaires dont le premier est piloté par le joueur humain et le second par l'ordinateur, d'autres mettent en présence de très nombreux joueurs et créatures dirigées par l'ordinateur [24]. Certains supposent la réalisation d'un objectif précis et très limité dans le temps et d'autres confrontent les joueurs à des problèmes beaucoup plus ouverts. Certains sont fondés sur des stratégies individuelles et d'autres mettent en présence des équipes de personnages. La liste des dimensions possibles ne s'arrête pas là, mais on trouve déjà dans cette énumération une richesse suffisante pour avoir une idée de la diversité des problématiques auxquelles le chercheur intéressé par de telles applications peut se confronter.

Face à une telle diversité, il est toujours possible de trouver dans ce vaste domaine une application « sur mesure » à laquelle des mécanismes adaptatifs spécifiques offrent une réponse adéquate. Afin de n'être pas confronté d'emblée à toutes les difficultés à la fois, on peut ainsi choisir de traiter d'un côté une application dans laquelle le monde est découpé en case mais les agents sont nombreux et, de l'autre, une application continue impliquant la mise au point de mécanismes de navigation opportunistes, mais dans laquelle l'agent doit satisfaire un objectif individuel. Cette stratégie permet de traiter séparément divers aspects des problèmes d'adaptation auquel un agent peut être confronté. Le lecteur verra que c'est cette stratégie que nous avons prudemment adoptée jusqu'à présent.

Cependant, dans le cadre de l'élaboration d'un programme de recherche, il est judicieux d'adopter une attitude ambitieuse. Cela consiste à nous fixer comme idéal à poursuivre la mise au point de mécanismes capables d'adapter le comportement des agents dans des environnements présentant simultanément toutes les difficultés énumérées ci-dessus. Outre l'ambition qu'elle nous permet d'afficher, cette démarche est destinée à nous permettre de nous tenir au plus près de nos objectifs les plus fondamentaux, à savoir la compréhension des capacités d'adaptation des êtres vivants dans leur milieu naturel.

Un examen des applications potentielles et notre connaissance des techniques adaptatives nous conduit à décomposer notre objectif global sous la forme d'une liste de capacités que devront exhiber les agents pour que cet objectif soit atteint :

- la capacité à évoluer intelligemment dans des univers de grande taille dans lesquels les perceptions et les actions possibles sont nombreuses ;
- la capacité à adapter son comportement dans le cadre d'environnements continus, dans lesquels la perception et l'action peuvent être continues ;
- la capacité à se localiser, à s'orienter, à naviguer dans son environnement en l'absence de référentiel absolu et d'informations externes sur sa position ;
- la capacité à gérer l'incertitude sur des informations importantes, à cause des limites de la perception et de la connaissance des agents ;
- la capacité à faire face à un environnement dynamique dans lequel divers agents évoluent simultanément et sont susceptibles de modifier l'environnement et leurs propres stratégies au cours du temps ;
- la capacité à anticiper les comportements des agents pilotés par les êtres humains, et des autres agents logiciels ;
- la capacité à se coordonner à la fois dans l'espace, le temps et le choix des actions, afin d'atteindre des objectifs collectifs.

Face à chacune de ces problématiques, il faudra être capable de mettre en œuvre un ensemble de mécanismes adaptatifs permettant aux agents d'élaborer des comportements adaptés. Nous allons voir dans le cadre de ce mémoire que l'ensemble des mécanismes adaptatifs étudiés dans les laboratoires de recherche est loin d'offrir des réponses satisfaisantes à la totalité de ces problèmes.

[24] *Non-player characters* ou NPC

L'ambition de notre programme de recherche consiste à contribuer à repousser ces frontières dans plusieurs directions à la fois. Dans ce but, le cœur de notre démarche consiste bien sûr à participer au développement de diverses techniques adaptatives répondant à certaines des problématiques spécifiques que nous avons énumérées, mais surtout à rechercher un cadre général visant l'intégration de ces techniques en un tout aussi cohérent que possible.

Ce travail s'effectue principalement par questionnement et modification des cadres formels déjà bien établis au sein desquels se sont développées les techniques adaptatives auxquelles nous nous intéressons. Conscient des limites de cette démarche, nous reviendrons à la fin de ce mémoire sur le fait que, puisque les êtres vivants sont dotés de toutes les capacités adaptatives que nous nous efforçons de rassembler, il apparaîtra nécessaire, lorsque des limites seront atteintes, de nous tourner à nouveau vers le vivant de façon approfondie pour nous efforcer de mieux comprendre d'où viennent les capacités que nous ne parvenons pas à reproduire.

1.3 Plan du mémoire

L'ensemble du mémoire est conçu comme une série de pistes de recherche complémentaires que nous suivons pour atteindre la liste d'objectifs énoncés ci-dessus. Ces pistes sont organisées dans l'ordre chronologique des travaux que nous avons encadrés. Elles sont reliées entre elles par la mise en évidence des limitations qui nous ont conduit à passer d'un thème à l'autre pour contourner divers obstacles conceptuels. De cette organisation découle le découpage ci-dessous. Les notions en italiques seront introduites dans les chapitres correspondants.

Le chapitre 2 se résume à une présentation didactique des fondements théoriques de l'*apprentissage par renforcement*. Destiné à préparer le lecteur aux nombreux développements que nous consacrerons à ce thème, il ne contient la mention d'aucun de nos travaux personnels.

Le chapitre 3 porte sur les *systèmes de classeurs*. Outre une présentation générale du domaine, il fait une large place aux travaux de thèse de P. Gérard sur les *systèmes de classeurs à anticipation*. En particulier, nous donnons une présentation détaillée du logiciel MACS, dans le but de faire ressortir la complexité et le caractère parfois *ad hoc* de certaines heuristiques.

Le chapitre 4 fait état de nos travaux récents dans le cadre des *processus de décision markoviens factorisés*. L'exposition de ce cadre nous permet de formaliser de façon plus satisfaisante une partie de nos recherches sur les systèmes de classeurs à anticipation. Nous présentons des expériences préliminaires réalisées par C. Lutz dans le cadre de son stage de DEA et mettons en évidence les nombreux développements auxquels cette piste devrait nous conduire. Ce cadre formel est destiné à jouer un rôle central dans le travail d'intégration auquel nous allons nous livrer en conclusion de ce mémoire.

Le chapitre 5 présente des travaux effectués en commun avec S. Landau dans le cadre des *processus de décision markoviens partiellement observables*, avec une approche fondée exclusivement sur des algorithmes évolutionnistes. Ce chapitre permet de faire état des difficultés liées à l'observabilité partielle, et prépare aux développements que nous mènerons au chapitre 8 pour dépasser les problèmes de lenteur de convergence liés aux approches évolutionnistes.

Le chapitre 6 est consacré à la thèse de F. Flacher, qui porte sur la *coordination spatiale* de plusieurs agents dans des environnements continus. F. Flacher utilise lui aussi des méthodes évolutionnistes, qui ont la particularité de s'appliquer à des problèmes *multi-critères*. Nous présentons quelques résultats expérimentaux significatifs et discutons de la nécessité d'abandonner une approche strictement évolutionniste au profit d'une approche impliquant des mécanismes d'apprentissage.

Le chapitre 7 propose brièvement une extension du travail mené par F. Flacher au cas où les agents sont dotés

de *variables essentielles* dont la valeur doit être régulée par leur comportement. Nous insistons sur la nécessité d'intégrer cette thématique dans notre plan de recherche à moyen terme.

Au chapitre 8, nous proposons une première perspective intégratrice avec une approche que nous appelons « *apprentissage par population* ». Cette approche, déjà présente dans le cadre des systèmes de classeurs, permet de combiner efficacement algorithmes évolutionnistes et apprentissage par renforcement. Nous faisons état des nouveaux développements du travail mené en commun avec S. Landau dans cette perspective, puis nous montrons comment elle pourrait s'appliquer aux problèmes de coordination spatiale traités par F. Flacher.

Au chapitre 9, nous présentons les derniers développements correspondant aux thèses respectives de T. Gourdin et T. Degris. Le premier s'efforce de traiter avec des techniques d'apprentissage par renforcement les problèmes que F. Flacher traitait avec des méthodes évolutionnistes, mais il le fait avec une approche qui s'éloigne sensiblement de l'apprentissage par population. Cela le conduit à résoudre des problèmes d'apprentissage par renforcement en environnement continu. Le second s'attaque à des problématiques générales d'apprentissage par renforcement dans un cadre multi-agents. Les deux problématiques abordées dans ce chapitre, le continu et le multi-agents, sont considérées comme les principaux points durs sur lesquels bute la recherche visant à étendre le domaine d'application des techniques d'apprentissage par renforcement.

Dans le dixième et dernier chapitre, destiné à conclure ce mémoire, nous faisons le bilan des travaux qui restent à mener pour intégrer les développements du chapitre précédent dans une perspective plus générale. Après avoir examiné toutes les techniques adaptatives avec un regard essentiellement théorique, nous nous tournons enfin vers le domaine d'application privilégié que nous visons, afin de montrer comment ce que nous avons présenté peut contribuer à résoudre les problématiques applicatives que nous avons énumérées. Nous concluons sur la nécessité de nous tourner à nouveau vers le vivant pour y trouver d'autres sources d'inspiration destinées à dépasser les limites que nous avons rencontrées.

Chapitre 2

Apprentissage par renforcement

> « Je vais vous montrer comment marche un vrai cerveau. Avec l'algo. »
>
> P. Pirim, le 26/112002

Le premier mécanisme adaptatif auquel nous allons nous intéresser est l'apprentissage par renforcement. L'objet de ce chapitre est de présenter le cadre théorique général de l'apprentissage par renforcement et de mettre en évidence certaines limites de ce cadre dont nous chercherons à nous affranchir par la suite.

En guise d'introduction, nous commençons par replacer l'apprentissage par renforcement dans le contexte qui lui a donné naissance, à savoir les recherches menées par la psychologie comportementaliste sur l'apprentissage du comportement animal. Nous présentons ensuite le cadre général des modèles de Markov, qui permettent de formaliser les phénomènes observés empiriquement par les psychologues.

Une fois ce cadre posé, nous exposons les principaux algorithmes de planification et d'apprentissage qui s'appliquent dans ce cadre, puis nous faisons succinctement état des relations entre ces travaux théoriques et les progrès de la modélisation neurobiologique de l'apprentissage animal. Enfin, nous énumérons les limites de l'apprentissage par renforcement que les recherches présentées dans ce mémoire visent à dépasser.

2.1 Origines psychologiques

2.1.1 Modèles associationnistes de l'apprentissage

L'apprentissage d'associations élémentaires entre des stimuli successifs ou entre des stimuli et des réponses est un facteur déterminant de la structuration de la vie cognitive. Par la formation d'associations, des situations initialement neutres pour un sujet se voient peu à peu attribuer une valeur en fonction de rapports de contingence qu'elles entretiennent avec d'autres situations qui ont une signification vitale pour lui, parce qu'elles jouent un rôle déterminant dans la satisfaction de ses besoins élémentaires. On assiste ainsi à une complication du comportement du sujet, complication qui reflète la structuration des représentations qu'il élabore au fil de ses interactions avec son environnement. De loin en loin, les valeurs associées à différentes situations se propagent au fur et à mesure de l'apprentissage par le sujet de nouvelles régularités dans son environnement, régularités qui s'expriment sous la forme de contingences statistiques. Parallèlement, des processus de catégorisation fondés sur ces valeurs contribuent à la formation de représentations plus structurées du monde. La modélisation des phénomènes d'apprentissage

d'associations est donc d'une importance théorique considérable pour la recherche en psychologie, puisque c'est la structuration de la vie cognitive qu'il s'agit d'expliquer.

Tous les phénomènes d'apprentissage d'associations se manifestent par des modifications de la probabilité de déclenchement d'une réponse à la suite de la présentation d'un stimulus ou, plus généralement, d'une situation perceptive. Les recherches empiriques sur ces modèles s'efforcent donc d'expliquer l'établissement de corrélations statistiques entre la présentation de stimuli ou de situations et la manifestation de comportements spécifiques.

On distingue en psychologie animale le conditionnement classique de I. Pavlov [1], le conditionnement opérant ou instrumental de B. Skinner [2], l'apprentissage par essais et erreurs de E. Thorndike [3] et l'apprentissage latent de E. Tolman [4]. Ces modèles de l'apprentissage rendent compte de la façon dont une association entre des stimuli et éventuellement des réponses ou des récompenses peut être renforcée. Alors que les premiers modèles relèvent au sens large de la tendance béhavioriste en psychologie, le dernier, bien que centré lui aussi sur l'étude des comportements, est considéré comme l'un des précurseurs du cognitivisme. Nous le traiterons à part au chapitre 3.

2.1.2 Le conditionnement classique

Ayant été mis en évidence par I. Pavlov [5], le phénomène de conditionnement classique est aussi appelé conditionnement pavlovien. À la naissance, le comportement des sujets est essentiellement constitué de réflexes. Les réflexes sont des comportements déclenchés en réponse à la présentation de stimuli très spécifiques, qui sont des événements dotés d'une signification biologique pour l'individu. Le conditionnement classique est le mécanisme par lequel ces mêmes réflexes peuvent se trouver associés à un répertoire plus large de stimuli. Pour mettre en évidence le phénomène de conditionnement pavlovien, le protocole expérimental consiste à présenter régulièrement un stimulus initialement neutre pour l'individu, c'est-à-dire n'induisant pas de réponse spécifique, de telle façon qu'il précède un second stimulus qui induit un réflexe. On constate qu'au bout d'un certain nombre d'expositions à une succession des deux stimuli, le premier stimulus, appelé « stimulus conditionnel », agit comme un signal annonçant le second stimulus, dit « inconditionnel », qui déclenche systématiquement la réponse. Une fois l'association entre les deux stimuli formée, la présentation du stimulus conditionnel suffit à déclencher le réflexe, alors même que le second stimulus n'a pas été présenté.

Le conditionnement classique opère donc uniquement sur des réflexes. Il permet d'expliquer l'extension du domaine de déclenchement de ces réflexes aux phénomènes qui annoncent de façon fiable les stimuli responsables de leur déclenchement.

Il s'agit en fin de compte de l'établissement d'associations entre des stimuli. En effet, plutôt que l'association entre des stimuli et des réponses, c'est l'association entre divers stimuli qui est mise en évidence, la réponse étant toujours associée au même stimulus inconditionnel. Par le conditionnement classique, les animaux apprennent la structure causale de leur environnement, indépendamment de leur action dans cet environnement, ce qui constitue une forme primitive de modèle du monde.

[1] [Pavlov, 1927]
[2] [Skinner, 1938]
[3] [Thorndike, 1911]
[4] [Tolman, 1932]
[5] [Pavlov, 1927]

2.1.3 Le conditionnement opérant

Mis au point par B. Skinner [6], le conditionnement opérant ou instrumental repose sur une procédure expérimentale distincte de celle du conditionnement classique. Il s'agit cette fois d'étudier l'apprentissage de relations entre des réponses ou des comportements volontaires et des conséquences de ces réponses ou comportements volontaires. On observe alors comment les organismes modifient leur comportement futur en fonction des conséquences de leur comportement passé.

Au lieu d'associer deux stimuli, la procédure fait en sorte que l'organisme associe une action à ses conséquences. Comme, en général, les conséquences d'une action dépendent de la situation dans laquelle elle a été initiée, l'association porte en fait sur trois termes, un stimulus initial, une action et un stimulus résultant.

Les conséquences des comportements étudiés par le conditionnement opérant sont classées en deux catégories : celles qui induisent une émission plus fréquente du comportement et celles qui induisent une émission moins fréquente. Les premières sont appelées des *renforcements*. On distingue d'une part les renforcements *positifs*, quand l'individu est récompensé en étant soumis à un stimulus qui lui est agréable et, d'autre part, les renforcements *négatifs*, quand l'individu cesse d'être soumis à un stimulus qui lui est désagréable. Les conséquences qui conduisent à une émission moins fréquente du comportement sont appelées des *punitions*. On distingue là aussi des punitions positives, quand on soumet le sujet à un stimulus pour lequel il éprouve une aversion, et les punitions négatives, qui consistent en la suppression d'un renforcement attendu. Le lecteur désireux d'en savoir plus trouvera dans la thèse de J. Jozefowiez [7] une présentation beaucoup plus détaillée du conditionnement opérant et des divers protocoles expérimentaux qui sont employés dans ce cadre.

2.1.4 L'apprentissage par essais et erreurs

La place de l'apprentissage « par essais et erreurs » d'E. Thorndike [8] au sein des théories associationnistes de l'apprentissage animal n'est pas clairement élucidée dans les ouvrages que nous avons consultés. Cette approche joue pourtant un rôle important pour nous car R. Sutton et A. Barto [9] rappellent comment les fondements théoriques de l'apprentissage par renforcement ont été essentiellement inspirés par cette théorie psychologique. Ils citent en particulier le passage suivant :

> « Parmi plusieurs réponses à une même situation, celles qui sont accompagnées ou suivies immédiatement d'une satisfaction pour l'animal seront, toutes choses égales par ailleurs, connectées plus fermement à la situation si bien que, quand celle-ci se reproduit, ces réponses auront plus de chances de se reproduire ; celles qui sont accompagnées ou suivies immédiatement d'une insatisfaction pour l'animal auront, toutes choses égales par ailleurs, leur connexion à la situation affaiblie si bien que, quand celle-ci se reproduit, ces réponses auront moins de chances de se reproduire. Plus la satisfaction ou l'insatisfaction est grande, plus le lien est renforcé ou affaibli. [a] » in [Thorndike, 1911] p. 244

[a] Of several responses made to the same situation, those which are accompanied or closely followed by satisfaction to the animal will, other things being equal, be more firmly connected with the situation, so that, when it recurs, they will be more likely to recur ; those which are accompanied or closely followed by discomfort to the animal will, other things being equal, have their connections with that situation weakened, so that, when it recurs, they will be less likely to recur. The greater the satisfaction or discomfort, the greater the strengthening or weakening of the bond.

Faute de pouvoir établir une filiation claire entre les écrits d'E. Thorndike et les travaux béhavioristes, disons simplement que les premiers sont antérieurs à la constitution du behaviorisme strict élaboré par J. Watson et B. Skinner.

[6] [Skinner, 1938]
[7] [Jozefowiez, 2001]
[8] [Thorndike, 1911]
[9] [Sutton & Barto, 1998]

Il faut retenir de cette présentation que les sources de l'apprentissage par renforcement dont nous allons exposer la formalisation théorique dans les sections suivantes sont d'origine psychologiques et non biologiques. Nous verrons à la fin de ce chapitre que les liens avec la modélisation neurobiologique se sont établis plus tard, une fois la théorie mathématique de l'apprentissage par renforcement bien développée, sans doute parce que la formalisation a permis de savoir quels étaient les mécanismes à rechercher dans un substrat plus difficile à interroger que le comportement directement observable.

2.1.5 Vers la modélisation informatique

La présentation adoptée dans la suite de ce chapitre va faire l'objet d'une reconstruction *a posteriori* qui ne rend pas compte de l'enchaînement historique des idées. Avec un souci de clarté des concepts, notre présentation s'appuiera très directement sur l'ouvrage de R. Sutton et A. Barto [10], qui constitue une synthèse d'une qualité telle qu'il est difficile de s'en démarquer. Cependant, avant d'en venir au cœur de cette présentation, nous donnons un bref aperçu de la succession des étapes qui ont conduit à la formulation actuelle des modèles théoriques de l'apprentissage par renforcement, en nous focalisant sur le développement de modèles informatiques.

Les premiers travaux en informatique représentatifs du cadre de l'apprentissage par renforcement datent approximativement de 1960. En 1961, D. Michie [11] décrit un système capable d'apprendre à jouer au morpion par essais et erreurs. Puis D. Michie et R. Chambers [12] écrivent en 1968 un programme capable d'apprendre à maintenir un pendule inversé à l'équilibre. Parallèlement, A. Samuel [13] réalise un logiciel qui apprend à jouer aux dames en utilisant une notion de différence temporelle. Les deux composantes, exploration par essais et erreurs et gestion de séquences d'action par différences temporelles vont être au cœur des modèles ultérieurs. La synthèse entre les deux courants est réalisée par H. Klopf [14]. Dans la lignée de ces travaux, les principaux acteurs du développement de l'apprentissage par renforcement en informatique, R. Sutton et A. Barto, implémentent en 1981 [15] un perceptron linéaire dont l'équation de mise à jour dérive directement des théories développées en psychologie expérimentale par R. Rescorla et A. Wagner [16]. J. Jozefowiez fait très clairement apparaître dans sa thèse [17] que l'équation de Rescorla-Wagner, qui est d'une importance considérable pour les modèles de l'apprentissage animal, n'est rien d'autre que la version approximée par un perceptron linéaire de l'algorithme $TD(0)$ présenté à la section 2.4.1, page 41. Cette équivalence explique que le recours aux algorithmes d'apprentissage par renforcement se soit appelé dans un premier temps « programmation neurodynamique » [18].

Sur ces bases, R. Sutton et A. Barto proposent en 1983 avec C. Anderson [19] une méthode, AHC-LEARNING [20], qui est considérée comme le véritable point de départ des travaux qui font l'objet de ce chapitre. De façon intéressante, AHC-LEARNING repose sur une architecture acteur-critique dont nous verrons à la section 2.4.5, page 44, qu'elle est au cœur du dialogue qui s'instaure à présent entre la modélisation informatique et la modélisation neurophysiologique de l'apprentissage par renforcement chez l'animal.

La formalisation mathématique des algorithmes d'apprentissage par renforcement telle que nous la connaissons

[10] [Sutton & Barto, 1998]
[11] [Michie, 1961]
[12] [Michie & Chambers, 1968]
[13] [Samuel, 1959]
[14] [Klopf, 1972, Klopf, 1975]
[15] [Sutton & Barto, 1981]
[16] [Rescorla & Wagner, 1972]
[17] [Jozefowiez, 2001]
[18] [Bertsekas & Tsitsiklis, 1996]
[19] [Barto et al., 1983]
[20] *Adaptive Heuristic Critic learning*

aujourd'hui s'est développée à partir de 1988 lorsque R. Sutton [21] puis C. Watkins [22] ont fait le lien entre leurs travaux et le cadre théorique de la commande optimale proposée par R. Bellman en 1957 avec la notion de processus de décision markovien stochastique [23]. Il va apparaître dans la suite que ce cadre théorique est à présent le socle sur lequel s'appuient tous les travaux théoriques relatifs à l'apprentissage par renforcement.

2.2 Les modèles de Markov

2.2.1 Une famille de modèles

Comme l'indique de façon très pédagogique J. Jozefowiez dans sa thèse [24], les modèles de Markov constituent un cadre mathématique très général parfaitement apte à exprimer formellement ces corrélations statistiques. On distingue en particulier les *chaînes de Markov*, les *chaînes de Markov cachées*, les *processus de décision markoviens* [25] et les *processus de décision markoviens partiellement observables*.

Les deux premiers modèles permettent d'exprimer des corrélations statistiques entre des états qui se présentent successivement au cours du temps dans un processus observé de l'extérieur. Les deux suivants ajoutent à ces modèles la prise en compte de l'action qui permet à un agent de passer d'un état à un autre. On considère alors que c'est l'agent qui observe la dynamique de ses propres interactions avec son environnement.

Dans les chaînes de Markov, l'observateur, qui doit établir un modèle des corrélations entre les états, dispose de toute l'information sur ces états. Dans les chaînes de Markov cachées, au contraire, l'observateur ne dispose que d'une information partielle sur les états réels par lesquels passe le système qu'il observe. Cette information partielle lui est fournie au travers d'une *fonction d'observation* [26].

De même, dans les processus de décision markoviens, l'agent connaît parfaitement les états par lesquels il passe au fil de son action, tandis que, dans les processus de décision markoviens partiellement observables, nous verrons au chapitre 5 qu'il n'a accès à cet état qu'au travers d'une fonction d'observation.

Les processus de décision markoviens partiellement observables sont donc aux processus de décision markoviens ce que les chaînes de Markov cachées sont aux chaînes de Markov.

Les chaînes de Markov et les chaînes de Markov cachées constituent le cadre adéquat pour rendre compte de l'apprentissage de la relation entre stimuli ne faisant pas intervenir l'action du sujet. Il s'agit donc de modèles appropriés pour étudier le conditionnement pavlovien. Pour étudier le conditionnement opérant, qui fait jouer un rôle central à un sujet, il vaut mieux se tourner vers les processus de décision markoviens et les processus de décision markoviens partiellement observables [27]. Dans la suite de ce mémoire, nous allons nous consacrer exclusivement à ces deux derniers modèles qui prennent en compte l'action des agents.

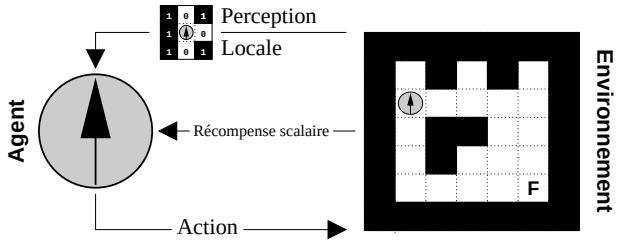


FIG. 2.1 – **Un problème d'apprentissage par renforcement met en scène un agent en interaction avec un environnement. L'agent reçoit de son environnement un ensemble de perceptions qui déterminent son état. Il décide en retour d'une action qui modifie sa situation dans l'environnement, donc son nouvel état. En outre, l'agent reçoit de l'environnement des récompenses lorsqu'il est dans une bonne situation et des punitions lorsqu'il est dans une mauvaise situation. Pour un agent donné, l'objectif assigné par un problème d'apprentissage par renforcement consiste en général à maximiser globalement la récompense sur un parcours éventuellement infini ou sur un ensemble de parcours finis.**

2.2.2 Les processus de décision markoviens

Comme nous venons de l'indiquer, les problèmes mettant en jeu un agent qui doit adopter un certain comportement dans un certain environnement se formalisent dans le cadre mathématique des *processus de décision markoviens*. Un processus de décision markovien se décrit par la donnée des éléments suivants :

- un ensemble fini S d'états discrets notés s ;
- un ensemble fini A d'actions discrètes notées a ;
- une fonction de transition $T : S \times A \rightarrow \Pi(S)$ où $\Pi(S)$ est l'ensemble des distributions de probabilité sur S. Une distribution de probabilité particulière $p(S)$ indique pour tout couple (s^t, a^t) la probabilité pour que l'agent se retrouve dans chacun des états s^{t+1} possibles quand il fait l'action a^t dans l'état s^t ;
- une fonction de renforcement $R : S \times A \rightarrow \mathbb{R}$ qui indique pour tout couple *(état, action)*, sous forme d'un nombre réel, le signal de renforcement (punition ou récompense) que reçoit l'agent quand il fait cette action dans cet état.

Il faut noter que le cadre déterministe est un cas particulier du cadre stochastique dans lequel, pour tout couple (s^t, a^t), la probabilité d'être dans l'un des états suivants s^{t+1} vaut 1 pour un état particulier et 0 pour tous les autres. Dans le cadre stochastique, on note usuellement $T(s^t, a^t)(s^{t+1})$ la probabilité d'atteindre l'état s^{t+1} lorsqu'on réalise l'action a^t dans l'état s^t. $T(s^t, a^t)(s^{t+1})$ est donc un nombre entre 0 et 1. Avec un regard probabiliste, que nous adopterons massivement au chapitre 4, on note aussi $Pr(s^{t+1}|s^t, a^t)$ la même information.

Le formalisme des processus de décision markoviens permet de décrire la structure d'un ensemble de problèmes auxquels est confronté un agent et ne préjuge en rien du choix des actions de l'agent au sein de son environnement. Il permet uniquement de déterminer quelle sera la situation future de l'agent selon qu'il effectue telle ou telle action

[21] [Sutton, 1988]
[22] [Watkins, 1989]
[23] [Bertsekas, 1995]
[24] [Jozefowiez, 2001]
[25] On parle aussi dans la communauté française de Processus Décisionnel de Markov, les deux termes étant des traductions équivalentes de *Markov Decision Process*.
[26] [Rabiner & Juang, 1986]
[27] J. Jozefowiez signale pourtant que, paradoxalement, c'est plutôt entre les travaux de psychologie sur le conditionnement pavlovien et les travaux sur les processus de décision markoviens que s'organise le dialogue, malgré l'affinité *a priori* plus grande de ce cadre formel avec le conditionnement opérant.

dans tel ou tel état et quelles récompenses il recevra en fonction de son comportement.

Les processus de décision markoviens définissent donc un cadre général pour représenter un problème auquel est susceptible d'être confronté un agent réalisant des actions discrètes dans un environnement discret. Dans ce cadre, rien n'a encore été défini pour décrire le comportement d'un agent particulier ni la capacité d'un agent à connaître son état.

En ce qui concerne la connaissance de l'état, le fait de définir la fonction de transition comme une distribution de probabilité implique une hypothèse forte sur la nature de l'état de l'agent. Cette hypothèse est connue sous le nom d'hypothèse de Markov. Elle stipule que la distribution de probabilité spécifiant l'état du système à l'instant $t+1$ (noté s^{t+1}) dépend uniquement de son état à l'instant t et de l'action de l'agent à l'instant t, mais pas du passé.

On a donc $Pr(s^{t+1}|s^t, a^t) = Pr(s^{t+1}|s^t, a^t, s^{t-1}, a^{t-1}, ...s^0, a^0)$ Pour un agent donné, dire que l'hypothèse de Markov est vérifiée revient à dire que la connaissance de son passé ne lui permet pas de déterminer plus précisément quel sera son état futur. Plus précisément, si l'hypothèse de Markov est vérifiée, une mémoire des actions réalisées précédemment ou des états parcourus ne permet pas de déterminer plus précisément avec quelle probabilité l'agent se trouvera dans chaque état possible à l'issue de son action.

Il apparaît donc que le cadre des processus de décision markoviens rend compte de situations dans lesquelles les agents dotés d'une mémoire de leur expérience passée ne disposent en théorie d'aucun avantage significatif. Il est donc fréquent dans un tel cadre de se restreindre à l'étude des agents dont l'organe de décision relie directement les perceptions aux actions, sans mettre en œuvre de variables internes susceptibles de leur apporter une mémoire du passé.

Nous indiquerons à la section 5.1, page 101, que si l'on fait l'hypothèse que l'agent n'a pas directement accès à une connaissance de son état et si l'état partiel auquel il a accès ne vérifie pas l'hypothèse de Markov, alors posséder une mémoire des perceptions passées peut devenir crucial pour préserver sa capacité à adopter un comportement optimal.

Définition d'une politique et de la fonction de valeur associée

On appelle politique d'un agent une fonction $\pi : S \rightarrow \Pi(A)$ qui indique pour tout état s^t la distribution de probabilité pour que l'agent choisisse les diverses actions $a \in A$ lorsqu'il se trouve dans l'état s^t. De même que pour la fonction de transition, la politique π étant définie de façon stochastique, $\pi(s)$ est une distribution de probabilité sur a^t et on note usuellement $\pi(s^t, a^t)$ la probabilité de réaliser l'action a^t dans l'état s^t lorsque l'on suit la politique π.

Une politique définit un comportement, elle indique pour tout état quel choix fera l'agent parmi toutes les actions qui se présentent à lui.

Dans la mesure où la réalisation de certaines actions dans certains états procure à l'agent une récompense, on peut associer à chaque politique une *valeur*. La fonction de valeur associée à une politique π indique pour tout état s^t la quantité de récompense que l'agent peut espérer recevoir sur le long terme s'il suit la politique π à partir de l'état s^t. Cette quantité se décompose en une récompense reçue immédiatement pour avoir accompli l'action stipulée par π dans l'état s^t, et d'autres récompenses reçues ultérieurement en continuant à se comporter conformément à la politique π.

La fonction de valeur $V^\pi(s^t)$ associe donc à chaque état s^t une mesure de la récompense cumulée qu'un agent recevra s'il suit la politique π à partir de l'état s^t. Si le comportement de l'agent est constitué d'une séquence finie d'actions, alors une bonne mesure de la récompense cumulée peut être simplement la somme des récompenses reçues. Par contre, si le comportement de l'agent est constitué d'une séquence infinie d'actions, alors la valeur associée à chaque état risque d'être infinie avec une telle mesure. On préfère donc choisir une définition de la

récompense cumulée dans laquelle les récompenses reçues sont d'autant moins prises en compte qu'on les recevra plus loin dans le temps.

Pour cela, on introduit un facteur multiplicatif γ, appelé « facteur d'actualisation [28] » et généralement compris entre 0 et 1. Le facteur γ mesure à quel point les récompenses ultérieures sont prises en compte dans le calcul de la récompense cumulée. On définit alors la récompense cumulée à l'instant t :

$$Rc_\pi(t) = \sum_{k=t}^{T_{max}} \gamma^{(k-t)} R_\pi(k)$$

où T_{max} peut être fini ou infini et $R_\pi(k)$ désigne la récompense immédiate reçue à l'instant k si l'on suit la politique π.

On voit que, si γ vaut 0, la récompense cumulée ne tient aucun compte du futur et, si γ vaut 1, on réalise une simple somme des récompenses immédiates reçues. On trouve dans la thèse de G. Laurent [29] une discussion de l'influence de γ sur les résultats des algorithmes d'apprentissage. L'effet du facteur d'actualisation γ sur les valeurs associées aux états est illustré sur la figure 2.2, dans le cas où γ=0.9.

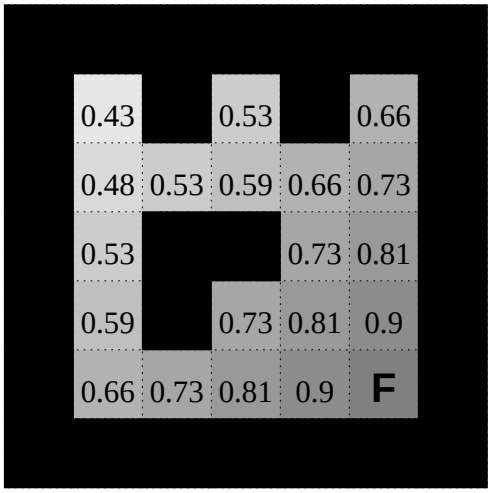


FIG. 2.2 – **Effet du facteur de dévaluation sur les valeurs optimales associées aux états. Ici, γ=0.9. On considère que l'agent reçoit une récompense qui vaut 1 lorsqu'il atteint l'état F et une récompense nulle partout ailleurs.**

Une fois le terme γ introduit, on peut définir la fonction de valeur en s^t pour la politique π récursivement, en se conformant exactement à ce que nous avons exprimé précédemment, à savoir que l'espérance de récompense en s^t, mesurée par $V(s)$, est la somme de la récompense $R(s^t, a^t)$ reçue immédiatement pour avoir accompli l'action a^t dans l'état s^t et de la récompense espérée par la suite si l'on continue à suivre la politique π. Il suffit de considérer que la récompense espérée dans les états successeurs est bien estimée par la valeur stockée en ces états. En prenant en compte les probabilités d'atteindre ces différents états, l'équation s'écrit :

$$\forall s \in S, V^\pi(s) = \sum_a \pi(s^t, a^t)[R(s^t, a^t) + \gamma \sum_{s^{t+1}} T(s^t, a^t)(s^{t+1}) V^\pi(s^{t+1})] \quad (2.1)$$

[28] *discount factor*
[29] [Laurent, 2002]

L'équation 2.1 est appelée équation de Bellman pour la politique π. Cette équation joue un rôle fondamental au cœur de toutes les méthodes d'optimisation qui permettent de définir des algorithmes d'apprentissage par renforcement.

Plutôt que de travailler sur la valeur des états, on peut associer à chaque couple (s^t, a^t) une *qualité* $Q^\pi(s^t, a^t)$ qui rend compte de l'espérance de renforcement sur le long terme si l'agent effectue l'action a^t dans l'état s^t puis se conforme à la politique π. De même que pour V, on peut écrire :

$$\forall s \in S, \forall a \in A, Q^\pi(s^t, a^t) = R(s^t, a^t) + \gamma \sum_{s^{t+1}} T(s^t, a^t)(s^{t+1}) V^\pi(s^{t+1}) \tag{2.2}$$

$V^\pi(s)$ est alors vu comme la moyenne sur les différentes actions possibles des $Q^\pi(s^t, a^t)$. De 2.1 et 2.2 on tire en effet :

$$\forall s \in S, V^\pi(s) = \sum_a \pi(s^t, a^t) Q^\pi(s^t, a^t) \tag{2.3}$$

ce qui permet d'établir l'équivalence :

$$\forall s \in S, V^\pi(s) = Q^\pi(s, \pi(s)) \tag{2.4}$$

Définition de la fonction de valeur optimale

Dans un processus de décision markovien, le problème posé à un agent consiste généralement à trouver la politique qui lui permet de maximiser la récompense cumulée qu'il reçoit. Pour définir une telle politique optimale, il faut commencer par définir la fonction de valeur optimale pour un problème donné.

La fonction de valeur optimale indique pour tout état s^t la récompense cumulée optimale que l'agent peut espérer recevoir sur le long terme s'il suit une politique optimale à partir de l'état s^t. Cette fonction est donc indépendante de toute politique. On la note V^* et on a :

$$\forall s \in S, V^*(s) = \max_\pi V^\pi(s)$$

ou, si l'on travaille sur les qualités,

$$\forall s \in S, \forall a \in A, Q^*(s^t, a^t) = \max_\pi Q^\pi(s^t, a^t)$$

L'équation de Bellman est vérifiée par toute politique π et, en particulier, par une politique optimale [30]. Donc on a

$$\forall s \in S, V^*(s) = \max_\pi [\sum_a \pi(s^t, a^t)[R(s^t, a^t) + \gamma \sum_{s^{t+1}} T(s^t, a^t)(s^{t+1}) V^*(s^{t+1})]] \tag{2.5}$$

ou

$$\forall s \in S, V^*(s) = \max_{a \in A} [R(s^t, a^t) + \gamma \sum_{s^{t+1} \in S} T(s^t, a^t)(s^{t+1}) V^*(s^{t+1})] \tag{2.6}$$

L'équation 2.6 est appelée équation d'optimalité de Bellman. On obtient une équation d'optimalité du même type en travaillant sur les qualités des actions associées aux états :

[30] [Bellman, 1957]

$$\forall s \in S, Q^*(s) = \sum_{s^{t+1}} T(s^t, a^t)(s^{t+1})[R(s^t, a^t) + \gamma \max_{a'} Q^*(s^{t+1}, a')] \tag{2.7}$$

ou

$$\forall s \in S, Q^*(s^t, a^t) = R(s^t, a^t) + \gamma \sum_{s^{t+1} \in S} T(s^t, a^t)(s^{t+1}) \max_{a' \in A} Q^*(s^{t+1}, a') \tag{2.8}$$

Si tous les $R(s^t, a^t)$ et $T(s^t, a^t)(s^{t+1})$ sont connus, alors l'équation 2.6 définit en fait un système de p équations à p inconnues, où $p = |S|$ est le nombre d'états du problème. Les inconnues sont les valeurs de V^* pour chaque état s^t. Cela suffit à justifier que, pour tout problème de décision markovien, il existe une fonction de valeur optimale et une seule.

Par contre, malgré l'unicité de la fonction de valeur optimale, il peut exister plusieurs politiques optimales, si la fonction de valeur définit plusieurs successions d'états de qualités égales.

Dès lors que l'hypothèse de Markov est vérifiée, la politique optimale est la politique *gloutonne* [31], qui consiste à choisir en chaque état l'action dotée de la plus grande qualité pour cet état :

$$\forall s \in S, \pi^*(s) = \arg\max_a Q^*(s^t, a^t) \tag{2.9}$$

On peut aussi dériver la politique optimale de la fonction de valeur, mais il faut rechercher par un regard en avant quelle est l'action qui conduit à l'état suivant de plus grande valeur. Il est donc plus aisé de travailler avec la qualité des actions.

2.3 Algorithmes d'optimisation du comportement

Jusqu'à présent, nous avons défini formellement un processus de décision markovien et nous avons posé les équations définissant la valeur d'une politique quelconque et la valeur d'une politique optimale. Nous nous penchons maintenant sur les algorithmes qui permettent à un agent de découvrir une politique optimale lorsqu'il est confronté à un processus de décision markovien. Il existe trois classes d'algorithmes qui remplissent cette fonction.

• Les algorithmes de programmation dynamique s'appliquent dans le cas où l'agent dispose d'un modèle de son environnement, c'est-à-dire lorsque les fonctions de transition T et de récompense R sont connues a priori.

• Les méthodes de Monte Carlo sont simples et ne présupposent pas la connaissance a priori d'un modèle, mais présentent le défaut de ne pas être incrémentales.

• Les méthodes de différence temporelle reposent sur une estimation incrémentale du modèle de l'environnement. Comme les méthodes de Monte Carlo, elles réalisent cette estimation sur la base de l'expérience de l'agent et se passent ainsi d'un modèle du monde. Néanmoins, elles combinent cette estimation avec des mécanismes de propagation locale d'estimation des valeurs tirées de la programmation dynamique, ce qui leur permet de conserver un caractère incrémental. Ces méthodes constituent le cadre de l'apprentissage par renforcement proprement dit.

Nous allons décrire succinctement les deux premières méthodes en suivant l'ouvrage de R. Sutton et A. barto [32], car elles nous serviront de base pour présenter la suite. Puis nous consacrerons une section plus détaillée aux méthodes de différence temporelle qui sont au cœur de notre propos.

[31] *greedy*
[32] [Sutton & Barto, 1998]

2.3.1 La programmation dynamique

La programmation dynamique recouvre un ensemble d'algorithmes qui permettent de trouver la politique optimale sur un problème de décision markovien, dans le cas où la fonction de transition et la fonction de récompense sont connues. Ces fonctions étant connues, on dit que ces méthodes permettent de résoudre un problème de *planification* et non pas d'apprentissage. On considère que l'agent dispose de toutes les données sur le problème qu'il doit résoudre et on lui fournit un algorithme qui lui permet de dériver une politique optimale en itérant sur ces données. Il n'y a aucun gain d'information dans un tel processus.

Afin de présenter le plus clairement possible l'ensemble des algorithmes relevant de la programmation dynamique dans un cadre unifié, R. Sutton et A. Barto [33] distinguent deux étapes élémentaires de ces algorithmes : l'évaluation d'une politique et l'amélioration d'une politique. Dans le cadre de la résolution de problèmes de décision markoviens, tous les algorithmes de programmation dynamique résultent de diverses combinaisons de ces deux étapes.

Évaluation d'une politique

Étant donné une politique, on appelle évaluation de cette politique le processus qui consiste à déterminer la valeur de chacun des états pour un agent qui se conforme à cette politique. Nous avons indiqué précédemment que la fonction de valeur associée à une politique π vérifie le système d'équations :

$$\forall s \in S, V^{\pi}(s) = \sum_{a} \pi(s^t, a^t)[R(s^t, a^t) + \gamma \sum_{s^{t+1}} T(s^t, a^t)(s^{t+1})V^{\pi}(s^{t+1})] \tag{2.10}$$

Une méthode incrémentale pour résoudre un tel système consiste à partir d'une fonction de valeur initiale V_0 quelconque, puis à appliquer la procédure itérative de mise à jour suivante :

$$\forall s \in S, V_{k+1}(s) = \sum_{a} \pi(s^t, a^t)[R(s^t, a^t) + \gamma \sum_{s^{t+1}} T(s^t, a^t)(s^{t+1})V_k(s^{t+1})] \tag{2.11}$$

On montre que, lorsque k tend vers l'infini, la suite V_k converge vers V^{π}. On sait donc approcher la fonction de valeur associée à une politique de cette façon.

Amélioration d'une politique

Étant donné une politique π quelconque, une façon de l'améliorer consiste à choisir un état s^t et à examiner si, en réalisant dans s^t une action a^t différente de celle spécifiée par $\pi(s)$ et en se conformant à la politique π une fois cette action effectuée, on n'obtiendrait pas une récompense cumulée plus élevée qu'en suivant la politique π dès l'état s^t.

Si c'est le cas, alors le théorème d'amélioration de la politique [34] prouve que la nouvelle politique que l'on obtient en remplaçant $\pi(s)$ par a^t en s^t est meilleure que la politique π initiale.

La méthode d'itération sur la politique [35] repose sur ce principe [36]. Pour chaque état s^t, elle calcule une nouvelle politique π' à partir de π en appliquant la mise à jour suivante :

[33] [Sutton & Barto, 1998]
[34] *policy improvement theorem*
[35] *policy iteration*
[36] [Howard, 1971]

$$\forall s \in S, \pi'(s) = \arg\max_a \sum_{s^{t+1}} T(s^t, a^t)(s^{t+1})[R(s^t, a^t) + \gamma V^\pi(s^{t+1})] \quad (2.12)$$

On démontre que cette méthode de mise à jour converge vers la politique optimale [37].

Méthode généralisée d'itération sur la politique

Pour appliquer l'équation 2.12, il faut connaître la fonction de valeur V^π associée à la politique précédente π. L'algorithme d'itération sur la politique est donc constitué d'une succession d'évaluations de la politique courante suivie d'une amélioration de cette politique, reposant sur le résultat des évaluations précédentes.

Pour obtenir une bonne évaluation d'une politique fixée π, on itère en général l'équation de mise à jour 2.11 jusqu'à ce que le plus grand écart entre $V_k(s)$ et $V_{k+1}(s)$ sur l'ensemble des s^t soit inférieur à une constante ϵ.

De même, pour améliorer une politique π à partir d'une fonction de valeur V^π donnée, il faut itérer jusqu'à ce que la plus grande amélioration de la politique soit inférieure à une autre constante ϵ'.

Au lieu de réaliser nécessairement un grand nombre d'étapes d'évaluations entre chaque série d'étape d'amélioration de la politique, on peut entrelacer un nombre quelconque de ces deux étapes sans remettre en question la convergence de l'algorithme.

Cette intuition constitue la méthode généralisée d'itération sur la politique [38].

Méthode d'itération sur la valeur

Il existe un cas particulier de la méthode généralisée exposée précédemment qui consiste à réaliser des étapes constituées d'une itération d'évaluation de la politique courante entrelacée avec une itération d'amélioration de la politique. On parvient alors à exprimer une équation de mise à jour qui condense les deux étapes en une seule. L'algorithme résultant est appelé algorithme d'itération sur la valeur [39]. L'équation de mise à jour est la suivante :

$$\forall s \in S, V_{k+1}(s) = \max_a \sum_{s^{t+1}} T(s^t, a^t)(s^{t+1})[R(s^t, a^t) + \gamma V_k(s^{t+1})] \quad (2.13)$$

Cette méthode est très employée. Comme les autres, elle converge à la limite vers la fonction de valeur optimale et on l'arrête lorsque le plus grand écart de valeurs entre deux itérations est inférieur à un seuil ϵ.

2.3.2 Les méthodes de Monte Carlo

Les méthodes de programmation dynamique présentent l'avantage d'être incrémentales. On peut réaliser des itérations successives qui convergent peu à peu vers la fonction de valeur optimale, ce qui permet d'agir sans attendre de réaliser toutes les itérations. Par contre, elles exigent une connaissance parfaite de la fonction de transition et de la fonction de récompense du processus de décision markovien associé. Les méthodes de Monte Carlo présentent les avantages et inconvénients opposés. Elles ne présupposent aucune connaissance a priori du problème de décision markovien à résoudre, mais elles ne sont pas incrémentales.

Le principe en est simple. L'agent réalise une série d'expériences. À chaque expérience réalisée, il mémorise les transitions qu'il a effectuées et les récompenses qu'il a reçues. Il met alors à jour une estimation de la valeur des

[37] [Bertsekas, 1995]
[38] [Howard, 1971, Puterman & Shin, 1978]
[39] *value iteration*

états parcourus en associant à chacun d'eux la part de la récompense reçue qui lui revient. Au fil de ses expériences, la valeur estimée associée à chaque état converge alors vers la valeur exacte de l'état pour la politique qu'il suit.

L'apport principal des méthodes de Monte Carlo réside donc dans la méthode qui permet d'estimer la valeur d'un état sur la base de la réception de plusieurs valeurs successives de récompense cumulée associées à cet état lors de trajectoires distinctes.

Une façon simple de réaliser une telle estimation consiste à réaliser la moyenne des récompenses cumulées reçues lors des différentes expériences. Si l'on note $E_k(s)$ la valeur estimée associée à l'état s^t après k expériences et R_k la récompense reçue à l'instant k, une estimation basée sur la moyenne donne :

$$\forall s \in S, E_{k+1}(s) = \frac{R_1 + R_2 + ... + R_k + R_{k+1}}{k+1} \tag{2.14}$$

Pour ne pas avoir à stocker chacune des récompenses reçues, on montre très simplement qu'un tel calcul peut se reformuler de manière incrémentale :

$$\forall s \in S, E_{k+1}(s) = E_k(s) + \frac{1}{k+1}[R_{k+1} - E_k(s)] \tag{2.15}$$

Si le nombre d'expériences devient grand, alors $E_k(s)$ converge vers $V^\pi(s)$. Plutôt que de stocker le nombre k d'expériences réalisées, on utilise généralement une formule d'estimation encore plus générique :

$$\forall s \in S, E_{k+1}(s) = E_k(s) + \alpha[R_{k+1} - E_k(s)] \tag{2.16}$$

et on a

$$\lim_{k \to \infty} E_k(s) = V^\pi(s) \tag{2.17}$$

où α est une constante qu'il faut fixer avec soin. Nous allons retrouver cette méthode d'estimation incrémentale dans les méthodes de différence temporelle.

Les méthodes de Monte Carlo sont peu utilisées en pratique, car elles exigent pour fonctionner qu'un grand nombre de contraintes soient remplies. En particulier, il est indispensable que l'apprentissage soit décomposé en une succession d'épisodes de longueur finie, faute de quoi la mise à jour de l'estimation de la valeur des états ne peut pas avoir lieu. Le fait qu'il faille attendre la fin d'une expérience pour apprendre quoi que ce soit justifie l'affirmation selon laquelle ces méthodes ne sont pas incrémentales.

Nous allons donc nous tourner à présent vers les méthodes de différence temporelle, qui combinent l'incrémentalité de la programmation dynamique avec le recours à l'expérience des méthodes de Monte Carlo.

2.3.3 Les méthodes de différence temporelle

Un apprentissage non supervisé

Dans le cadre que nous avons présenté jusqu'ici, la nécessité de l'apprentissage se fait jour quand l'agent ne connaît pas *a priori* la fonction de renforcement ni la fonction de transition qui détermine les successions d'états par lesquels il est susceptible de passer en fonction de ses actions.

En informatique, on distingue d'une part l'apprentissage dit « supervisé », dans lequel un « instructeur » indique à l'apprenant quelle réponse il aurait dû fournir dans un contexte donné, et d'autre part l'apprentissage dit « non supervisé », dans lequel l'apprenant doit identifier par lui-même la meilleure réponse possible.

Dans le cadre de l'apprentissage supervisé la meilleure réponse possible est fournie à l'apprenant si bien qu'il n'a pas besoin de la rechercher. C'est le cas, par exemple, avec l'algorithme de rétro-propagation du gradient dans les réseaux de neurones à couches.

Parmi les méthodes d'apprentissage non supervisé, il faut distinguer les méthodes dans lesquelles l'apprentissage se fait sans évaluation, par exemple en mémorisant simplement des associations observées, et les méthodes dans lesquelles une évaluation est fournie à l'apprenant, comme c'est le cas pour l'apprentissage par renforcement. L'apprentissage par renforcement est donc une forme d'apprentissage non supervisé reposant sur une évaluation.

L'apprentissage par renforcement se distingue de l'apprentissage supervisé par le fait que, lorsqu'il reçoit un premier signal d'évaluation, l'apprenant ne sait toujours pas si la réponse qu'il a donnée est la meilleure possible ; il doit essayer d'autres réponses pour déterminer s'il peut recevoir une meilleure évaluation.

Au contraire de l'apprentissage supervisé, l'apprentissage non supervisé et, en particulier, l'apprentissage par renforcement induisent par nature une activité d'exploration de la part de l'agent. Il faut que cet agent explore son environnement pour déterminer dans quelles circonstances il est puni ou récompensé et quelles sont les séquences d'action qui lui permettent d'atteindre les récompenses plutôt que les punitions.

Cette nécessité d'explorer est à la source de la présence dans tous les travaux d'apprentissage par renforcement du dilemme posé par le compromis entre exploration et exploitation

Le compromis entre exploration et exploitation

Pour régler la politique de façon à maximiser sa récompense sur le long terme, la phase d'apprentissage se trouve confrontée à la nécessité de trouver un compromis entre l'exploitation, qui consiste à refaire les actions dont on sait déjà qu'elles donnent lieu à une récompense, et l'exploration, qui consiste à parcourir de nouveaux couples *(état, action)* à la recherche d'une récompense cumulée plus grande, mais au risque d'adopter parfois un comportement sous optimal. En effet, tant que l'agent n'a pas exploré la totalité de son environnement, il n'est pas certain que la meilleure politique qu'il connaît est la politique optimale.

En conséquence, toutes les preuves de convergence des algorithmes d'apprentissage par renforcement exigent en théorie que chaque transition soit expérimentée un nombre infini de fois [40]. En pratique, toutefois, on se contente d'une exploration partielle qui suffit en général à découvrir une politique satisfaisante.

Cette présence d'un compromis induit l'existence d'un grand nombre d'algorithmes qui font des choix différents sur ce compromis. Comme le dit élégamment J. Jozefowiez [41], compte tenu de cet état de fait, la résolution d'un problème d'apprentissage par renforcement reste un art plutôt qu'une science.

Les méthodes d'exploration les plus employées sont d'une part les méthodes *softmax* et d'autre part les méthodes ϵ*-greedy*.

Les méthodes *softmax* utilisent un facteur de température, noté τ, au sein d'une distribution de Boltzmann ou de Gibbs. La probabilité de choisir une action a^t parmi n est égale à

$$\frac{e^{Q^t(a^t)/\tau}}{\sum_{b=1}^{n} e^{Q^t(b)/\tau}}$$

où le facteur τ décroît au cours du temps. La méthode de la roue de la fortune [42] est un cas particulier de cette méthode dans laquelle le facteur τ de température est constant au lieu de décroître.

[40] [Watkins & Dayan, 1992]
[41] [Jozefowiez, 2001]
[42] *roulette wheel selection*

Les méthodes ϵ-*greedy* choisissent une action aléatoirement si un tirage aléatoire est en dessous de ϵ, et choisissent la meilleure action sinon.

Ces différentes méthodes d'exploration peuvent être utilisées indifféremment quel que soit l'algorithme d'apprentissage par différence temporelle auquel on fait appel. En effet, pour tous les algorithmes de différence temporelle que nous allons présenter dans la suite, si l'hypothèse de Markov est vérifiée et si l'on parcourt tous les états un nombre infini de fois, alors les valeurs $V(s^t)$ ou les qualités des actions $Q(s^t, a^t)$ convergent vers les valeurs optimales.

Si l'hypothèse de Markov n'est pas vérifiée, en revanche, alors il se peut que le modèle de l'agent ne lui permette pas de distinguer deux états qui sont pourtant différents. Nous verrons dans la section 5.1, page 101, ce qu'il advient dans un tel cas et le type de solution qu'il faut mettre en œuvre pour résoudre ces problèmes plus difficiles.

2.4 Principaux algorithmes d'apprentissage par renforcement

2.4.1 L'algorithme TD(0)

L'algorithme de différence temporelle TD(0) repose sur une comparaison entre la récompense que l'on reçoit effectivement et la récompense que l'on s'attend à recevoir en fonction des estimations construites précédemment.

Si les estimations des fonctions de valeurs aux états s^t et s^{t+1}, notées $V(s^t)$ et $V(s^{t+1})$, étaient exactes, on aurait :

$$V(s^t) = R^{t+1} + \gamma R^{t+2} + \gamma^2 R^{t+3} + ... \tag{2.18}$$

$$V(s^{t+1}) = R^{t+2} + \gamma R^{t+3} + \gamma^2 R^{t+4} + ... \tag{2.19}$$

Donc on aurait :

$$V(s^t) = R^{t+1} + \gamma V(s^{t+1}) \tag{2.20}$$

Par conséquent, le terme $\delta = R^{t+1} + \gamma V(s^{t+1}) - V(s^t)$, que l'on appelle « erreur de différence temporelle [43] » mesure l'erreur entre les valeurs effectives des estimations $V(s)$ et les valeurs qu'elles devraient avoir.

La méthode de différence temporelle consiste à corriger peu à peu cette erreur en modifiant la valeur de $V(s^t)$ selon une équation de Widrow-Hoff :

$$V(s^t) \leftarrow V(s^t) + \alpha[R^{t+1} + \gamma V(s^{t+1}) - V(s^t)] \tag{2.21}$$

Cette équation de mise à jour permet de comprendre immédiatement comment les algorithmes de différence temporelle combinent les propriétés de la programmation dynamique avec celles des méthodes de Monte Carlo. En effet, elle fait apparaître les deux caractéristiques suivantes :

- comme dans les algorithmes de programmation dynamique, la valeur estimée de $V(s^t)$ est mise à jour en fonction de la valeur estimée de $V(s^{t+1})$. Il y a donc propagation des valeurs estimées à partir des valeurs estimées des états successeurs ;
- comme dans les méthodes de Monte Carlo, chacune de ces valeurs résulte d'une estimation qui repose sur l'expérience accumulée par l'agent au fil de ses interactions avec son environnement.

[43] *temporal difference error, [Sutton, 1988]*

On voit donc que les méthodes de différence temporelle et, en particulier, TD(0), reposent sur deux processus de convergence couplés, le premier estimant de plus en plus précisément la récompense immédiate reçue dans chacun des états et le second approchant de mieux en mieux la fonction de valeur résultant de ces estimations en les propageant de proche en proche.

Dans le cas de TD(0), les mises à jour se font localement à chaque fois que l'agent réalise une transition dans son environnement, à partir d'une information se limitant à son état courant s^t, l'état successeur s^{t+1} et la récompense R^{t+1} reçue dans cet état successeur. Une preuve de convergence de l'algorithme a été proposée par P. Dayan et T. Sejnowski [44].

Il faut noter, par contre, que comme TD(0) estime la fonction de valeur d'un problème, faute d'un modèle des transitions entre les états, l'agent est incapable d'en déduire quelle politique suivre, car il ne peut réaliser un pas de regard en avant pour déterminer quel est l'état suivant de plus grande valeur. Ce point explique que l'on préfère avoir recours aux algorithmes qui travaillent sur la fonction de qualité que nous allons présenter dans ce qui suit.

2.4.2 L'algorithme SARSA

L'algorithme SARSA est similaire à l'algorithme TD(0) à ceci près qu'il travaille sur les qualités des couples (s^t, a^t) plutôt que sur la valeur des états. Son équation de mise à jour est la suivante :

$$Q(s^t, a^t) \leftarrow Q(s^t, a^t) + \alpha[R^{t+1} + \gamma Q(s^{t+1}, a^{t+1}) - Q(s^t, a^t)] \tag{2.22}$$

L'information nécessaire pour réaliser une telle mise à jour alors que l'agent réalise une transition est le quintuplet $(s^t, a^t, R^{t+1}, s^{t+1}, a^{t+1})$, d'où découle le nom de l'algorithme.

Effectuer ces mises à jour implique que l'agent détermine avec un pas de regard en avant quelle est l'action a^{t+1} qu'il réalisera lors du pas de temps suivant, lorsque l'action a^t dans l'état s^t l'aura conduit dans l'état s^{t+1}.

2.4.3 L'algorithme Q-LEARNING

L'algorithme Q-LEARNING se présente comme une simplification de l'algorithme SARSA par le fait qu'il n'est plus nécessaire pour l'appliquer de déterminer un pas de temps à l'avance quelle sera l'action réalisée au pas de temps suivant. Son équation de mise à jour est la suivante :

$$Q(s^t, a^t) \leftarrow Q(s^t, a^t) + \alpha[R^{t+1} + \gamma \max_a Q(s^{t+1}, a) - Q(s^t, a^t)] \tag{2.23}$$

Le terme $Q(s^{t+1}, a^{t+1})$ apparaissant dans l'équation 2.22 a été remplacé par le terme $max_a Q(s^{t+1}, a)$ dans l'équation 2.23. Cela pourrait sembler équivalent si la politique suivie était gloutonne (on aurait alors $a^{t+1} = \arg\max_a Q(s^{t+1}, a)$). Toutefois, compte tenu de la nécessité de réaliser un compromis entre exploration et exploitation, ce n'est généralement pas le cas. Il apparaît donc que l'algorithme SARSA effectue les mises à jour en fonction des actions choisies effectivement alors que l'algorithme Q-LEARNING effectue les mises à jour en fonction des actions optimales, ce qui est à la fois plus simple et plus efficace.

Cette simplicité a fait la réputation de l'algorithme Q-LEARNING. Il s'agit sans doute de l'algorithme d'apprentissage par renforcement le plus connu en raison des preuves formelles de convergence qui ont accompagné sa publication [45].

[44] [Dayan & Sejnowksi, 1994]
[45] [Watkins & Dayan, 1992]

2.4.4 Les algorithmes TD(λ), Sarsa(λ) et Q(λ)

FIG. 2.3 – **Q-learning (Premier et deuxième essai).**

Les algorithmes TD(0), SARSA et Q-LEARNING présentent le défaut de ne mettre à jour qu'une valeur par pas de temps, à savoir la valeur de l'état que l'agent est en train de visiter. Comme il apparaît sur la figure 2.3, cette procédure de mise à jour est particulièrement lente. En effet, pour un agent ne disposant d'aucune information a priori sur la structure de la fonction de valeur ou de la fonction de qualité, il faut au moins n expériences successives pour que la récompense immédiate reçue dans un état donné soit propagée jusqu'à un état distant du premier de n transitions. En attendant le résultat de cette propagation, tant que toutes les valeurs sont identiquement nulles, le comportement de l'agent s'apparente à une marche aléatoire.

Une façon d'améliorer cet état de fait consiste à doter l'algorithme d'une mémoire des transitions qu'il a effectuées au cours d'une expérience afin d'effectuer toutes les propagations possibles à la fin de cette expérience. Cette mémoire des transitions effectuées précédemment est appelée une *trace d'éligibilité*. Ainsi, R. Sutton et A. Barto ont proposé une classe d'algorithmes appelés « TD(λ) » qui généralisent l'algorithme TD(0) au cas où l'agent dispose d'une mémoire des transitions. De même, les algorithmes SARSA et Q-LEARNING ont été généralisés en SARSA(λ) et Q(λ), le second l'ayant été de deux façons différentes par deux auteurs différents [46]. Ces algorithmes sont plus efficaces que la version correspondante dépourvue de trace d'éligibilité, mais requièrent plus de mémoire. Il faut noter qu'à l'heure où nous rédigeons ce chapitre, ni SARSA(λ), ni Q(λ), ni TD(λ) ne disposent d'une preuve de convergence publiée. F. Garcia [47] dispose cependant d'une telle preuve pour TD(λ), soumise à publication.

Un premier procédé naïf pour accélérer l'apprentissage consiste à stocker directement une liste des couples (état, action) parcourus par l'agent puis, à chaque fois que l'agent reçoit une récompense, propager celle-ci en parcourant la mémoire des transitions en marche arrière depuis la récompense reçue. Avec un tel procédé, plus la mémoire des transitions est longue, plus une récompense reçue est propagée efficacement. Il apparaît donc un compromis entre la quantité de mémoire mobilisée pour apprendre et la vitesse d'apprentissage.

La méthode mise en œuvre dans TD(λ), SARSA(λ) et Q(λ) est plus sophistiquée et fonctionne pour des horizons infinis. La trace d'éligibilité est représentée en associant à chaque état s^t un indicateur $e(s)$ qui mesure l'écoulement du temps depuis que l'état a été visité. L'indicateur $e(s)$ est augmenté de 1 à chaque fois que l'état s^t est visité, et diminué d'un facteur $\gamma\lambda$ à chaque pas de temps pour tous les états.

Par ailleurs, l'erreur de différence temporelle δ calculée à chaque pas de temps [48] corrige simultanément l'estimation de chacun des états en fonction de la valeur de $e(s^t)$. Par exemple, dans TD(λ), on applique pour tout état s^t

[46] [Watkins & Dayan, 1992, Peng & Williams, 1996]
[47] Communication personnelle
[48] Selon une formule différente selon qu'il s'agit de TD(λ), SARSA(λ) ou Q(λ)

la mise à jour $V(s^t) \leftarrow V(s^t) + \alpha\delta e(s^t)$, et la formule est similaire pour SARSA(λ) et Q(λ) où ce sont les qualités que l'on met à jour.

On peut noter que si $\lambda = 0$, l'indicateur s'annule instantanément si bien que l'on retombe effectivement sur TD(0), SARSA ou Q-LEARNING. Par ailleurs, si $\lambda = 1$, les algorithmes s'apparentent à des algorithmes de type Monte Carlo.

Nous discuterons à la section 3.3.1, page 59, une autre solution qui permet d'effectuer plusieurs mises à jour à chaque pas de temps, dans le cadre de l'apprentissage par renforcement indirect.

2.4.5 Les architectures acteur-critique

Historiquement, les architectures acteur-critique ont été les premières architectures informatiques imaginées par des chercheurs pour modéliser l'apprentissage par renforcement [49].

La caractéristique de ces architectures réside dans la séparation de deux zones de mémoire bien distinctes qui stockent respectivement la politique réalisée par l'agent et la fonction de valeur associée au processus de décision markovien auquel est confronté l'agent.

La structure représentant la politique est appelée « l'acteur », car elle détermine les actions de l'agent, tandis que la structure représentant la fonction de valeur est appelée « le critique », car elle évalue les actions réalisées par l'agent. Cette évaluation est retournée sous la forme d'un écart entre la récompense effectivement reçue et la récompense prédite par la fonction de valeur stockée par le critique. Plus précisément, l'évaluation transmise à l'acteur est le terme $\delta_t = R^{t+1} + \gamma V(s^{t+1}) - V(s^t)$.

Si ce terme est positif, cela signifie que la récompense reçue est plus importante que prévue. En conséquence, l'action réalisée mérite d'être renforcée. S'il est négatif, au contraire, il faut diminuer la tendance de l'acteur à réaliser cette action.

Cette spécification présente le mérite d'être très générale, elle impose très peu de contraintes sur les formalismes de représentation respectifs de l'acteur et du critique. La seule contrainte forte est que l'acteur soit capable de prendre en compte le signal d'erreur δ_t pour modifier sa propension à réaliser telle ou telle action dans un contexte donné. Quant au critique, tout formalisme capable de fournir une approximation de la fonction de valeur fait a priori l'affaire.

Cependant, un grand nombre d'architectures acteur-critique s'appuient sur des réseaux de neurones formels, en raison de la relative plausibilité biologique de cette architecture. Le terme δ_t vient alors modifier le poids des connexions qui gouvernent la propension d'un réseau à réaliser telle ou telle action.

Avant de conclure ce chapitre sur les modèles théoriques de l'apprentissage par renforcement, nous allons dire un mot de la relation en train de s'établir entre ces modèles théoriques et la recherche neurophysiologique et non plus psychologique. Il faut noter que c'est précisément sur la base des architectures acteur-critique que cette relation s'établit, compte tenu du plus grand degré de plausibilité biologique dont elles bénéficient vis-à-vis des autres modèles de l'apprentissage par renforcement.

2.5 Architectures biomimétiques et apprentissage par renforcement

Les recherches qui ont permis d'établir la relation entre les modèles formels et les processus neurophysiologiques portent sur un ensemble de noyaux sous-corticaux appelés « *ganglions de la base* ». Ces ganglions jouent un rôle essentiel au cœur du système dopaminergique mésencéphalique commun aux vertébrés et sont considérés

[49] [Witten, 1977, Barto et al., 1983]

comme impliqués dans le contrôle du comportement [50]. Par exemple, des mammifères dont des sites proches des voies dopaminergiques sont stimulés lorsqu'ils appuient sur un levier, actionnent compulsivement le levier quitte à se laisser mourir de faim ou de soif, plutôt que de renoncer à l'excitation de ce qui doit être compris comme une aire cérébrale liée à la sensation de plaisir [51]. L'examen du rôle joué par diverses drogues comme la cocaïne ou des amphétanimes qui modifient le métabolisme de la dopamine vient confirmer ces faits expérimentaux.

Pour être plus précis, M. Khamassi [52] invite à distinguer deux mécanismes d'apprentissage différents au sein de deux parties différentes des ganglions de la base, à savoir le striatum dorsal et le striatum ventral :

- l'apprentissage stimulus-réponse, s'apparentant au conditionnement instrumental, qui concerne l'association entre deux événements immédiats et permettant la formation de procédures comportementales dont l'exécution serait supportée par le striatum dorsal ou moteur [53] ;
- l'apprentissage de la sélection des stratégies de navigation pour les comportements orientés vers un but, fonctionnant pour des associations à plus long terme et supposé siéger dans le noyau accumbens, recouvrant les parties associative, limbique et motrice du striatum ventral [54].

De nombreux travaux consacrés au rôle de la dopamine dans les *ganglions de la base* s'efforcent de mettre en évidence le rapport complexe entre la perception d'une récompense et la sécrétion de dopamine, neurotransmetteur chimique considéré comme jouant un rôle déterminant dans la réalisation du renforcement.

Cependant, il y a divergence sur l'interprétation de la fonction exacte de la dopamine dans l'apprentissage par renforcement. W. Schultz considère qu'il s'agit d'un médiateur du renforcement [55], sécrété lorsque le renforcement perçu ne correspond pas au renforcement attendu. En effet, la dopamine est couramment considérée comme un signal d'erreur de prédiction qui permet la formation de procédures motrices dépendantes du contexte sensoriel envoyé du cortex vers le striatum dorsal [56]. W. Schultz et ses collègues mettent alors en rapport l'architecture des ganglions de la base avec l'architecture acteur-critique utilisée dans les modèles formels de l'apprentissage par renforcement dont nous avons parlé à la section 2.4.5 et tentent d'établir une équivalence entre le processus biologique du renforcement et l'algorithme TD(λ) issu de ces modèles formels. Plus précisément, la quantité de dopamine émise pourrait coïncider avec le signal d'erreur δ_t calculé dans les architectures acteur-critique. Plusieurs modèles informatiques tentent de valider ces hypothèses biologiques [57], mais de telles données sont encore largement controversées. Par exemple, C. Pennartz ou S. Ikemoto considèrent que la dopamine code pour la saillance des percepts associés à la récompense [58] et non pas pour l'écart à la récompense attendue.

Plutôt que d'entrer dans le détail de toutes ces discussions, nous préférons renvoyer le lecteur aux travaux respectifs de J. Jozefowiez [59], B. Girard [60] et M. Khamassi [61] qui proposent séparément une revue étendue de différents modèles de neurophysiologie qui ont trait à l'apprentissage par renforcement, mettant en jeu le cervelet ainsi que les ganglions de la base.

Pour conclure sur les rapports entre formalisation mathématique de l'apprentissage par renforcement et neurophysiologie de l'apprentissage, il apparaît clairement lorsqu'on consulte les dates de publications des travaux

[50] [Graybiel, 1995, DeLong, 2000]
[51] [Robbins & Everitt, 1996]
[52] [Khamassi, 2003]
[53] [Houk et al., 1995]
[54] [Pennartz et al., 1994, Ikemoto & Panksepp, 1999, Wiener et al., 2002]
[55] [Schultz et al., 1997, Schultz, 1998]
[56] [Khamassi, 2003]
[57] [Barto, 1995, Montague et al., 1996]
[58] [Berridge & Robinson, 1998, Pennartz et al., 2000, Ikemoto & Panksepp, 1999]
[59] [Jozefowiez, 2001]
[60] [Girard, 2002]
[61] [Khamassi, 2003]

que nous avons cités dans ce chapitre que la mise au point des modèles théoriques et informatiques de l'apprentissage par renforcement est antérieure aux travaux de neurophysiologie auxquels nous venons de faire référence. La modélisation informatique issue de la psychologie a donc probablement permis d'orienter l'investigation neurophysiologique en indiquant ce qu'il fallait chercher au sein du cerveau des mammifères. Dans cette hypothèse, les modèles informatiques ont servi de relais entre la psychologie et la neurophysiologie.

À présent, on est en droit d'espérer que le transfert de connaissances puisse s'effectuer dans l'autre sens. En effet, une prise en compte approfondie des connaissances actuelles sur l'architecture cérébrale des êtres vivants doit permettre de réaliser des avancées significatives sur la mise au point de mécanismes informatiques d'apprentissage par renforcement.

La difficulté principale pour une démarche visant à s'inspirer davantage de la physiologie provient de ce que les aires cérébrales qui semblent impliquées dans les mécanismes d'apprentissage par renforcement sont aussi celles pour lesquelles il existe quantités de modèles plus ou moins contradictoires leur attribuant des fonctions souvent hétérogènes [62].

Malgré sa difficulté, cette voie de recherche est porteuse d'un grand espoir concernant la réalisation prochaine de modèles informatiques de l'apprentissage par renforcement au sein d'architectures neuronales complexes. À lui seul, cet espoir permet de saisir tout l'intérêt de l'insistance caractéristique de l'approche animat à se tenir au plus près des modèles issus de la biologie pour construire des systèmes dotés de propriétés intéressantes du point de vue de l'ingénieur.

Nous reviendrons sur ce point au moment de conclure notre mémoire, une fois que nous auront fait le tour des approches formelles et mis en évidence leurs limites.

2.6 Conclusion : limites du cadre de l'apprentissage par renforcement

L'apprentissage par renforcement constitue assurément une technique privilégiée pour la modélisation des comportements adaptatifs. En tant que technique d'apprentissage, elle permet à un agent d'adapter son comportement au cours d'expériences successives en extrayant de l'information de ses interactions avec son environnement. Étant non supervisée, elle permet de ne pas violer la contrainte d'autonomie de l'agent qui apprend, le signal de renforcement étant fourni de façon naturelle par l'environnement de cet agent.

Cependant, le cadre théorique que nous avons exposé au cours de ce chapitre n'est pas exempt de limites vis-à-vis des problèmes généraux et applicatifs que nous nous proposons d'aborder. Dans ce qui suit, nous allons nous efforcer de présenter ces limites en commençant par celles dont nous espérons nous affranchir le plus facilement.

• Tout d'abord, nous avons vu à la section 2.4.4 qu'il existait un compromis entre la vitesse d'apprentissage et la mémoire utilisée pour apprendre. La solution consistant à mettre en œuvre des traces d'éligibilité reste limitée en ceci que l'apprentissage n'opère qu'à partir d'informations extraites du passé immédiat de l'agent. Nous allons voir au chapitre suivant qu'une rupture plus radicale avec le cadre béhavioriste permet de déboucher sur une approche différente de l'apprentissage. Dans cette approche, l'agent élabore un modèle de ses interactions avec son environnement puis peut apprendre sur la foi de ce modèle, indépendamment de son expérience courante.

• Ensuite, le cadre théorique général que nous venons de présenter suppose de la part de l'agent une parfaite connaissance de son état considéré comme un tout indifférencié. Dans la plupart des situations où l'on représente un agent en interaction avec son environnement, l'agent ne dispose pas d'un tel accès à son état, mais plutôt à

[62] [Pennartz et al., 2000]

des perceptions différenciées qui le renseignent exhaustivement ou non sur sa situation vis-à-vis de son environnement. Dans les deux chapitres suivants, nous allons nous intéresser en particulier au cas où l'état de l'agent peut être décomposé sous la forme d'une liste d'attributs qui prennent des valeurs distinctes. Cette première remise en question de la représentation de l'état de l'agent sera restreinte au cas où les attributs fournis à l'agent lui suffisent effectivement à connaître sans ambiguïté son état.

• Dans un deuxième temps, nous nous attaquerons au cas beaucoup plus difficile où les perceptions instantanées de l'agent ne lui suffisent pas à déterminer son état. Ce second cas nous écarte davantage du cadre standard des processus de décision markoviens dans la mesure où l'hypothèse de Markov, qui joue un rôle central dans les preuves de convergence de tous les algorithmes que nous avons présentés, n'y est plus vérifiée. Nous consacrerons donc le chapitre 5 à l'étude des cas où les agents doivent s'appuyer sur une mémoire de leur passé pour espérer adopter un comportement optimal.

• Une autre limitation inhérente au cadre des processus de décision markoviens dont nous allons tenter de nous affranchir est le caractère discret des transitions entre états. De nombreux problèmes d'interaction entre un agent et son environnement se représentent de façon plus naturelle par l'évolution de grandeurs continues. Dans un tel cas, il faut choisir entre adapter le cadre de l'apprentissage par renforcement aux problèmes continus comme l'ont fait R. Munos [63] et R. Coulom [64], ou bien adapter la représentation du problème en discrétisant la dynamique des interactions entre l'agent et son environnement. Nous présenterons au chapitre 6 une approche qui s'écarte largement de la problématique de l'apprentissage par renforcement pour traiter des problèmes de comportements adaptatifs collectifs avec des représentations continues, puis nous indiquerons au chapitre 8 comment réintroduire la problématique de l'apprentissage par renforcement au cœur de cette approche.

• Enfin, les problèmes liés à la dimension de l'espace d'états et d'action constituent une dernière source de limitation incontournable de la modélisation par des processus de décision markoviens. Face à cette difficulté, la seule issue consiste à élaborer les algorithmes les plus efficaces possibles et, surtout, à travailler sur la représentation des problèmes pour limiter l'espace de recherche dans lequel les algorithmes travaillent. Sur ce dernier point, il nous semble qu'il n'y a pas de recette générale, chaque application requiert une expertise particulière pour être représentée de la façon la plus adéquate possible. Au chapitre 9, nous exposerons les travaux que nous entreprenons pour traiter avec des algorithmes d'apprentissage par renforcement les problèmes difficiles que constituent le cadre continu et le cadre multi-agents.

[63] [Munos, 1997]
[64] [Coulom, 2002]

Chapitre 3

Systèmes de classeurs à anticipation

> « Moi, je suis en train de ne rien comprendre, mais j'aimerai bien participer à la discussion. »
>
> P. Gérard, le 17/09/2002

Nous avons conclu le chapitre précédent sur les limitations inhérentes au cadre théorique des processus de décision markoviens pour modéliser des phénomènes d'apprentissage par renforcement. Dans ce chapitre, nous présentons un travail de recherche mené dans le cadre de la thèse de P. Gérard destiné à dépasser deux de ces limitations.

En adoptant le formalisme des systèmes de classeurs, nous passons d'une représentation d'état monolithique à une représentation dans laquelle l'agent a accès à des observables différenciées qu'il peut choisir de prendre en compte ou non, ce qui lui permet de bâtir une représentation compacte du processus de décision markovien auquel il est confronté.

Par ailleurs, nous présentons les architectures DYNA *qui, tout en restant tributaires des représentations tabulaires décrites au chapitre précédent, résolvent le problème de lenteur de la propagation des valeurs apprises en élaborant un modèle interne des transitions effectuées par l'agent. Cela leur permet d'appliquer des algorithmes de programmation dynamique pour propager efficacement les valeurs ou les qualités indépendamment de la situation courante de l'agent.*

Nous nous tournons alors vers les systèmes de classeurs à anticipation, dont nous montrons qu'ils combinent à la fois les propriétés des architectures DYNA *et celles des systèmes de classeurs. Nous décrivons en particulier MACS, un des systèmes élaborés par P. Gérard, et nous comparons ses performances à celles des architectures* DYNA. *Cela permet de mettre en évidence les difficultés supplémentaires posées par la problématique de la compacité de la représentation, puis de présenter les limites qui restent à dépasser dans cette nouvelle perspective.*

3.1 Les systèmes de classeurs

3.1.1 Insuffisance de la notion d'état

Nous avons indiqué au chapitre précédent que l'apprentissage par renforcement en informatique est issu de la modélisation de processus psychologiques d'association entre des stimuli et des réponses. Cependant, les modèles formels nés du recours à la théorie de la commande optimale associent à la situation d'un agent un « état » plutôt que des stimuli. Les travaux formels en apprentissage par renforcement considèrent en effet l'état comme un tout

indifférencié, une « case » dans laquelle se situe l'agent à un instant donné. Ils laissent de côté la question de la détermination par l'agent de l'état dans lequel il se trouve à partir des stimuli qu'il reçoit.

Si cette simplification a permis de poser les fondations théoriques du domaine, elle engendre aussi un certain nombre de problèmes qui conduisent à sa remise en question. Notamment, de nombreux auteurs [1] s'accordent sur le fait que, pour la plupart des problèmes issus du monde réel, une modélisation qui repose sur une énumération explicite de tous les états possibles est extrêmement inefficace. Nous approfondirons ce point au chapitre suivant.

En attendant, nous allons considérer le cas où les informations dont dispose un agent sont constituées d'un ensemble de variables renvoyées par des capteurs, que nous appellerons « observables ». Les observables sont des attributs de son environnement auxquels l'agent a accès et qui lui permettent, dans le cas markovien, de déterminer l'état de ses interactions avec l'environnement avec certitude. Cela signifie que, si l'hypothèse de Markov est vérifiée, la donnée d'un vecteur d'observables est équivalente à la donnée de l'état ou, pour le dire autrement, que l'état de l'agent est *défini* par le vecteur d'observables [2].

Dans le cadre des processus de décision markoviens, on pourrait donc appliquer l'ensemble des algorithmes décrits au chapitre précédent à des vecteurs d'observables plutôt qu'à des états, en considérant chaque vecteur d'observables possible comme l'étiquette qui désigne un état donné.

Des considérations algorithmiques montrent toutefois que ce procédé est inefficace. En effet, le nombre de situations envisageables est une fonction exponentielle du nombre d'observables. Par exemple, si les observables sont booléennes, un vecteur de n observables permet de représenter 2^n situations. En conséquence, lorsque le nombre d'observables dont dispose l'agent est élevé, associer un état à chaque combinaison d'observables conduit à une explosion du nombre d'états considérés [3].

Dans des problèmes concrets, il s'avère que l'ensemble des vecteurs d'observables qu'un agent peut rencontrer est généralement beaucoup plus faible que l'ensemble des combinaisons engendrées par l'énumération de toutes les valeurs possibles pour chaque observable. Ce que nous venons d'avancer revient à dire que, dans les problèmes réels, les observables ne sont pas indépendantes les unes des autres. Par exemple, dans un problème de simulation multi-agents qui nous a servi de banc de test [4], nous avons constaté que l'agent pouvait rencontrer 290 situations possibles sur les 8192 situations auxquelles conduirait la combinatoire des valeurs possibles pour chaque observable.

Un examen attentif permet d'identifier des corrélations entre les observables, si bien qu'il doit exister des modes de représentation plus compacts dont on peut tirer parti pour travailler sur des espaces de recherche plus restreints. Ce constat a donné naissance à deux domaines de recherche qui se sont développés indépendamment l'un de l'autre, malgré un certain nombre de similitudes sur lesquelles nous reviendrons à la section 4.2, page 81.

Le premier domaine est constitué par la recherche sur les réseaux bayésiens dynamiques [5]. Le cœur du formalisme de représentation sous-jacent consiste à faire apparaître graphiquement sous forme de liens des dépendances probabilistes entre les observables qui décrivent un problème au temps t et les mêmes observables au temps $t + 1$, puis à exprimer sous forme de tables de vérité ou d'arbres de décision les corrélations statistiques effectives entre ces observables. Une fois qu'un problème a été représenté de cette façon par un expert ou appris à partir d'exemples par un algorithme de type Monte Carlo [6], on peut appliquer à la représentation résultante les techniques classiques

[1] [Boutilier et al., 1995, Koller & Parr., 2000]

[2] Nous verrons au chapitre 5 ce qu'il advient lorsque le vecteur d'observables dont dispose l'agent à un instant donné ne suffit pas à déterminer son état à cet instant.

[3] [Boutilier et al., 1999]

[4] [Sigaud & Gérard, 2001]

[5] *Dynamic Bayes Networks*, ou DBN

[6] voir la section 2.3.2, page 38

de planification élaborées dans le cadre des modèles de Markov, telle que l'itération sur la politique [7].

Le second domaine, à savoir les systèmes de classeurs, s'appuie lui aussi sur une représentation des états sous forme d'un ensemble d'observables, mais restreint l'espace engendré par leur énumération au moyen d'un processus de généralisation qui s'applique sur un ensemble de règles.

Dans ce chapitre, nous allons commencer par étudier dans le détail ce second domaine. Nous reviendrons au rapport avec les réseaux bayésiens dynamiques au chapitre suivant, une fois que nous aurons présenté le formalisme des systèmes de classeurs à anticipation.

3.1.2 Historique succinct des systèmes de classeurs

Les systèmes de classeurs ont été inventés par J. Holland [8] en vue de proposer un modèle de l'émergence de la cognition fondé sur des mécanismes adaptatifs. La première implémentation concrète, appelée « CS1 », a été publiée par J. Holland et J. Reitman [9]. Ce premier modèle était assez complexe, il utilisait notamment un mécanisme de messages internes et un algorithme d'apprentissage par renforcement fondé sur les différences temporelles [10] nommé BUCKET BRIGADE [11]. Une formalisation plus rigoureuse des différents mécanismes impliqués dans ces premiers modèles a permis de dégager le rôle spécifique joué par les *algorithmes génétiques* [12], qui ont été utilisés par la suite indépendamment des systèmes de classeurs.

Les algorithmes génétiques constituent un modèle informatique extrêmement simplifié du processus d'évolution génétique qui opère au sein d'une population. De façon très succincte, ces algorithmes se dotent d'une population d'agents qui évolue de génération en génération. Les agents disposent d'une *fitness* qui mesure l'adéquation de leur comportement au problème qui leur est posé. Lors du passage d'une génération à la suivante, on élimine un certain nombre d'agents dont la *fitness* est faible, et on les remplace par des nouveaux individus qui sont les descendants d'agents généralement performants. Au fil des générations, la *fitness* moyenne de la population se rapproche de la *fitness* optimale, ce qui permet d'affirmer que les algorithmes génétiques constituent des méthodes très générales d'optimisation.

Cette présentation lapidaire des algorithmes génétiques était destinée à nous permettre d'exposer l'opposition entre les approches de type Pittsburgh et les approches de type Michigan qui ont structuré les premiers pas des recherches sur les systèmes de classeurs.

Dans les systèmes proposés par J. Holland et ses doctorants successifs, constituant l'école Michigan, les agents dont la *fitness* doit évoluer au cours du temps sont les classeurs individuels. La *fitness* d'un classeur est mesurée par sa capacité à proposer une action efficace au moment opportun, et l'algorithme génétique sert avant tout à proposer, par recombinaison, des nouveaux classeurs issus de classeurs performants. Dans l'approche de type Pittsburgh défendue par S. Smith [13], au contraire, les individus sur lesquels travaille l'algorithme génétique sont des systèmes de classeurs entiers, dont la *fitness* est donnée par l'adéquation du comportement de l'agent qu'ils dirigent. La voie de recherche constituée par l'approche de type Pittsburgh a été absorbée dans la rapide expansion des travaux sur les algorithmes génétiques, à laquelle elle se rattachait plus directement. Même si l'on assiste à l'heure actuelle à un renouveau des approches de type Pittsburgh dans le domaine des systèmes de classeurs, le cadre le plus standard pour ces systèmes reste tout de même l'approche de type Michigan.

[7] voir la section 2.3.1, page 37
[8] [Holland, 1975]
[9] [Holland & Reitman, 1978]
[10] voir la section 2.4.1, page 41
[11] [Holland et al., 1986]
[12] [Booker et al., 1989, Goldberg, 1989]
[13] [Smith, 1980]

Un historique du développement des systèmes de classeurs a été publié par S. Wilson et D. Goldberg en 1989 [14], marquant ainsi un premier jalon de l'histoire de ces systèmes. Nous ne détaillerons pas le fonctionnement de ces systèmes historiques. À cette époque, la recherche sur les systèmes de classeurs a connu un certain essoufflement en raison de la difficulté à obtenir des résultats avec des architectures dotées de messages internes. Un renouveau s'est produit lorsque S. Wilson est parvenu à proposer une version radicalement simplifiée de ces systèmes [15], dans laquelle les messages internes avaient disparu et l'algorithme BUCKET BRIGADE était remplacé par le Q-LEARNING, mieux formalisé. C'est ce système simplifié, nommé « ZCS » [16], que nous allons choisir comme référence pour décrire ce qu'est exactement un système de classeurs.

Après avoir proposé ZCS, qui assimilait la force d'un classeur à l'espérance de renforcement reçu en déclenchant le classeur concerné, S. Wilson a imaginé un nouveau système, « XCS » [17], dans lequel la force représentait cette fois la capacité du classeur à *prédire* précisément la quantité de renforcement que son déclenchement permet de recevoir. Ce système s'est avéré beaucoup plus performant que tous les systèmes de classeurs précédents et constitue la nouvelle base sur laquelle la plupart des systèmes se développent aujourd'hui.

Il faut noter que, bien que l'impulsion initiale donnée à la recherche sur les systèmes de classeurs ait porté sur les problèmes d'interaction entre un agent et son environnement dont nous avons parlé au chapitre précédent, les excellents résultats du système XCS sur des problèmes de fouille de données [18] [19] conduisent actuellement à une extension spectaculaire des travaux en direction des problèmes de classification automatique, direction amorcée par L. Booker [20] et J. Holmes [21].

Nous invitons le lecteur désireux d'en apprendre davantage sur l'histoire du développement des systèmes de classeurs à consulter les articles respectifs de S. Wilson et D. Goldberg en 1989 [22] puis P.-L. Lanzi et R. Riolo en 1999 [23]. Pour étudier les développements les plus récents, il faut se reporter aux actes des conférences GECCO [24] et IWLCS [25], qui rassemblent chaque année l'essentiel des travaux consacrés à ce domaine.

3.1.3 Description des systèmes de classeurs

Formalisme de représentation

Un système de classeurs simple est composé d'une population de règles appelées « *classeurs* ». Ces règles comportent généralement une partie `[Condition]` et une partie `[Action]`. On note sous la forme `[Condition]` → `[Action] (Qualité)` un classeur qui signifie « si la condition est vérifiée, alors faire l'action devrait être d'autant plus bénéfique que la qualité est plus élevée ».

Afin de présenter ce qui fait la spécificité des systèmes de classeurs vis-à-vis du cadre de l'apprentissage par renforcement, il est nécessaire de rapprocher les deux types de représentation.

Dans le cadre standard, la structure de données qui permet l'apprentissage de la politique optimale est une table de la forme `[Etat] x [Action]` → `(Qualité)`, appelée *Q-Table*. On qualifie ce type de représentation de

[14] [Wilson & Goldberg, 1989]
[15] [Wilson, 1994]
[16] *Zeroth-level Classifier System*
[17] [Wilson, 1995]
[18] *Data Mining*
[19] [Bernardó et al., 2001]
[20] [Booker, 2000]
[21] [Holmes, 2002]
[22] [Wilson & Goldberg, 1989]
[23] [Lanzi & Riolo, 2000]
[24] [Banzhaf et al., 1999, Whitley et al., 2000, Spector et al., 2001, Langdon et al., 2002, Cantu-Paz et al., 2003, Deb et al., 2004]
[25] [Lanzi et al., 2000, Lanzi et al., 2001, Lanzi et al., 2002a, Lanzi et al., 2002b, Stolzmann et al., 2002]

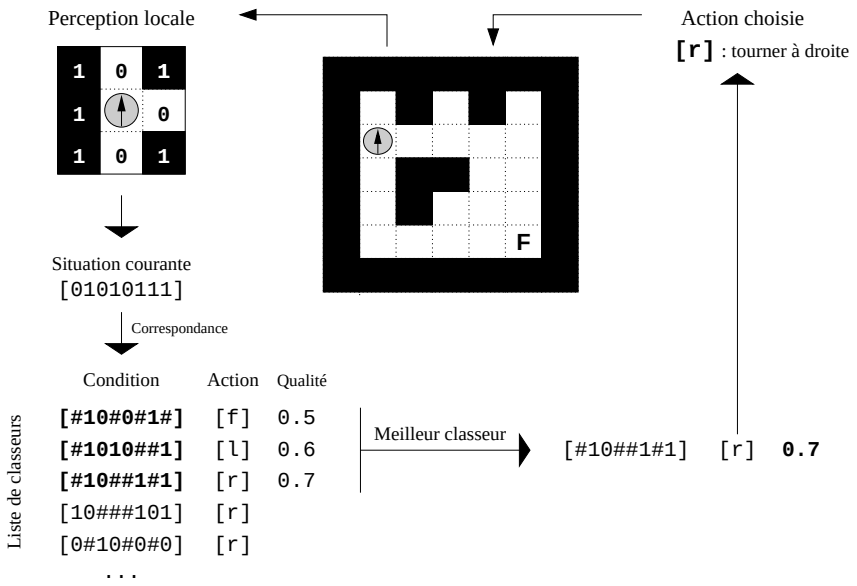


FIG. 3.1 – **Un système de classeurs contient une liste de classeurs. L'agent perçoit une situation sous la forme d'un ensemble d'attributs. Dans cet exemple, l'agent est situé dans un labyrinthe et perçoit la présence (symbole 1) ou l'absence (symbole 0) de blocs dans les huit cases environnantes. Les cases sont considérées dans le sens des aiguilles d'une montre, en commençant par le nord, si bien qu'il perçoit `[01010111]` dans l'exemple ci-dessus. L'agent doit choisir à chaque instant entre avancer (`[f]`) tourner à droite (`[r]`) ou tourner à gauche (`[l]`). Parmi les classeurs appris, le système identifie ceux dont les conditions sont compatibles avec la situation courante, puis choisit dans cet ensemble un classeur dont la qualité est élevée. L'action spécifiée par ce classeur est activée si bien que, dans l'exemple, l'agent se tourne vers sa droite.**

« tabulaire ».

Pour certains systèmes de classeurs comme ZCS, la qualité d'un classeur est égale à la qualité de l'action qu'il stipule pour le contexte spécifié par la partie `[Condition]`. Si la partie `[Condition]` portait sur l'état considéré comme un tout indifférencié, alors la représentation de ZCS serait strictement équivalent à une *Q-Table*, mais ce n'est pas le cas. La différence essentielle entre les deux formalismes réside dans le codage de la partie `[Condition]`.

Dans le cadre des systèmes de classeurs, on considère que l'agent a accès à un vecteur d'observables et la partie `[Condition]` des classeurs est constituée d'une série de tests individuels portant sur les valeurs de chacune des observables. Plus précisément, la partie `[Condition]` d'un classeur stipule pour chaque observable la valeur qu'elle doit prendre au sein de son domaine pour que la condition soit vérifiée.

Avec un tel formalisme, si la donnée d'un vecteur d'observables est équivalente à la donnée d'un état, alors la partie `[Condition]` d'un classeur n'est vérifiée que pour un état donné. On se retrouve alors dans le cas où l'on gère des états « étiquetés » par un ensemble de valeurs d'observables. Nous avons dit que cette représentation n'est pas satisfaisante.

L'innovation proposée par les systèmes de classeurs vis-à-vis de cette représentation provient du symbole « # », appelé « *don't care symbol* », dans la partie `[Condition]` des classeurs. Le fait que la partie `[Condition]` d'un classeur contienne un # pour une observable donnée signifie que la condition stipulée par le classeur sera vérifiée quelle que soit la valeur de l'observable correspondante. Seules les observables dont la valeur n'est pas # sont prises en compte pour vérifier que le classeur peut être déclenché. Formellement, la partie `[Condition]` d'un classeur est donc un vecteur de valeurs parmi lesquelles on peut trouver un certain nombre de symboles #.

Historiquement, les premiers systèmes de classeurs fonctionnaient uniquement avec des vecteurs d'observables booléennes. Chaque test de la partie `[Condition]` prend alors sa valeur dans l'ensemble $\{0, 1, \#\}$. Plus récemment, de nombreux systèmes ont été étendus pour prendre en compte des valeurs discrètes [26] ou continues [27]. Dans ce second cas, un test spécifie généralement un intervalle dans lequel doit se situer la valeur de l'observable pour que la condition soit vérifiée. Il existe par ailleurs un système de classeurs dans lequel chaque condition individuelle est exprimée sous la forme d'une expression symbolique [28] que doit vérifier l'observable pour que la condition soit vérifiée [29].

Un système d'apprentissage par renforcement est confronté à deux problèmes. D'une part, il doit estimer aussi précisément que possible la valeur de chacune des actions que l'agent peut entreprendre pour tout contexte dans lequel il est susceptible de se trouver. D'autre part, il doit permettre à l'agent de déterminer à chaque pas de temps quelle action déclencher.

Nous avons vu que, dans le cadre classique de l'apprentissage par renforcement, la solution du second problème découle directement de la solution du premier en associant une qualité estimée à chaque couple (état, action) et en choisissant à chaque pas de temps l'action dont la qualité estimée est la plus grande.

Dans le cadre des systèmes de classeurs, si le schéma d'ensemble reste le même, l'usage du symbole # pour obtenir une représentation plus compacte induit un certain nombre de complications.

Le problème général provient de ce que la correspondance entre des états et des parties `[Condition]` dans lesquelles apparaissent des # n'est pas triviale : une même partie `[Condition]` peut désigner plusieurs états et plusieurs parties `[Condition]` peuvent désigner le même état.

Une première préoccupation apparaît donc, qui consiste à rechercher l'ensemble de règles couvrant l'ensemble des états possibles de la façon la plus compacte possible, compte tenu de l'objectif général qui est de produire un agent dont le comportement converge vers un optimum. La prise en compte de cette préoccupation conduit à définir des mécanismes de généralisation et de spécialisation, que nous allons décrire dans la section suivante.

Une seconde préoccupation consiste à se demander à quel classeur se fier dans un état donné, puisque plusieurs classeurs peuvent voir leur partie `[Condition]` vérifiée pour le même état. Cela suppose de mettre en œuvre un mécanisme de sélection de l'action, que nous décrirons par la suite.

Généralisation et spécialisation

La notion d'état telle qu'elle est définie dans le cadre des processus de décision markoviens ne permet pas de rendre compte de l'idée qu'il y a des situations analogues dans lesquelles l'action à réaliser est analogue. Á chaque situation différente est associée un état différent et à chaque état une réponse, si bien que toute idée de proximité entre les situations est perdue. Dans le cadre des systèmes de classeurs, on prend en compte le fait que, au sein de ce qu'il perçoit, un classeur ne réagit qu'à un ensemble limité d'attributs de la situation. Deux situations sont alors considérées comme équivalentes lorsque les observables auxquelles le classeur est sensible ont les mêmes valeurs dans ces situations.

Cette propriété fondamentale du formalisme général des systèmes de classeurs vis-à-vis du cadre classique des modèles de Markov dérive de la présence de symboles # dans la partie `[Condition]` des classeurs.

Un classeur qui dispose d'un # pour l'une des observables de sa partie `[Condition]` est général au sens où il subsume tous les classeurs qui prennent une valeur spécifique donnée pour cette même observable et qui spécifient les mêmes valeurs pour les autres observables et la même action à accomplir.

[26] Par exemple, MACS [Gérard et al., 2005]
[27] Par exemple, XCSF [Wilson, 2001]
[28] *S-expression*
[29] [Lanzi, 2002]

Cette propriété, reconnue comme la spécificité centrale des systèmes de classeurs vis-à-vis des autres formalismes permettant de mettre en œuvre de l'apprentissage par renforcement, est identifiée comme une capacité de *généralisation*[30]. Un classeur qui ne possède que des # dans sa partie `[Condition]` et spécifie une certaine action à accomplir stipule que cette action doit être accomplie en toute généralité, quelle que soit la situation effective de l'agent.

Pour la plupart des problèmes, un classeur dont la partie `[Condition]` est constituée uniquement de # sera sans doute trop général. En effet, il est rare que la même action soit la meilleure chose à faire quelle que soit la situation. Au contraire, les classeurs qui spécifient trop de contraintes sur la valeur des observables sont trop spécifiques. Dans ce cas, en effet, il existe sans doute un classeur plus général qui permet de prendre la même décision sans se tromper. La préoccupation consistant à associer le bon degré de généralité à chaque classeur est donc cruciale pour la mise au point d'un système de classeurs.

Le fait qu'un classeur soit trop général se manifeste par le fait que l'action qu'il propose a tantôt des conséquences positives, tantôt des conséquences négatives sur le comportement global de l'agent. Il faut alors préciser davantage l'ensemble des situations dans lesquelles l'accomplissement de l'action est souhaitable, afin d'éviter son déclenchement dans les situations défavorables. Cela se traduit par l'introduction de valeurs spécifiques là où la partie `[Condition]` contenait un #. Une telle opération revient à disjoindre l'espace des situations auxquelles les descriptions des nouvelles parties `[Condition]` font référence.

Le fait qu'un classeur soit trop spécifique est plus difficile à détecter, puisque la règle stipulée est correcte dans l'ensemble des situations qu'il couvre. De nombreux systèmes reposent sur un algorithme génétique qui prend en compte le degré de généralité des classeurs pour sélectionner de préférence des classeurs généraux.

Le système MACS que nous allons présenter à la section 3.4, page 63, se donne pour indice qu'un classeur est trop spécifique le fait qu'il existe un autre classeur plus général qui couvre un sur-ensemble des situations couvertes par le classeur et dont le déclenchement est au moins aussi favorable à l'agent. Une stratégie pour atteindre le degré de généralité optimal consiste alors à éliminer les classeurs trop généraux qui conduisent l'agent à faire des erreurs en les spécialisant et à éliminer les classeurs trop spécifiques quand il existe un classeur plus général qui donne d'aussi bons résultats.

Choix de l'action

En raison de la capacité de généralisation des classeurs, il se peut que plusieurs classeurs différents voient leur partie `[Condition]` vérifiée dans la même situation. Il se peut en outre que ces différents classeurs stipulent des actions différentes. Il faut donc adjoindre à un système de classeurs un mécanisme d'arbitrage qui décide de l'action qui sera effectivement accomplie par l'agent en fonction des classeurs dont la partie `[Condition]` est vérifiée dans la situation courante.

Pour résoudre un tel problème, on appelle « ensemble des classeurs activables » [31] l'ensemble des classeurs susceptibles d'être activés dans une situation donnée. Dans tous les systèmes de classeurs, la première étape du choix de l'action consiste à construire l'ensemble des classeurs activables. Ensuite, différents mécanismes d'arbitrage sont possibles, mais tous reposent sur la qualité associée aux différents classeurs. Certains systèmes activent simplement le classeur doté de la plus grande qualité, d'autres font la somme des qualités associées à chacune des actions possibles par les classeurs candidats et choisissent l'action dont la somme associée est la plus élevée. Récemment, M. Butz a montré que l'on obtenait une augmentation significative de la performance du système XCS

[30] [Lanzi, 2002]
[31] *match-set*

en choisissant l'action par un mécanisme de tournoi [32] au sein de l'ensemble des classeurs activables.

Le point commun à tous les systèmes de classeurs est que la qualité d'un classeur est égale à l'espérance de récompense associée à l'action stipulée par le classeur dans tous les états dans lesquels sa partie `[Condition]` est vérifiée. Pour calculer cette espérance, les systèmes de classeurs font appel aux algorithmes d'apprentissage par renforcement que nous avons décrits au chapitre précédent. Ce choix permet de garantir que l'agent dirigé par le système de classeurs verra son comportement converger vers le comportement optimal au fur et à mesure que la qualité associée à chaque classeur s'affine.

Les systèmes de classeurs standards

Les systèmes que nous qualifions de « standards » s'écartent à la fois des systèmes de classeurs à anticipation que nous allons décrire dans la suite de ce chapitre et des systèmes de classeurs « historiques », proposés par J. Holland. Ces systèmes ont été inventés par S. Wilson [33] et nous avons vu qu'ils sont plus simples que les systèmes précédents. Comme les systèmes originaux de J. Holland [34], ils reposent sur une combinaison d'apprentissage par renforcement et d'algorithmes génétiques, mais ils ne contiennent pas de liste de messages.

Tout d'abord, à chaque classeur est associé une estimation de la récompense que l'on peut attendre si l'on déclenche l'action stipulée dans les états dans lesquels la partie `[Condition]` est vérifiée. Cette estimation est obtenue par un algorithme d'apprentissage par renforcement, comme nous l'avons décrit au chapitre 2.

Par ailleurs, la spécialisation et la généralisation des classeurs existants sont entièrement réalisées par un algorithme génétique qui fait évoluer les parties `[Condition]` des classeurs dans l'espoir de trouver des règles à la fois plus efficaces et plus générales. On fait donc confiance à une heuristique de type essais et erreurs pour converger vers le bon degré de généralité.

L'algorithme génétique fonctionne sur une population de classeurs de taille limitée et induit entre ces classeurs une compétition dans laquelle seuls les plus adaptés subsistent, les plus faibles étant éliminés au profit de nouveaux individus qui entrent dans la compétition.

Dans un système comme ZCS, la *fitness* des classeurs est directement égale à la qualité du classeur, donc à l'estimation de récompense attendue calculée par l'apprentissage par renforcement. Ainsi, l'algorithme génétique élimine les classeurs qui conduisent l'agent à atteindre moins de récompenses que d'autres.

Le mécanisme d'XCS est plus subtil. Les classeurs déclenchés sont choisis eux aussi en fonction de l'estimation de récompense attendue, mais l'algorithme génétique s'applique sur un critère différent. Il évalue les classeurs en fonction de leur capacité à prédire exactement la récompense qu'ils vont recevoir. Ainsi, des classeurs dont la partie `[Condition]` est vérifiée à la fois dans des états où la récompense sera faible et dans des états où elle sera forte ne parviennent pas à prédire correctement la récompense qu'ils vont recevoir, si bien qu'ils sont éliminés. Ainsi, seuls sont conservés les classeurs dont la partie `[Condition]` est vérifiée pour des états dans lesquels la récompense à long terme sera sensiblement homogène.

La différence entre ZCS et XCS est que le second conserve au sein de sa population les classeurs qui se déclenchent loin d'une source de récompense, et prévoient donc une récompense faible, pourvu qu'il la prévoie exactement. Ainsi, la couverture de l'espace d'état est plus complète avec XCS qu'avec ZCS.

[32] *tournament selection*
[33] [Wilson, 1994, Wilson, 1995]
[34] [Holland & Reitman, 1978]

3.1.4 Récapitulation

Nous venons de dire que la plupart des systèmes de classeurs associent des algorithmes d'apprentissage par renforcement à des algorithmes génétiques, mais nous allons voir que ce n'est pas le cas des systèmes que nous allons présenter dans la suite. Il faut surtout retenir de ce qui précède que les systèmes de classeurs sont des systèmes d'apprentissage par renforcement spécifiquement conçus pour apprendre des comportements optimaux et trouver une représentation compacte de la solution dans des processus de décision markoviens dont l'état est représenté sous la forme d'une collection d'observables.

Ils répondent ainsi à la première limitation du cadre de l'apprentissage par renforcement que nous avons énoncée en conclusion du chapitre précédent. Avant de comparer au prochain chapitre ce mode de représentation compact à celui proposé par les réseaux bayésiens dynamiques, nous allons nous tourner vers la seconde difficulté identifiée au chapitre précédent, à savoir la lenteur de l'apprentissage en ligne et l'intérêt d'apprendre un modèle des interactions entre l'agent et son environnement.

Ce point nous permettra de nous rapprocher considérablement des réseaux bayésiens dynamiques, ce qui facilitera la comparaison. Nous allons donc présenter dans ce qui suit des systèmes de classeurs qui apprennent un modèle des interactions avec leur environnement, à savoir les systèmes de classeurs à anticipation.

3.2 L'apprentissage latent

3.2.1 Sources psychologiques

La seconde limitation à laquelle nous allons nous intéresser dans cette section concerne la lenteur de l'apprentissage. En effet, les algorithmes d'apprentissage par renforcement propagent très lentement la valeur des différentes situations. La valeur d'une situation donnée est mise à jour uniquement lorsqu'un agent réalise effectivement une transition à partir de cette situation, et la mise à jour s'effectue uniquement en fonction de l'estimation courante de la valeur de la situation d'arrivée. En conséquence, comme nous l'avons illustré avec la figure 2.3, page 43, pour que la valeur se propage depuis une source de récompense jusqu'à une situation placée à une distance de n états de cette source, il faut que l'agent réalise au moins n trajectoires entre la situation en question et la source de récompense. Cela est loin d'être optimal.

Nous avons indiqué à la section 2.4.4, page 43, qu'il était possible d'atténuer les conséquences de ce problème en ayant recours à des traces d'éligibilité. Toutefois, il existe une autre approche de ce problème qui conduit à une remise en question beaucoup plus profonde du cadre général de l'apprentissage par renforcement et qui repose sur un courant épistémologique sensiblement distinct. C'est ce courant que nous allons évoquer à présent.

L'apprentissage par renforcement tabulaire associe une qualité à chaque action dans chaque état dans lequel peut se trouver un agent. Une fois l'apprentissage réalisé, le formalisme permet à un agent de choisir l'action de meilleure qualité dans chaque état. De même, les systèmes de classeurs que nous avons présentés jusqu'ici associent une réponse à une situation, représentée sous la forme d'un vecteur d'observable, que l'on peut assimiler à des stimuli. À ce titre, ce formalisme semble tomber sous le coup du reproche adressé au behaviorisme d'avoir limité ses investigations à la recherche d'associations entre des stimuli et des réponses.

Une approche du comportement animal résolument opposée au behaviorisme et défendue par E. Tolman est parvenue à mettre en évidence des phénomènes d'apprentissage trop complexes pour être attribués au conditionnement classique et dans lesquels, pourtant, aucune récompense n'intervient dans l'apprentissage. E. Tolman [35] présentait l'apprentissage comme un processus d'acquisition d'un modèle interne des relations entre situations et

[35] [Tolman, 1932]

actions. Là où le béhaviorisme supposait le renforcement du déclenchement d'une réponse à partir d'un stimulus donné, E. Tolman considérait plutôt la mémorisation de relations indiquant que telle réponse déclenchée dans telle situation débouche sur telle autre situation.

La plupart des expériences étaient menées sur des rats conduits à explorer des labyrinthes. J. Seward [36], par exemple, a réalisé l'expérience suivante. On commence par laisser à un rat l'opportunité d'explorer un labyrinthe dans lequel il n'y a pas de nourriture, mais il y a une zone d'une couleur spécifique, facile à distinguer. Puis on place de la nourriture dans la zone en question et on introduit le rat dans le labyrinthe après lui avoir montré que la zone colorée contient de la nourriture. On observe alors que le rat se rend là où se trouve la nourriture plus vite que s'il n'avait pas eu l'occasion de se promener dans le labyrinthe auparavant. E. Tolman conclut de cette expérience que le rat a appris la configuration du labyrinthe en l'absence de récompense, lors de la phase d'exploration initiale. Ce phénomène a été dénommé « *apprentissage latent* ». Pour l'expliquer, E. Tolman a développé une *théorie de l'anticipation* [37] qui consiste à considérer, au lieu de simples associations *Stimulus* → *Réponse*, des associations plus complexes de la forme *Stimulus* → *Réponse* → *Stimulus*. Dans le cadre des conditionnements pavlovien et opérant, l'apprentissage prend en compte les conséquences des actions du sujet, mais ces conséquences ne sont considérées que relativement aux récompenses ou punitions reçues. Dans le modèle proposé par E. Tolman, au contraire, la situation résultant de chaque action est représentée dans sa totalité.

J. Hoffmann est un second psychologue convaincu de l'insuffisance du paradigme béhavioriste et qui a joué un grand rôle dans l'élaboration du modèle d'apprentissage latent que nous allons présenter. Faisant remonter sa théorie à des intuitions initiales de J. Herbart [38], J. Hoffmann a bâti une théorie de l'apprentissage appelée « théorie du contrôle anticipatif du comportement » [39].

Cette théorie stipule que la sensibilité aux attributs observables d'une situation en vue de l'action est régie principalement par l'anticipation. Plus précisément, les anticipations incorrectes conduisent à une différenciation des situations perçues – si le sujet a mal anticipé, c'est qu'il n'a pas distingué suffisamment entre deux situations qui lui paraissaient identiques mais ne l'étaient pas –, tandis que les anticipations correctes conduisent à un renforcement du lien entre la situation anticipée et la situation initiale. La pertinence d'une observable donnée dans une situation donnée n'est donc plus déterminée relativement à un but à atteindre, elle est déterminée en fonction de la nécessité de distinguer une situation particulière parmi plusieurs dans l'élaboration d'un modèle fidèle des interactions du sujet avec son environnement. Il s'agit donc pour l'agent d'un processus de différenciation entre diverses situations en fonction de la structure des contingences entre ces situations.

Les systèmes de classeurs à anticipation que nous allons présenter à la section 3.3.2, page 61, sont directement issus d'une modélisation informatique de la théorie du contrôle anticipatif du comportement, puisque W. Stolzmann, leur inventeur, était un étudiant de J. Hoffmann. Nous allons voir que, grâce à leur capacité de généralisation et de spécialisation, les systèmes de classeurs possèdent le formalisme adéquat pour représenter le processus de différenciation mis en évidence par J. Hoffmann.

Cette différenciation est rendue possible par le fait que la situation à laquelle est confrontée l'agent est perçue sous la forme d'un ensemble d'observables dont quelques-unes seulement sont utilisées par l'agent pour déterminer son action ultérieure.

Nous allons voir que l'opérateur qui rend compte de la différenciation consistent à spécialiser les conditions dans lesquelles une action doit être déclenchée en étant sensible à davantage d'observables. Cette spécialisation opère lorsque les anticipations du système s'avèrent erronées.

[36] [Seward, 1949]
[37] *expectancy theory*
[38] [Herbart, 1825]
[39] *Anticipatory Behavioral Control*

3.3 Modèles informatiques de l'apprentissage latent

3.3.1 Les architectures DYNA

La théorie d'E. Tolman n'est pas restée sans écho du côté de la simulation informatique. En effet, au sein de la communauté des chercheurs qui élaborent des algorithmes d'apprentissage par renforcement, les limites induites par le cadre trop restreint *Stimulus* → *Réponse* n'ont pas tardé à se faire sentir et à être bousculées. Ainsi, M. Witkowski [40] identifie les premières réalisations informatiques inspirées du modèle d'E. Tolman : JCM, de J. Becker [41], ALP, de D. Mott [42] et le travail de G. Drescher [43], plus directement inspiré de la théorie piagétienne du développement de l'enfant.

Tous les systèmes qui appartiennent à cette famille sont motivés par le même constat. Plutôt que d'attendre qu'un agent effectue des transitions dans son environnement réel, une façon d'accélérer la propagation de la qualité des situations consiste à construire un modèle des transitions réalisées par l'agent et à se servir de ce modèle pour appliquer un algorithme de propagation indépendamment du comportement de l'agent. En effet, quand un agent interagit avec son environnement, ses actions ne débouchent pas seulement sur une possible punition ou récompense, mais aussi sur une situation ultérieure. L'agent peut donc construire un modèle des transitions dont il fait l'expérience, indépendamment de tout modèle des récompenses. Ce procédé a été nommé « apprentissage par renforcement indirect » [44].

L'idée de construire un modèle des transitions par apprentissage a été réalisée dans le cadre de l'apprentissage par renforcement tabulaire avec les architectures DYNA [45]. En effet, R. Sutton a proposé cette famille d'architectures pour doter un agent de la capacité à mettre à jour plusieurs qualités lors du même pas de temps, indépendamment de la situation courante de l'agent, de façon à accélérer sensiblement l'apprentissage de la fonction qualité [46]. Cette capacité est réalisée en appliquant au modèle un nombre fixé d'étapes de planification, qui consistent à appliquer un algorithme de programmation dynamique [47] au sein du modèle de l'agent.

En parfait accord avec ce que nous avons dit du modèle proposé par E. Tolman, plutôt que d'exprimer simplement le choix d'une action, le modèle des transitions des architectures DYNA exprime de façon explicite ce que sera l'état du monde perçu par l'agent lors de l'instant suivant s'il effectue une certaine action. Pour exprimer la même idée dans le cadre formel de l'apprentissage par renforcement, l'apprentissage latent consiste pour un agent, moyennant une mémoire de l'état dans lequel il était lors du pas de temps précédent, à mémoriser l'ensemble des transitions (s_t, a_t, s_{t+1}) par lesquelles il passe pendant le déroulement de ses interactions avec son environnement.

Les architectures DYNA, présentées sur la figure 3.2, construisent un modèle des transitions effectuées par l'agent dans son environnement et un modèle des récompenses qu'il peut y obtenir.

Le modèle des transitions prend la forme d'une liste de triplets (s_t, a_t, s_{t+1}) qui indiquent que, si l'agent effectue l'action a_t dans l'état s_t, il atteindra immédiatement l'état s_{t+1}.

Des transitions de la forme (s_t, a_t, s_{t+1}) constituent un modèle interne des interactions d'un agent avec son environnement. Au lieu d'apprendre que tel stimulus doit être suivi de telle action, l'agent apprend alors que, s'il effectue telle action après avoir reçu tel stimulus, alors il doit s'attendre à recevoir tel autre stimulus lors du pas de temps suivant. En conséquence, le modèle interne est indépendant de toute notion de récompense assignée à

[40] [Witkowski, 1997]
[41] [Becker, 1970]
[42] [Mott, 1981]
[43] [Drescher, 1991]
[44] *model-based reinforcement learning*
[45] [Sutton, 1990]
[46] Voir le chapitre précédent
[47] voir la section 2.3.1, page 37

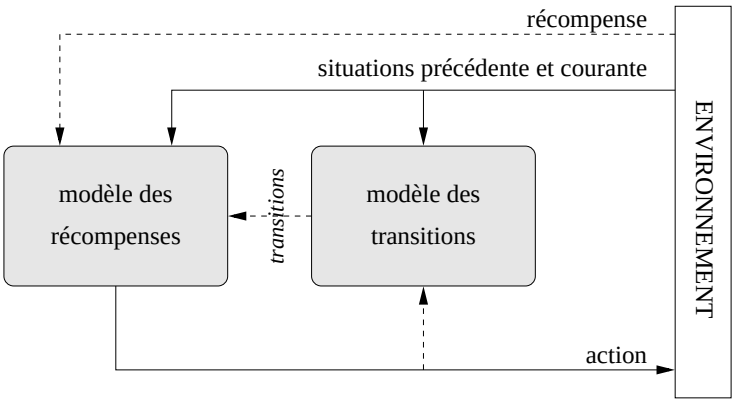


FIG. 3.2 – **Les architectures DYNA combinent un modèle des récompenses et un modèle des transitions. Le modèle des transitions est utilisé pour accélérer le processus d'apprentissage par renforcement. La construction des deux modèles requiert une mémorisation de la situation précédente, ce qui n'est pas explicite sur la figure.**

l'agent. En effet, plutôt que d'exprimer ce qu'il faut faire pour recevoir une certaine récompense sans préciser dans quel état elle sera reçue, le nouveau formalisme exprime des connaissances sur la dynamique d'évolution de l'état en fonction des actions de l'agent.

Une fois ce modèle construit par apprentissage, on peut appliquer des algorithmes de programmation dynamique tels que l'itération sur la valeur ou sur la politique [48] pour accélérer l'apprentissage de la fonction de qualité. Ainsi, le modèle des transitions peut être utilisé indépendamment de la situation courante de l'agent pour propager les qualités de différentes situations à partir de diverses sources de récompense. En pratique, ce processus de propagation consiste à réaliser des actions simulées à partir de situations fictives, éventuellement plusieurs fois par pas de temps.

Lorsque l'agent est un robot réel dont chaque action effective peut être longue et difficile à mettre en œuvre, voire dangereuse pour son fonctionnement, réaliser des transitions simulées dans un modèle interne est beaucoup moins coûteux pour propager des qualités que refaire chaque action laborieusement. En outre, si les buts attribués à l'agent changent, ce qui a pour effet de modifier les récompenses qu'il recevra, alors l'agent est capable de reconfigurer immédiatement sa stratégie de comportement plutôt que de « désapprendre » tout ce qu'il a péniblement appris et d'apprendre à nouveau une autre stratégie comportementale.

Par ailleurs, la construction d'un tel modèle peut doter un agent de capacités d'anticipation. En effet, s'il enregistre les associations entre les états dans lesquels il se trouve d'une part et les récompenses qu'il reçoit d'autre part, l'agent peut choisir son action en fonction du caractère désirable pour lui de l'état auquel cette action doit le conduire. Au lieu d'avancer aveuglément vers une récompense attendue, l'agent peut alors mettre en œuvre des capacités de planification à plus long terme. En parcourant par un regard en avant le graphe des transitions d'état en état dont il dispose, l'agent devient capable de prévoir l'évolution de ses interactions avec son environnement en fonction de ses actions après un nombre indéfini d'actions. Si bien que, s'il veut atteindre un but dans un certain état, il peut rechercher au sein du graphe la séquence d'action qui le conduit à ce but, avant d'effectuer la séquence d'actions correspondante.

Les architectures DYNA constituent une famille de systèmes qui vérifient les principes que nous venons de décrire. Au sein de cette famille, on distingue cependant des variations sur l'algorithme d'apprentissage ou la mé-

[48] voir la section 2.3.1, page 38

thode d'exploration utilisés. Le système original, DYNA-PI [49] repose sur un algorithme d'itération sur les politiques. R. Sutton [50] montre que ce système est moins flexible que le système DYNA-Q, qui repose sur le Q-LEARNING. Il propose en outre une version dotée de capacités d'exploration active, nommée DYNA-Q+ et présente la différence de performance entre ces trois systèmes sur des environnements changeants dans lesquels soit des chemins optimaux sont bloqués, soit, au contraire, des raccourcis apparaissent.

Enfin, tous ces systèmes reposant sur un algorithme de programmation dynamique pour propager efficacement la fonction de valeur ou la fonction de qualité, il est possible de rendre cette programmation dynamique plus efficace avec un algorithme de « balayage prioritaire » tel que PRIORITIZED SWEEPING [51], qui met à jour en priorité les valeurs des états dans lesquels l'agent est susceptible de se trouver, ou d'autres variantes telles que FOCUSED DYNA [52], EXPERIENCE REPLAY [53] et TRAJECTORY MODEL UPDATES [54].

Le travail de R. Sutton consiste en une modification du cadre classique de l'apprentissage par renforcement prenant en compte la conséquence des actions. Malheureusement cette modification dans le cadre tabulaire implique une aggravation du problème que pose l'explosion du nombre d'entrées dans la table. En effet, les lignes de la nouvelle table contiennent des informations supplémentaires – la conséquence de chaque action – et sont aussi nombreuses. L'intérêt d'avoir recours à une représentation compacte est donc renforcée dans ce cadre.

3.3.2 Les systèmes de classeurs à anticipation

La capacité d'apprentissage de la plupart des systèmes de classeurs repose sur le *Q-learning*. Par conséquent, ils souffrent du même problème de lenteur de convergence que cet algorithme, si bien que les raisons d'avoir recours à un modèle des transitions sont aussi valides dans le cadre des systèmes de classeurs que dans le cadre tabulaire.

Cependant, la capacité à élaborer un modèle interne n'a pas immédiatement été recherchée en tant que telle mais, dans un premier temps, pour doter les systèmes de classeurs de capacités d'anticipation.

Tout d'abord, comme nous l'avons dit dans la section 3.1, page 49, S. Wilson [55] a reconnu l'importance du caractère prédictif des classeurs en remplaçant dans son nouveau système, XCS, le critère de sélection des classeurs, qui reflétait jusqu'alors l'espérance de récompense, par un critère mesurant la capacité des classeurs à prédire avec précision la récompense reçue à l'issue de leur déclenchement. XCS s'est révélé plus performant que les systèmes précédents.

C'est R. Riolo [56] qui semble être le premier à avoir présenté un système de classeurs intégrant une forme d'anticipation explicite. Son système, CFSC2, est directement inspiré du cadre original des systèmes de classeurs à messages internes proposé par J. Holland [57], dont il était un doctorant.

Dans un second temps, transposant en cela le travail de R. Sutton au cas des systèmes de classeurs, W. Stolzmann [58], qui a réalisé ses recherches dans le laboratoire de psychologie de J. Hoffmann, a proposé de remplacer les classeurs de la forme `[Condition]` → `[Action] (Qualité)` par des classeurs de la forme `[Condition] [Action]` → `[Effet] (Qualité)` La partie `[Effet]` représente les effets attendus de

[49] PI pour *Policy Iteration*. À noter que dans [Sutton & Barto, 1998], R. Sutton appelle ce même système DYNA-AC, avec AC pour *Actor Critic*

[50] [Sutton, 1990]

[51] [Moore & Atkeson, 1993]

[52] [Peng & Williams, 1992]

[53] [Lin, 1993]

[54] [Kuvayev & Sutton, 1996]

[55] [Wilson, 1995]

[56] [Riolo, 1991]

[57] [Holland & Reitman, 1978]

[58] [Stolzmann, 1998]

l'action dans toutes les situations qui vérifient la partie `[Condition]` du classeur. Ce formalisme est proche des règles de la forme `[contexte-action-résultat]` de Drescher [59]. Par ailleurs, on retrouve les transitions de la forme (s_t, a_t, s_{t+1}) des architectures DYNA, mais s'y ajoute la capacité de généralisation spécifique des systèmes de classeurs.

Au cœur du système de W. Stolzmann nommé ACS [60], l'algorithme ALP (*Anticipatory Learning Process*) est une formalisation de la théorie psychologique du contrôle anticipatif du comportement développée par J. Hoffmann [61]. La problématique de l'anticipation y joue donc un rôle central. Au fil des années, ACS a été enrichi par M. Butz pour devenir ACS2 [62]. Parallèlement, C. Witkowski, s'appuyant sur les arguments mis en avant par E. Tolman en faveur de l'apprentissage latent, proposait un système mettant en œuvre des règles de la même forme et disposant d'une capacité de généralisation similaire à celle des systèmes de classeurs, bien qu'elle soit exprimée différemment [63].

YACS [64] et MACS [65] sont les systèmes de classeurs à anticipation que nous avons conçus avec P. Gérard. Ils sont proches d'ACS et ACS2. Leurs classeurs disposent aussi d'une partie `[Condition]`, une partie `[Action]` et une partie `[Effet]`. Leur principale originalité consiste en une séparation plus claire entre deux processus d'apprentissage bien distincts : l'apprentissage d'un modèle des transitions et l'apprentissage d'un modèle des récompenses. C'est en développant YACS et MACS que nous avons établi plus explicitement le rapport entre l'apprentissage par renforcement indirect réalisé dans les architectures DYNA et les systèmes de classeurs à anticipation.

Nous avons dit que la spécificité des systèmes de classeurs réside dans leur capacité de généralisation. Dans les systèmes de classeurs comme XCS, la généralisation consiste à regrouper sous une même partie `[Condition]` les situations pour lesquelles l'espérance de récompense est similaire.

Dans un système de classeurs à anticipation, au contraire, il est possible de chercher à généraliser le modèle des transitions, en regroupant sous une même partie `[Condition]` les états pour lesquels la même action conduit au même effet.

Le constat qui a donné naissance à YACS et MACS est que ces deux processus de généralisation sont indépendants l'un de l'autre et gagnent à être effectués séparément. Il en résulte la possibilité de séparer complètement l'apprentissage du modèle des transitions et celui du modèle des récompenses[66].

Cette clarification se traduit concrètement par des performances sensiblement meilleures sur une série de bancs de test, puisque YACS a besoin de réaliser trois fois moins d'expériences qu'ACS pour apprendre un modèle complet des transitions qui s'avère en outre sensiblement plus compact.

À partir du point de vue qui vient d'être développé, les systèmes de classeurs à anticipation peuvent être vus comme combinant deux propriétés cruciales pour les systèmes d'apprentissage par renforcement. Comme les architectures DYNA, ils apprennent un modèle des transitions, ce qui les dote de propriétés d'anticipation et leur permet d'apprendre plus vite. Comme les systèmes de classeurs classiques, ils sont dotés d'une capacité de généralisation, ce qui leur permet de construire un modèle plus compact que les architectures DYNA tabulaires.

---

[59] [Drescher, 1991]
[60] [Stolzmann, 1998, Butz et al., 2000]
[61] [Hoffmann, 1993]
[62] [Butz, 2002]
[63] [Witkowski, 1997]
[64] [Gérard & Sigaud, 2001, Gérard, 2001, Gérard et al., 2002]
[65] [Gérard & Sigaud, 2003, Gérard et al., 2005]
[66] Plus précisément, dans YACS et MACS nous avons cherché à atteindre le degré maximal de généralisation dans le modèle des transitions, sans nous soucier de généraliser le modèle des récompenses. Récemment, M. Butz [67] a proposé un modèle qui utilise conjointement les deux formes de généralisation.

3.4 Présentation détaillée de MACS

3.4.1 Architecture générale

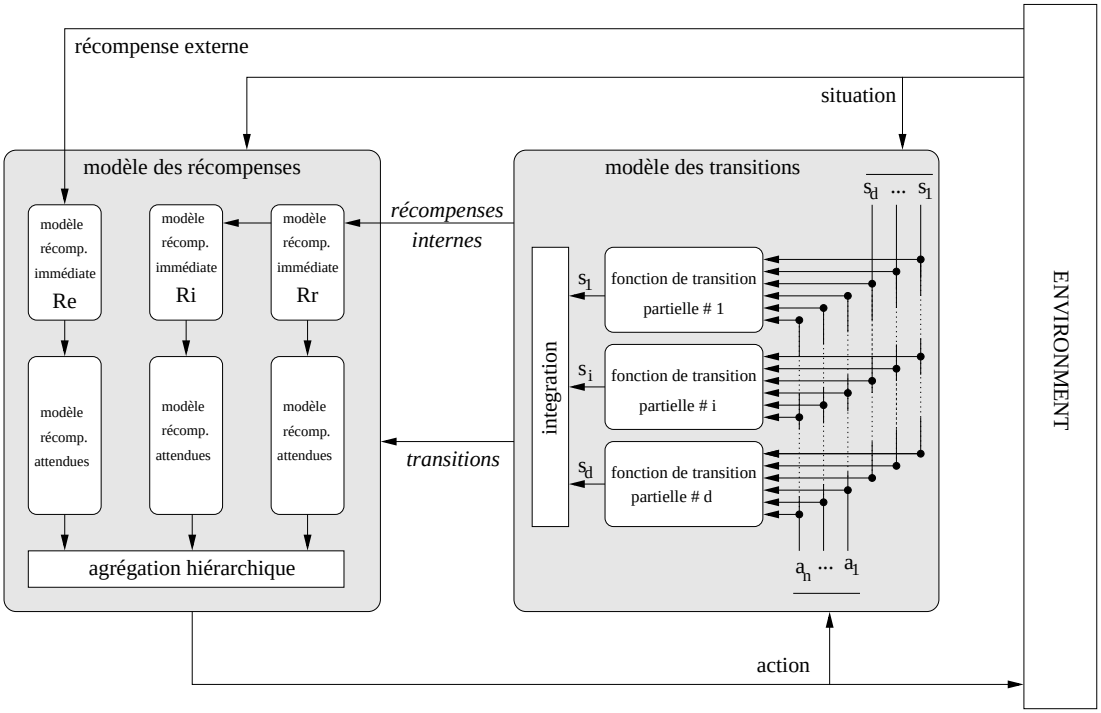


FIG. 3.3 – **L'architecture globale de MACS. On reconnaît le schéma général des architectures** DYNA **présenté sur la figure 3.2.**

L'architecture de contrôle générale de MACS apparaît sur la figure 3.3. Cette figure a été réalisée de façon a mettre en évidence la similitude d'organisation entre cette architecture et celle des systèmes de la famille DYNA.

De même que son prédécesseur, YACS, MACS est un système de classeurs à anticipation qui présente la particularité de ne pas faire appel à des algorithmes génétiques, mais à des heuristiques plus spécialisées visant à maximiser son efficacité.

La contrepartie induite par le recours à ces heuristiques est une complexité plus grande du logiciel correspondant et un domaine d'application plus restreint. Les heuristiques ont été définies spécifiquement pour traiter des problèmes déterministes plutôt que stochastiques et, comme P. Gérard y insiste dans sa thèse [68], l'extension à la résolution de problèmes dans lesquels l'hypothèse de Markov n'est pas vérifiée s'avère particulièrement problématique.

Nous allons décomposer la présentation de MACS en quatre points :

– Tout d'abord, nous allons présenter la principale innovation du formalisme de représentation de MACS vis-à-vis de ses prédécesseurs, ACS et YACS, et indiquer les modifications induites par cette innovation dans les mécanismes de fonctionnement de MACS.
– Ensuite, nous allons montrer comment MACS élabore un modèle des transitions compact à l'aide d'heuristiques de généralisation et de spécialisation.

[68] [Gérard, 2002]

– Puis nous présenterons le modèle des récompenses élaboré par MACS pour réaliser un mécanisme d'exploration active.
– Enfin, restera à présenter le mécanisme de choix de l'action qui tire parti de ces deux modèles pour choisir à tout instant l'action entreprise par l'agent.

3.4.2 Formalisme de MACS

Dans ACS, ACS2 et YACS, la partie `[Effet]` de chaque classeur est organisée de façon à indiquer quels sont les attributs qui changent et quels sont ceux qui ne changent pas si l'on applique l'action dans une situation compatible avec la partie `[Condition]` du classeur. À cette fin, les parties `[Effet]` peuvent contenir un symbole "`=`", qui signifie que l'attribut ne change pas [69]. Par exemple, le classeur `[#0#1] [0] [=10=]` appliqué à la situation `[1031]` prédit que la situation résultant de son application sera `[1101]`. Appliqué à `[2011]`, il prédit `[2101]`.

Ce formalisme permet de représenter des régularités comme par exemple *« quand l'agent perçoit un mur au nord, quoi qu'il perçoive dans les autres directions, se déplacer vers le nord le conduit à rester dans la même case, donc aucun changement ne sera perçu dans sa situation »*, ce qui s'exprime par le classeur suivant : `[1#######] [Nord] [========]`.

L'originalité la plus manifeste de MACS provient de la transformation opérée sur la sémantique de la partie `[Effet]` des classeurs. Alors que les systèmes précédemment cités ne peuvent exprimer une relation qu'entre la valeur d'un attribut et la valeur du même attribut au pas de temps suivant, il existe des régularités qui portent sur des relations entre les valeurs de plusieurs attributs différents lors de deux pas de temps successifs.

Par exemple, si l'agent représenté sur la figure 3.1 se déplace vers l'Est, le mur qu'il voyait au Nord-Est est à présent au Nord. Dans MACS, on représente une régularité de ce type avec un classeur de la forme `[#1#######] [Est] [1????????]`, où le symbole "`?`" signifie que le classeur ne sait pas prédire la valeur de l'attribut considéré. L'introduction de ce nouveau symbole permet donc de prédire séparément la valeur des différents attributs au pas de temps suivant.

Dans MACS toujours, nous avons fait le choix de prédire la valeur de chaque attribut indépendamment de celle des autres. Comme cela apparaît sur la figure 3.3, cela permet de décomposer le modèle des transitions complet en un ensemble de modèles dont chacun réalise une fonction qui prédit la valeur d'un attribut particulier. L'avantage de ce nouveau formalisme est qu'il permet d'exprimer un nombre de régularités beaucoup plus grand que le précédent, si bien que le modèle des transitions que l'on obtient est plus compact et s'apprend plus vite. Nous verrons au chapitre suivant que le formalisme des réseaux bayésiens dynamiques fait exactement le même choix de représentation.

Intégration des anticipations partielles

Chaque classeur de MACS ne prédit la valeur que d'une observable à la fois. Même si l'on ne prend en compte pour prédire la situation courante que les classeurs qui ont anticipé correctement jusque là, il se peut que certains s'avèrent erronés pour la situation en question, si bien que la liste des classeurs activables produit pour chaque observable non pas une valeur, mais une liste de valeurs que cette observable est susceptible de prendre.

Si une telle liste est vide, alors on est dans l'incertitude totale pour cette observable. Si la liste ne contient qu'une valeur, on s'attend à ce que l'observable prenne la valeur prédite. Si elle contient plusieurs valeurs, alors apparaît

[69] [Gérard et al., 2002]

[01020100]	←	*Situation*
[0#######]	[Est]	[???????0]
[#1######]	[Est]	[1???????]
[##02####]	[Est]	[????2???]
[###2####]	[Est]	[????1???]
[####0###]	[Est]	[?????0??]
[#####1##]	[Est]	[?1??????]
[######0#]	[Est]	[??0?????]
[#######0]	[Est]	[???1????]
[########]	[Est]	[??????0?]
Anticipations	→	[11012000]
	ou	[11011000]

TAB. 3.1 – **Exemple d'intégration des anticipations partielles. Le mécanisme construit toutes les situations anticipées possibles en fonction des valeurs des attributs spécifiés dans les classeurs compatibles avec la situation courante. Ici, le système prédit que, en faisant l'action [Est] dans la situation [01020100], on atteint soit la situation [11012000], soit la situation [11011000]. Si tous les classeurs étaient précis, on n'obtiendrait qu'une seule situation anticipée possible.**

une combinatoire d'où résulte une liste des situations anticipées possibles. On peut cependant filtrer ces situations en ne prenant en compte que celles qui ont été effectivement rencontrées par l'agent jusque-là.

Dans MACS, pour déterminer la liste des situations anticipées possibles, plutôt que d'engendrer la combinatoire puis filtrer *a posteriori*, nous avons fait le choix de déterminer pour chacune des situations rencontrées jusque-là si elle fait partie des anticipations possibles.

Une fois construite la liste des situations anticipées possibles, reste à choisir celle que l'agent va tenter d'atteindre sur la base des critères précédemment décrits. C'est ce que nous présentons dans la section suivante.

Il faut noter que l'on associe à chacune des situations anticipées construites par cet algorithme l'action qui permet de l'atteindre, afin de simplifier le choix de l'action une fois choisie la situation visée.

3.4.3 Apprentissage du modèle des transitions

Le processus d'apprentissage du modèle des transitions réalisé dans YACS et MACS s'apparente à celui qui est réalisé dans les architectures DYNA, à ceci près que le modèle appris est constitué de classeurs aussi généraux que possible. Compte tenu de l'introduction d'une problématique de généralisation, le modèle des transitions doit être à la fois aussi général que possible et aussi précis que possible, ce qui nous conduit à poser les définitions suivantes :

- un classeur est dit « maximalement précis » s'il ne fait aucune erreur de prédiction, c'est-à-dire si sa partie `[Effet]` prédit sans erreur la situation suivante dans tous les contextes où sa partie `[Condition]` est vérifiée.
- un classeur est dit « maximalement général » si sa partie `[Condition]` ne pourrait pas être plus générale sans perdre en précision.

Afin d'obtenir une liste de classeurs maximalement précis et généraux, la plupart des systèmes de classeurs reposent sur un algorithme génétique, dans lequel la pression de sélection favorise les classeurs les plus généraux.

Notre approche est différente. Pour obtenir un tel modèle, nous nous efforçons d'obtenir le bon degré de généralité grâce à la combinaison de deux heuristiques :

- on applique aux classeurs qui ne sont pas assez précis une heuristique de *spécialisation des conditions* ;

– on applique aux classeurs qui ne sont pas assez généraux une heuristique de *généralisation des conditions*.

L'utilisation conjointe de ces deux heuristiques permet de faire converger la généralité et la précision des classeurs vers un niveau optimal. Chacune des heuristiques repose sur une mémoire partielle des dernières prédictions correctes et incorrectes de chaque classeur, ainsi que sur des estimations destinées à déterminer si un classeur donné doit être généralisé davantage ou spécialisé davantage et, le cas échéant, sur quel attribut doit porter la modification.

Une troisième heuristique, la couverture des transitions [70], est utilisée pour remédier aux cas où aucun classeur ne s'applique dans une situation donnée. Dans les sections suivantes, nous allons présenter successivement chacune de ces trois heuristiques.

Évaluation et sélection des classeurs précis

Afin d'évaluer les classeurs constituant le modèle des transitions et de supprimer ceux qui doivent l'être, deux entiers `g` [71] et `b` [72] sont associés à chaque classeur :

– `g` contient le nombre de bonnes anticipations du classeur depuis sa création ;

– `b` contient le nombre d'anticipations erronées du classeur depuis sa création.

À tout instant, le système conserve en mémoire la dernière situation perçue s_{t-1} et la dernière action effectuée a_{t-1}. Avec cette information, il parcourt la liste des classeurs et sélectionne tous ceux dont la partie `[Condition]` est vérifiée par s_{t-1} et dont la partie `[Action]` spécifie a_{t-1}. Pour chacun de ces classeurs, il vérifie si la partie `[Effet]` anticipe correctement la situation courante s_t [73].

– si le classeur anticipe correctement, `g` est augmenté de 1 ;

– si le classeur anticipe mal, `b` est augmenté de 1 ;

Un classeur qui anticipe systématiquement mal à la suite d'un nombre e_r d'évaluations successives est supprimé. En général, on prend $e_r = 5$ dans toutes les expériences réalisées avec YACS et MACS.

Spécialisation des conditions

On dit d'un classeur qui anticipe parfois bien et parfois mal qu'il oscille. Une telle situation traduit le fait que la partie `[Condition]` du classeur est trop générale et doit être spécialisée. Le processus de spécialisation s'appuie sur des estimations i_s associées à chaque attribut de la partie `[Condition]` des classeurs portant la valeur #. Ces estimations mesurent le gain qu'il y aurait à spécialiser le classeur selon l'attribut auquel elles sont associées. Ainsi, lorsque le système décide de spécialiser un classeur, il le spécialise selon l'attribut dont l'estimation i_s est la plus élevée.

L'algorithme général pour calculer i_s est le suivant. Si la valeur d'un attribut particulier est systématiquement différente selon que le classeur anticipe bien ou mal, alors cet attribut est particulièrement pertinent pour distinguer les situations dans lesquelles le classeur anticipe bien de celles dans lesquelles il anticipe mal. Donc, dans un tel cas la valeur de i_s doit être très élevée.

Plus précisément, pour calculer i_s, chaque classeur mémorise la situation s_b précédent la dernière erreur d'anticipation et la situation s_g précédent la dernière anticipation correcte. Par ailleurs, pour tout classeur dont la partie `[Condition]` est vérifiée par s_{t-1} et dont la partie `[Action]` spécifie a_{t-1}, on met à jour la valeur de i_s de l'attribut correspondant selon une règle de Widrow-Hoff [74] :

[70] *transition covering*

[71] pour *good*

[72] pour *bad*

[73] Cette vérification diffère selon qu'il s'agit de YACS ou MACS, la sémantique des parties `[Effet]` étant différente.

[74] [Widrow & Hoff, 1960]

- si le classeur anticipe correctement, pour chaque attribut :
  - si l'attribut correspondant dans s_b a la même valeur que l'attribut correspondant dans s_{t-1}, i_s est décrémenté : $i_s = (1 - \beta_s)i_s$
  - si l'attribut correspondant dans s_b diffère de l'attribut correspondant dans s_{t-1}, i_s est incrémenté : $i_s = (1 - \beta_s)i_s + \beta_s$
- si le classeur anticipe mal, pour chaque attribut :
  - si l'attribut correspondant dans s_g a la même valeur que l'attribut correspondant dans s_{t-1}, i_s est décrémenté : $i_s = (1 - \beta_s)i_s$
  - si l'attribut correspondant dans s_g diffère de l'attribut correspondant dans s_{t-1}, i_s est incrémenté : $i_s = (1 - \beta_s)i_s + \beta_s$

La valeur initiale de i_s est fixée à 0.5 et β_s est un paramètre du système.

L'opérateur de spécialisation est appliqué aussitôt que le système détecte qu'un classeur oscille. On dit d'un classeur qu'il oscille lorsque $\mathsf{g}+\mathsf{b} > e_o$ et $\mathsf{g}\times\mathsf{b} > 0$, où e_o est un paramètre du système qui représente le nombre d'évaluations nécessaires avant de décider qu'un classeur oscille.

Cet opérateur est inspiré de l'opérateur *mutspec* de M. Dorigo [75]. Il s'en distingue par deux points :

- l'attribut selon lequel on spécialise un classeur n'est pas choisi aléatoirement, mais en fonction des valeurs de i_s ;
- on ne crée pas autant de classeurs qu'il y a de valeurs possibles pour l'observable que l'on spécialise, on crée uniquement le classeur dont la valeur à spécialiser est la valeur de la dernière anticipation correcte du classeur que l'on spécialise.

Il faut noter que, lors de la spécialisation, un classeur qui ne coïncide avec aucune situation précédemment rencontrée n'est pas créé.

Généralisation des conditions

En essayant d'engendrer des classeurs aussi précis que possible, le processus de spécialisation des classeurs risque de produire des classeurs dont la partie `[Condition]` est moins générale que possible, en particulier lorsque l'agent n'a pas encore exploré la totalité de son environnement. Un processus de généralisation des conditions est donc nécessaire pour revenir sur les spécialisations initialement excessives.

Notre approche d'un tel processus consiste à identifier, parmi les classeurs qui ne sont pas compatibles avec la situation précédente mais qui l'auraient été si un de leurs attributs spécialisés avait été un #, ceux dont la partie `[Effet]` est compatible avec la situation courante. De tels classeurs pourraient être plus généraux sans faire d'erreur d'anticipation, donc ils méritent d'être généralisés.

La réalisation du processus de généralisation résultant de cette intuition est très proche de celle du processus de spécialisation que nous venons de décrire.

Nous associons à chaque attribut spécialisé de la partie `[Condition]` d'un classeur une estimation i_g qui mesure l'impact éventuel du remplacement de la valeur spécifiée par un #.

À chaque pas de temps, le système parcourt les classeurs dont la partie `[Action]` spécifie a_{t-1} et dont la partie `[Condition]` n'est pas compatible avec s_{t-1}. Parmi ces classeurs, il identifie ceux dont la partie `[Condition]` serait compatible avec s_{t-1} si on remplaçait un attribut par un #.

Pour chacun de ces classeurs, on met à jour la valeur de i_g l'attribut correspondant, là aussi selon une règle de Widrow-Hoff [76] :

[75] [Dorigo, 1993]
[76] [Widrow & Hoff, 1960]

– Si la partie [Effet] du classeur résultant de la généralisation de l'attribut associé à i_g est compatible avec s_t, alors i_g est augmenté : $i_g = (1 - \beta_g)i_g + \beta_g$
– Sinon, i_g est diminué : $i_g = (1 - \beta_g)i_g$

Ainsi, la valeur de i_g reste faible lorsque le classeur généralisé anticipe parfois bien et parfois mal, si bien qu'on ne le généralise pas.

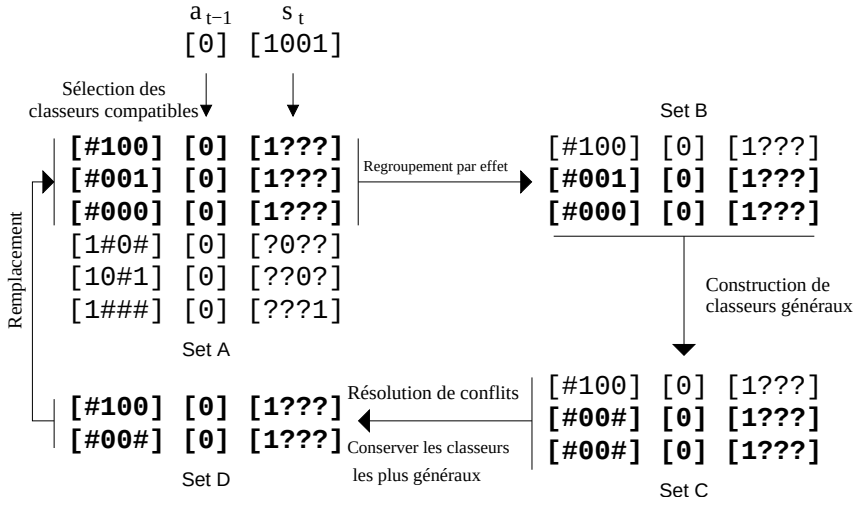


FIG. 3.4 – **Le processus de généralisation dans MACS. Ce processus est identique dans YACS, à ceci près que la compatibilité entre la situation s_t et la partie [Effet] est calculée différemment.**

Pour généraliser effectivement les classeurs, on sélectionne uniquement ceux pour lesquels une valeur i_g vient d'être augmentée. Ces classeurs constituent l'ensemble A sur la figure 3.4. Les classeurs de l'ensemble A dotés de la même partie [Effet] sont regroupés au sein d'ensembles notés B sur la figure 3.4.

Si un ensemble B contient au moins un classeur qui anticipe parfois mal – c'est-à-dire dont l'entier b est supérieur à 0 – le processus s'arrête là pour cet ensemble. Cela garantit que l'on ne généralise que des classeurs maximalement précis.

Si tous les classeurs d'un ensemble B anticipent toujours bien, on engendre un ensemble C de la façon suivante. Pour chaque classeur de B :

– Si aucune estimation i_g n'est supérieure à 0.5, alors le classeur est ajouté tel quel à l'ensemble C ;
– Sinon, l'attribut dont l'estimation i_g est la plus élevée est généralisé et le classeur résultant est ajouté à l'ensemble C .

On élimine ensuite de l'ensemble C les classeurs qui entrent en conflit avec des classeurs de la liste globale. Deux classeurs sont en conflit s'il existe au moins une situation à partir de laquelle ils prédiraient deux effets différents pour la même action. Lorsqu'on élimine de l'ensemble C un classeur parce qu'il est en conflit avec un autre, on le remplace par le classeur de l'ensemble A dont il dérive. La conjonction de tous ces remplacements donne naissance à l'ensemble D.

Enfin, pour obtenir les classeurs les plus généraux possibles, on compare deux à deux tous les classeurs de D et on supprime ceux pour lesquels il existe au moins un classeur plus général dans D. Il ne reste plus qu'à remplacer au sein de la liste globale les classeurs de l'ensemble A par ceux auxquels ils donnent naissance dans divers ensembles D et on obtient une liste de classeurs plus généraux sans perdre en précision et sans engendrer de conflits entre classeurs.

Couverture des transitions

Pour décrire parfaitement un environnement, le modèle des transitions doit disposer d'au moins un classeur pour couvrir chacune des transitions possibles. Une telle condition peut ne pas être remplie à un moment donné. En effet, le modèle des transitions est initialisé avec une liste de classeurs vide et les mécanismes de spécialisation n'ajoutent un classeur que s'il est compatible avec une situation déjà perçue.

Le rôle du processus de couverture des transitions est d'ajouter des classeurs dans le modèle des transitions lorsqu'aucun classeur n'est compatible avec la dernière transition effectuée. Pour cela, il construit des classeurs dont la partie `[Condition]` et la partie `[Action]` sont compatibles respectivement avec s_{t-1} et a_{t-1}, et dont la partie `[Effet]` est compatible avec l'effet observé. Ces nouveaux classeurs sont calculés pour être aussi généraux que possible.

Dans le cas de MACS, le processus de couverture des transitions consiste à ajouter un classeur pour chaque attribut, de telle façon que sa partie `[Effet]` contienne la valeur observée pour l'attribut en question, à condition qu'un tel classeur n'existe pas déjà.

Le degré de généralité de la partie `[Condition]` de ces classeurs est conditionné par les contraintes suivantes : leur partie `[Condition]` n'est compatible avec la partie `[Condition]` d'aucun autre classeur doté des mêmes parties `[Action]` et `[Effet]`, mais elle doit être compatible avec s_{t-1}.

3.4.4 Apprentissage du modèle des récompenses

Un mécanisme d'exploration active

Il y a deux motivations principales pour s'intéresser à l'exploration active dans le contexte des systèmes de classeurs à anticipation. Tout d'abord, tant qu'il n'a reçu aucune récompense, un agent dont le mécanisme d'exploration repose sur un choix aléatoire entre des actions équivalentes réalise une marche aléatoire, toutes les actions étant considérées comme de valeur nulle. Or un agent qui réalise une marche aléatoire atteint un état donné en un nombre de pas exponentiel en fonction de sa distance initiale à cet état. Par conséquent, doter un agent d'un mécanisme d'exploration active peut raccourcir considérablement la phase initiale de recherche d'une première source de récompense.

Ensuite, lorsque l'agent fait reposer l'apprentissage de son modèle des récompenses sur un modèle des transitions, il est crucial, pour que le comportement soit efficace, que le modèle des transitions soit le plus exact possible le plus vite possible. Tant que l'environnement de l'agent n'est pas entièrement connu, les mécanismes de généralisation et de spécialisation vont commettre des erreurs vis-à-vis des situations qui n'ont pas encore été rencontrées. Par conséquent, l'objectif du mécanisme d'exploration active est de diriger l'agent de façon à ce qu'il maximise l'information qu'il peut tirer de ses interactions avec son environnement.

Afin de combiner la politique d'exploration active avec la politique exploitant le modèle des récompenses, nous avons défini une architecture qui résulte de la combinaison hiérarchique de plusieurs modules de programmation dynamique.

Chacun de ces modules s'efforce de maximiser une récompense provenant d'une source qui lui est propre. En effet, nous distinguons trois sources de récompense immédiate :

- la récompense en information correspond à un gain en information sur le modèle des transitions ;
- la récompense externe correspond à la récompense effective que l'agent peut obtenir dans son environnement ;
- la récompense de rafraîchissement correspond à une mesure du temps écoulé depuis la dernière visite d'un état.

Nous notons respectivement $R_i(s_0)$, $R_e(s_0)$ et $R_r(s_0)$ les récompenses immédiates en information, externe et de rafraîchissement obtenues par l'agent lorsqu'il se trouve dans l'état s_0.

La récompense externe représente la fonction de récompense classique des modèles de Markov. Les deux autres sources de récompense méritent une définition plus précise.

Définition de la récompense immédiate en information

La récompense immédiate en information est définie de façon à mesurer si l'activation d'une action particulière permettra ou non d'améliorer le modèle des transitions. Plus précisément, nous définissons un estimateur $El(c)$ qui mesure le niveau d'évaluation de chaque classeur c. $El(c)$ vaut 0 si le classeur n'a pas du tout été évalué et vaut 1 si le classeur n'a plus du tout besoin d'être évalué. On pose $El(c) = min((b+g)/\theta_e, 1)$, où θ_e est le nombre d'évaluations requises pour décider si un classeur est précis, imprécis, ou oscillant, et `b` et `g` sont les nombres d'échecs et de succès collectés par le classeur lors d'évaluations précédentes.

Les estimateurs $El(c)$ sont associés à des classeurs, or nous souhaitons associer des mesures de récompense immédiate à des situations. Afin de calculer la récompense immédiate en information $Ri(s_0)$ associée à une situation s_0, le processus est le suivant.

Les classeurs dont la partie `[Condition]` est compatible avec s_0 sont regroupés par partie `[Action]`. Parmi ces classeurs, on élimine ceux qui prédisent à partir de s_0 une situation qui n'a jamais été rencontrée. De l'ensemble des classeurs restants, on déduit l'ensemble des triplets (s_0, a, s_1) représentant les transitions possibles à partir de s_0.

À chaque triplet (s_0, a, s_1), on associe un niveau d'évaluation $El(s_0, a, s_1)$ qui est le produit des niveaux d'évaluation de tous les classeurs qui prédisent cette transition :

$$El(s_0, a, s_1) = \prod_{c \approx (s_0, a, s_1)} El(c)$$

Moins une transition a été évaluée, plus le gain en information sera grand si on l'effectue. Donc, si la transition est effectuée, la récompense immédiate en information associée est :

$$R_i(s_0, a, s_1) = 1 - El(s_0, a, s_1)$$

Nous définissons la récompense immédiate en information associée à un couple (s_0, a) comme le maximum des récompenses immédiates en information qui peuvent être obtenues à partir de cette situation en faisant cette action :

$$R_i(s_0, a) = \max_{s1 \in S_{s_0,a}} R_i(s_0, a, s_1)$$

Si pour une action a, il n'y a aucune situation anticipée s_1 possible à partir de s_0 à cause de l'incomplétude du modèle, alors la récompense immédiate en information associée au couple (s_0, a) est maximale, elle vaut 1.

Enfin, nous définissons la récompense immédiate en information associée à une situation s_0 comme le maximum des récompenses immédiates en information qui peuvent être obtenues à partir de cette situation :

$$R_i(s_0) = \max_{a \in A} R_i(s_0, a)$$

Définition de la récompense immédiate de rafraîchissement

La récompense immédiate de rafraîchissement sert à inciter l'agent à retourner visiter de temps en temps les situations qu'il n'a pas visitées depuis longtemps. Cela permet d'assurer la persistance de comportements exploratoires, qui sont nécessaires lorsque l'environnement évolue au cours du temps. Plus une situation s_0 a été visitée

récemment, plus la récompense immédiate de rafraîchissement $R_r(s_0)$ est faible. On annule $R_r(s_0)$ à chaque fois qu'un agent visite une situation s_0. À chaque pas de temps où elle n'est pas visitée, la récompense immédiate de rafraîchissement associée à s_0 augmente selon l'équation de Widrow-Hoff : $R_r(s_0) = (1 - \beta_r)R_r(s_0) + \beta_r$.

3.4.5 Algorithme de choix de l'action

À partir des différentes récompenses que nous venons de décrire, il reste à choisir l'action effectuée par l'agent.

Dans le cadre de l'apprentissage par renforcement classique que nous avons présenté au chapitre précédent, l'action choisie dans un état donné est celle qui permet d'obtenir la plus grande récompense sur le long terme à partir de cet état, à ceci près que l'on fait intervenir un mécanisme d'exploration qui conduit à choisir parfois une action sous optimale (voir la section 2.3.3, page 40).

Lorsque l'algorithme estime directement une fonction de qualité $Q(s, a)$, il suffit de choisir l'action dont la qualité est la plus élevée pour l'état courant. Lorsque l'algorithme estime une fonction de valeur $V(s)$, en revanche, le choix de l'action est plus complexe. Il faut déterminer quels sont les états atteignables à partir de l'état courant, trouver parmi ces états celui qui a la plus grande valeur associée, puis réaliser l'action qui permet d'atteindre cet état.

Dans le contexte de MACS, c'est le second choix qui a été fait. Dans la partie du système consacrée à la construction du modèle des récompenses, on stocke pour chaque situation effectivement rencontrée les trois valeurs de récompense décrites à la section précédente.

La réalisation du mécanisme de choix de l'action dans MACS suppose donc de résoudre deux difficultés :

– d'une part, il faut déterminer quelle est la valeur d'une action donnée en fonction des valeurs des différentes récompenses que nous avons définies. Nous avons choisi le mécanisme d'agrégation hiérarchique décrit à la section suivante ;
– d'autre part, il faut déterminer quelles sont les situations immédiatement atteignables en fonction de la situation courante et du modèle des transitions, puis choisir l'action qui permet d'atteindre la situation atteignable pour laquelle la valeur résultant de l'agrégation hiérarchique est la plus élevée.

Cette seconde difficulté pose des problèmes spécifiques dans le cadre de MACS. En effet, les classeurs de MACS ne fournissent que des anticipations partielles, si bien que pour une situation donnée et une action donnée, il faut intégrer l'information portée par plusieurs classeurs pour déterminer une liste des situations que l'on espère atteindre. Il faut donc s'appuyer sur le mécanisme d'intégration des anticipations partielles décrit à la section 3.4.2, page 64.

Dès lors que le modèle des transitions est incomplet ou partiellement erroné, ce qui peut arriver au début de l'apprentissage, on n'a aucune garantie que la liste des situations atteignables sera déterminée correctement. Cela explique que l'on choisisse de privilégier dans un premier temps la complétude du modèle des transitions, comme expliqué à la section suivante.

Agrégation hiérarchique des récompenses

Une fois que l'on a construit la liste des situations que l'on peut atteindre à partir de la situation courante, reste à choisir celle que l'on vise sur la base des différentes récompenses associées à chaque situation. Pour cela, nous avons choisi un modèle d'agrégation hiérarchique qui donne la priorité à la récompense en information. Il apparaît en effet qu'un modèle des transitions erroné conduit inexorablement à une estimation erronée de la récompense externe.

Il est donc préférable de commencer par chercher à obtenir un modèle des transitions aussi exact que possible, avant de se fier à des récompenses estimées qui peuvent être fausses. En conséquence, s'il existe parmi les situations

atteignables une situation dont le gain estimé en information est supérieur à celui des autres, c'est l'action qui conduit à cette situation qui est préférée. Si toutes les situations atteignables sont équivalentes du point de vue de ce critère, alors on choisit d'atteindre celle qui possède la plus grande estimation de récompense externe.

En particulier, une fois que l'agent possède une information parfaite sur son environnement, il choisit toujours l'action qui permet de maximiser la récompense, ce qui constitue le pré-requis élémentaire de tous les mécanismes de gestion du compromis entre exploration et exploitation.

Enfin, si l'agent est dans une situation où toutes les situations atteignables sont équivalentes du point de vue des deux critères précédents, il déclenche le classeur qui lui permet d'atteindre la situation qui n'a pas été visitée depuis le plus longtemps. Cela permet de mettre à jour régulièrement la connaissance de l'environnement lorsque celui-ci est susceptible d'évoluer au cours du temps.

En fin de compte, l'agent déclenche l'action qui lui permet d'atteindre la situation dont la valeur est la plus élevée compte tenu de l'agrégation hiérarchique que nous venons de décrire.

3.5 Quelques résultats marquants

Des résultats expérimentaux sur la compacité du modèle bâti par MACS et sur sa vitesse de convergence [77] ont montré que ce système produit un modèle un peu plus compact que YACS, qui lui-même construit des modèles quatre fois plus compacts qu'ACS. En outre, MACS construit ce modèle trois fois plus vite que YACS, et 9 fois plus vite qu'ACS en nombre de pas de temps.

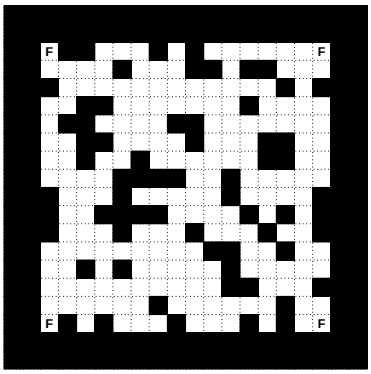


FIG. 3.5 – **La topologie de Maze2232**

La conséquence de ces performances est que MACS est capable de traiter dans des délais raisonnables des problèmes dont la structure est plus complexe que celle des problèmes traités par les systèmes concurrents. Par exemple, MACS peut résoudre le problème **Maze2232**, qui apparaît sur la figure 3.5, alors que la perception de l'agent est composée de 25 attributs, ce qui constitue un problème de taille tout à fait respectable.

Les performances correspondantes sont présentées sur la figure 3.6, sans comparaison car les systèmes concurrents demandent un temps prohibitif pour produire une solution à ce problème.

La conclusion générale que l'on peut tirer de ces résultats est que MACS apparaît à l'heure actuelle comme la référence en matière de performances parmi les systèmes de classeurs à anticipation.

[77] [Gérard et al., 2005]

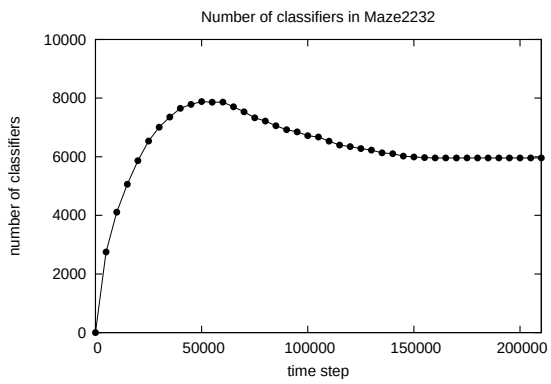


FIG. 3.6 – **Evolution du nombre de classeurs sur Maze2232**

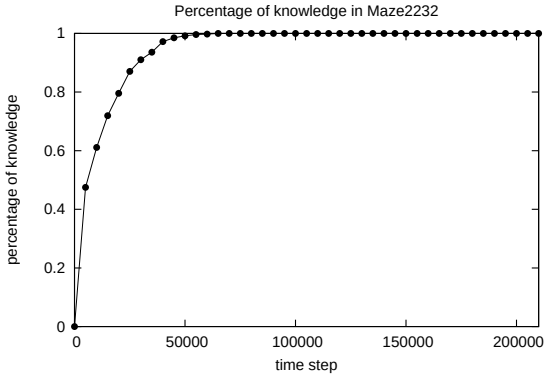


FIG. 3.7 – **Evolution du pourcentage de connaissance sur Maze2232**

Cette efficacité provient de choix conceptuels plus clairs que ceux réalisés par les concurrents, mais s'obtient aussi au prix d'une complexité du logiciel sensiblement plus grande que celle des autres systèmes.

3.6 Comparaison avec les architectures DYNA

Plutôt que de se restreindre à une comparaison avec les autres systèmes de classeurs à anticipation, il apparaît utile de comparer MACS avec les systèmes initiaux fondés sur l'architecture DYNA conçus par R. Sutton [78].

Nous avons affirmé que MACS combinait les capacités de généralisation des systèmes de classeurs avec les capacités d'anticipation fournies par l'apprentissage par renforcement indirect. Pour valider cette affirmation, nous devons donc montrer à la fois que sa capacité à construire un modèle augmente sa vitesse d'apprentissage, comme c'est le cas pour les systèmes fondés sur l'architecture DYNA et que le modèle interne qu'il construit est plus compact que celui que bâtirait un DYNA tabulaire. Nous réalisons cette comparaison dans des environnements standards qui ont servi à R. Sutton à illustrer les propriétés de l'architecture DYNA [79]. Le premier d'entre eux apparaît sur la figure 3.8.

Bien qu'il s'agisse du même environnement, la perception qu'en ont un agent commandé par MACS et un agent commandé par un système tabulaire fondé sur l'architecture DYNA sont fondamentalement différentes. En effet, pour le second, tous se passe comme si chaque case était dotée d'un numéro unique qui la rend immédiatement distincte des autres cases. Pour le premier, au contraire, la distinction se fait en utilisant des perceptions discriminantes parmi celles qui sont fournies à l'agent. Il faut donc s'assurer que ces perceptions contiennent suffisamment d'information pour permettre de distinguer toutes les cases. C'est le cas si l'agent perçoit les 25 cases environnantes, comme il apparaît sur la figure 3.8.

3.6.1 Comparaison des vitesses d'apprentissage

Dans le cadre d'une comparaison, nous nous efforçons d'assigner les mêmes valeurs aux paramètres fonctionnellement équivalents pour les deux systèmes. Lorsque les paramètres donnés dans [Sutton, 1990] diffèrent de ceux donnés dans [Sutton & Barto, 1998], nous prenons ceux de [Sutton & Barto, 1998]. Nous posons donc γ=0.95 et β=0.5.

[78] [Sutton, 1990, Sutton & Barto, 1998]
[79] [Sutton & Barto, 1998]

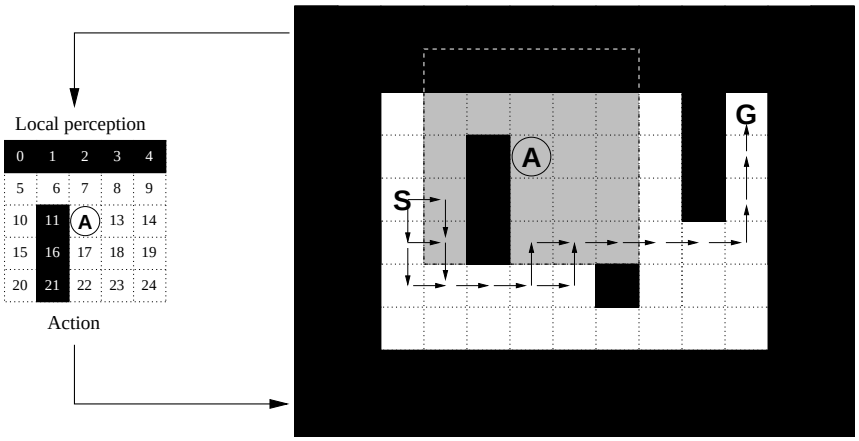


FIG. 3.8 – **Le labyrinthe de Sutton. L'agent démarre de la case marquée « S » et doit rejoindre la case marquée « G ». Les chemins optimaux, de longueur 14, sont indiqués par les flèches. La perception de l'agent est restreinte aux 25 cases environnantes, numérotées comme il apparaît sur la partie gauche de la figure. L'agent peut se déplacer dans les quatre directions cardinales. Atteindre le but permet de recevoir une récompense de +1, il n'y a pas de punition pour heurter un mur.**

Par ailleurs, dans un premier temps, nous remplaçons la fonctionnalité d'exploration active de MACS par une politique d'exploration fondée sur un mécanisme de Boltzmann pour comparer avec les architectures DYNA dans un cadre identique.

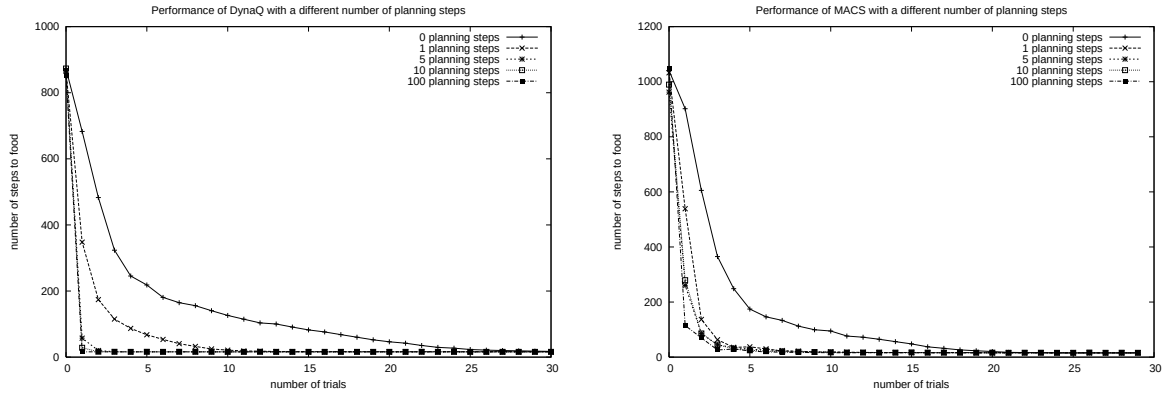


FIG. 3.9 – **Performance de** DYNA-Q **et MACS sur le labyrinthe de Sutton. Chaque courbe fournie pour** DYNA-Q **a été obtenue par une moyenne sur 1000 essais. Celles pour MACS l'ont été pour une moyenne sur 500 essais, car le logiciel est plus lent. La différence principale apparaît dans le coin en bas à gauche des courbes.**

On observe que les courbes ont la même allure générale, mais des différences mineures ressortent de cette comparaison :

– comme un même classeur modélise plusieurs transitions à la fois, l'algorithme de programmation dynamique utilisé dans les étapes de planification peut converger plus vite sur le modèle construit par MACS que sur le modèle tabulaire. Cela est d'autant plus vrai que le nombre d'étapes de planification est faible devant le nombre de pas passés à améliorer le modèle.
– au contraire, au début des expériences, lorsque le modèle des transitions contient encore des généralisations fausses, la programmation dynamique risque de faire plus de mises à jour erronées que sur un modèle ta-

bulaire. Donc quand il y a beaucoup d'étapes de planification, la performance est moins bonne en début d'expérience que sur un modèle tabulaire.

Il ressort de cette comparaison que le problème posé à MACS est beaucoup plus difficile que celui posé à DYNA-Q. En effet, alors que le second se contente d'approximer efficacement la fonction de qualité grâce à la programmation dynamique, le premier doit simultanément généraliser correctement son modèle du monde, ce qui engendre des interactions fâcheuses entre les deux processus.

Notre choix de conception a consisté à découpler autant que possible ces deux processus, ce qui contribue selon nous à expliquer les bonnes performances de MACS vis-à-vis des autres systèmes de classeurs à anticipation.

3.6.2 Comparaison de la compacité des modèles

Le principal avantage des systèmes de classeurs sur les systèmes d'apprentissage par renforcement tabulaire est leur capacité à construire une représentation compacte du problème qu'ils traitent.

Nous espérons donc que MACS peut bâtir une représentation plus compacte que DYNA-Q.

Le labyrinthe de Sutton contient 47 états et 4 actions. Le problème étant déterministe, chaque couple (s_t, a_t) donne lieu à une seule transition (s_t, a_t, s_{t+1}). En conséquence, DYNA-Q construit un modèle tabulaire contenant $47 \times 4 = 288$ transitions.

Les résultats expérimentaux obtenus avec MACS montrent qu'un modèle des transitions complet contient en moyenne 900 classeurs. Par conséquent, le modèle des transitions bâti par MACS ne peut être considéré *a priori* comme plus compact que celui construit par DYNA-Q. Ce résultat décevant provient de ce que chaque classeur de MACS ne cherche à anticiper qu'une valeur d'attribut à la fois, alors que les systèmes tabulaires fondés sur l'architecture DYNA prédisent la situation suivante comme un tout.

En effet, dans MACS, chacun des 25 modules prédit la valeur d'un attribut particulier. Donc chaque module contient en moyenne $900/25 = 36$ classeurs, ce qui est beaucoup moins que les 288 classeurs par module qui seraient nécessaires sans la généralisation.

On voit donc que le taux de généralisation est élevé, mais qu'une représentation à base d'attributs est intrinsèquement plus gourmande en espace mémoire qu'une représentation basée sur l'énumération des états. Par ailleurs, dans MACS comme dans YACS, nos modèles de récompense consistent à stocker la fonction de valeur sous la forme d'une valeur de récompense immédiate et une valeur de récompense attendue pour chaque état effectivement rencontré par l'agent. Quand le nombre d'états devient grand, une telle représentation s'avère trop gourmande en mémoire.

Il semble donc utile de chercher à élaborer un modèle compact de la fonction de valeur ou de la fonction de qualité. Une première approche de ce problème est proposée par M. Butz [80] qui utilise XCS comme outil d'approximation compacte de la fonction de récompense attendue. La capacité de généralisation d'XCS permet en effet de regrouper efficacement les états pour lesquels la valeur est identique.

Une solution plus prometteuse consiste à se tourner vers les travaux de C. Boutilier [81] sur les réseaux bayésiens dynamiques. L'auteur propose en effet une méthode qui permet à la fois d'extraire un modèle compact de la fonction qualité sous forme d'arbre et de déduire efficacement de cette représentation arborescente une politique optimale pour l'agent. La présentation de cette voie de recherche sera au cœur du chapitre suivant.

[80] [Butz & Goldberg, 2003]
[81] [Boutilier et al., 1999]

3.6.3 Expériences dans des environnements variables

Disposer d'un modèle des transitions de l'environnement est un avantage tant que ce modèle est correct, car cela permet d'accélérer le processus de propagation des qualités des différentes actions. En revanche, dès lors que l'environnement change au cours du temps, l'apprentissage par renforcement indirect pose le problème supplémentaire de la mise à jour du modèle des transitions à chaque évolution de l'environnement.

Nous avons testé la capacité de MACS à affronter deux types de problèmes dans lesquels l'environnement change :

- dans un cas, ce sont les sources de récompenses qui se déplacent. P. Gérard a montré dans le cadre de sa thèse [82] que MACS était capable de mettre à jour son modèle de la fonction de qualité très rapidement, si bien qu'il gérait efficacement ces problèmes ;
- dans le second cas, c'est la topologie de l'environnement qui change. Nous exposons ici une comparaison de la performance de MACS avec divers systèmes tabulaires fondés sur l'architecture DYNA sur ce problème.

Les expériences à partir desquelles nous allons évaluer MACS ont été réalisées initialement par R. Sutton [83] dans des environnements qui apparaissent à la figure 3.6.3.

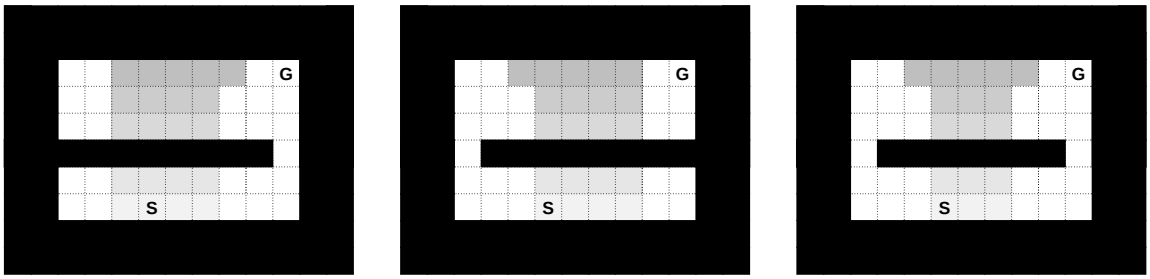


FIG. 3.10 – **Environnements pour tester les architectures DYNA et MACS dans le cas où la topologie change au cours du temps. À gauche : environnement initial pour le blocage ; Au centre : environnement final pour le blocage et environnement initial pour le raccourci ; À droite : environnement final pour le raccourci. Les cases grisées sont perçues identiques par l'agent dirigé par MACS.**

Une difficulté posée par ces environnements est que la perception des 25 cases environnantes ne suffit pas à distinguer exactement toutes les cases, si bien que le problème est non markovien pour MACS avec de telles perceptions. Ceci dit, l'action optimale est la même dans toutes les situations perçues comme identiques, si bien que le non respect de l'hypothèse de Markov n'a pas d'effet notable sur la performance.

Avec le même cadre expérimental que précédemment, nous obtenons les résultats de la figure 3.6.3.

Ces expériences permettent de tirer les enseignements suivants :

- MACS doté du seul modèle de la récompense externe, appelé MACS-, se comporte comme DYNA-Q sur les deux problèmes, bien qu'il soit moins efficace au début des expériences ;
- la version complète de MACS, appelée MACS+, est de loin le système le moins efficace sur ces problèmes. À partir des courbes, il est impossible de déterminer si l'agent parvient à trouver le raccourci dans les expériences correspondantes ;
- En revanche, en basculant de MACS+ à MACS- lorsque 1000 pas de temps se sont écoulés après la modification de l'environnement, on obtient un système plus performant.

Le premier point s'explique aisément. En effet, nous avons montré sur la figure 3.6.1 que MACS- converge plus lentement que DYNA-Q lorsqu'on réalise 10 étapes de planification par pas de temps. Il en résulte qu'il accumule

[82] [Gérard, 2002, Gérard et al., 2005]
[83] [Sutton, 1990]

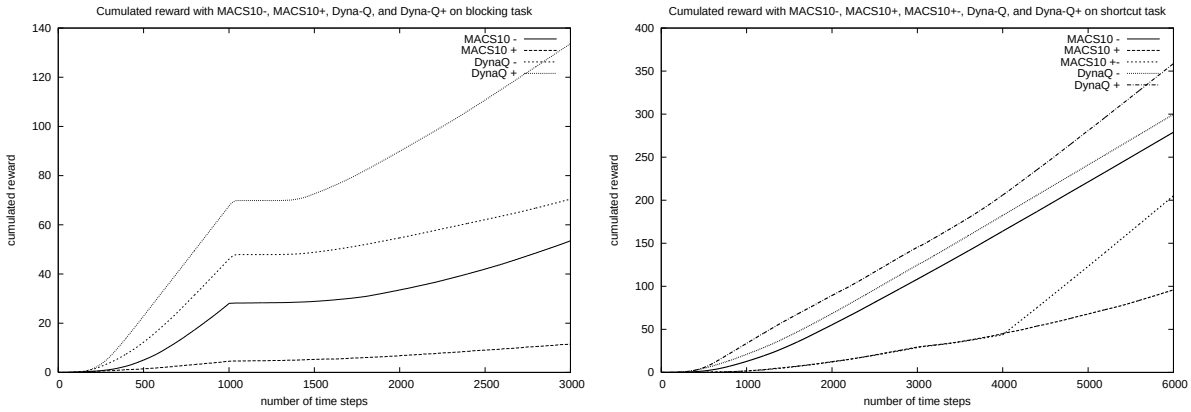


FIG. 3.11 – **Récompense cumulée moyenne obtenues par** DYNA-Q**,** DYNA-Q+ **et différentes versions de MACS dans le cas du blocage et dans le cas du raccourci. MACS- est la version de MACS dotée d'un seul modèle de récompense externe, comme nous l'avons utilisé dans la section 3.6.1 ; MACS+ est la version complète de MACS avec agrégation hiérarchique des trois modèles de récompense décrits à la section 3.4.5. MACS+- présente la courbe obtenue en basculant de MACS+ à MACS- 1000 pas de temps après la modification de l'environnement. Dans tous les cas, la performance moyenne est calculée sur 100 essais. Tous les systèmes font 10 étapes de planification par Ils reçoivent une récompense de 1 pour trouver le but. La modification de l'environnement a lieu au bout de 1000 pas de temps dans le cas du blocage et 3000 pas de temps dans le cas du raccourci.**

moins de récompense.

Les deux autres points font apparaître les limites du mécanisme de gestion du compromis entre exploration et exploitation sur lequel repose MACS.

En effet, notre choix de donner la priorité à la recherche d'un modèle des transitions aussi complet que possible se justifie lorsque ce modèle peut être acquis en peu de temps, mais il devient contestable lorsque la taille du problème augmente ou lorsque la violation de l'hypothèse de Markov empêche le modèle de converger. À cause d'un tel choix, MACS continue à explorer son environnement là où DYNA-Q+ se contente d'exploiter avec un taux d'exploration résiduel le chemin qu'il a trouvé vers la récompense, si bien que le second accumule beaucoup plus de récompense que le premier. D'ailleurs, on observe grâce au troisième point que, dès lors qu'on utilise MACS en exploitation pure, sa capacité à accumuler la récompense devient identique à celle de DYNA-Q+.

Il apparaît donc nécessaire de rechercher un meilleur mécanisme de gestion du compromis entre exploration et exploitation. M. Kearns a proposé avec S. Singh [84] puis avec D. Koller [85] un mécanisme appelé E^3 [86] qui semble pouvoir s'appliquer directement à MACS et pour lequel il existe des preuves de convergence vers une exploitation optimale. Nous présenterons ce mécanisme à la section 4.7.1 et nous indiquerons comment nous envisageons de l'intégrer dans le nouveau système qui résulte de nos réflexions.

3.7 Conclusion

Dans ce chapitre, nous avons présenté les systèmes de classeurs, qui ajoutent à l'apprentissage par renforcement classique la possibilité de représenter les états sous la forme de vecteurs d'observables, et de trouver une expression compacte de la représentation ainsi obtenue. Puis nous avons présenté les systèmes de classeurs à anticipation,

[84] [Kearns & Singh, 1998]
[85] [Kearns & Koller, 1999]
[86] Pour *Explicit Exploration and Exploitation*

qui complètent les précédents par la construction d'un modèle des transitions qui permet d'accélérer sensiblement la convergence de l'apprentissage. Nous avons donc proposé des solutions à deux des insuffisances du modèle standard de l'apprentissage par renforcement.

Pour étayer notre propos, nous avons présenté le système MACS, mis au point dans le cadre de la thèse de P. Gérard. MACS obtient des performances supérieures aux autres systèmes de classeurs à anticipation sur des problèmes déterministes de type labyrinthe, mais il souffre d'un certain nombre de problèmes que nous recensons ici avant de les dépasser dans les chapitres suivants.

- Le premier problème qui saute aux yeux est que MACS est un système sensiblement plus complexe que les autres systèmes de classeurs. Il repose sur un grand nombre d'heuristiques qui ont montré leur efficacité, mais il a du apparaître au cours du chapitre que donner une description de MACS à la fois suffisamment concise et suffisamment précise pour donner à quiconque l'envie de le recoder, ne va pas de soi.
- En outre, l'ensemble d'heuristiques qui assure son fonctionnement est fondé sur le plan conceptuel, mais sa complexité est telle qu'il est improbable que l'on puisse donner à ces fondements une validité théorique susceptible de déboucher sur des preuves de convergence.
- Ensuite, de façon plus locale, nous avons vu que le choix de stocker des modèles de la fonction de valeur plutôt que des qualités complique singulièrement le processus de choix de l'action, qui exige un pas de regard en avant. Il y a là une source d'inefficacité qui demande à être remise en cause.
- Par ailleurs, les modèles de la fonction de valeur ainsi construits e sont pas compacts, les valeurs sont associées à des états énumérés. Cela contribue à l'inefficacité à la fois en mémoire et en temps de calcul du processus de choix de l'action.
- Enfin, les heuristiques sur lesquelles repose MACS sont conçues pour fonctionner dans un cadre déterministe et ne savent être transposées telles quelles au cadre stochastique.

Au chapitre suivant, nous allons examiner une classe de systèmes, les réseaux bayésiens dynamiques, qui proposent des solutions élégantes à tous ces problèmes à la fois. Nous verrons comment l'examen de ce courant de recherche nous conduit à proposer un nouveau système qui améliore MACS sur de nombreux points.

Par ailleurs, nous avons énuméré les limitations propres à MACS, mais de nombreuses autres limitations, dont souffre le cadre général des systèmes de classeurs, restent à traiter. Ce sera l'objet des chapitres ultérieurs.

Chapitre 4

Processus de Décision Markoviens Factorisés

> « Quand tu crois que ça marche, il faut crier de joie très vite. Si tu attends, même quelques instants, tu t'aperçois que ça ne marche pas et alors c'est dommage, parce que tu as perdu une occasion de te remonter le moral. »
>
> S. Gourichon, le 12/03/2002

Nous avons conclu le chapitre précédent sur un certain nombre de limitations inhérentes aux travaux développés autour de MACS. En particulier, nous avons critiqué le caractère ad hoc *de quelques heuristiques utilisées par MACS sans prise de recul théorique sur la signification de ces heuristiques.*

Dans ce chapitre, nous profitons de l'existence d'un cadre théorique adapté à la formalisation des problèmes auxquels s'attaque MACS pour prendre ce recul qui nous faisait défaut. Nous établissons en effet un parallèle entre le formalisme de MACS et celui des réseaux bayésiens dynamiques [1]*. Le cadre des réseaux bayésiens dynamiques a donné lieu à de nombreux développements théoriques et à la mise au point d'algorithmes efficaces dont nous montrons qu'ils permettent de traiter une partie des problématiques abordées par les systèmes de classeurs à anticipation.*

Nous faisons tout d'abord un état des lieux de ces développements. Nous établissons alors des comparaisons entre les représentations des systèmes de classeurs à anticipation et celles des réseaux bayésiens dynamiques et entre les algorithmes qui en découlent. Nous présentons les principaux algorithmes de planification développés dans le cadre des processus de décision markoviens factorisés et nous mettons en évidence le fait qu'il manque aux systèmes correspondant la capacité à apprendre le modèle des transitions et des récompenses qui caractérise MACS.

Nous décrivons alors un nouveau système dont la capacité d'apprentissage d'un modèle des transitions et la capacité de recherche d'une politique optimale sur le modèle appris s'avèrent plus efficaces que les capacités correspondantes de MACS. Cela nous conduit à énumérer un certain nombre de pistes de recherche qui constituent l'un des principaux axes de développement de nos recherches futures.

[1] *Dynamic Bayes Networks, ou DBNs*

4.1 Les processus de décision markoviens factorisés

Les recherches sur les réseaux bayésiens dynamiques sont motivées par le fait que les situations auxquelles sont confrontés les agents sont souvent dotées d'une certaine structure dont la représentation sous forme d'état monolithique fait abstraction. Cette abstraction est d'autant plus fâcheuse que, si le nombre d'états est très élevé, la probabilité pour que l'agent repasse deux fois par le même état est très faible, donc une connaissance qui lui dit quoi faire dans un état précis est très rarement sollicitée. Cela est d'autant plus problématique que le comportement d'un agent ne s'approche du comportement optimal avec un facteur de précision donné que si chaque état est visité un nombre de fois qui augmente exponentiellement avec ce facteur de précision, ce qui implique une explosion du temps de calcul.

Or, dans de nombreux problèmes, on peut représenter l'état de l'agent par un vecteur de valeurs discrètes plutôt que par un label unique [2]. La représentation qui résulte de cette transformation est appelée « *processus de décision markovien factorisé* » [3].

L'idée qui a donné naissance aux processus de décision markoviens factorisés remonte à une publication de C. Boutilier [4]. Cet auteur a consacré un certain nombre de travaux à la mise en évidence de liens entre la planification telle qu'on la pratique dans le cadre de l'intelligence artificielle classique et celle qui a cours dans le cadre des processus de décision markoviens [5]. Ce regard transversal lui a permis de mettre en évidence l'intérêt de représenter de façon compacte l'état sur lequel opèrent les algorithmes de programmation dynamique.

Dans cette nouvelle représentation, l'ensemble des états est défini par une collection de variables aléatoires. Cet ensemble s'écrit $S = \{X_1, ..., X_n\}$. Chaque variable prend sa valeur au temps t (notée x_i ou x_i^t) dans un domaine : $\forall i, x_i \in Dom(X_i)$. Un état s donné est défini par un vecteur de valeurs de chacune de ces variables aléatoires : $s = \{x_1, ..., x_n\}$. Le reste du formalisme des processus de décision markovien peut être associé à cette nouvelle représentation des états sans autre modification.

Si les dépendances entre les variables sont faibles, cette façon de procéder permet de réaliser des économies substantielles de calcul vis-à-vis de la représentation fondée sur les états. Comme l'indique très clairement C. Boutilier [6], s'il y a n observables telles que la valeur de chacune d'elles ne dépend au plus que des valeurs de l autres observables au pas de temps précédent, alors la taille d'une représentation factorisée est en $O(n2^l)$ au lieu de $O(n^2)$.

En effet, supposons que l'on dispose d'un espace d'états S dont les états peuvent se présenter sous la forme d'une collection de trois variables binaires X, Y et Z, telles que chacune de ces variables à $t+1$ ne dépend que de cette même variable à t, sauf Y qui dépend de X et de Y.

Alors stocker directement $P(S^{t+1}|S^t)$ suppose une énumération exhaustive de toutes les combinaisons. En pratique, on obtiendra une table qui contient $2^6 = 64$ entrées.

On peut aussi écrire $P(S^{t+1}|S^t) = P(X^{t+1}, Y^{t+1}, Z^{t+1}|S^t) = P(X^{t+1}|S^t)P(Y^{t+1}|S^t)P(Z^{t+1}|S^t)$,

ou encore $P(X^{t+1}, Y^{t+1}, Z^{t+1}|S^t) = P(X^{t+1}|X^t)P(Y^{t+1}|X^t, Y^t)P(Z^{t+1}|Z^t)$, ce qui n'implique plus que des énumérations partielles. On a cette fois $4 + 8 + 4 = 16$ entrées réparties dans trois tables de probabilités, ce qui fait ressortir l'intérêt de la factorisation.

Nous montrerons plus loin que le cadre des processus de décision markovien factorisés est le cadre formel adéquat pour décrire la représentation adoptée par les systèmes de classeurs. Néanmoins, nous allons voir tout d'abord que d'autres représentations sont possibles. En effet, dès l'article original de 1995, C. Boutilier suggère

[2] [Boutilier et al., 1999]
[3] *factored MDP, ou FMDP*
[4] [Boutilier et al., 1995]
[5] [Boutilier & Goldszmidt, 1996]
[6] [Boutilier et al., 1999]

d'utiliser le formalisme des réseaux bayésiens dynamiques pour traiter ces problèmes. Par la suite, l'équipe de D. Koller, spécialisée dans l'étude de ce type de réseaux, est venue apporter un certain nombre de contributions majeures à ce qui est devenu le domaine des processus de décision markoviens factorisés. En particulier, cette équipe a développé une technique de programmation linéaire pour résoudre efficacement les problèmes de planification dans le cadre des processus de décision markoviens factorisés sur laquelle se sont appuyés les travaux successifs [7].

Les collaborateurs de C. Boutilier et ceux de D. Koller restent les acteurs principaux de l'avancement des travaux dans ce domaine. Nous nous intéresserons particulièrement dans ce chapitre aux prolongements apportés par C. Guestrin dirigé par D. Koller en direction de l'apprentissage par renforcement et des problèmes multi-agents.

4.2 Les réseaux bayésiens dynamiques

Le cadre des réseaux bayésiens [8] inclut un formalisme graphique qui permet de représenter des relations de dépendance conditionnelle entre des variables.

Graphiquement, les variables sont représentées par des nœuds d'un graphe orienté acyclique. Un lien entre deux nœuds représente une relation de dépendance probabiliste entre les variables correspondantes, de telle façon que la probabilité conjuguée de toutes les variables peut être décomposée en un produit des probabilités de chaque variable connaissant la valeur de ses parents dans le graphe. Ainsi, les réseaux bayésiens fournissent un cadre formel élégant pour représenter des indépendances conditionnelles entre des variables. En outre, il est facile de paramétrer de manière compacte le modèle des relations entre les variables.

Les réseaux bayésiens dynamiques [9] sont des réseaux bayésiens particuliers qui représentent les données temporelles engendrées par des processus stochastiques [10]. Ils peuvent être vus comme un cadre général englobant les modèles de Markov cachés [11], les systèmes dynamiques linéaires [12] et les filtres de Kalman [13]. Ils étendent les réseaux bayésiens en permettant de représenter des évolutions au cours du temps de variables aléatoires de la forme $X_i^{t+1} = f(X_i^t)$. La séquence engendrée de cette façon est potentiellement infinie mais, si le processus est stationnaire, il suffit de représenter une unique fonction f reliant X_i^{t+1} à X_i^t pour tout t pour représenter la totalité du processus.

L'hypothèse de stationnarité du processus sous-jacent peut être relâchée pour ce qui concerne aussi bien la valeur des paramètres que la structure des dépendances entre variables. Néanmoins, dans le cas stationnaire, on peut représenter un réseau bayésien dynamique à l'aide d'un graphe faisant apparaître uniquement deux pas de temps successifs et n'impliquant des dépendances qu'entre ces deux pas de temps. Cette représentation restreinte est connue sous le nom de « 2TBN » [14]. Au sein du cadre des 2TBNs, nous nous restreindrons en outre au cas où il n'apparaît pas de relations dites « synchrones », c'est-à-dire des relations entre les variables définies dans le même pas de temps.

La classe de problèmes résultant de cette restriction est beaucoup moins vaste que la classe générale, mais elle reste suffisamment riche pour représenter efficacement les problèmes qui nous intéressent, à savoir les processus de décision markoviens factorisés. En outre, cette classe de problèmes donne lieu à des algorithmes efficaces, alors

[7] [Kearns & Koller, 1999, Koller & Parr., 2000, Guestrin et al., 2003b]
[8] [Pearl, 1988]
[9] *Dynamic Bayesian Networks (DBNs)*
[10] [Dean & Kanazawa, 1989]
[11] Hidden Markov Models (HMMs) [Rabiner & Juang, 1986]
[12] Linear Dynamical Systems (LDSs) [Luenberger, 1979]
[13] [Kalman, 1960]
[14] *2 Time (slices) Bayesian Network*

que les représentations moins restreintes souffrent en général de problèmes d'explosion combinatoire, qui rendent inapplicables les algorithmes qui les manipulent.

4.3 Exemple de processus de décision markoviens factorisé

Afin de montrer comment on utilise les 2TBNs pour modéliser des processus de décision markoviens factorisés, nous reproduisons dans cette section un exemple élémentaire proposé par C. Boutilier [15]. Dans cet exemple, un robot doit aller acheter un café pour le donner à sa propriétaire qui reste dans son bureau. Il peut éventuellement pleuvoir pendant le trajet entre le café et le bureau, auquel cas le robot se fait mouiller, à moins qu'il n'ait pris un parapluie.

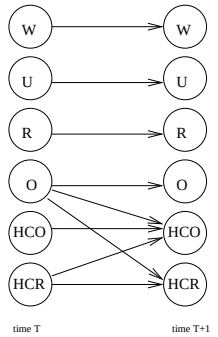
L'état du système peut être décrit à l'aide de six propositions, donc six variables aléatoires qui sont binaires dans cet exemple :

- O : le robot est au bureau ($\overline{O}$ signifie qu'il est au café) ;
- R : il pleut ($\overline{R}$ signifie qu'il ne pleut pas) ;
- W : le robot est sec ($\overline{W}$ signifie qu'il est mouillé) ;
- U : le robot a un parapluie ($\overline{U}$ signifie qu'il n'en a pas) ;
- HCR : le robot a un café ($\overline{HCR}$ signifie qu'il n'en a pas) ;
- HCO : la propriétaire a un café ($\overline{HCO}$ signifie qu'elle n'en a pas).

Par ailleurs, le robot peut réaliser quatre actions :

- Go : se déplacer dans le lieu opposé (café ou bureau) ;
- $GetU$: prendre un parapluie ;
- $BuyC$: acheter du café ;
- $DelC$: donner le café à l'utilisateur.

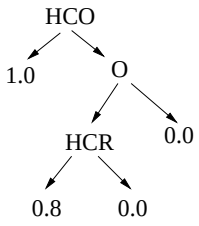
Un 2TBN permet de modéliser l'effet de chaque action sur l'état du problème. Chacun des 2TBN ainsi représenté est appelé « réseau d'action » [16].



(a)

O^t	HCR^t	HCO^t	$HCO^{t+1} = V$
V	V	V	1.0
F	V	V	1.0
V	F	V	1.0
F	F	V	1.0
V	V	F	0.8
F	V	F	0.0
V	F	F	0.0
F	F	F	0.0

(b)



(c)

FIG. 4.1 – **(a) Le réseau d'action de $DelC$, sous forme d'un 2TBN ; (b) La table de probabilité conditionnelle (CPT) de HCO si le robot fait l'action $DelC$; (c) La représentation arborescente correspondante**

Sur la figure 4.1.a, on montre le réseau d'action correspondant à l'action « $DelC$ ». À partir de ce réseau, on peut constater, par exemple, que quand le robot remet un café à sa propriétaire, le fait qu'il soit mouillé dépend uniquement de s'il était mouillé auparavant ou pas, puisque l'action de donner le café s'effectue dans le bureau, où

[15] [Boutilier et al., 2000]

[16] *action network, [Darwiche & Goldszmit, 1994]*

il est impossible de se faire mouiller. Il faut à la fois qu'il pleuve et que le robot réalise l'action Go sans parapluie pour qu'il devienne mouillé.

Par contre, quand le robot effectue l'action « $DelC$ », le fait que la propriétaire ait finalement un café dépend de trois propositions :

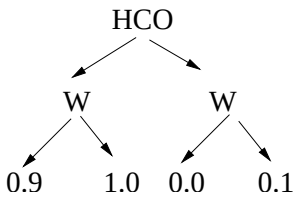
– le robot est-il dans le bureau ?
– le robot a-t-il un café ?
– la propriétaire a-t-elle déjà un café ?

Pour représenter le problème complet, il faut construire un réseau analogue pour chaque action. Toutefois, ces réseaux indiquent uniquement la présence ou l'absence de relations de dépendances conditionnelles – on dit qu'ils spécifient la structure du problème –, ils ne donnent pas l'information quantitative nécessaire pour connaître la dynamique effective du problème.

L'information manquante est représentée sous la forme de « *tables de dépendances probabilistes* » ou CPTs [17].

Un exemple de CPT apparaît sur la figure 4.1.b. Cette table donne la probabilité pour que HCO soit vrai au pas de temps suivant en fonction de O, HCR et HCO si l'agent effectue l'action $DelC$. La même information peut être représentée de façon plus compacte sous forme arborescente, comme indiqué par la figure 4.1.c.

Pour spécifier complètement la dynamique d'un processus de décision markovien factorisé, il faut fournir une table de ce type pour chaque couple (variable, action).



HCO^t	W^t	R^{t+1}
F	F	0.1
F	V	0.0
V	F	1.0
V	V	0.9

FIG. 4.2 – **L'arbre de récompense associé à l'exemple et la table de valeurs correspondante**

Par ailleurs, on peut aussi donner une représentation plus compacte de la fonction de récompense d'un processus de décision markovien factorisé sous forme arborescente, comme indiqué par la figure 4.2. La possibilité de représenter ainsi la dynamique d'un processus de décision markovien sous forme compacte permet d'élaborer des algorithmes efficaces que nous présenterons à la section 4.5.1. En attendant, nous allons mettre en évidence le lien entre ce type de représentation arborescente et la représentation élaborée par MACS que nous avons présentée au chapitre précédent.

4.4 Comparaison des approches

4.4.1 Comparaison des représentations

Les données représentées dans MACS sont très similaires à celles représentées dans une architecture à base de 2TBNs, bien qu'elles soient organisées différemment. Dans MACS, chaque module du modèle des transitions prédit la valeur d'un attribut particulier pour toutes les situations possibles et pour toutes les actions. Dans une architecture à base de 2TBNs, au contraire, on représente un réseau d'action pour chaque action, puis une CPT pour chaque attribut prédit au sein de ce réseau d'action. Donc dans les 2TBNs la distinction entre actions intervient avant la distinction des attributs prédits, alors que c'est le contraire dans MACS.

[17] *Conditional Probabilistic Tables*

Cela étant, hormis cette différence d'organisation, pour un couple (variable prédite, action), on retrouve la même information, représentée sous forme d'un arbre dans les 2TBNs et sous forme d'une liste de classeurs dans MACS, comme indiqué par le tableau 4.4.1.

W	U	R	O	HCR	HCO	Action	W	U	R	O	HCR	HCO	**proba. absente**
#	#	#	#	#	1	DelC	?	?	?	?	?	1	**1.0**
#	#	#	1	1	0	DelC	?	?	?	?	?	1	**0.8**
#	#	#	#	0	0	DelC	?	?	?	?	?	0	**1.0**
#	#	#	0	#	0	DelC	?	?	?	?	?	0	**1.0**

TAB. 4.1 – **Le CPT de $HCO/DelC$ représenté dans le formalisme de MACS**

En conséquence, traduire une CPT sous la forme d'un ensemble de classeurs au format de MACS est immédiat : pour chaque feuille de l'arbre, il suffit d'ajouter un classeur qui spécifie les conditions qui ont permis d'atteindre cette feuille au sein de l'arbre, avec un # pour les observables non pertinentes, puis d'ajouter l'action correspondante et la partie `[Effet]` correspondant à l'attribut prédit.

Une fois cette traduction effectuée, il apparaît une différence de taille : la CPT donne la probabilité pour que l'attribut ait la valeur prédite, il s'agit d'un modèle stochastique, alors que MACS donne une information déterministe de type « tout ou rien ».

Par ailleurs, la fonction de récompense est représentée dans MACS sous la forme d'une table exhaustive qui associe à chaque situation effectivement rencontrée une valeur. Nous avons dit au chapitre précédent que cet aspect de MACS constituait l'une de ses faiblesses car, dès lors que le problème traité fait intervenir un grand nombre d'états, cette liste de situations observées peut devenir très lourde à gérer. Dans les réseaux bayésiens dynamiques, au contraire, la fonction de valeur est représentée sous forme compacte, comme nous allons le voir à la section 4.5.

4.4.2 Comparaison des mécanismes

MACS est doté de la capacité à apprendre le modèle des transitions d'un problème, ce qui constitue un atout indiscutable vis-à-vis des réseaux bayésiens dynamiques, dans lequel ce modèle doit être donné a priori.

En revanche, certain des mécanismes et heuristiques de MACS sont moins efficaces que les traitements correspondant dans le domaine des processus de décision markovien factorisés. En particulier, le calcul de la fonction de valeur suppose, comme nous l'avons montré au chapitre précédent, la mise en œuvre d'algorithmes qui parcourent la liste des classeurs existants et réalisent des intégrations des différentes partie `[Effet]` pour déterminer la liste des états futurs atteignables. Ces algorithmes sont particulièrement inefficaces. Nous allons voir que les représentations arborescentes utilisées dans le cadre des réseaux bayésiens dynamiques permettent de gagner considérablement en efficacité.

Ces quelques éléments de comparaison font apparaître la nécessité de concevoir un nouveau système qui dispose d'une capacité d'apprentissage d'un modèle des transitions comme MACS, mais qui repose sur les algorithmes de planification plus clairs, plus efficaces et applicables dans un cadre stochastique que l'on trouve dans le cadre des processus de décision markovien factorisés.

Nous allons donc passer en revue quelques-uns de ces algorithmes avant de présenter notre proposition.

4.5 Programmation dynamique sur des représentations compactes

4.5.1 L'algorithme SPI

Le premier algorithme de planification proposé pour engendrer une politique optimale dans un processus de décision markovien factorisé est l'algorithme *Structured Policy Iteration* (SPI) [18]. Cet algorithme est directement inspiré de *Modified Policy Iteration* (MPI) [19], algorithme de programmation dynamique tabulaire que nous avons déjà présenté à la section 2.3.1, page 38. Toutefois, il s'agit d'une version conçue pour manipuler exclusivement des structures arborescentes, ce qui se traduit dans les cas favorables par une réduction exponentielle du coût d'application de l'algorithme vis-à-vis de son équivalent tabulaire [20].

L'opération centrale commune à tous les algorithmes de programmation dynamique et d'apprentissage par renforcement est la propagation des valeurs ou qualités au sein des états du processus de décision markovien. C. Boutilier appelle « régression théorique [21] » cette opération.

Cette propagation s'effectue en calculant les qualités conformément à l'équation de Bellman :

$$\forall s^t \in S, Q_a^V(s^t) = R(s^t, a^t) + \beta \sum_{s^{t+1} \in S} P(s^{t+1}|a^t, s^t) V(s^{t+1}) \tag{4.1}$$

puis en revenant aux valeurs avec

$$\forall s^t \in S, V(s^t) = arg\max a^t Q(s^t, a^t) \tag{4.2}$$

L'idée centrale de SPI consiste à réaliser cette propagation de façon plus efficace que dans le cas tabulaire en regroupant les états qui partagent les mêmes valeurs au sein des mêmes feuilles d'un arbre.

L'algorithme SPI maintient donc à jour un arbre de décision dont les feuilles contiennent tous les états qui ne méritent pas d'être distingués du point de vue de l'algorithme de propagation des valeurs.

Dans le cadre de cette représentation arborescente, on note :

- $QTree_a^V(s)$ l'arbre des qualités de l'action a étant donnée la fonction de valeur V ;
- $RTree(s, a)$ l'arbre des récompenses que l'on obtient en réalisant l'action a dans l'état s ;
- $VTree(s)$ l'arbre des valeurs de l'état s.

L'équation de propagation précédente devient alors :

$$\forall s^t \in S, QTree_a^V(s^t) = RTree(s^t, a^t) + \beta \sum_{s^{t+1} \in S} P(s^{t+1}|a^t, s^t) VTree(s^{t+1}) \tag{4.3}$$

L'algorithme de propagation des valeurs suit rigoureusement l'algorithme standard déjà présenté dans le cas tabulaire dans la section 2.3.1 à la page 38 :

- il commence avec une fonction de valeur initiale $VTree_0(s^t) = RTree(s^t, a^t)$.
- puis, pour mettre à jour $QTree_a^V(s^t)$, il applique l'équation 4.3 ci-dessus, en allant chercher les transitions $P(s^{t+1}|a^t, s^t)$ dans les CPTs de chacune des actions concernées, ce qui donne directement sous forme arborescente l'information utile à la propagation.

Cet algorithme peut calculer directement la représentation arborescente de la qualité de chacune des actions à partir de la représentation arborescente de la fonction de valeur courante et réciproquement, ce qui permet de

[18] [Boutilier et al., 1995, Boutilier et al., 2000]
[19] [Howard, 1971, Puterman & Shin, 1978]
[20] [Boutilier et al., 2000]
[21] *Decision-Theoretic Regression*

réaliser un algorithme arborescent d'itération sur les valeurs, que nous appelons « *Structured Value Iteration* » (SVI). Nous ne décrirons pas cet algorithme plus en détail, le lecteur désireux d'en savoir plus peut se reporter à l'article original [22].

En s'appuyant sur SVI comme brique de base, les auteurs proposent une décomposition de l'algorithme SPI en deux phases, comme c'est le cas pour tout algorithme d'itération sur la politique :

- une phase d'évaluation de la politique courante [23], qui consiste à évaluer la récompense à long terme que l'agent peut obtenir à partir d'une politique fixée ;
- une phase d'amélioration de la politique courante [24], qui consiste à améliorer la politique courante en fonction de la nouvelle fonction de valeur calculée lors de la phase précédente. Cette phase est réalisée de manière arborescente de façon similaire à la précédente, à ceci près que, au lieu d'étiqueter les feuilles des arbres $VTree(s)$ avec les valeurs maximales estimées, on les étiquette avec les actions qui permettent d'atteindre ces valeurs maximales.

4.5.2 L'algorithme SPUDD

L'algorithme SPUDD [25] est une variante de SPI qui utilise des « Diagrammes de Décision Algébriques » ou ADDs [26] [27] plutôt que des arbres, permettant de prendre en compte des symétries dans l'arbre afin d'engendrer une représentation encore plus compacte. C. Boutilier et ses collègues proposent sur internet une version du logiciel APRICODD [28] qui implémente l'algorithme SPUDD [29].

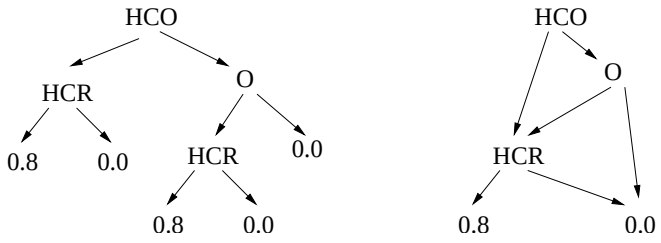


FIG. 4.3 – **L'arbre de la figure 4.1.c et le diagramme de décision algébrique correspondant**

Les ADDs constituent une extension des diagrammes de décision binaires [30] à l'espace des réels. La figure 4.3 donne un exemple d'arbre et l'ADD correspondant. L'idée générale est que les ADDs permettent de construire une représentation sous forme de graphe encore plus compacte que les arbres car, partout où un arbre fait apparaître des sous-arbres identiques, l'ADD correspondant ne fait apparaître ce sous-arbres qu'une seule fois, en tant que fils de plusieurs nœuds. La représentation à base d'ADDs est donc intéressante dès lors qu'il y a plusieurs sous-arbres identiques dans le même arbre.

Nos expérimentations sur des problèmes jouets dans lesquels il y a une seule source de récompense semblent montrer que c'est souvent le cas. J. Hoey et ses collègues montrent que, dans de tels cas, SPUDD est beaucoup plus

[22] [Boutilier et al., 2000]
[23] appelée « *Structured Successive Approximation* » (SSA) dans [Boutilier et al., 2000]
[24] appelée « *Structured Policy Improvement* » dans [Boutilier et al., 2000]
[25] *Stochastic Planning Using Decision Diagrams*
[26] *Algebraic Decision Diagrams*
[27] [R.I. Bahar et al., 1993, Hoey et al., 1999]
[28] [St-Aubin et al., 2000]
[29] http ://www.cs.ubc.ca/spider/staubin/Spudd/
[30] *Binary Decision Diagrams* ou BDDs

efficace que SPI, mais cette efficacité s'obtient au prix de complications supplémentaires de l'algorithme.

Nous allons voir que, dans le cadre du stage de DEA de C. Lutz [31], l'utilisation de SPI a été préférée à celle de SPUDD car nous disposons d'un algorithme capable d'apprendre un modèle des transitions sous forme d'arbres et non d'ADDs. Il serait cependant possible, à condition de pouvoir convertir les arbres appris en ADDs, d'utiliser ensuite SPUDD pour le calcul de la fonction de valeur. Il y a toutefois plusieurs obstacles à l'utilisation de SPUDD.

Le premier est que, dans SPUDD, les variables manipulées doivent être booléennes, ce qui n'est pas le cas de SPI qui peut manipuler des variables quelconques. Il est possible de transformer une variable n-aire en $log(n)$ variables binaires, mais la taille de l'espace d'états est alors augmentée (sauf si n est une puissance de 2).

Le deuxième inconvénient de SPUDD est qu'il prend en compte des variables n'ayant aucune influence sur la dynamique du système. Si, par exemple, on ajoute à la description d'un problème une variable $color$ donnant la couleur du sol là où se déplace un agent, et n'ayant aucune influence sur les autres variables, SPI ne prendra pas en compte cette variable, tandis que le temps de calcul de SPUDD sera multiplié par $Dom(color)$. Ce problème ne se pose pas pour des problèmes jouets mais dans le cas de systèmes correspondants à des problèmes réels, il peut arriver que de telles variables soient présentes.

Enfin, nous soupçonnons que, lorsque le nombre de récompenses augmente, les sous-arbres identiques tendent à disparaître, si bien que cette technique sera moins intéressante dans le cadre des simulations complexes auxquelles nous nous intéressons.

4.6 Résolution de FMDPs par programmation linéaire

Une autre classe d'algorithmes pour effectuer une planification efficace dans le cadre des processus de décision markoviens factorisés provient de l'équipe de D. Koller et s'appuie sur la programmation linéaire.

4.6.1 La programmation linéaire

La programmation linéaire est une méthode d'optimisation qui permet de maximiser une fonction linéaire d'un ensemble de variables, appelée fonction objectif, tout en vérifiant un ensemble de contraintes linéaires qui restreignent le domaine des valeurs possibles pour ces variables. On ramène toujours ces contraintes à des inégalités, une égalité pouvant s'exprimer sous la forme de deux inégalités.

L'algorithme standard qui permet de résoudre un programme linéaire est appelé « méthode du simplexe ». Pour expliquer son fonctionnement, il est bon d'avoir recours à une représentation géométrique.

Dans l'espace engendré par les variables du problème, chaque contrainte définit un hyperplan qui sépare le domaine des ensembles de valeurs possibles de celui des ensembles de valeurs interdites. L'ensemble des valeurs qui satisfont toutes les contraintes peut être vide, si les contraintes sont mutuellement exclusives, s'étendre à l'infini si les contraintes ne bornent pas certaines variables, ou bien être délimité par un polyèdre convexe, dans le cas qui nous intéresse. La fonction objectif définit une direction de l'espace dans laquelle il est bon de se déplacer pour maximiser l'objectif. L'optimum est atteint pour le sommet du polyèdre le plus extrême dans la direction d'optimisation. Ce point définit en effet les valeurs des variables pour lesquelles la fonction objectif sera optimale. La méthode du simplexe consiste à réaliser l'équivalent algébrique d'un déplacement de sommet en sommet vers ce sommet optimal, en suivant la plus grand pente relativement à la direction définie par la fonction objectif. Nous n'entrerons pas davantage dans les détails de la méthode dans le cadre de ce document, le lecteur peut se référer à [Chvatal, 1983].

[31] [Lutz, 2004]

4.6.2 Application directe aux processus de décision markoviens

Il est possible de trouver la fonction de valeur optimale d'un processus de décision markovien en représentant la recherche de cette fonction sous la forme d'un programme linéaire. Dans ce cas, les variables du programme linéaire sont les valeurs $V(s^t)$ associées à chacun des états s^t. Les états étant discrets, nous choisissons de noter V_s la fonction de valeur à l'état s, plutôt que $V(s)$. Cette notation pourrait faire croire que l'on dispose d'une expression analytique de la fonction alors qu'il s'agit en fait d'un vecteur donnant une valeur pour chaque état.

On sait que les valeurs optimales $V^*_{s^t}$ que l'on cherche définissent un point fixe pour l'opérateur de Bellman, ré-écrit dans la nouvelle notation :

$$\forall a^t \in A, \forall s^t \in S, V^*_{s^t} = R_{s^t}(a^t) + \gamma \sum_{s^{t+1} \in S} P(s^{t+1}|s^t, a^t) V^*_{s^{t+1}} \tag{4.4}$$

La recherche de ces valeurs optimales sous la forme d'un programme linéaire s'exprime en cherchant à minimiser le terme $\sum_s \alpha_s V_s$, où les poids α_s sont tous positifs, en posant l'ensemble de contraintes :

$$\forall a^t \in A, \forall s^t \in S, V_{s^t} \geq R_{s^t}(a^t) + \gamma \sum_{s^{t+1} \in S} P(s^{t+1}|s^t, a^t) V_{s^{t+1}} \tag{4.5}$$

L'intuition sous-jacente est que les variables V_{s^t} qui minimisent le mieux la fonction objectif sont celles qui vérifient l'équation de point fixe de l'opérateur de Bellman, donc celles pour lesquelles les inégalités exprimées par la contrainte deviennent des égalités [32].

Il faut noter que, dans cette approche, le déplacement d'un sommet à un autre consiste à maximiser une seule variable à la fois, alors que dans un algorithme de type itération sur la politique, on maximise plusieurs variables à la fois, comme l'indique M. Puterman [33]. En conséquence, le premier réalise a priori plus d'itérations que le second, même si ces itérations sont a priori plus courtes.

Trouver la fonction de valeur optimale d'un processus de décision markovien de cette façon est possible en principe, mais le programme linéaire correspondant comporte autant de variables qu'il y a d'états dans le processus de décision markovien et un nombre de contraintes égal au produit du nombre d'états par le nombre d'actions, si bien que présenter les données sous cette forme à un outil de résolution automatique par la méthode du simplexe est très coûteux en pratique.

4.6.3 Application compacte pour les FMDPs

Dans le cadre factorisé, chaque état s'écrit comme une collection de variables aléatoires et, pour n variables qui peuvent prendre k valeurs, il y a k^n états possibles, si bien que les énumérations ci-dessus sont exponentielles en fonction du nombre de variables.

Il y a trois sources d'explosion combinatoire dans la représentation décrite ci-dessus.

- la fonction à minimiser fait intervenir une valeur par état du problème, donc k^n valeurs ;
- chaque contrainte fait intervenir une probabilité de transmission pour chaque couple d'états du problème, donc $(k^n)^2$ valeurs de probabilité ;
- enfin, le nombre de contraintes est linéaire en fonction du nombre d'états du problème et linéaire en fonction du nombre d'actions du problème, donc on a $|A| \times (k^n)$ contraintes, où $|A|$ est le nombre d'actions.

[32] On dit alors que les contraintes sont saturées

[33] [Puterman, 1994]

On va donc s'efforcer de tirer parti de la structure des dépendances entre les variables pour limiter chacune de ces explosions. Le principe général de réduction de la complexité consiste à remplacer des énumérations de groupes n variables ayant chacune k valeurs possibles, donc k^n termes à énumérer, par des séquences d'énumérations conjointes de sous-parties de ces groupes de variables, en faisant l'hypothèse qu'il y a indépendance entre les énumérations de chaque sous-partie. On remplace alors une combinatoire en k^n par une combinatoire en $k^{n_1} + k^{n_2} + ... + k^{n_m}$, où $n_1 + n_2 + ... + n_m \geq n$.

La méthode proposée par D. Koller et ses collaborateurs sur la base de ce principe permet de réduire la taille du problème en ayant recours à trois opérations, dont chacune s'attaque respectivement aux trois sources d'explosion combinatoire identifiées ci-dessus :

- d'une part, approximer la fonction de récompense et la fonction de valeur globale par un ensemble de fonctions linéaires dont les supports sont réduits à une partie des variables ;
- d'autre part, utiliser la représentation factorisée du processus de décision markovien pour bénéficier de la structure des dépendances entre les variables, afin de réduire le nombre de termes dans chacune des contraintes ;
- enfin, avoir recours à un processus d'élimination de variables pour réduire le nombre de contraintes à traiter effectivement.

Il faut noter qu'une quatrième source d'explosion, relative au nombre d'actions, n'est pas traitée par les développements qui vont suivre. Une telle explosion est en particulier susceptible de se produire dans le cadre des problèmes multi-agents. L'article de C. Guestrin sur lequel nous nous appuyons propose un moyen de le traiter en combinant programmation linéaire et itération sur la politique, nous évoquerons ce moyen lorsque nous aborderons spécifiquement la classe des problèmes multi-agents, au chapitre 9.

L'algorithme que proposent C. Guestrin et ses collègues peut être vu comme une succession d'étapes visant à présenter la dynamique d'un processus de décision markovien factorisé sous la forme d'un programme linéaire de taille aussi faible que possible, au prix d'une approximation dont on maîtrise l'impact sur l'écart entre la fonction de valeur obtenue et la fonction de valeur optimale.

Seule la première étape induit une approximation sur la fonction de valeur obtenue. Les deux suivantes sont des optimisations qui ne changent pas le résultat, mais permettent de l'obtenir plus vite. Ces étapes successives sont décrites dans les sections qui suivent.

4.6.4 Approximation linéaire de la fonction de valeur

La première étape consiste à approximer la fonction de récompense et la fonction de valeur recherchée par des sommes pondérées de fonctions h_i dont le support est restreint à un sous-ensemble des variables qui définissent le processus de décision markovien. Si un état s^t s'écrit sous la forme d'une collection de variables aléatoires instanciées $s^t = \{x_1, x_2, ..., x_n\}$, on note $y_i \in s^t$ un sous-ensemble de s^t, par exemple $\{x_2, x_5, x_n\}$. Nous reviendrons dans une partie de discussion sur le fait que trouver une décomposition de la fonction de récompense et de la fonction de valeur qui permette cette opération ne va pas de soi.

Pour tout état $s \in S$ on pose $V_{s^t} = \sum_i w_i h_i(s^t)$, de telle façon que $h_i(s^t) = h_i(y_i^t)$, où y_i^t est une instanciation de Y_i qui est strictement inclus dans X.

Les fonctions h_i étant posées, les variables sont à présent les poids w_i et le programme linéaire revient à minimiser le terme $\sum_s \alpha_{s^t} \sum_i w_i h_i(y_i^t)$, sous les contraintes :

$$\forall a^t \in A, \forall s^t \in S, \sum_{s^t} w_i h_i(s^t) \geq R(s^t, a^t) + \gamma G_{s^t}^{a^t} \tag{4.6}$$

où

$$G_{a^t}(s^t) = \sum_{s^{t+1}} P(s^{t+1}|s^t, a^t) \sum_i w_i h_i(s^t)$$

ou encore

$$G_{a^t}(s^t) = \sum_i w_i \sum_{y_i^{t+1}} P(y_i^{t+1}|s^t, a^t) h_i(y_i^{t+1})$$

Dans ce nouveau programme linéaire, le nombre de variables est égal au nombre de fonctions h_i que l'on a posées. Le terme à minimiser est donc de dimension réduite par rapport au nombre d'états. Pour ce qui est des contraintes, par contre, le terme $g_i^{a^t}(s^t) = \sum_{y_i^{t+1}} P(y_i^{t+1}|s^t, a^t) h_i(y_i^{t+1})$ suppose encore une énumération exhaustive du nombre d'états du problème pour déterminer les probabilités de transition. Pour réduire cette énumération, on fait appel au caractère factorisé de la représentation, ce que nous examinons dans la section suivante.

4.6.5 Prise en compte de la structure des dépendances entre variables

On cherche à présent à réduire le terme $g_i^{a^t}(s^t) = \sum_{y_i^{t+1}} P(y_i^{t+1}|s^t, a^t) h_i(y_i^{t+1})$. Dans une représentation factorisée, la valeur de chacune des variables définissant y_i^{t+1} à chaque instant ne dépend que d'une partie des mêmes variables à l'instant précédent.

On note $\Gamma_a(y_i^{t+1})$ l'ensemble des variables d'état à l'instant t qui ont une influence sur la valeur de y_i^{t+1} à l'instant $t+1$. Ces $\Gamma_a(y_i^{t+1})$ sont donnés sous forme arborescente à partir des CPT des variables intervenant dans y_i^{t+1} (voir section 4.4.1, page 83).

On a alors $g_i^{a^t}(\Gamma_a(y_i^{t+1})) = \sum_{y_i^{t+1}} P(y_i^{t+1}|\Gamma_a(y_i^{t+1}), a^t) h_i(y_i^{t+1})$, ce qui se calcule beaucoup plus rapidement que l'expression précédente.

Les contraintes du programme linéaire deviennent alors

$$\forall a^t \in A, \forall s^t \in S, \sum_i w_i h_i(s^t) \geq R(s^t, a^t) + \gamma \sum_i w_i g_i^{a^t}(\Gamma_a(y_i^{t+1})) \tag{4.7}$$

ou encore

$$\forall a^t \in A, \forall s^t \in S, 0 \geq R(s^t, a^t) + \sum_i w_i (\gamma g_i^{a^t}(\Gamma_a(y_i^{t+1})) - h_i(y_i^{t+1})) \tag{4.8}$$

Le calcul de cette expression est devenu beaucoup plus compact, même s'il fait encore intervenir un nombre exponentiel de facteurs. On s'efforce de gagner encore en compacité grâce à une technique d'élimination de variables qui permet cette fois de réduire le nombre de contraintes.

4.6.6 Réduction du nombre de contraintes

La dernière opération pour réduire la taille du programme linéaire qui permet de résoudre le processus de décision markovien est la partie de l'algorithme la plus délicate à comprendre.

On dispose pour l'instant d'un ensemble de contraintes dont le cardinal est encore égal au produit du nombre d'états par le nombre d'actions du problème :

$$\forall a^t \in A, \forall s^t \in S, 0 \geq R(s^t, a^t) + \sum_i w_i (\gamma g_i^{a^t}(\Gamma_a(y_i^{t+1})) - h_i(y_i^{t+1})) \tag{4.9}$$

Le terme de droite étant toujours négatif ou nul dans toutes ces contraintes, le maximum de ces termes l'est aussi. Une première étape consiste donc à remplacer ce grand nombre de contraintes par une seule contrainte par action qui les résume toutes :

$$\forall a^t \in A, 0 \geq \max_{s^t} [R(s^t, a^t) + \sum_i w_i(\gamma g_i^{a^t}(\Gamma_a(y_i^{t+1})) - h_i(y_i^{t+1}))] \tag{4.10}$$

Les nouvelles contraintes sont beaucoup moins nombreuses, mais elles ne sont plus linéaires, puisqu'elles impliquent une maximisation, donc elles ne peuvent plus être résolues par la programmation linéaire. En outre, chercher à les satisfaire sous cette forme n'est pas moins coûteux en calcul pour l'instant, puisqu'il faut énumérer les termes pour trouver leur maximum.

Pour réaliser l'étape suivante, on tire parti de la décomposition des états x et y_i^{t+1} sous forme d'un ensemble de variables aléatoires. On note $c_a^i(y_i^{t+1}, \Gamma_a(y_i^{t+1})) = \gamma g_i^{a^t}(\Gamma_a(y_i^{t+1})) - h_i(y_i^{t+1})$.

On considère que, pour tout s, $R(s^t, a^t)$ et $c_a^i(s, \Gamma_a(s^t))$ peuvent s'écrire sous la forme d'une somme de fonctions de ces variables aléatoires dans laquelle chaque terme ne fait intervenir qu'une partie de ces variables.

L'objectif de l'étape qui suit est de transformer l'équation 4.10, qui fait intervenir une maximisation, en un ensemble de contraintes linéaires beaucoup plus petit que l'ensemble de contraintes dont on disposait au départ, pour trouver le maximum par programmation linéaire.

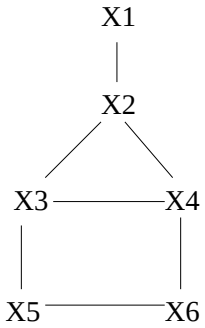


FIG. 4.4 – **Le graphe des relations entre variables correspondant à l'équation 4.11**

Considérons un exemple dans lequel les états s'écrivent $s^t = \{x_1, x_2, x_3, x_4, x_5, x_6\}$ et l'équation 4.10 s'écrit sous la forme :

$$M = \max_{s^t} [f_a(x_1, x_2) + f_b(x_2, x_3, x_4) + f_c(x_3, x_5) + f_d(x_5, x_6) + f_e(x_4, x_6)] \tag{4.11}$$

À l'équation 4.11, on peut associer le graphe de la figure 4.4, appelé « graphe de coût [34] », dans lequel chaque clique maximale [35] indique que les variables qu'elle relie sont les variables d'une même fonction f_k. Eliminer une variable d'une fonction dans une équation du type de l'équation 4.11 consiste à maximiser la fonction sur cette variable, ce qui permet d'obtenir une nouvelle fonction qui porte sur le reste des variables. Un graphe de coût permet de déterminer l'ordre dans lequel éliminer les variables de façon à simplifier au maximum les calculs. En effet, à chaque fois qu'on élimine un nœud du graphe, ce qui se traduit par l'élimination d'une variable dans l'équation, il faut relier ensemble tous ses voisins qui ne le sont pas encore, ce qui se traduit par l'introduction d'une nouvelle fonction de tous ses voisins dans l'équation. Or on cherche à introduire le minimum d'arcs nouveaux, donc le minimum de termes dépendants de plusieurs variables dans la somme.

[34] *cost network*

[35] On appelle « clique » d'un graphe un ensemble de nœuds de ce graphe tels que tous les nœuds sont tous reliés les uns aux autres. Une clique est dite « maximale » s'il n'existe pas de nœud du graphe hors de la clique qui soit relié à tous les nœuds de la clique.

Ainsi, dans le graphe de la figure 4.4, si l'on commençait par éliminer x_2, il faudrait ajouter un arc entre x_1 et x_3 et un arc entre x_1 et x_4. En effet, on a $\max_{x_2} f_a(x_1, x_2) + f_b(x_2, x_3, x_4) = F_{ab}(x_1, x_3, x_4)$ si bien qu'éliminer la variable x_2 fait apparaître une nouvelle clique x_1, x_3, x_4. Toujours dans cet exemple, il apparaît judicieux d'éliminer d'abord x_1, puis x_2, puis les variables x_3, x_4, x_5 et x_6 dans un ordre quelconque. Si l'ordre optimal est évident sur cet exemple, trouver l'ordre optimal pour un graphe quelconque est un problème NP-complet [36]. Ce problème est laissé ouvert par C. Guestrin [37], qui se contente de renvoyer à d'autres travaux [38] pour déterminer des heuristiques efficaces.

Pour transformer l'équation 4.11 en un ensemble de contraintes linéaires, on commence donc par éliminer x_1 en posant $e_a(x_2) = \max_{x_1} f_a(x_1, x_2)$. L'équation 4.11 devient

$$M = \max_{x_2 x_3 x_4 x_5 x_6} [e_a(x_2) + f_b(x_2, x_3, x_4) + f_c(x_3, x_5) + f_d(x_5, x_6) + f_e(x_4, x_6)]$$

dans laquelle x_1 n'apparaît plus. On élimine ensuite x_2 en posant $e_b(x_3, x_4) = \max_{x_2} [f_b(x_2, x_3, x_4) + e_a(x_2)]$. L'équation précédente devient alors

$$M = \max_{x_3 x_4 x_5 x_6} [e_b(x_3, x_4) + f_c(x_3, x_5) + f_d(x_5, x_6) + f_e(x_4, x_6)]$$

d'où x_2 a disparu.

On applique ainsi l'algorithme variable après variable jusqu'à obtenir le résultat $e_e = \max_{x_6} e_e(x_6)$. À chaque variable que l'on élimine, on ajoute au programme linéaire les contraintes qui permettent de trouver la valeur optimale pour cette variable.

L'idée est que calculer un maximum sur des variables isolées comme dans le cas de $\max_{x_6} e_e(x_6)$ ne fait intervenir d'énumération que sur les variables en question, si bien qu'on va présenter au programme linéaire un nombre de contraintes linéaires beaucoup plus petit en isolant ainsi les termes un à un par rapport à une énumération globale.

Tous les termes isolés ainsi supposent le calcul d'un maximum, donc ils ne s'expriment pas directement sous forme de contraintes linéaires.

Pour se ramener à des contraintes linéaires, il reste quelques opérations à réaliser. Tout d'abord, on pose des variables matricielles, par exemple $U^{f_a}_{x_1,x_2} = f_a(x_1, x_2)$, où chaque valeur de U est donnée par la valeur que donne la fonction f pour les valeurs correspondantes de x_1 et x_2. On a aussi $e_a(x_2) = \max_{x_1} f_a(x_1, x_2)$ et on note $U^{e_a}_{x_2}$ la variable matricielle – ici un vecteur – correspondant à la fonction $e_a(x_2)$.

Etant donné la forme du programme linéaire que l'on construit, il existe un couple de valeurs pour x_1 et x_2 tel que l'on ait $U^{e_a}_{x_2} = \max_{x_1} U^{f_a}_{x_1,x_2}$.

On peut alors remplacer chacune des équations de la forme ci-dessus, obtenues en introduisant des variables matricielles, par des contraintes de la forme $\forall x_1, U^{f_a}_{x_1,x_2} \leq U^{e_a}_{x_2}$, qui est une contrainte linéaire en x_2.

On fait de même pour chacune des équations, ce qui permet de ramener notre ensemble initial de 2^n contraintes à un ensemble de k groupes de n contraintes, donc $k \times n$ contraintes au total.

Dans le cas qui nous intéresse, à savoir la transformation de l'équation 4.10 en un programme linéaire, les fonctions f_k sont issues de la décomposition additive des termes $R(s^t, a^t)$ et $c^i_a(s^t, \Gamma_a(s^t))$. La décomposition additive des termes $R(s^t, a^t)$ est une donnée du problème qu'il faut fournir à l'algorithme et celles des termes $c^i_a(s^t, \Gamma_a(s^t))$ s'obtient à partir de la décomposition en fonctions h_i en calculant les termes $\gamma g^{a^t}_i(\Gamma_a(y^{t+1}_i)) - h_i(y^{t+1}_i)$.

[36] [Rose, 1970, Arnborg et al., 1987, Kjærulff, 1993]

[37] [Guestrin et al., 2003b]

[38] [Bertele & Brioschi, 1972, Reed, 1992, Becker & Geiger, 2001]

C. Guestrin [39] donne alors une description algorithmique directe qui permet de réaliser le plus efficacement possible les étapes que nous avons présentées ici dans un ordre didactique plutôt qu'algorithmique. Il n'en reste pas moins que cet algorithme est difficile à comprendre.

4.6.7 Discussion

Outre les difficultés à le comprendre et à le mettre en œuvre, l'algorithme que nous venons de présenter pose un certain nombre de problèmes. Tout d'abord, pour l'appliquer il faut commencer par fournir une bonne décomposition additive de la fonction de récompense, ce qui ne va pas de soi. Notre intuition pour l'instant est que la fonction de récompense se décompose sans doute aisément dans le cadre de certains problèmes multi-agents, mais cette intuition reste à vérifier. Il faut ensuite trouver une décomposition du même type pour la fonction de valeur. Une solution heuristique peut consister à utiliser la même décomposition que pour la fonction de récompense, mais cette heuristique médiocre risque de donner d'assez mauvais résultats. Comme le suggère C. Guestrin, une décomposition bien meilleure peut être obtenue en considérant l'arbre représentant la fonction de valeur engendré par l'algorithme SPI lors de ses itérations successives. On peut décomposer selon le découpage imposé à la fonction de récompense, ce qui correspond à l'itération O, ou selon les variables qui figurent dans les états mesurant immédiatement à la récompense, c'est-à-dire selon les parents dans les CPT des variables impliquées dans les arbres de récompense, ce qui correspond à l'itération 1, ou bien selon les parents de ces parents, etc., jusqu'au niveau où l'arbre des valeurs n'évolue plus, auquel cas on devrait obtenir une décomposition réellement satisfaisante.

Une force de l'algorithme vis-à-vis du délicat problème de décomposition que l'on pose ici, est qu'il dispose d'un moyen explicite de calculer l'erreur d'approximation engendrée par la mauvaise qualité d'une décomposition jusqu'à atteindre un niveau d'erreur acceptable. Par ailleurs, il faut avoir à l'esprit le fait que les décompositions dont nous parlons ici ne sont pas de même nature que la décomposition qui opère dans le cadre des algorithmes SPI et SPUDD. Il apparaît possible de combiner ces deux types de décomposition, ce à quoi nous devrons réfléchir dans le cadre de nos travaux futurs.

Enfin, il doit être clair que les algorithmes SPI, SPUDD et l'algorithme que nous venons de présenter supposent que la structure de dépendance entre les variables est connue. Nous allons à présent nous pencher sur la possibilité d'apprendre la structure de ces dépendances, ce qui impose de réfléchir à la capacité qu'auront ces algorithmes à fonctionner dans un cadre où le modèle des transitions est incomplet, voire erroné.

4.7 Apprentissage par renforcement dans les FMDPs

Le point faible le plus manifeste de la méthode reposant sur la programmation linéaire provient de ce qu'il est nécessaire pour l'appliquer de disposer a priori d'une bonne décomposition additive de la fonction de valeur et de la fonction de récompense, faute de quoi les approximations réalisées par l'algorithme sont susceptibles d'être très médiocres. Or identifier une telle décomposition ne va pas de soi, sauf dans des cas particuliers, comme certains problèmes multi-agents. Il faut donc travailler sur les méthodes qui permettent d'obtenir automatiquement une bonne décomposition.

D'une façon plus générale, l'un des principaux obstacles à l'utilisation des réseaux bayésiens est la nécessité de fournir à la fois la structure des relations de dépendance entre les variables aléatoires qui les constituent et les paramètres qui permettent de quantifier ces dépendances. Une voie de recherche fructueuse dans ce domaine

[39] [Guestrin et al., 2003b]

consiste à essayer d'obtenir soit les paramètres, soit à la fois les paramètres et la structure du processus de décision markovien factorisé à l'aide de techniques d'apprentissage.

Un certain nombre d'auteurs se sont focalisés sur l'apprentissage de structures complexes dans le cadre général des réseaux bayésiens dynamiques [40]. Ces travaux consistent pour la plupart à généraliser au cadre des réseaux bayésiens dynamiques les algorithmes développés pour l'apprentissage de modèles de Markov cachés [41]. Nous ne les détaillerons pas dans ce document car les hypothèses sur lesquelles ils reposent sont incompatibles avec les contraintes d'incrémentalité que le contexte de l'apprentissage en ligne nous impose.

Cependant, le développement des processus de décision markoviens factorisés a récemment donné naissance à une voie de recherche spécifique visant à transposer les algorithmes classiques d'apprentissage par renforcement au cadre factorisé. Ce sont ces algorithmes que nous allons examiner à présent.

4.7.1 Les algorithmes E^3 et Factored-E^3

Les premiers travaux qui ont permis d'appliquer le principe de l'apprentissage par renforcement dans le cadre des processus de décision markoviens factorisés sont le fait de M. Kearns. En effet, après avoir collaboré avec S. Singh à la mise au point d'un algorithme d'apprentissage par renforcement dont la convergence est polynomiale en fonction du nombre d'états – et non pas exponentielle [42] –, M. Kearns s'est efforcé sous la direction de D. Koller de généraliser son travail au cadre des processus de décision markoviens factorisés [43].

L'algorithme publié avec S. Singh s'appelle E^3, pour « *Explicit Explore and Exploit* » et celui publié avec D. Koller s'appelle Factored-E^3. Les deux algorithmes, ainsi que l'algorithme R_{max} dont nous parlerons plus loin, disposent d'une preuve de convergence, dans le cas où l'on cherche à maximiser la récompense moyenne sur un horizon fini.

De même que les architectures DYNA ou les systèmes de classeurs à anticipation, l'algorithme E^3 repose sur la construction d'un modèle des transitions et des récompenses. Néanmoins, au lieu de chercher à construire un modèle de l'environnement tout entier, il ne mémorise qu'un sous-ensemble des transitions rencontrées, à savoir celles qui jouent un rôle dans la définition de la politique optimale.

Pour construire ce modèle, l'algorithme visite les états de façon homogène, ce qui signifie que, quand il arrive dans un nouvel état, il choisit l'action qu'il a effectuée le moins souvent dans cet état. Il tient alors à jour une probabilité observée de transition vers des états suivants, ce qui revient à se donner un modèle approché du processus de décision markovien sous-jacent.

Au cœur de l'algorithme E^3 se trouve un mécanisme de gestion du compromis entre exploration et exploitation qui repose sur le principe dit « de la case courrier [44] ». L'idée est que, si l'on visite un nombre fini d'états de façon homogène, il finit toujours par y avoir un état qui a été visité suffisamment souvent pour que l'estimation des probabilités de transition construite sur la base des observations à partir de cet état soit proche de la distribution de probabilité effective engendrée par l'environnement. Les auteurs définissent alors une notion d'état « connu » de telle façon que le nombre de visites suffisant pour connaître un état reste polynomial en fonction de la taille du problème.

Le modèle construit par E^3 est un processus de décision markovien dans lequel figurent tous les états « connus » avec les probabilités de transition observées et un état absorbant qui représente à lui seul tous les états non encore « connus ». L'algorithme est alors face à une alternative :

[40] [Heckerman et al., 1995, Ghahramani, 1997, Friedman et al., 1998]
[41] [Rabiner & Juang, 1986]
[42] [Kearns & Singh, 1998]
[43] [Kearns & Koller, 1999]
[44] *The pigeon hole principle*

– soit il existe une politique proche de l'optimum qui repose uniquement sur des états connus, et l'algorithme peut trouver cette politique en appliquant un algorithme de planification sur le modèle qu'il a construit ;
– soit cette condition n'est pas vérifiée et il faut explorer davantage, donc choisir des actions qui mènent à l'état absorbant afin de mieux connaître les états de l'environnement qu'il regroupe.

Cet algorithme est simple et présente l'intérêt de disposer de preuves de convergence qui garantissent le caractère polynomial de cette convergence. Pour déterminer dans quelle branche de l'alternative ci-dessus on se trouve, l'algorithme est capable de calculer si explorer davantage au-delà d'un horizon fixé pour mieux connaître le modèle permet d'améliorer de façon significative la meilleure politique connue. Si ce n'est pas le cas, il vaut mieux exploiter qu'explorer.

L'algorithme Factored-E^3 est simplement une généralisation de l'algorithme E^3 au cas factorisé. Alors que l'algorithme E^3 est limité par nature aux processus de décision markoviens de petite dimension puisqu'il s'efforce d'en bâtir un modèle en énumérant les états, Factored-E^3 peut s'appliquer à des problèmes de taille beaucoup plus grande car il tire parti du caractère factorisé de la représentation pour bâtir un modèle beaucoup plus compact. Nous ne présenterons pas la version factorisée, qui bénéficie des mêmes preuves de convergence que la version tabulaire.

4.7.2 R_{max}, Factored-R_{max} et la proposition de C. Guestrin

L'algorithme R_{max} [45] apparaît comme une alternative intéressante à l'algorithme E^3. Il repose sur le même principe général que E^3 consistant à construire un modèle des transitions et à rechercher une politique qui maximise la récompense moyenne sur un horizon fini. Cependant, d'une part, il simplifie la gestion du dilemme entre exploration et exploitation en initialisant toutes les espérances de récompense à une valeur optimiste égale au maximum des récompenses immédiates atteignables dans l'environnement [46] et, d'autre part, il étend le cadre des processus de décisions markoviens dans lequel travaille E^3 au cadre des jeux stochastiques à somme nulle, ce qui permet de prendre en compte la présence d'un adversaire. La principale différence provient de ce que l'initialisation des récompenses à une valeur optimiste permet à l'algorithme R_{max} d'éviter d'avoir à gérer explicitement le compromis entre exploration et exploitation. En effet, l'agent se contente d'aller vers les états dans lesquels il s'attend à recevoir une récompense élevée. La récompense attendue peut être élevée pour deux raisons : soit parce que l'état est mal connu et dans ce cas le comportement de l'agent relève de l'exploration, soit parce que l'agent sait qu'il a déjà reçu des récompenses dans l'état en question et son comportement relève de l'exploitation.

Les auteurs argumentent la supériorité de leur approche sur celle de l'algorithme E^3 en soulignant que, en présence d'un adversaire, on ne maîtrise pas complètement les transitions du système, donc on ne maîtrise pas totalement le choix entre exploration et exploitation. Avec R_{max}, l'algorithme explore ou exploite « au mieux » compte tenu des choix réalisés par l'adversaire. L'algorithme est globalement plus simple et constitue donc un excellent candidat pour gérer efficacement le compromis entre exploration et exploitation.

R_{max} a lui aussi été adapté au cadre factorisé par C. Guestrin [47], dans un souci de comparaison avec sa propre proposition.

Comme l'algorithme E^3, R_{max} repose sur deux hypothèses irréalistes qui le rendent difficilement applicable en pratique. D'une part, on suppose connu l'horizon T au-delà duquel chercher à améliorer le modèle en explorant ne permet plus d'améliorer la politique de façon significative. D'autre part, on suppose qu'à chaque pas de temps, on est capable de déterminer efficacement la politique optimale sur cet horizon T, compte tenu de la connaissance dont on dispose dans le modèle. La difficulté provient de ce que T peut être grand, si bien que trouver une politique

[45] [Brafman & Tennenholtz, 2001]
[46] d'où le nom de l'algorithme, R_{max}
[47] [Guestrin et al., 2002b]

optimale sur cet horizon peut être très long et que donner à T une valeur arbitraire suppose de prendre une valeur encore plus grande que la valeur idéale, ce qui ne fait qu'aggraver le problème.

En vue d'applications pratiques de ces algorithmes conçus essentiellement en fonction de la possibilité de disposer de preuves théoriques de convergence, des variantes ont été proposées pour accélérer la recherche d'une politique optimale en faisant un échantillonnage heuristique [48] des transitions plutôt qu'une exploration systématique homogène [49].

Pour sa part, C. Guestrin propose un algorithme d'apprentissage qui s'appuie sur la méthode de programmation linéaire que nous venons de présenter et dirige l'exploration de façon à tirer parti au maximum de la validité courante de l'estimation fournie par le programme linéaire.

L'idée générale, que nous ne détaillerons pas, consiste à se doter d'une borne supérieure et d'une borne inférieure sur la qualité de l'estimation de la valeur en chaque état, puis à diriger l'exploration préférentiellement vers les états pour lesquels l'écart entre ces bornes est le plus grand, afin de les faire converger peu à peu l'une vers l'autre. Il montre sur un problème jouet que son algorithme, bien que beaucoup plus complexe, est nettement plus utilisable en pratique que les précédents. Nous vérifierons ce point dans le cadre de travaux futurs.

4.8 Apprentissage de la structure dans les FMDPs

S'il apparaît possible de déterminer par apprentissage la grandeur numérique des relations de dépendance conditionnelle entre les variables d'un processus de décision markovien factorisé, toutes les techniques que nous avons examinées jusqu'à présent supposent connue la structure des relations entre ces variables. Un système de classeurs à anticipation tel que MACS, en revanche, est capable d'extraire cette structure par apprentissage, sans disposer d'aucune connaissance a priori sur le problème auquel il est confronté. Nous sommes donc confronté à la situation suivante :

- d'un côté, nous avons MACS, un système capable d'apprendre la structure des relations entre les variables d'un processus de décision markovien factorisé, mais dont la représentation, à savoir une liste de classeurs, ne se prête pas à des algorithmes efficaces de planification. En outre, la gestion du compromis entre exploration et exploitation dans MACS n'est qu'à moitié satisfaisante ;
- de l'autre, nous avons un ensemble d'algorithmes efficaces reposant sur des fondements théoriques solides et s'appliquant sur des représentations arborescentes, mais ces algorithmes présupposent que la structure des relations entre les variables soient données a priori sous forme d'arbres.

Une voie de recherche qui s'impose consiste à intégrer les capacités d'apprentissage de la structure des relations entre les variables fournies par MACS au sein d'une architecture qui s'appuie sur les algorithmes efficaces de planification précédemment décrits. Ces algorithmes reposant exclusivement sur des représentations arborescentes, deux solutions s'offrent à nous :

- soit utiliser les capacités de construction d'un modèle des transitions de MACS en l'état, puis introduire un algorithme *ad hoc* pour reformuler les classeurs obtenus sous forme d'arbres ;
- soit utiliser un équivalent de ces capacités d'apprentissage qui opère directement sur des arbres.

La seconde solution nous est apparue comme indiscutablement plus efficace. Un survol de la littérature sur l'apprentissage d'arbres de décision nous a permis, avec les conseils avisés de V. Corruble, d'identifier l'algorithme ITI [50] comme répondant exactement à nos besoins.

[48] *heuristic sampling*
[49] [Peret & Garcia, 2004, Kearns et al., 2002]
[50] [Utgoff et al., 1997]

ITI est un algorithme de construction d'arbres de décision qui possède la particularité d'être incrémental. Cela signifie qu'il met à jour l'arbre de décision à chaque fois qu'il reçoit une nouvelle entrée, au lieu de construire l'arbre d'un coup à partir de l'ensemble des entrées, comme c'est le cas de la plupart des algorithmes classiques associés aux arbres de décision. Le prix à payer pour cette propriété d'incrémentalité est le fait que, lorsque les premières données reçues ont conduit à une certaine organisation de l'arbre et les données suivantes remettent en question cette organisation, il faut appliquer un opérateur de réorganisation de l'arbre qui peut être coûteux. On voit au passage que l'avantage d'une représentation « à plat » de type « liste de classeurs » est de ne jamais impliquer ce type de réorganisation, mais cet avantage est amoindri par le fait que l'accès à toute information est ralenti par la nécessité de parcourir l'ensemble des classeurs. Nous présentons l'algorithme ci-dessous.

```
ajouter_exemple (nœud, exemple)

si nœud := NULL alors
| convertir nœud en une feuille vide
fin
si nœud est une feuille alors
| enregistrer exemple dans la feuille
| si nœud devrait être transformé en nœud de décision alors
| | transformer nœud en nœud de décision
| | construire les informations de nœud
| | marquer nœud comme « stale »
| | pour chaque exemple j enregistrer dans nœud faire
| | | si j vérifie le test alors
| | | | ajouter_exemple(nœud->fils_gauche, j)
| | | sinon
| | | | ajouter_exemple(nœud->fils_droit, j)
| | | fin
| | fin
| fin
sinon
| mettre informations de nœud à jour
| marquer nœud « fresh »
| si exemple vérifie test alors
| | ajouter_exemple(nœud->fils_gauche, exemple)
| sinon
| | ajouter_exemple(nœud->fils_droit, exemple)
| fin
fin
```

Algorithme 1 : **Procédure d'apprentissage de l'algorithme ITI**

La prise en compte d'un nouvel exemple se fait de la façon suivante. L'algorithme commence par faire « descendre »[51] l'exemple dans l'arbre courant. À chaque étape de la descente, les statistiques conservées au nœud sont mises à jour. Une fois que l'exemple arrive à une feuille, l'algorithme vérifie s'il est bien classé. Si c'est le cas, les statistiques de la feuille sont mises à jour et on passe à l'exemple suivant. Si ce n'est pas le cas, l'algorithme tente de

[51] Si l'exemple vérifie le test du nœud, il est propagé dans le fils gauche du nœud et sinon dans le fils droit.

transformer la feuille en nœud de décision pour que tous les exemples soient classés correctement. L'algorithme ne pourra pas créer de nouveau nœud de décision si l'exemple qu'il tente d'incorporer entre en contradiction [52] avec un exemple précédent. Dans ce cas, il crée une feuille impure, c'est-à-dire qui contient des exemples contradictoires.

Après l'intégration d'un nouvel exemple dans un arbre, ITI s'assure que l'arbre qu'il a produit est le plus compact possible. Si ce n'est pas le cas, la restructuration de l'arbre est lancée pour que celui-ci soit toujours le plus simple possible.

Ainsi, ITI construit bien des arbres de décision de façon incrémentale et il possède quelques propriétés intéressantes. La première est que l'arbre construit ne dépend pas de l'ordre des exemples. La seconde est qu'il est capable de créer des feuilles impures, c'est-à-dire contenant un exemple qui a pris plusieurs classes distinctes. Lorsque l'on fait de la classification, on prend la classe majoritaire pour cette feuille. Toutefois, dans le cadre de l'apprentissage d'un modèle de transition, cette capacité à produire des feuilles impures est intéressante car elle permet d'apprendre un modèle non déterministe. Un modèle de transition est non déterministe lorsqu'un même état de départ s et une même action a peuvent mener à plusieurs état résultants s_i, la probabilité de passer à s_i étant p_i. Ces différents états seront enregistrés dans une seule feuille par ITI. Par ailleurs, pour obtenir p_i, il suffit de compter le nombre d'exemples correspondants à s_i et de le diviser par le nombre total d'exemples enregistrés à la feuille.

La nouvelle architecture que nous avons construite dans le cadre du stage de DEA de C. Lutz [53] a donc l'ensemble des propriétés exhibées par MACS, mais elle repose sur ITI pour l'apprentissage de la structure, sur Factored-R_{max} pour la gestion du compromis entre exploration et exploitation et sur SPI pour le composant de planification.

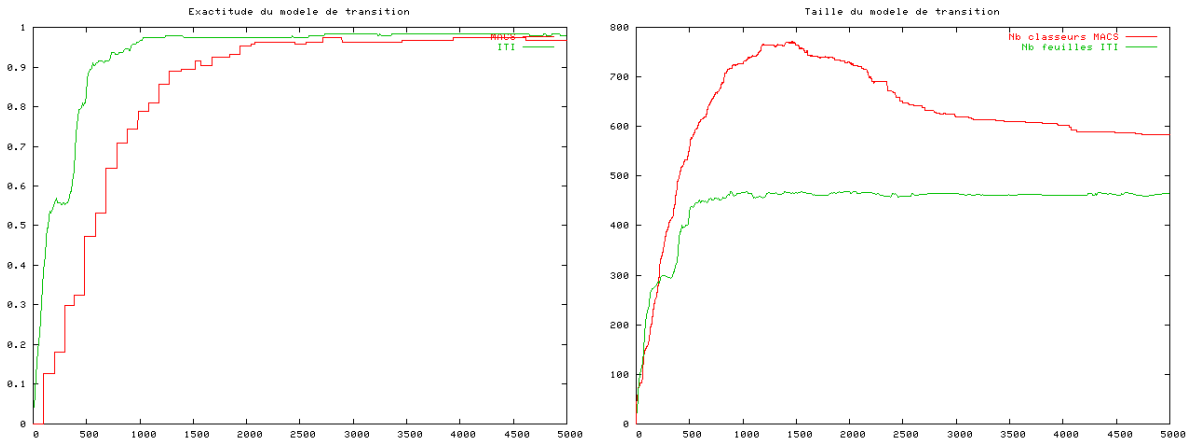


FIG. 4.5 – **La partie gauche de la figure présente l'évolution de l'exactitude des modèles des transitions construits respectivement par ITI et par MACS au cours de 5000 pas de temps successifs sur le labyrinthe de Sutton (la courbe de ITI est au-dessus). La partie droite représente l'évolution du nombre de classeurs engendrés par MACS et du nombre de feuilles dans les arbres construits par ITI dans les mêmes conditions expérimentales (la courbe de ITI est en dessous).**

Comme il apparaît sur la figure 4.5, des travaux encore très préliminaires menés sur le labyrinthe de R. Sutton décrit sur la figure 3.8, page 74, ont permis de montrer que ITI apprenait un modèle des transitions sensiblement plus rapidement que MACS et que le modèle qu'il obtient est sensiblement plus compact. Ces résultats ont été obtenus alors que le modèle construit par ITI est fondé sur les perceptions d'un agent qui se déplace aléatoirement, tandis que le modèle construit par MACS est fondé sur les perceptions d'un agent dirigé par le mécanisme d'exploration

[52] Des exemples sont en contradiction s'ils ont la même valeur pour chaque variable mais sont de classes différentes.
[53] [Lutz, 2004]

active de MACS. Ces résultats sont donc très encourageants.

Cependant, des essais préliminaires pour appliquer SPI aux arbres construits par ITI ont été plus décevants. En effet, faute d'un mécanisme de filtrage des situations qui n'ont jamais été rencontrées, SPI manipule de nombreux états correspondants à des situations qui n'existent pas et ses performances s'effondrent sur des problèmes tels que le labyrinthe de R. Sutton. Une solution à ce problème est en cours de développement.

4.9 Travaux futurs et conclusion

La voie de recherche que nous venons de présenter est destinée à fournir un outil d'apprentissage puissant et efficace, mais limité à l'apprentissage du comportement optimal par un agent dans un environnement markovien.

Il se trouve que les travaux que nous avons décrits, en particulier ceux de C. Boutilier et ses collègues ainsi que ceux de C. Guestrin, font à l'heure actuelle l'objet d'extensions en direction du cadre multi-agents [54] qui nous intéresse particulièrement compte tenu de nos objectifs de recherche.

L'outil d'apprentissage que nous venons de présenter est donc destiné à nous servir de socle sur lequel nous nous appuierons pour intégrer ces nouveaux développements en direction du cadre multi-agents, toujours en préservant dans la mesure du possible la spécificité de ne pas avoir à fournir a priori la connaissance de la structure du problème. Nous décrirons nos premiers efforts dans cette direction au chapitre 9.

En attendant, nous allons montrer au chapitre suivant que le type de comportement des agents que l'on est capable de produire à partir du modèle que nous venons de présenter est massivement restreint par l'absence de variables internes à leur organe de décision, ce qui les empêchent de tenir compte de leur passé pour prendre de bonnes décisions.

[54] [Guestrin et al., 2001, Guestrin et al., 2002a, Guestrin et al., 2003a]

Chapitre 5

Observabilité partielle

« Tu as perdu le mot de passe root de ton rasoir ? »

G. Bayard, croisant B. Girard velu au matin, le 08/01/2002

La plupart des travaux décrits jusqu'ici s'appliquent au cadre formel des processus de décision markoviens. Nous avons indiqué que ce cadre est intrinsèquement restreint par l'hypothèse de Markov, qui stipule que chaque agent dispose à tout instant dans ses perceptions immédiates d'une information suffisante pour choisir une action optimale. La plupart des algorithmes de décision présentés jusqu'ici ne disposent d'une preuve formelle de convergence que si cette hypothèse est vérifiée.

Dans ce chapitre, nous allons examiner les techniques qui s'appliquent dans le cas – malheureusement très fréquent dans la pratique – où l'hypothèse de Markov n'est pas vérifiée. Nous commençons par présenter le cadre général des processus de décision markoviens partiellement observables. Nous explicitons la différence entre la notion de modèle interne et celle d'état interne, puis nous énumérons différents types de problèmes dans lesquels l'observabilité partielle se manifeste. Nous nous focalisons ensuite sur le traitement du cas particulier de la perception ambiguë. Nous faisons le tour des travaux réalisés dans le cadre des systèmes de classeurs pour traiter ces problèmes, puis nous présentons ATNoSFERES, un système mis au point par S. Landau pour traiter la même classe de problèmes.

Nous revenons à la fin du chapitre sur un certain nombre de limitations inhérentes à la classe des problèmes visés ici et aux algorithmes évolutionnistes sur lesquels repose ATNoSFERES.

5.1 Les processus de décision markoviens partiellement observables

Un processus de décision markovien partiellement observable ou POMDP [1], se compose d'un processus de décision markovien classique comprenant les éléments $< S, A, T, R >$ décrits à la section 2.2.2, page 32, auquel s'ajoute un ensemble discret Ω de symboles ω et une *fonction d'observation* $O : S \rightarrow \Pi(\Omega)$.

La fonction d'observation sert à prendre en compte le fait que l'agent ne sait pas dans quel état il se trouve, mais dispose de capteurs qui permettent de le renseigner au moins partiellement. Dans le cadre général, un symbole $\omega \in \Omega$ correspond à ce que l'agent perçoit lorsqu'il se trouve dans l'état $s \in S$. Cela correspond à ce que ses capteurs lui indiquent sur son environnement et sur lui-même. La fonction d'observation O a pour objet de définir pour tout état s dans lequel se trouve l'agent la distribution de probabilité de recevoir les différents symboles

[1] *Partially Observable Markov Decision Process*

$\omega \in \Omega$. Cette fonction d'observation permet donc de spécifier la probabilité des différentes perceptions de l'agent en fonction de son état réel, auquel il n'a pas accès.

Dans le cas général, les symboles observés par l'agent peuvent appartenir à un alphabet totalement différent de l'alphabet selon lequel est défini son état, ce qui ouvre la porte à la possibilité d'inclure un processus de traduction au sein de la fonction d'observation. Cependant, dans le cadre des processus de décision markoviens que l'on rencontre en modélisation informatique, l'état s de l'agent est généralement assimilé à un vecteur d'observables et la fonction d'observation O est une projection qui transforme un élément de S en un élément de Ω en ne retenant que les observables accessibles à l'agent et en éliminant les autres.

La difficulté de l'observation partielle provient de ce que l'agent peut recevoir le même symbole $\omega \in \Omega$ dans deux états différents. On parle de perception ambiguë pour décrire les situations dans lesquelles l'agent peut recevoir la même observation dans plusieurs états distincts.

Dans le cas où la fonction d'observation est une projection d'un vecteur d'observables, cela signifie que les traits qui permettent de distinguer les deux états sous-jacents font justement partie de ceux qui ne sont pas accessibles à l'agent. Un exemple simple permet de bien comprendre ce qui est en jeu dans le contexte de perceptions ambiguës.

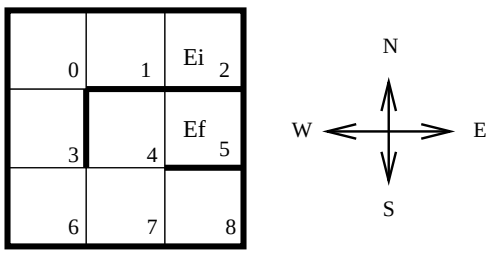


FIG. 5.1 – **Un problème élémentaire à perception ambiguë. L'agent part de E_i et doit se rendre en E_f. Sur son chemin, il perçoit la même chose dans la case 0 et la case 4, alors qu'il doit choisir une action différente dans ces deux cases.**

Considérons le problème jouet de la figure 5.1. L'agent doit se rendre de E_i à E_f.

À chaque instant, l'agent a le choix entre quatre actions : aller au nord (N), à l'est (E), au sud (S) ou à l'ouest (W). Il peut très bien entreprendre de se diriger vers un mur et s'y cogner, auquel cas il reste sur place.

Il ne peut observer que la présence ou l'absence de murs autour de la case dans laquelle il se situe. Il dispose de quatre observables indiquant, pour chaque direction, s'il voit un mur ou non dans cette direction. Chacune de ces observables vaut V quand il y a un mur, F sinon. Les quadruplets formés par les quatre observables prises dans l'ordre NESW constituent l'ensemble des observations possibles. Ainsi, dans la case 0 de la figure 5.1, l'agent observe $VFFV$, dans la case 1 il observe $VFVF$ et dans la case 4 il observe $VFFV$, comme dans la case 0.

Lorsqu'il observe $VFFV$, l'agent peut en fait être dans la case numérotée 0 ou celle numérotée 4. Cette dernière information, le numéro de la case, lui est inaccessible alors que c'est elle qui définit son état au sens des processus de décision markoviens.

Du point de vue de l'agent qui exécute l'action, la présence d'une information cachée se manifeste par le fait que, dans ce qu'il considère à partir de son observation comme le même état, la même action peut le conduire à faire deux observations différentes au pas de temps suivant.

Ainsi, dans l'exemple du labyrinthe, lorsque l'agent observe $VFFV$ et qu'il se déplace vers le sud, il observe parfois $FVFV$ – lorsqu'il était en 0 et descend vers 3 – et parfois $FFVF$ – lorsqu'il était en 4 et descend vers 7.

Il faut bien avoir à l'esprit le fait qu'être non markovien n'est pas une propriété intrinsèque d'un environnement, c'est une propriété qui dépend à la fois de l'environnement et des organes sensoriels de l'agent plongé dans cet

environnement. Par exemple, dans le labyrinthe ci-dessus, si l'on considère que l'agent ne perçoit pas la présence de quatre murs mais de cinq, le cinquième étant l'un des murs latéraux d'une case qui se situe au sud de sa position, alors la case 0 et la case 4 ne sont plus ambiguës pour l'agent. En théorie, il est donc possible de rendre markovien un problème qui ne l'était pas en ajoutant des capteurs à l'agent. Malheureusement, cette solution n'est pas toujours réalisable en pratique.

5.1.1 État interne et modèle interne

Il est important de lever ici une confusion possible entre la notion d'état interne et celle de modèle interne.

Les deux notions semblent à première vue extrêmement proches, dans la mesure où état interne et modèle interne permettent tous deux à l'agent d'accéder à des informations relatives à son passé. Dans les deux cas, le processus implique une mémorisation.

À la section 3.3, page 59, nous avons défini un *modèle interne* comme consistant en un ensemble de relations de la forme $(s(t), a(t), s'(t+1))$, où les $s(t)$ désignent des états et les $a(t)$ désignent des actions. Cette définition formelle vaut pour le cas où il n'y a pas de généralisation dans la représentation du problème. Dans un tel cas, un modèle interne consiste en l'inscription en mémoire de l'ensemble des expériences effectuées au cours du temps par l'agent, indépendamment du contexte temporel dans lequel ont été effectuées ces expériences.

Dans le cas où l'agent dispose d'une capacité de généralisation, la mise en œuvre d'un modèle interne résulte cette fois de la mémorisation par l'agent des régularités observées au sein de son environnement. Ces régularités sont exprimées en faisant abstraction du contexte temporel dans lequel l'agent les a observées. Puisqu'il s'agit de régularités, l'agent doit s'attendre à ce que la même transition se produise s'il effectue la même action dans le même état, à n'importe quel autre instant.

La différence avec la notion d'état interne est alors manifeste. Alors que le modèle interne est détaché de tout contexte historique spécifique, l'usage de l'état interne consiste pour l'agent à enregistrer des états discriminants par lesquels il est passé dans son histoire récente, afin de lever une éventuelle ambiguïté sur l'état dans lequel il se trouve actuellement quand les informations sensorielles immédiatement disponibles n'y suffisent pas.

À ce titre, posséder un état interne ou un modèle interne sont deux moyens distincts pour l'agent de mieux choisir son action présente en faisant appel à une information qui n'est pas immédiatement accessible dans son environnement.

Pour construire les modèles internes de la forme $(s(t), a(t), s'(t+1))$, un agent doit nécessairement posséder une mémoire des observables qu'il a reçues – ou de l'état par lequel il est passé – au pas de temps précédent. De même, pour stocker le modèle interne qu'il construit, l'agent doit nécessairement disposer d'une mémoire.

Dans un cas comme dans l'autre, cette mémoire sert à faire transiter l'information acquise lors de pas de temps précédents à destination des processus qui vont s'exercer au cours du pas de temps qui définit le présent.

Toutefois, le modèle interne permet à l'agent de déterminer, sur la base de sa seule situation présente, quelles sont les situations futures auxquelles il peut être confronté, alors qu'il ne lui permet en aucun cas de déterminer plus exactement quelle est sa situation présente. À l'inverse, l'état interne permet à l'agent de mieux tirer parti de son passé récent pour lever une éventuelle ambiguïté sur sa situation présente. Par contre, à moins de lever effectivement une telle ambiguïté, l'état interne ne lui donne aucune visibilité supplémentaire sur les conséquences de ses prochaines actions.

Il faut donc être clair sur le fait qu'un agent qui construit un modèle interne de la forme $(s(t), a(t), s'(t+1))$ et fait usage de ce modèle interne pour déterminer son action n'est pas capable pour autant de résoudre un problème dans lequel l'hypothèse de Markov n'est pas vérifiée, s'il n'utilise pas la mémoire de son état ou de ses observables du pas de temps précédent pour choisir son action. En effet, s'il se trouve à un pas de temps donné dans une situation

ambiguë pour ses capteurs, alors son modèle interne ne lui fournit aucune information supplémentaire pour lever cette ambiguïté.

Nous avons donc fait apparaître que, s'il existe une définition commune des informations mémorisées par l'agent, il importe de bien distinguer les types d'informations en fonction de l'usage qui en est fait au sein de l'architecture de l'agent, car cet usage détermine le type de capacités supplémentaires dont ils dotent l'agent.

Ainsi, il faut distinguer :

- la mémoire à court terme, qui sert à stocker des informations sur le ou les pas de temps précédents en vue de construire un modèle interne ;
- la mémoire à long terme, qui sert à stocker un modèle interne décrivant les régularités dont peut se servir un agent pour mieux choisir son action ;
- la mémoire pour l'action, qui apporte de l'information supplémentaire sur le passé récent de l'agent pour choisir l'action quand l'information présente n'y suffit pas.

5.1.2 Types de problèmes non markoviens

Nous avons dit que les problèmes de perception ambiguë que nous avons présentés à la section précédente pouvaient en général être résolus en dotant l'agent de capteurs supplémentaires destinés à lui fournir une information qui lève l'ambiguïté. Cette stratégie ne s'applique pas en général à d'autres classes de problèmes non markoviens dans lesquelles l'information cachée n'est pas de nature à être perçue. D'une façon générale il faut en effet distinguer au moins quatre classes de problèmes dans lesquels l'hypothèse de Markov n'est pas vérifiée :

- les cas de perception ambiguë, que nous avons déjà présentés ;
- les cas où l'agent est doté de variables internes qui évoluent au cours du temps ;
- les cas où l'environnement de l'agent évolue au cours du temps ;
- les cas où les sources de récompense évoluent au cours du temps.

Il faut bien entendu considérer aussi les cas qui combinent plusieurs de ces difficultés et, en particulier, les cas où l'évolution de la récompenses au cours du temps est liée à l'évolution de l'environnement. La seconde classe, dans laquelle l'agent est doté de variables internes, se ramène à la troisième en considérant que ces variables sont externes au processus de décision de l'agent et font donc partie de son environnement.

Dans la troisième et la quatrième classes, on parle alors d'une fonction de transition ou d'une fonction de récompense non stationnaire. La plupart des applications multi-agents peuvent être considérées comme appartenant à l'une ou l'autre de ces classes. Dans l'autre sens, on peut ramener tout problème doté d'un environnement non stationnaire à un problème multi-agent, en considérant que ce qui change au sein de l'environnement est le fruit de l'action d'un agent. Par contre, il ne suffit pas qu'un problème fasse intervenir plusieurs agents pour que l'hypothèse de Markov n'y soit plus vérifiée. En effet, si tous les agents réalisent toujours la même action pour le même état de l'environnement, alors l'état suivant du système peut être entièrement déterminé en fonction de l'état précédent et de l'action de l'agent auquel on s'intéresse, si bien que l'hypothèse de Markov est toujours vérifiée.

En revanche, si les agents ont des états internes, et si leur action dépend de ces états internes, ils sont susceptibles de réaliser différentes actions dans la même situation perçue par l'agent auquel on s'intéresse, si bien que l'état suivant n'est plus fonction uniquement de cet état perçu et de l'action de l'agent. L'hypothèse de Markov n'est plus vérifiée et les états internes des autres agents constituent une information cachée par l'agent auquel on s'intéresse.

La seule solution pour l'agent, s'il veut être capable de choisir une action optimale en toute situation, consiste à se doter lui-même d'états internes dont la dynamique reflète exactement celle des états cachés auxquels il n'a pas accès. Si les différentes classes de problèmes énumérés ci-dessus se manifestent de la même façon pour l'agent auquel on s'intéresse – la situation dans laquelle il se trouve au pas de temps suivant ne dépend pas seulement de

l'état qu'il perçoit et de son action – nous verrons à la fin de ce chapitre que différentes techniques de résolution, qui s'appliquent avec succès dans certaines classes, sont moins efficaces pour d'autres. D'une façon générale, les applications multi-agents dans lesquelles la récompense dépend du comportement des autres agents peuvent être considérées comme les plus difficiles. Nous verrons au chapitre 9 que, malheureusement, les applications que nous visons ont toutes les chances d'appartenir à cette classe difficile. En attendant, nous allons commencer par examiner des techniques conçues pour traiter le cas des perceptions ambiguës.

5.1.3 Résolution classique

La résolution classique d'un POMDP [2] consiste à supposer que l'agent ne sait pas dans quel état il est [3] puis à introduire l'état dans lequel l'agent « croit être » [4] Plus précisément, l'agent est doté d'une distribution de probabilité reflétant le degré de plausibilité pour lui d'être dans tel ou tel état. Une distribution uniforme correspond à l'incertitude la plus grande et une distribution valant 1 pour un seul état et 0 partout ailleurs correspond à la certitude – éventuellement infondée – d'être dans l'état en question.

En théorie, cette représentation du problème permet de trouver le comportement optimal pour l'agent compte tenu de son incertitude quant à l'état dans lequel il se trouve effectivement. En effet, il existe des algorithmes qui permettent à l'agent, au fil des observations qu'il reçoit de son environnement, d'affiner sa croyance jusqu'à atteindre, à la limite, une certitude absolue et fondée sur l'état dans lequel il est vraiment. De tels algorithmes impliquent toutefois, comme c'est le cas pour les processus de décision markoviens sans information cachée, que l'agent réalise un nombre d'itérations infini pour atteindre un tel résultat. Néanmoins, il est possible de calculer le nombre d'itérations qu'il faut réaliser pour que l'erreur sur la distribution de probabilité correspondant au *belief state* soit bornée par une constante ϵ [5].

Malgré ce résultat théorique, la difficulté persistante provient de ce que la complexité de ces algorithmes est exponentielle en fonction de la taille du problème, si bien qu'avec des algorithmes issus de ce cadre mathématique bien formalisé, on ne peut obtenir une erreur ϵ raisonnable que sur des problèmes de très petite dimension.

En outre, ces algorithmes classiques résolvent un problème de planification et non pas d'apprentissage. Ils supposent connues les tables de probabilités de transition entre les états et les sources de récompense, ce qui nous écarte du cas de l'apprentissage, sur lequel nous nous focalisons.

5.2 Solutions heuristiques aux problèmes non markoviens

Etant donné le caractère inopérant de la résolution classique des POMDPs, les solutions utilisées en pratique reposent sur diverses heuristiques qui permettent de dépasser les limites des solutions dépourvues de variables internes sans toutefois offrir de garanties d'optimalité dans le cadre non markovien.

Dans les sections qui suivent, nous allons recenser les principales approches utilisées, en particulier dans les travaux qui s'appuient sur des systèmes de classeurs.

[2] [Kaelbling et al., 1998]

[3] On suppose tout de même qu'il sait de combien d'états est constitué le processus de décision markovien sous-jacent à ses observations, ce qui constitue une hypothèse de modélisation largement inapplicable en pratique.

[4] Cet état supposé est appelé « *belief state* ».

[5] [Kaelbling et al., 1998]

Gestion explicite d'un état interne

La première solution pour résoudre des problèmes non markoviens consiste à ajouter un mécanisme de gestion explicite d'un état interne. L'état interne fournit une information supplémentaire sur le passé de l'agent qui vient s'ajouter à ses perceptions pour lui permettre de prendre une meilleure décision.

Cette approche était utilisée d'emblée à l'origine des systèmes de classeurs dans les architectures définies par J. Holland [6], sous la forme d'une liste de messages internes. R. Riolo a effectué quelques expériences sur des problèmes non markoviens avec ce type d'architecture, mais avec G. Robertson [7], il fait état de performances insatisfaisantes avec une telle architecture sur les problèmes non markoviens, ce que confirme R. Smith [8].

Une réflexion approfondie sur ce sujet, menée dans le cadre du stage de DEA de J. Villers, montre que disposer d'une liste de messages internes ne permet de traiter des problèmes non markoviens qu'à condition de définir des mécanismes très spécifiques sur la persistance des messages et les interactions entre les différents types de message, ce qui s'avère délicat à mettre au point. En particulier si tous les messages sont supprimés au bout d'un pas de temps comme c'est le cas dans les propositions initiales, alors la liste de messages internes n'apporte rien de plus qu'une liste de registres, dont nous allons voir qu'ils ont donné lieu à des propositions plus simples.

Dans le contexte des systèmes de classeurs simplifiés proposés par S. Wilson, dont nous avons dit qu'ils ont engendré un renouveau de la recherche dans ce domaine, une forme plus simple de gestion explicite d'un état interne a été adoptée d'une part par D. Cliff [9] avec ZCSM et par P.-L. Lanzi [10] avec XCSM et XCSMH.

D. Cliff est le premier à avoir ajouté des registres internes à ZCS peu après la publication de cet algorithme par S. Wilson [11]. Le système conçu selon un principe déjà proposé par S. Wilson en 1989 [12] s'appelle ZCSM, où M signifie « *Mémoire* ».

En appliquant la même idée, P.-L. Lanzi [13] a proposé XCSM en tant qu'extension de XCS avec des registres internes. Nous faisons le choix de plutôt décrire ce dernier, qui a donné lieu à davantage de publications.

XCSM gère une mémoire interne sous la forme d'un registre de plusieurs bits qui représentent l'état interne du système de classeurs. Le registre mémoire permet à un agent dirigé par XCSM d'ajouter une mémoire de son passé à ses perceptions immédiates lorsqu'il est confronté à des perceptions ambiguës.

Pour intégrer ce registre mémoire dans XCS, P.-L. Lanzi transforme les classeurs de telle façon qu'ils incorporent les éléments suivants :

– une partie `[Condition externe]` qui porte sur la situation perçue instantanément ;
– une partie `[Condition interne]` qui porte sur la valeur des bits du registre mémoire ;
– une partie `[Action externe]` qui spécifie l'action à accomplir dans l'environnement ;
– une partie `[Action interne]` qui permet de modifier les valeurs des bits du registre mémoire ;

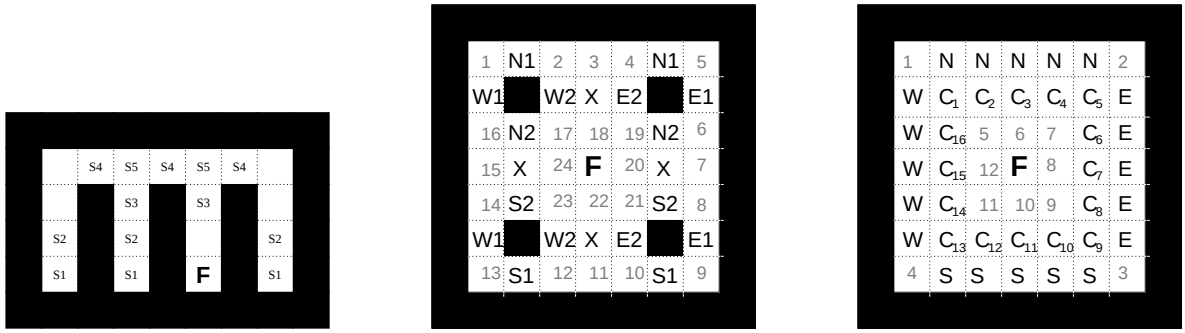
Les parties `[Condition interne]` et `[Action interne]` contiennent autant d'attributs qu'il y a de bits dans le registre mémoire.

Les modifications qui en résultent sur le fonctionnement du système de classeurs sont les suivantes.

Pour qu'un classeur puisse être déclenché, il faut que sa partie `[Condition externe]` soit vérifiée dans l'environnement et que sa partie `[Condition interne]` soit vérifiée par les valeurs du registre mémoire.

[6] [Holland & Reitman, 1978]
[7] [Robertson & Riolo, 1988]
[8] [Smith, 1994]
[9] [Cliff & Ross, 1994]
[10] [Lanzi & Wilson, 2000]
[11] [Wilson, 1994]
[12] [Wilson & Goldberg, 1989]
[13] [Lanzi, 1998]

Par ailleurs, une fois le classeur choisi, son déclenchement engendre à la fois une action externe et la modification des bits du registre mémoire. Concernant ce second point, les bits prennent les valeurs spécifiée dans la partie `[Action interne]` du classeur ou ne changent pas lorsque l'attribut est un symbole *don't change* « # ». Par ailleurs, comme XCS, XCSM tire partie de son mécanisme de généralisation pour construire une représentation compacte, mais il le fait aussi bien dans les parties externes que dans les parties internes. [14]



(a) Environnement **Maze10** (b) Environnement **E1** (c) Environnement **E2**

FIG. 5.2 – **Les environnements Maze10, E1 et E2. F représente le but, à savoir de la nourriture. Les autres marques dans les cases signalent les situations ambiguës, sachant que la même marque pour différentes cases indiquent que les cases sont perçues comme identiques.**

P.-L. Lanzi a présenté les résultats obtenus avec XCSM sur plusieurs environnements simplifiés dont **Maze10** (voir la figure 5.2) est le plus complexe. La performance très médiocre de XCSM sur ce problème non markovien a conduit P.-L. Lanzi à proposer une extension supplémentaire, XCSMH, où le H signifie « *hiérarchique* ».

Les seules différences entre XCSM et XCSMH proviennent de ce que la dynamique d'évolution du registre mémoire est déterministe dans XCSMH même pendant la phase d'exploration, alors qu'elle était stochastique dans XCSM, et que l'état interne ne peut pas changer si les perceptions ne changent pas dans XCSMH alors qu'un tel changement était possible dans XCSM.

Ces modifications mineures se sont traduites par une amélioration significative des résultats [15].

Fenêtre de mémoire

Le second mécanisme de résolution des problèmes non markoviens, qui consiste à gérer une fenêtre de mémoire des perceptions tout juste passées, est un cas particulier de gestion explicite d'un état interne dans lequel l'état interne est restreint à une mémoire immédiate des perceptions passées.

Quelques systèmes utilisent une fenêtre de taille fixe [16], tandis que d'autres utilisent une fenêtre de taille variable. Dans cette seconde catégorie, le travail de A. McCallum [17] apparaît comme l'un des plus représentatifs. Son algorithme, U-TREE, bâtit des arbres de décisions dont les conditions aux noeuds portent soit sur des valeurs de variables au pas de temps courant, soit sur des valeurs de variables aux pas de temps précédents. L'algorithme

[14] donner un exemple, via une figure
[15] [Lanzi & Wilson, 2000]
[16] voir [Lin & Mitchell, 1992] pour une revue
[17] [McCallum, 1995]

remplace ces feuilles de l'arbres par des nouvelles conditions portant sur des valeurs de variables à des pas de temps de plus en plus éloignés, à chaque fois qu'un test statistique montre qu'ajouter de nouvelles distinctions est nécessaire pour prendre la bonne décision. Il apparaît toutefois des situations dans lesquelles le comportement de l'agent se fige dans un cycle inefficace, et la même branche de l'arbre s'étend indéfiniment jusqu'à une profondeur potentiellement infinie.

Mécanismes de chaînage

Le troisième mécanisme, à savoir le chaînage de décisions, peut être vu comme un regard alternatif porté sur le mécanisme précédent.

Il consiste à faire dépendre chaque décision prise à un instant donné des décisions et perceptions réalisées auparavant, de telle façon que la mémoire des situations précédentes permet de lever d'éventuelles ambiguïtés sur la situation courante. Parmi les systèmes de classeurs, ce mécanisme a été utilisé notamment dans ZCCS [18], CXCS [19] et ACS [20]. Les systèmes de classeurs qui utilisent ce mécanisme sont appelés « systèmes de classeurs corporatifs » [21].

L'idée d'un tel mécanisme a été publiée par S. Wilson dans un article qui fait l'état des recherches sur les systèmes de classeurs et souligne le besoin d'un bon compromis entre les architectures de type Michigan et les architectures de type Pittsburgh [22]. Il suggère l'idée de chaîner les classeurs les uns aux autres afin de créer des « corporations » de classeurs souvent déclenchés les uns à la suite des autres, qui seraient évaluées ensemble, constituant ainsi un juste milieu entre le type Michigan où chaque classeur est évalué individuellement et le type Pittsburgh où tous les classeurs sont évalués comme un seul individu. Il semble que S. Wilson n'ait jamais programmé le système correspondant. Par contre, R. Riolo a fait des expériences dans ce sens avec son système CFSC2, sans obtenir de résultats probants [23].

Plus récemment, cependant, A. Tomlinson et L. Bull [24] ont développé ZCCS, un système de classeurs corporatif conforme aux suggestions de S. Wilson et basé sur ZCS [25]. ZCCS relie des classeurs en « séquences comportementales » en fonction de la probabilité pour que deux classeurs soient déclenchés l'un à la suite de l'autre. La longueur des séquences comportementales est limitée a priori par un paramètre indiquant une longueur maximale.

Lorsqu'un classeur inscrit au début d'une séquence comportementale est déclenché, les classeurs suivants dans la séquence sont déclenchés tour à tour sans repasser par un mécanisme de choix parmi des candidats.

Cette solution permet de résoudre les problèmes non markoviens non pas en levant les ambiguïtés, mais en évitant d'avoir à prendre des décisions dans les situations ambiguës. En effet, il suffit de construire une séquence qui se déclenche avant une situation ambiguë de telle façon qu'elle se termine seulement une fois la situation ambiguë franchie pour que le système de classeurs n'ait jamais à se demander quelle action déclencher au moment où il est dans la situation ambiguë.

De même que les classeurs élémentaires, les séquences comportementales sont conservées par l'algorithme génétique quand leur déclenchement permet à l'agent de recevoir davantage de récompenses. Sinon, elles sont éliminées. Cela suffit à ce que les séquences comportementales qui permettent d'éviter les décisions néfastes dans les

[18] [Tomlinson & Bull, 2000b]
[19] [Tomlinson & Bull, 2000a]
[20] [Stolzmann, 1999]
[21] *Corporate Classifier Systems*
[22] [Wilson & Goldberg, 1989]
[23] [Riolo, 1991]
[24] [Tomlinson & Bull, 2000b]
[25] [Wilson, 1994]

situations ambiguës soient conservées, alors que les séquences qui ne procurent aucun avantage à l'agent disparaissent.

Les mêmes auteurs ont ensuite proposé CXCS [26], un système similaire basé cette fois sur XCS. Ils ont testé un grand nombre de variations sur le mécanisme de chaînage, les taux d'exploration et la façon de représenter les séquences comportementales [27], mais toujours sur des problèmes trop simples pour pouvoir donner lieu à des comparaisons sérieuses avec d'autres familles de systèmes.

Un mécanisme de chaînage similaire est utilisé dans ACS [28] pour résoudre des problèmes non markoviens. Le cas de ACS se distingue du précédent d'une part en ce que le mécanisme de chaînage ne s'applique qu'aux environnements déterministes et, d'autre part, en ce que le chaînage n'est appliqué que dans des circonstances spécifiques en prenant en compte les indications fournies par la partie `[Effet]` des classeurs, alors que dans CXCS, l'algorithme tente un grand nombre de chaînages et compte sur l'élimination de ceux qui ne sont pas pertinents pour ne conserver que les chaînages utiles.

Malgré cet usage plus parcimonieux, le mécanisme de chaînage utilisé dans ACS le rend capable de résoudre efficacement des problèmes non markoviens déterministes, comme l'ont montré M. Métivier et C. Lattaud [29] sur les environnements **E1** et **E2** décrits sur la figure 5.2.

Découpage en sous-problèmes

Le quatrième mécanisme consiste à découper un problème non markovien en plusieurs sous-problèmes tous markoviens, de telle façon que les états ambigus se trouvent répartis dans les différents sous-problèmes. Cette solution a fait l'objet des recherches de M. Wiering [30], puis elle a été améliorée par R. Sun [31]. À l'heure actuelle, nous ne connaissons pas de systèmes de classeurs qui reproduisent ce type de solutions, même si l'architecture hiérarchique de J.-Y. Donnart [32] constitue un pas intéressant dans cette direction. Nous reviendrons sur cette classe de solutions au chapitre 8.

Automates à états finis

La dernière solution consiste à construire un automate à états finis dont la structure reflète celle du problème. C'est la solution choisie par E. Hansen [33] ou N. Meuleau [34], lorsque la structure du problème est connue à l'avance. Jusqu'à présent, cette solution n'avait pas été appliquée dans un cadre où il faut apprendre la structure des problèmes. C'est cette innovation que nous avons proposée avec ATNoSFERES, comme nous le verrons à la section 5.3.

[26] [Tomlinson & Bull, 2000a]
[27] [Tomlinson & Bull, 2002]
[28] [Stolzmann, 1999]
[29] [Métivier & Lattaud, 2002]
[30] [Wiering & Schmidhuber, 1997]
[31] [Sun & Peterson, 2000, Sun & Sessions, 2000]
[32] [Donnart, 1998]
[33] [Hansen, 1998]
[34] [Meuleau et al., 1999]

5.3 ATNoSFERES

SFERES est une plate-forme générique de mise au point d'algorithmes génétiques développée par S. Landau et S. Doncieux [35] dans le cadre de leurs travaux de thèse respectifs [36]. Construit sur la base de SFERES, ATNoSFERES [37] est un modèle spécifique dans lequel les individus manipulés par les algorithmes génétiques sont des automates à états finis. Les automates engendrés par ATNoSFERES ont été conçus de telle façon qu'ils peuvent représenter une architecture de contrôle pour des agents confrontés à des problèmes de décision dans lesquels l'hypothèse de Markov n'est pas vérifiée.

Un travail de recherche mené en commun avec S. Landau a consisté à établir la relation entre le formalisme des contrôleurs engendrés par ATNoSFERES et des contrôleurs représentés sous forme de systèmes de classeurs, puis à comparer les performances des deux types de systèmes sur quelques problèmes non markoviens représentatifs.

Dans la suite, nous présentons le fonctionnement général d'ATNoSFERES, nous mettons en évidence le lien avec les systèmes de classeurs, puis nous exposons les résultats des principales comparaisons effectuées. Nous terminons par l'énoncé des conclusions auxquelles nous ont conduit ces comparaisons et l'exposé succinct du programme de recherche qui en découle.

5.3.1 Codages génétiques de structures

La structure engendrée par ATNoSFERES pour commander des agents est un automate à états finis. Avant de décrire le processus de construction qui permet d'engendrer de tels automates, nous commençons par passer en revue les différents types de codage utilisés dans les algorithmes évolutionnistes pour engendrer des structures.

Les codages paramétriques

Le premier type de codage consiste à considérer une structure fixe et à utiliser le génome pour coder la valeur des paramètres associés à la structure. On peut utiliser pour cela les algorithmes évolutionnistes standards tels que les algorithmes génétiques [38] ou les stratégies évolutionnistes [39], qui permettent une optimisation fine des valeurs des paramètres. Ces structures peuvent être par exemple des réseaux de neurones [40]. Le problème de cette approche est qu'elle impose de travailler avec une structure fixe, qu'il faut déterminer a priori alors que la bonne structure n'est pas nécessairement connue.

Les codages directs

Le second type de codage consiste à utiliser des génotypes structurés de telle façon qu'un génome est directement équivalent à une structure. C'est le cas par exemple de la programmation génétique [41] ou de la programmation évolutionniste [42] qui produisent directement des structures arborescentes.

Les difficultés que l'on rencontre avec ce type de codage proviennent des dépendances entre les éléments au sein de la structure, qui imposent de construire les opérateurs de mutation et de croisement en prenant en compte de nombreuses contraintes de cohérence structurelle et sémantique. À leur tour, ces contraintes induisent des biais

[35] [Landau et al., 2002a]
[36] [Landau, 2003, Doncieux, 2003]
[37] [Landau & Picault, 2001, Picault & Landau, 2001]
[38] [Holland, 1975, De Jong, 1975, Goldberg, 1989]
[39] [Schwefel, 1995]
[40] [Yao, 1999]
[41] [Koza, 1992, Montana, 1995]
[42] [Fogel et al., 1966]

dans la façon dont l'espace de recherche sera parcouru par l'algorithme d'exploration associé à la sélection. On est conduit par exemple à borner la profondeur des arbres engendrés par la programmation génétique, ce qui peut avoir des conséquences difficiles à évaluer sur la capacité de l'algorithme de recherche à trouver une bonne solution.

Ces questions commencent à recevoir un début de réponse théorique, avec les travaux de R. Poli et B. Langdon [43], mais beaucoup reste à faire dans cette direction.

Les codages indirects

Le troisième type de codage, dit « indirect » s'affranchit des contraintes des deux précédents en représentant dans le génome un langage spécifiant des opérations de construction des structures visées [44]. Cette approche permet de réaliser une optimisation fine des paramètres d'une structure tout en n'imposant pas de contraintes a priori sur la structure elle-même ni sur les opérateurs qui la manipulent. Les seules contraintes résident dans la cohérence syntaxique du langage qui permet de manipuler les structures.

C'est le cas par exemple des algorithmes de développement de réseaux de neurones qui associent à chaque neurone des instructions de croissance, de duplication, de connexion et de suppression qui font qu'un réseau se construit par interprétation du programme qui lui est associé [45].

La plupart des systèmes qui reposent sur ce type de codage sont soit des règles de réécriture inspirées des « $\mathcal{L}$-*systems* » de Lindenmayer [46], soit des grammaires non contextuelles [47].

La difficulté associée aux règles de réécriture est qu'elles sont difficiles à contrôler, les structures engendrées peuvent être désespérément pauvres ou au contraire croître de façon démesurée. Il est donc indispensable d'associer des mécanismes de régulation à ce type de codages.

Pour ce qui est des grammaires non contextuelles, elles imposent des contraintes fortes sur la syntaxe des génomes. Il est plus facile de faire en sorte que les opérateurs d'évolution génétique (mutation et croisement) respectent ces contraintes que dans le cas des codages directs mais, en contrepartie, l'effet de l'évolution d'un génome sur la structure produite est plus difficile à maîtriser. Par exemple, la mutation d'un opérateur qui s'applique à proximité de la racine d'une structure sera beaucoup plus important que celles d'un opérateur qui s'applique aux feuilles d'un arbre.

La solution retenue dans le cadre d'ATNoSFERES est un codage indirect qui ne repose ni sur les règles de réécriture, ni sur les grammaires non contextuelles, mais sur un mécanisme d'expression génétique à base de piles que nous allons décrire dans la section suivante.

5.3.2 Construction d'une structure de commande par ATNoSFERES

La méthode de construction d'automates retenue dans le cadre d'ATNoSFERES repose sur une pile d'opérateurs de construction.

Le langage codé par les génomes est constitué d'un ensemble de « *tokens* ». Il contient à la fois des tokens qui représentent des opérateurs de construction d'un automate (création de nœuds, création d'arcs entre nœuds, etc.) et des tokens qui représentent des opérateurs de gestion d'une pile.

Une première phase de la construction consiste à traduire le génomes en une suite de tokens puis une seconde phase consiste à appliquer les opérateurs correspondant, ce qui se traduit par l'évolution d'une pile et la construction

[43] [Poli & Langdon, 1998]
[44] [Kodjabachian & Meyer, 1995]
[45] [Gruau, 1994, Luke & Spector, 1996, Kodjabachian & Meyer, 1998]
[46] [Lindenmayer, 1968]
[47] [Chomsky, 1962, Chomsky & Miller, 1963]

progressive d'un automate. Le schéma de principe général est représenté sur la figure 5.3.2. Nous détaillons les deux phases correspondantes dans les deux sous-sections suivante.

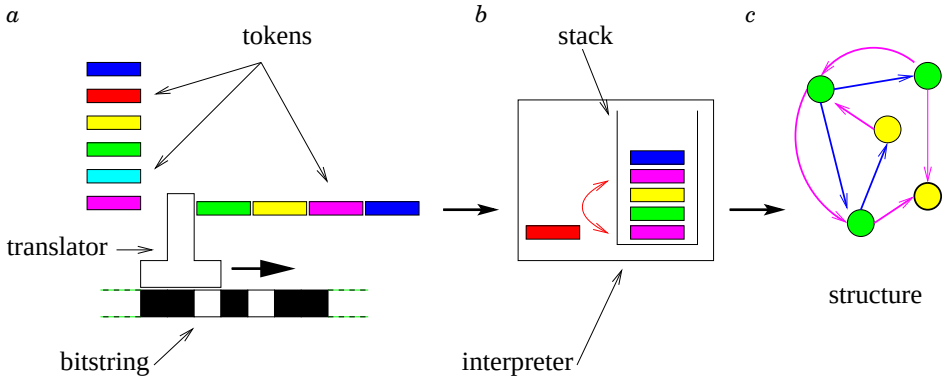


FIG. 5.3 – **Principe de l'expression génétique qui permet de traduire un génotype sous forme de bits en un graphe de comportement. Le génotype est d'abord décodé en tokens (*a*), puis ces tokens sont interprétés en tant qu'instructions (*b*) pour créer des nœuds, des arcs, etc. Enfin, quand tous les tokens ont été interprétés, les conditions et les actions restantes sont ajoutées à la structure et celle-ci est extraite de la pile (*c*).**

Traduction du génome

Le génome est traduit sous la forme d'une séquence de tokens. La traduction s'appuie sur un codage génétique qui est une fonction :

$$\mathcal{G} : \{0,1\}^n \longrightarrow \mathcal{T} \quad (|\mathcal{T}| \leq 2^n) \tag{5.1}$$

où $\mathcal{T}$ est l'ensemble des tokens du langage. Une sous-chaîne binaire de longueur n est appelée un « codon ». À chaque codon correspond un token, comme il apparaît dans la table 5.3.2.

Nous donnons ci-dessous dans la table 5.3.2 la liste des tokens utilisés pour comparer les performances d'AT-NoSFERES à celles de différents systèmes de classeurs [48].

Il y a trois catégories de tokens :

- **les tokens comportementaux** (actions, conditions), qui permettent de spécifier un type de comportement particulier. Ces tokens sont simplement empilés pour être consommés par la suite.
- **les tokens de structure** (*node*, *connect*, etc.), qui réalisent les opération de construction de la structure, en utilisant éventuellement des tokens déjà présents dans la pile. Ces tokens sont indépendants de la spécification des comportements.
- **les tokens de gestion de pile** (*swap*, *dup*, etc.), qui manipulent la pile. Ces tokens sont indépendants à la fois de la structure et de la spécification des comportements.

Etant donné une liste de tokens, il faut aussi décider du code binaire attribué à chaque token. Nous donnons dans la table 5.3.2 le codage utilisé dans les expériences de comparaison précédemment citées. Le code de chaque token étant de longueur fixe, le nombre de tokens disponibles est nécessairement une puissance de 2. Cela induit la possibilité de coder certains tokens plusieurs fois, comme il apparaît dans la table ci-dessous.

[48] [Landau et al., 2002b, Landau & Sigaud, 2004]

token	interprétation
tokens comportementaux	
condition ?	ajoute une condition sur la pile
`action !`	ajoute une action sur la pile
tokens de structure	
node	ajoute un nouveau nœud sur la pile
connect	ajoute un arc entre le premier nœud et le second sur la pile
	s'il s'agit de copies du même nœud, le résultat est un arc ré-entrant
startConnect	crée un arc entre le nœud Start et le premier nœud sur la pile
endConnect	crée un arc entre le premier nœud sur la pile et le nœud End
selfConnect	crée un arc entre le premier nœud sur la pile et lui-même
defaultSelfConnect	crée un arc entre tous les nœuds présents dans la pile et eux-mêmes
tokens de gestion de pile	
nop	token simplement ignoré
swap	permute les deux premiers tokens
dup	empile une copie du premier token de condition ou d'action
del	supprime le premier token de condition ou d'action
dupNode	empile une copie du premier token de nœud
delNode	supprime le premier nœud
popRoll	envoie le premier token à la fin de la pile
pushRoll	met le dernier token au sommet de la pile

TAB. 5.1 – **Le langage de construction d'automates. Tous les opérateurs de type *connect* associent à l'arc les conditions et les actions portés par les tokens entre les nœuds qu'ils connectent, et suppriment les tokens en question de la pile. Dans le cas des arcs connectant un nœud à lui-même, on consomme les conditions et les actions situées en haut de la pile. Nous avons montré que la performance d'ATNoSFERES était dépendante des tokens de structure employés. En particulier, l'ajout du token *selfConnect* qui favorise la connexion de nœuds à eux-mêmes s'avère bénéfique pour la résolution de problèmes de décision non markoviens. La plate-forme expérimentale permet de faire varier très facilement le langage en ajoutant ou supprimant à volonté divers tokens.**

000000	*swap*	000001	*swap*	000010	*swap*	000011	*swap*
000100	*dup*	000101	*dup*	000110	*dupNode*	000111	*dupNode*
001000	*del*	001001	*del*	001010	*delNode*	001011	*delNode*
001100	*popRoll*	001101	*popRoll*	001110	*pushRoll*	001111	*pushRoll*
010000	*node*	010001	*node*	010010	*node*	010011	*node*
010100	*connect*	010101	*connect*	010110	*connect*	010111	*connect*
011000	*selfConnect*	011001	*selfConnect*	011010	*selfConnect*	011011	*startConnect*
011100	*startConnect*	011101	*startConnect*	011110	*endConnect*	011111	*endConnect*
100000	*goNorth !*	100001	*goSouth !*	100010	*goWest !*	100011	*goEast !*
100100	*goNorthEast !*	100101	*goSouthEast !*	100110	*goNorthWest !*	100111	*goSouthWest !*
101000	*freeNorth ?*	101001	*foodNorth ?*	101010	*occupiedNorth ?*	101011	*freeSouth ?*
101100	*foodSouth ?*	101101	*occupiedSouth ?*	101110	*freeWest ?*	101111	*foodWest ?*
110000	*occupiedWest ?*	110001	*freeEast ?*	110010	*foodEast ?*	110011	*occupiedEast ?*
110100	*freeNorthEast ?*	110101	*foodNorthEast ?*	110110	*occupiedNorthEast ?*	110111	*freeSouthEast ?*
111000	*foodSouthEast ?*	111001	*occupiedSouthEast ?*	111010	*freeNorthWest ?*	111011	*foodNorthWest ?*
111100	*occupiedNorthWest ?*	111101	*freeSouthWest ?*	111110	*foodSouthWest ?*	111111	*occupiedSouthWest ?*

TAB. 5.2 – **Code génétique des tokens utilisé dans le cadre des comparaisons avec les systèmes de classeurs. Certains tokens sont codés plusieurs fois.**

Interprétation des tokens

La phase d'interprétation des tokens est réalisée par un interpréteur doté d'une pile. Le principe de l'interprétation consiste à traiter un à un chaque token dans l'ordre d'apparition spécifié par le génome et à appliquer l'opérateur correspondant sur l'état de la pile.

La plupart des opérations s'appliquent sur le sommet de la pile, mais certaines opérations peuvent référencer directement des données ailleurs dans la pile, comme c'est le cas pour les autres langages à base de pile tels que Forth [49] ou PostScript [50].

Les langages à base de pile présentent l'intérêt de reposer sur des grammaires contextuelles et de ne pas souffrir des problèmes de dépendances hiérarchiques caractéristiques des grammaires non contextuelles [51]. Par conséquent, ce choix nous affranchit des contraintes qui pèsent sur l'évolution directe de grammaires [52]. En outre, comme c'est le cas pour les autres langages à base de pile déjà cités, l'interpréteur peut accepter n'importe quelle séquence de tokens et produira toujours une structure correcte, notamment parce qu'il se contente d'ignorer tous les opérateurs qui ne peuvent pas être exécutés dans le contexte courant.

Grâce à ce choix d'un langage à base de pile, le processus de construction est extrêmement robuste aux variations qui affectent le génotype au cours du processus d'évolution. En effet, il n'y a aucune contrainte spécifique sur les opérateurs de manipulation du génome (mutation et croisement) et tous les génomes engendrés peuvent être interprétés et produisent une structure exécutable.

En outre, la structure produite ne dépend que partiellement de l'ordre d'application des tokens et une même structure peut être produite par des génomes éventuellement très différents. Enfin, il apparaît qu'il n'est pas nécessaire d'introduire des mécanismes de régulation externe sur la forme des structures produites, telles que des

[49] [Moore & Leach, 1970]
[50] [Adobe Systems, 1985]
[51] [Landau & Picault, 2002]
[52] [Ryan & O'Neill, 1998]

contraintes de taille, car il semble que des mécanismes de régulation implicites soient déjà à l'œuvre dans la conjonction du processus d'interprétation et de l'évolution génétique [53].

Le processus de construction de la structure proprement dite dépend des tokens spécifiques dédiés à cette construction. Par exemple, l'opérateur *node* crée un nouveau nœud et l'ajoute au sommet de la pile, tandis que l'opérateur *connect* relie par un arc les deux premiers nœuds qu'il trouve à partir du sommet de la pile.

Dans le cas qui nous intéresse, l'arc porte une partie `[Condition]` et une partie `[Action]`, qui sont remplies avec les conditions et les actions présentes dans la pile entre le premier et le second nœud, ces conditions et actions étant « consommées » dans la pile lors de leur attribution à un arc.

Lorsque l'ensemble du génome a été interprété, la structure ainsi engendrée est extraite de la pile et peut être évaluée au travers de son exécution, comme nous le décrivons à la section 5.3.4.

Avant de décrire le processus d'exécution, nous commençons par présenter le processus d'évolution génétique auquel est soumise une population de génomes dans le cadre du modèle ATNoSFERES.

5.3.3 Mécanisme de sélection d'un automate performant

À des fins de comparaison avec les systèmes de classeurs, l'algorithme d'évolution est utilisé pour trouver un contrôleur capable de doter un agent d'un comportement efficace dans un environnement éventuellement non markovien.

Chaque contrôleur est un individu doté d'un génome sous forme d'une chaîne de bits. La *fitness* d'un contrôleur est donnée par la qualité du comportement de l'agent qu'il dirige dans un environnement. Classiquement, les contrôleurs destinés à se reproduire sont sélectionnés en fonction de leur *fitness* pour produire de nouveaux contrôleurs.

Le génome d'un nouvel individu est produit par une opération de croisement classique entre les génomes de deux parents. De plus, on utilise deux stratégies de mutation différentes pour introduire des variations dans les nouveaux génomes :

– des inversions classiques de valeurs de bits ;
– des insertions ou suppressions aléatoires de codons.

Cela permet de modifier les séquences de tokens produites par le mécanisme de traduction, si bien que la structure résultante évolue.

5.3.4 Mécanisme d'exécution d'un automate

Pour évaluer un contrôleur, on l'utilise de la façon suivante (voir la figure 5.4) :

– À l'initialisation de l'agent, l'état courant est le nœud *Start* (noté S).
– À chaque pas de temps, l'état courant franchit un arc :

1. On calcule l'ensemble des arcs éligibles parmi ceux qui partent de l'état courant, sachant qu'un arc est éligible soit lorsque les conditions qu'il porte sont vérifiées par les perceptions de l'agent, soit lorsqu'il ne porte aucune condition.
2. On choisit le premier arc dans cet ensemble (ce choix déterministe donne de meilleurs résultats qu'un choix aléatoire) et l'agent exécute la première action portée par l'arc si elle existe. Si l'ensemble est vide ou si l'arc choisi ne porte pas d'action, l'agent exécute une action au hasard.
3. L'état courant devient l'état de destination de l'arc.

[53] [Landau & Picault, 2001]

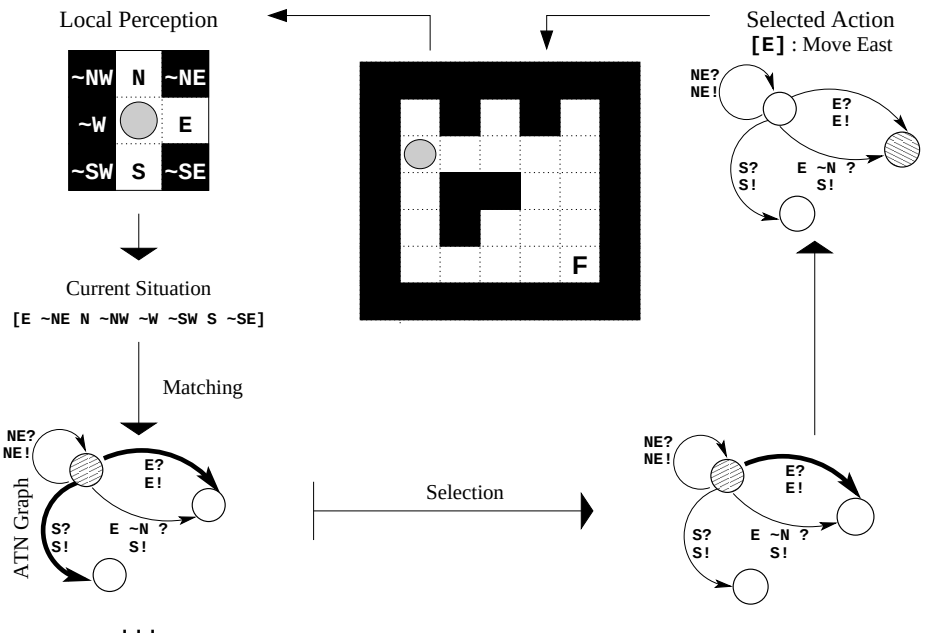


FIG. 5.4 – **Éxécution d'un automate de comportement. L'agent perçoit la présence ou l'absence de murs dans les 8 cases qui l'entourent. Il doit décider vers laquelle de ces 8 cases il se déplace. Dans l'exemple, il perçoit [E ¬NE N ¬NW ¬W ¬SW S ¬SE] (le token E est vrai quand la case à l'est est vide). À partir du nœud courant de son automate, les deux arcs en gras sont éligibles, puisque leur partie [Condition] est vérifiée. L'un de ces arcs est choisi, l'action correspondante est exécutée, et l'état courant de l'automate est mis à jour.**

– Le processus se termine lorsque l'état atteint le nœud *End* (noté E). En général, dans les expériences réalisées pour comparer avec les systèmes de classeurs, un arrêt définitif de l'agent est fortement pénalisé, donc les automates performants ont une structure telle que le nœud *End* n'est jamais atteint.

5.4 Équivalence avec les systèmes de classeurs

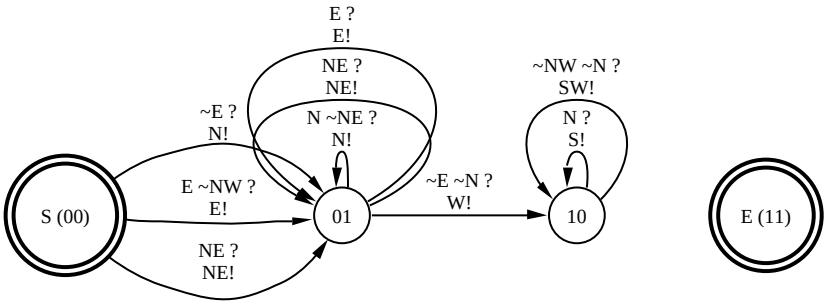


FIG. 5.5 – **Graphe d'un bon automate sur Maze10**

La figure 5.5 et la table 5.3 représentent respectivement un automate de contrôle construit par ATNoSFERES et un système de classeurs avec état interne du type XCSM réalisant un comportement strictement équivalent.

La traduction d'un formalisme vers l'autre est immédiate : à chaque arc correspond un classeur et à chaque nœud du graphe correspond un état interne. Par ailleurs, il est possible d'établir une équivalence du même type pour faire correspondre à un automate un grand nombre de représentations possibles fondées sur le chaînage des classeurs.

EC								IC	EA	IA
E	NE	N	NW	W	SW	S	SE			
1	#	#	#	#	#	#	#	00	N	01
0	#	#	1	#	#	#	#	00	E	01
#	0	#	#	#	#	#	#	00	NE	01
#	1	0	#	#	#	#	#	01	N	##
#	0	#	#	#	#	#	#	01	NE	##
0	#	#	#	#	#	#	#	01	E	##
1	#	1	#	#	#	#	#	01	W	10
#	0	#	#	#	#	#	#	10	S	##
#	#	1	1	#	#	#	#	10	SW	##

TAB. 5.3 – **Une représentation du graphe de la figure 5.5 au format de XCSM. EC : conditions externes, IC : conditions internes, EA : actions externes, IA : actions internes**

5.5 Synthèse des principaux résultats

Les travaux expérimentaux menés avec ATNoSFERES pour comparer aux divers systèmes de classeurs capables de traiter des problèmes non markoviens font apparaître les résultats suivants :

- sur **Maze10**, ATNoSFERES atteint une performance nettement supérieure à celle de XCSM et comparable à celle de XCSMH ;
- par ailleurs, ATNoSFERES est meilleur qu'ACS sur **E2** et moins bon sur **E1**.

Etudier de plus près la structure des meilleurs automates engendrés par ATNoSFERES sur **E1** et **E2** nous a permis de mettre en évidence le fait que, malgré leur allure similaire, ces deux environnements ont des structures sensiblement différentes. En effet, nous avons pu montrer qu'un mécanisme de chaînage ne peut pas produire un contrôleur optimal dans l'environnement **E2**, alors qu'il le peut dans l'environnement **E1**. Nous avons donc fait apparaître des sous-classes des problèmes, dans lesquelles l'agent est confronté à des perceptions ambiguës. Certains mécanismes s'avèrent plus adaptés à la recherche d'un contrôleur optimal dans certaines sous-classes de problèmes que dans d'autres.

Il se dégage de ces comparaisons le fait que la représentation sous forme d'automates produite par ATNoSFERES subsume à la fois les mécanismes de gestion explicite d'un état interne et les mécanismes de chaînage, si bien que le modèle est efficace sur une classe de problème plus vaste que chacun des précédents modèles. Notamment, nous avons montré que le mécanisme de chaînage des classeurs utilisé dans ACS est intrinsèquement incapable de trouver un comportement optimal sur **E2** alors qu'il est suffisant pour engendrer un comportement très efficace sur **E1**.

Ces résultats encourageants nous incitent à persévérer dans la voie de la production d'automates de contrôle pour résoudre des problèmes non markoviens.

Cependant, la comparaison des itérations nécessaires pour obtenir avec ATNoSFERES une performance comparable à celle de ses concurrents nous conduit à revoir en profondeur le mécanisme de recherche des automates performants. En effet, ATNoSFERES s'avère 7.000 fois plus lent qu'ACS pour trouver une solution équivalente sur **E2**, et 182.000 fois plus lent sur **E1**, comme il apparaît sur le tableau 5.5, tiré de nos travaux précédents [54].

envir.	LCS type	perf.	LCS	PR(%)	NG	NT	NT/LCS
Maze10	XCSM	15.1	7,000	100	8.42	45,000	6.42
Maze10	XCSMH	6.12	6,500	7.30	4909	360.10^6	55,400
E1	ACS $_{(BS_{max}=1)}$	4	1,200	30.92	5115	218.10^6	182,000
E2	ACS $_{(BS_{max}=2 \text{ or } 3)}$	6.5	2,000	83.84	835	14.10^6	7,000

TAB. 5.4 – **Comparaison de performances entre ATNoSFERES et divers systèmes de classeurs en termes de nombre de pas de temps nécessaire pour atteindre la performance indiquée dans la colonne « perf. ». PR est le pourcentage d'exécutions d'ATNoSFERES qui font mieux que le système de classeurs correspondant. NG est le nombre moyen de générations nécessaires pour atteindre ladite performance. Pour faire mieux que le système de classeurs correspondant avec 300 individus par génération, il faut réaliser $NE = 300 * NG * 100/PR$ exécutions. Le nombre de lancements élémentaires NT pour un environnement doté de NS positions de départ ($NS = 18$ pour Maze10, 44 pour E1 et 48 pour E2) est $NT = NS * NE$. La variable NT donne le nombre moyen de lancements élémentaires nécessaires pour qu'ATNoSFERES fasse mieux que le système de classeurs correspondant. Enfin, NT/LCS donne une bonne approximation du facteur par lequel le système de classeurs correspondant est plus rapide qu'ATNoSFERES à atteindre sa meilleure performance.**

[54] [Landau & Sigaud, 2004]

5.6 Conclusion

Nous avons montré au cours de ce chapitre que le formalisme manipulé par ATNoSFERES est à la fois riche et puissant, si bien que l'algorithme génétique sous-jacent est capable de trouver des solutions efficaces à une classe de problèmes plus vastes que les systèmes concurrents sur lesquels nous nous sommes penché. Ceci dit, ATNoSFERES s'avère aussi très sensiblement plus lent que ces mêmes concurrents.

Cet écart considérable s'explique très simplement par le fait que la recherche des automates performants s'effectue à l'aide d'un algorithme génétique aveugle qui repose uniquement sur des modifications aléatoires guidées par la seule fonction de *fitness*, alors que dans les systèmes de classeurs, les mécanismes d'apprentissage par renforcement permettent de guider la recherche de façon beaucoup plus fine et donc de trouver des solutions satisfaisantes beaucoup plus efficacement.

Nous présenterons au chapitre 8 l'évolution majeure du modèle qui nous a permis d'introduire des mécanismes d'apprentissage par renforcement dans une architecture dérivée du modèle initial d'ATNoSFERES.

Par ailleurs, il faut avoir conscience du fait que la classe des problèmes non markoviens que nous avons abordée dans ce chapitre se limite ici aux problèmes de perception ambiguë, qui ne posent pas les mêmes difficultés que d'autres sous-classes auxquelles nous allons nous intéresser dans la suite et, en particulier, la classe des problèmes multi-agents. Pour s'en convaincre, il suffit de constater que le mécanisme de chaînage, qui permet de franchir une série de situations ambiguës jusqu'à retrouver une situation sans ambiguïté, est totalement inopérante face à la plupart des problèmes multi-agents. En effet, la séquence des actions réalisées par les autres agents ne fait qu'accroître l'incertitude sur ce que seront les situations futures au fur et à mesure que l'on regarde plus loin dans le futur, donc il y a très peu de chances pour qu'une séquence d'actions permette de retourner à coup sûr sur une situation dans laquelle l'hypothèse de Markov est vérifiée. En conséquence, il sera nécessaire de nous tourner vers d'autres mécanismes plus spécifiquement consacrés à la gestion des problèmes multi-agents. Nous allons faire une première incursion dans ce domaine au chapitre suivant, puis nous y reviendrons de façon beaucoup plus centrale au chapitre (9).

Chapitre 6

Champs de forces pour la coordination spatiale

> « P..., comment j'le kiffe mon code, maintenant ! Non mais tu sais ce que ça fait... Des fois, t'es trop content rien que de le lire ! »
>
> F. Flacher, le 28/01/2002
>
> « Je suis entré dans la période où j'ai fini de m'enthousiasmer pour ce que je fais »
>
> F. Flacher, le 14/05/2003

Les mécanismes adaptatifs étudiés jusqu'ici étaient conçus pour permettre à un agent isolé de résoudre des problèmes dont les états et les actions sont représentés de manière discrète. Dans ce chapitre, consacré aux travaux de thèse de F. Flacher, nous abordons deux nouvelles problématiques à la fois, puisque nous allons nous intéresser à des environnements continus et à la coordination du comportement de plusieurs agents. Considérés de manière générale et abstraite, le cas continu et le cas multi-agents sont connus pour poser des problèmes très difficiles aux chercheurs qui s'efforcent de mettre au point des mécanismes adaptatifs.

Plutôt que d'affronter d'emblée ces difficultés dans un cadre général, nous allons nous restreindre à un cas particulier et concret, à savoir la mise au point de méthodes de navigation permettant la coordination spatiale d'un ensemble d'agents. En outre, pour traiter ce problème particulier, nous allons nous écarter pour l'instant du cadre formel des processus de décision markoviens et des techniques d'apprentissage, pour nous tourner vers des méthodes à base de champs de force dont le caractère adaptatif est assuré par des algorithmes évolutionnistes. Nous reviendrons au cadre markovien appliqué en environnement continu et à des problèmes multi-agents au chapitre 9 de ce mémoire.

Nous commençons par situer plus précisément la problématique de la coordination spatiale, par justifier notre choix d'une approche fondée sur des interactions de type attraction/répulsion, et par argumenter notre recours à des algorithmes évolutionnistes pour ajuster précisément ces attractions et ces répulsions. Puis nous présentons le formalisme proposé par F. Flacher et les algorithmes évolutionnistes auxquels il a recours.

Nous présentons ensuite quelques résultats obtenus avec la plate-forme qui en découle, GACS. En particulier, nous comparons les performances obtenues avec GACS aux résultats publiés par T. Balch et M. Hybinette dans le

cadre d'une démarche similaire sur des problèmes de déplacement en patrouille. Nous montrons que notre méthode permet d'obtenir des résultats équivalents ou meilleurs alors qu'elle est plus générale.

Nous concluons sur la nécessité de renforcer les aspects méthodologiques de la démarche et sur la nécessité d'avoir recours à une autre classe d'algorithmes adaptatifs susceptibles de converger plus rapidement.

6.1 Introduction

Lorsque nous nous donnons pour objectif scientifique la mise au point de comportements adaptatifs pour des agents immergés dans des environnements de simulation de grande envergure, la difficulté centrale à laquelle nous nous attaquons provient de ce que ces simulations sont souvent peuplées de nombreux agents, véhicules simulés ou créatures artificielles, qui remettent en question l'hypothèse de stationnarité sur laquelle s'appuient les techniques d'apprentissage par renforcement.

Si l'on aborde cette difficulté du point de vue d'un agent isolé qui apprend à se comporter au mieux dans une telle simulation, cette caractéristique ajoute des contraintes supplémentaires : il faut réagir de façon adaptée au comportement des autres agents, qui peut être plus ou moins prévisible, si bien que, comme nous l'avons souligné au chapitre 5, se faire un modèle de la dynamique de l'environnement peut être très difficile.

Par opposition, on peut aussi aborder cette difficulté du point de vue d'un ensemble d'agents qui doivent agir ensemble au sein d'un même environnement dynamique. On s'attaque alors à une autre classe de problèmes, dans lesquels il faut trouver des mécanismes de coordination entre les agents. Le problème que nous abordons dans ce chapitre est celui, minimal mais déjà difficile, de la coordination spatiale entre les agents. En effet, pour exhiber un comportement de groupe et satisfaire des objectifs collectifs, chaque agent, à chaque instant, doit au moins être capable de se positionner et s'orienter correctement en fonction du déplacement de tous les autres agents. Il en résulte une forte interdépendance entre tous ces déplacements.

Nos travaux sur cette question s'inscrivent dans le cadre de la thèse CIFRE de F. Flacher, financée par Dassault Aviation. L'application sur laquelle il travaille concerne la coordination d'un ensemble d'aéronefs sans pilote dans le cadre de simulations d'opérations militaires sur le champ de bataille. Il s'agit de démontrer sur un plan très conceptuel ce que des capacités adaptatives peuvent apporter de concret dans ce type de contexte.

Cette application conduit F. Flacher à travailler à un niveau d'abstraction relativement élevé vis-à-vis des problèmes que nous poserons par la suite dans le cadre des jeux vidéo. En effet, les aéronefs étant équipés de GPS, les « capteurs » des agents qui simulent ces véhicules leur donnent directement la position relative des objets pertinents dans leur environnement. Un grand nombre de problématiques qui relèvent de la perception sensorielle de l'environnement et de la localisation au sein de cet environnement ne sont donc pas abordées dans le cadre de ce travail. Par ailleurs, seule la position des aéronefs est pertinente, on les assimile à des points dont la dynamique est éventuellement contrainte, donc on ne se préoccupe pas de la réalisation physique des mouvements.

Une fois le cadre posé, il faut admettre comme y insiste H. Van Parunak [1], que les méthodes de décomposition classiques ne s'appliquent pas de façon satisfaisante sur cette classe de problèmes. En effet, la multiplicité des agents ne permet pas d'utiliser les algorithmes classiques de planification qui reposent sur une connaissance complète de l'environnement. De plus, la forte interdépendance des déplacements rend inopérante la démarche qui consiste à identifier, à partir des spécifications du problème, toutes ces configurations possibles puis à déterminer une solution définie à l'avance. Enfin, la complexité de cette interdépendance augmente rapidement avec le nombre d'agents impliqués. Ce phénomène pose également un problème de robustesse au changement d'échelle face à la multitude des cas pour lesquels il faut concevoir une solution.

[1] [Van Parunak, 1997]

Face à ces défis, une approche alternative, dite ascendante, consiste à définir des mécanismes d'interaction locale entre des agents puis à adapter ces interactions jusqu'à obtenir le comportement global attendu. Plusieurs solutions de cette famille reposent sur des formalismes à base de champs de potentiel ou de champs de force. Avant de faire le tour des travaux mettant en œuvre de tels champs pour la coordination spatiale, nous commençons par une brève présentation historique de l'application de tels champs pour la navigation d'agents isolés.

6.2 Champs de potentiel et planification de chemins

0. Khatib [2] est le premier à avoir appliqué les champs de potentiel, en simulation et sur des bras manipulateurs, à la planification de chemins et à l'évitement d'obstacles. Sa méthode consiste à établir un champ de potentiel à partir d'une carte des obstacles donnée *a priori*. Chaque obstacle génère une force répulsive, le but génère une force attractive et la résultante de ces forces permet de construire une carte vectorielle. Ainsi, si l'agent sait déterminer à chaque instant sa position dans la carte, il peut déduire du champ un déplacement le conduisant de sa position courante à une position à atteindre en évitant les obstacles.

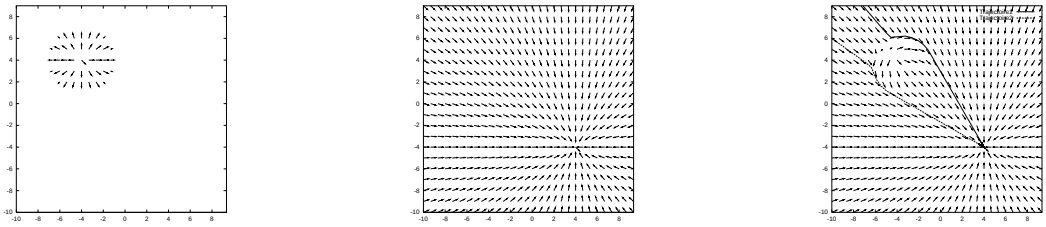


FIG. 6.1 – **a) champ répulsif autour d'un obstacle en B(-4,4) ; b) champ attractif vers une cible en C(4,-4) ; c) champ combiné et mouvements résultants**

Comme il apparaît sur la figure 6.1, le champ influe sur le comportement de l'agent de la même manière qu'un champ de potentiel électrique influe sur un électron. Toutefois, malgré le caractère éclairant de cette analogie, il faut noter que la définition du champ de potentiel n'implique pas que le champ soit calculé en tout point de l'espace, mais uniquement là où se trouve l'agent à un instant donné.

Cette proposition a été ensuite reprise et généralisée au cas de la planification de chemins par B. Krogh [3]. Un certain nombre de problèmes inhérents à la méthodologie des champs de potentiel ont été identifiés, notamment par Y. Korenz et J. Borenstein [4], en particulier le risque qu'un agent soit piégé dans un minimum local ou adopte une trajectoire oscillante. Depuis, de nombreuses méthodes ont été mises au point pour faire face à ces problèmes, comme l'utilisation de bruit [5], de méthodes évolutionnistes [6], ou de systèmes de moniteurs qui détectent une situation bloquée et appliquent une heuristique pour en sortir [7].

Le principal promoteur de l'approche fondée sur les champs de potentiel est R. Arkin [8]. Inspiré par les recherches comportementales et neurophysiologiques menées sur les batraciens par M. Arbib [9], qui permettaient

[2] [Khatib, 1985]. Ses premiers travaux datent de 1980, mais cet article postérieur est la synthèse la plus fréquemment citée.
[3] [Krogh, 1984]
[4] [Korenz & Borenstein, 1991a, Korenz & Borenstein, 1991b]
[5] [Arkin, 1989]
[6] [Pearce et al., 1992]
[7] [Korenz & Borenstein, 1991b, Piaggio et al., 2000]
[8] [Arkin, 1989]
[9] [Arbib, 1981, Arbib & House, 1985]

d'interpréter les mouvements de certains animaux comme résultant d'une combinaison d'attractions et de répulsions induites par l'environnement (une proie est un attracteur, un prédateur est un répulseur, etc.), R. Arkin a proposé une *théorie des schémas* qui décompose le comportement en un ensemble de *schémas* indépendants. Ces schémas, sortes d'atomes comportementaux, peuvent être perceptifs ou moteurs et engendrent des champs vectoriels attractifs ou répulsifs. La mise en œuvre de cette théorie consiste à combiner des schémas, attractifs pour les buts et répulsifs pour les obstacles.

Appliquée à la navigation des robots, cette méthode a démontré sa capacité à générer des trajectoires efficaces, continues et lissées avec une simplicité et une rapidité de calcul tout à fait satisfaisantes. Ces travaux ont donné lieu à plusieurs applications dans le domaine de la robotique mobile [10].

6.3 Champs de potentiel et coordination spatiale

Pour ce qui est de l'application de ces techniques à des problèmes de coordination spatiale, B. Partridge [11], lui aussi influencé par M. Arbib, est l'un des pionniers avec ses travaux sur la simulation de bancs de poissons.

Des résultats importants dans ce domaine proviennent des travaux de C. Reynolds [12]. Il définit pour chaque agent une combinaison de vecteurs vitesse construits à partir de perceptions locales. Ces vecteurs permettent par exemple d'éviter la collision entre agents ou de s'aligner sur la vitesse moyenne et le cap moyen de ses voisins. Avec les *boïds* [13], il a démontré la faisabilité de la génération de comportements coordonnés à partir de l'auto-organisation d'un petit ensemble de modules comportementaux. Malheureusement, l'ensemble des vecteurs vitesses possibles est prédéfini manuellement, ce qui réduit considérablement la variété des comportements collectifs potentiellement réalisables.

K. Zeghal [14] a apporté une contribution théorique et pratique au domaine en montrant que la combinaison de forces d'attraction, de répulsion et de contournement dérivant d'un potentiel forme une base permettant de réaliser toute trajectoire. Sur la base des principes théoriques qu'il a dégagés, il a pu montrer en simulation comment une combinaison simple de telles forces permettait de coordonner le mouvement de plusieurs avions dans un même espace aérien encombré de zones interdites.

Des résultats comparables ont également été produits en robotique collective, notamment ceux de M. Matarić [15] qui a montré l'existence d'un ensemble de comportements élémentaires – exploration prudente, agrégation, dispersion et retour au nid – qui, correctement utilisés, suffisent à résoudre des problèmes complexes de coordination spatiale robotique comme l'encerclement ou le déplacement en troupeau. Cependant, comme dans le cas de C. Reynolds, ces comportements élémentaires sont prédéfinis manuellement.

R. Arkin et son équipe ont aussi examiné la conception des mécanismes de coordination spatiale à l'aide de la théorie des schémas [16]. En effet, la combinaison des schémas permet de faire émerger des comportements collectifs utiles. Ainsi, T. Balch [17] a montré que la génération et le maintien de formation peuvent être synthétisés en élargissant un peu la théorie des schémas. Il montre qu'un groupe de véhicules autonomes évoluant parmi un ensemble de pôles attractifs s'auto-organise en diverses formations qui sont maintenues le long d'un parcours encombré d'obstacles. Cependant, malgré ces innovations, cette approche souffre des mêmes inconvénients que ceux

[10] [Barraquand et al., 1992, Matarić, 1994a, Balch & Arkin, 1995]
[11] [Partridge, 1982]
[12] [Reynolds, 1987]
[13] Contraction de *bird-oïds*
[14] [Zeghal, 1994]
[15] [Matarić, 1994a, Matarić, 1995]
[16] [Arkin, 1989, Arkin, 1992]
[17] [Balch & Arkin, 1998, Balch & Hybinette, 2000]

de C. Reynolds ou M. Matarić, à savoir la nécessité de définir à l'avance et régler manuellement un ensemble de schémas élémentaires.

Parmi les approches qui étendent la problématique de la coordination spatiale, C. Panatier [18] met en œuvre des champs de potentiel pour construire une représentation interne de la dynamique du comportement d'autres agents sous forme de tropismes. Parallèlement, généralisant l'approche classique au cas dynamique en utilisant des champs qui évoluent en fonction du temps, O. Simonin et J. Ferber [19] ont montré l'efficacité d'une transposition à la communication entre agents dans le cadre d'une coordination multi-agents réactive s'effectuant par émission de signaux.

Toutes les techniques de coordination spatiale à l'aide de champs mis au point manuellement que nous avons passées en revue reposent toutes sur une méthode générale décomposable en deux parties :

– la génération d'une variété de contrôleurs potentiels de coordination spatiale ;
– leur ajustement vers un contrôleur solution, correspondant aux spécifications du problème.

Plutôt que de réaliser la seconde partie à la main, on peut étendre la méthode en utilisant des procédés de mise au point automatique tels que les algorithmes évolutionnistes.

Le modèle de F. Flacher permet de représenter un ensemble de contrôleurs de coordination spatiale suffisamment grand et facile à explorer pour qu'il soit possible de trouver en un temps raisonnable des solutions efficaces à divers problèmes en utilisant un algorithme d'exploration et de sélection automatique. Divers chercheurs que nous avons déjà cités ont fait des tentatives dans ce sens [20]. Dans ce chapitre, après avoir montré la généricité de notre modèle sur une application simple inspirées des travaux de R. Vaughan [21], nous montrons que nous avons généralisé le modèle de coordination spatiale proposé par T. Balch et R. Arkin pour obtenir un ensemble de contrôleurs potentiels plus vaste, tout en maîtrisant la durée d'exploration de l'espace de recherche nettement plus important qui en résulte. En reproduisant une expérience de T. Balch et M. Hybinette [22], nous montrons que notre méthode permet la découverte automatique de contrôleurs de coordination spatiale qui, d'une part, surpassent ceux construits manuellement par les auteurs et, d'autre part, sont équivalents à des contrôleurs optimisés par évolution à partir des contrôleurs construits manuellement.

6.4 Présentation de GACS

6.4.1 Présentation générale

L'approche choisie par F. Flacher pour traiter des problèmes de coordination spatiale dans des univers continus et dynamiques est une approche ascendante fondée sur la combinaison de forces d'attraction et de répulsion entre tous les agents et les objets pertinents dans leur environnement. Comme dans toute approche ascendante, ces forces sont définies à un niveau élémentaire puis modifiées localement jusqu'à atteindre le comportement global recherché. Néanmoins, au lieu de nécessiter une fastidieuse modification manuelle des forces par un processus d'essais et erreurs, le système GACS [23] fait appel à un algorithme génétique pour trouver automatiquement des forces qui permettent aux agents d'atteindre un objectif de coordination qui s'exprime comme l'optimisation d'un ensemble de critères numériques de performance.

L'originalité du travail de F. Flacher se manifeste dans trois domaines complémentaires.

[18] [Panatier et al., 2000]
[19] [Simonin & Ferber, 2000]
[20] [Reynolds, 1994, Clark et al., 1992, Matarić, 1994b]
[21] [Vaughan et al., 1998]
[22] [Balch & Hybinette, 2000]
[23] pour Génération Ascendante de la Coordination Spatiale

– Le formalisme employé pour exprimer numériquement les forces d'attraction et de répulsion est proche de celui employé dans le cadre de la théorie des schémas de R. Arkin et T. Balch, mais il est plus général, et plus souple. Nous allons présenter ce formalisme dans la section suivante.
– Pour sélectionner un contrôleur de coordination spatiale adapté à un problème donné au sein de l'espace des contrôleurs possibles, l'approche repose sur un algorithme génétique, ce qui suppose d'exprimer numériquement le fait qu'une solution satisfait mieux le problème qu'une autre. Pour exprimer numériquement des objectifs de coordination qui peuvent être complexes, nous faisons appel à une approche multi-critères, ce qui nous permet de sélectionner des contrôleurs optimaux vis-à-vis d'un ensemble de critères donnés. Nous aborderons ce point à la section 6.4.3.
– Enfin, le choix d'exprimer nos contrôleurs sous la forme d'une combinaison linéaire de forces élémentaires d'attraction et de répulsion induit la définition d'une structure spécifique dans le génome des individus sélectionnés par l'algorithme. Les opérateurs de recombinaison génétique proposés par F. Flacher tirent parti de cette structure pour réaliser une exploration plus efficace de l'espace de recherche engendré par les contrôleurs, ce que nous exposerons aussi à la section 6.4.3.

6.4.2 Un formalisme pour la coordination spatiale

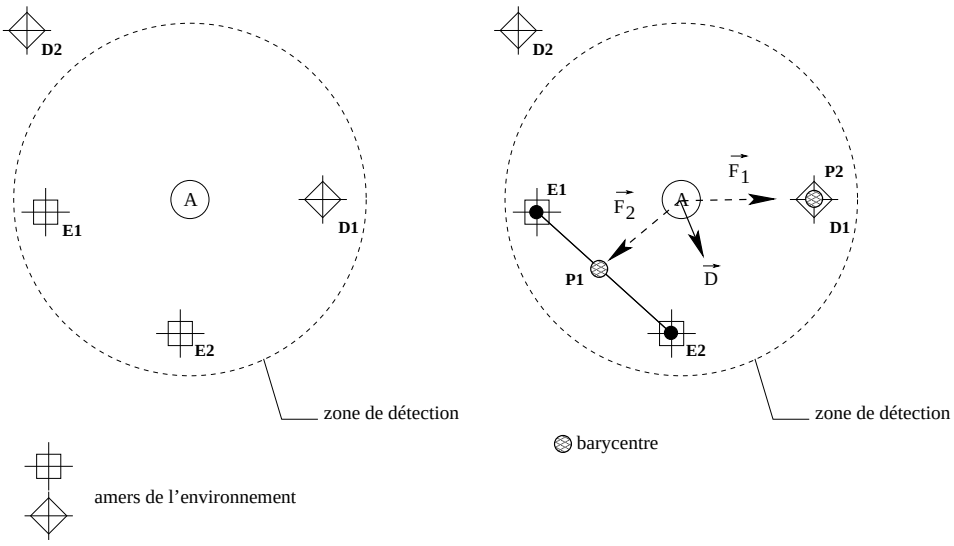


FIG. 6.2 – **Exemple où deux types différents d'objets pertinents sont présents dans l'environnement de l'agent : le type α (losange) et le type β (carré). L'agent détecte trois objets (D_1, E_1 et E_2) Les points d'intérêt P_1 et P_2 sont construits comme des barycentres des objets, respectivement entre E_1 et E_2 pour P_1 et sur D_1 seul pour P_2 ; le déplacement de l'agent est la résultante des forces déduites des points d'intérêt P_1 et P_2 et des fonctions de sensibilité associées.**

De même que dans la théorie des schémas de R. Arkin [24], nos agents sont dirigés par un ensemble de forces relatives à des points pertinents dans leur environnement, qui sont soit des objets, soit d'autres agents.

Ces forces sont attractives ou répulsives selon que les entités auxquelles elles sont attachées attirent ou repoussent l'agent, en fonction de la distance qui sépare l'agent de l'entité considérée. Elles représentent sous forme

[24] [Arkin, 1989]

vectorielle une influence particulière sur le comportement global de l'agent, le déplacement étant déduit de la résultante de leur combinaison.

Le comportement global d'un agent A est modélisé par un ensemble de N forces $\overrightarrow{F_i}$. Chacune de ces forces est un vecteur prenant l'agent pour origine, construit à partir d'une direction et d'une intensité.

La somme vectorielle de ces forces, modulées par un gain, permet de décrire le déplacement que l'agent doit s'efforcer de réaliser. Plutôt que d'associer ces forces directionnelles aux objets pertinents, nous avons fait le choix de les associer à des barycentres d'un ensemble d'objets et agents pertinents qui se situent dans la zone de détection de l'agent (voir figure 6.2). Nous appelons « points d'intérêt » d'un agent les barycentres auxquels sont associés des forces dans le contrôleur de cet agent.

Cet élément de notre formalisme permet d'exprimer simplement le fait que le déplacement d'un agent doit parfois être relatif à un ensemble d'objets considérés simultanément plutôt que dirigé vers chacun des objets considérés séparément. Par exemple, un joueur de football qui cherche à marquer un but doit se diriger vers un point situé derrière le ballon par rapport à la cage adverse. Ce point, « derrière le ballon par rapport à la cage », s'exprime très simplement comme un barycentre associé à la cage et au ballon auxquels sont affectés des coefficients bien choisis.

Jusqu'ici, nous avons expliqué comment les divers objets et agents pertinents qui entourent un agent sont pris en compte dans le formalisme pour construire un contrôleur sensible à la position relative de ces objets et agents. Il nous reste à voir comment le formalisme exprime les forces d'attraction et de répulsion associées aux divers points d'intérêts pris en compte par un contrôleur et comment il prend en compte la présence de η agents.

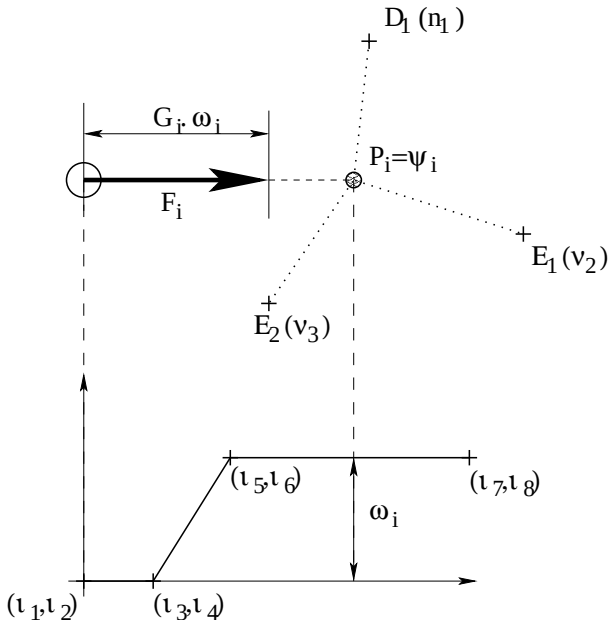


FIG. 6.3 – **Schéma explicatif de la construction d'une force F_i. La partie haute de la figure représente un agent attiré par un point d'intérêt dans son environnement. La partie basse représente la fonction d'attraction/répulsion associée à ce point d'intérêt. Le point d'intérêt P_i, visé par F_i, est le barycentre des éléments de l'environnement perçus D_1, E_1 et E_2, respectivement associés aux poids ν_1, ν_2 et ν_3. La fonction ω_i, paramétrée par $\iota_1, \cdots, \iota_8$ renvoie l'intensité partielle de F_i, qui est ensuite modulée par le gain G_i.**

Formellement, le déplacement $\overrightarrow{D_i}$ d'un agent A_i est modélisé par une somme de δ forces $\overrightarrow{F_k}$. Chacune de ces forces est un vecteur prenant l'agent A_i pour origine et construit à partir d'une direction et d'une intensité comme indiqué sur la figure 6.3. On commence par déterminer parmi tous les objets et agents pertinents au sein

de l'environnement de l'agent ceux qui sont perçus par l'agent. On les ordonne par type et par ordre de proximité décroissante à l'agent. On définit ensuite pour chaque force une fonction ψ_k, paramétrée par α réels, qui prend en entrée les positions relatives de α objets ou agents pertinents, notés O_j, et qui rend le point d'intérêt P_k :

$$\psi_k^{\nu_1,\cdots,\nu_\alpha}(O_1,\cdots,O_\alpha) = P_k = \sum_{j=1}^{\alpha} \nu_j.O_j,\ \sum_{j=0}^{\alpha} \nu_j = 1 \tag{6.1}$$

La fonction ψ_k est donc la fonction qui construit le point d'intérêt P_k en fonction des objets et agents pertinents perçus par l'agent.

L'intensité d'une force individuelle $\overrightarrow{F_k}$ est obtenue en multipliant une fonction normalisée ω_k par un gain G_k.

En pratique, la fonction ω_k est définie comme une fonction linéaire par morceaux, paramétrée par β réels, prenant en entrée la distance $\| \overrightarrow{AP_k} \|$ qui sépare l'agent du point d'intérêt P_k. Définir ω_k comme une fonction linéaire par morceaux revient à définir linéairement la valeur de ω_k sur β intervalles successifs :

$$\omega_k^{\iota_1,\cdots,\iota_\beta} = \begin{cases} \frac{\iota_4-\iota_2}{\iota_3-\iota_1}(x-\iota_1)+\iota_2, & x \in [\iota_1,\iota_3] \\ \vdots & \\ \frac{\iota_{i+3}-\iota_{i+1}}{\iota_{i+2}-\iota_i}(x-\iota_i)+\iota_{i+1}, & x \in [\iota_i,\iota_{i+2}] \\ \vdots & \\ \frac{\iota_\beta-\iota_{\beta-2}}{\iota_{\beta-1}-\iota_{\beta-3}}(x-\iota_{\beta-3})+\iota_{\beta-2}, & x \in [\iota_{\beta-3},\iota_{\beta-1}] \end{cases} \tag{6.2}$$

Le déplacement $\overrightarrow{D_i}$ de l'agent A_i résulte d'une combinaison linéaire de forces. Il est donc donné par l'équation suivante :

$$\overrightarrow{D_i} = \sum_{k=1}^{\delta} G_k \times \omega_k(\| \overrightarrow{A_iP_k} \|) \times (\sum_{j=1}^{\alpha} \nu_j.\overrightarrow{A_iO_j}) \tag{6.3}$$

Finalement, selon notre formalisme, un mécanisme de coordination spatiale est construit globalement pour η agents à partir de η fonctions de déplacement $\overrightarrow{D_i},\ i \in [1,\eta]$, où η est le nombre d'agents. Nous allons voir que, dans le cadre de sa thèse, F. Flacher n'a traité que le cas particulier où tous les agents sont identiques et partagent le même contrôleur afin de restreindre le nombre de paramètres à faire évoluer. Il construit donc une unique fonction $\overrightarrow{D_1}$ qui vaut pour tous les agents.

6.4.3 Sélection automatique d'un contrôleur performant

Les objectifs associés à différents problèmes de coordination spatiale peuvent être plus ou moins complexes et faire appel à des contraintes de différents types. Ayant fait le choix de régler les paramètres de nos contrôleurs à l'aide d'un algorithme évolutionniste, nous sommes contraints de mesurer la satisfaction d'un problème de coordination spatiale donné sous la forme de critères numériques. Il nous est rapidement apparu que représenter différents objectifs complexes à l'aide d'un critère unique n'était pas satisfaisant sur le plan méthodologique, car cela imposait un travail de réglage de la fonction d'évaluation correspondante au moins aussi difficile que de régler directement des contrôleurs à la main. Nous avons donc choisi une approche faisant appel à plusieurs critères à la fois, ce qui nous a naturellement conduit à nous tourner vers le domaine des algorithmes évolutionnistes multi-critères, qui sont en plein essor depuis quelques années [25].

[25] [Coello, 2003]

Méthodes évolutionnistes multi-critères

Alors que dans un problème d'optimisation simple, toutes les solutions peuvent être ordonnées en fonction de la valeur qu'elles atteignent pour le critère qu'elles optimisent, le trait caractéristique des problèmes multi-critères est que l'espace de recherche n'est que partiellement ordonné, c'est-à-dire que plusieurs solutions peuvent être équivalentes au regard des critères qu'elles optimisent. Par exemple, si l'on cherche à maximiser deux critères c_1 et c_2, une première solution peut être maximale sur c_1 et nulle sur c_2. Inversement, une autre solution peut être maximale sur c_2 et nulle sur c_1. Aucune des deux solutions n'est meilleure que l'autre.

Il arrive cependant qu'une solution soit moins bonne qu'une autre sur chacun des critères qui caractérisent le problème. On dit alors que la première solution est dominée par la seconde. On appelle « solutions non dominées » l'ensemble des solutions qui ne sont dominées par aucune autre solution. Pour un problème d'optimisation multi-critères donné, toutes les solutions non dominées forment un ensemble de compromis optimaux, nommé « front de Pareto » [26].

Les algorithmes évolutionnistes multi-critères partent d'une population de solutions quelconques et cherchent, en adaptant les mécanismes classiques des méthodes évolutionnistes, à atteindre une population d'individus qui appartiennent tous au front de Pareto du problème posé. La plupart des techniques cherchent en outre à faire en sorte que la population finale soit répartie le plus uniformément possible sur le front de Pareto afin d'en donner la meilleure approximation possible. Nous verrons que cette seconde contrainte n'est pas toujours pertinente pour les applications complexes que traite F. Flacher.

On distingue classiquement deux générations d'algorithmes, que nous n'allons pas passer en revue dans ce document [27]. La seconde génération d'algorithmes [28] se distingue de la première [29] par des algorithmes sensiblement plus performants, mais aussi plus complexes et plus préoccupés d'obtenir une meilleure couverture du front de Pareto sur un certain nombre de problèmes jouets choisis pour les difficultés spécifiques qu'ils posent.

Pour notre part, nous utilisons un algorithme génétique librement inspiré de l'algorithme NPGA de J. Horn [30], qui appartient à la première génération. Ce choix est justifié par l'efficacité satisfaisante de cette proposition, par sa souplesse et surtout par sa simplicité de mise en œuvre [31].

Comme l'algorithme NPGA, notre algorithme de sélection repose sur la détermination de strates de solutions non dominées. La première de ces strates est l'ensemble des solutions non dominées (le front de Pareto). La seconde strate est l'ensemble des solutions non dominées lorsqu'on a retiré la première strate, etc.

Dans les algorithmes évolutionnistes les plus standards, l'évolution de la population se fait génération après génération en conservant un certain pourcentage des meilleurs individus de la génération précédente dans la génération suivante, puis en recombinant entre eux ces individus pour créer de nouveaux individus qui viennent compléter la nouvelle génération. Dans notre algorithme, de la même façon, on attribue à chaque individu un pourcentage de chance d'être conservé d'une génération à la suivante en fonction de la strate à laquelle il appartient – ce pourcentage étant d'autant plus élevé qu'il appartient à une strate plus élevée de la population – puis on remplace les individus qui n'ont pas été conservés par des descendants de ceux qui l'ont été.

Un tel mécanisme n'inclut aucune pression vers une diversité maximale entre les individus. Toutefois, nous allons voir qu'il donne des résultats satisfaisants. Avant cela, il nous reste à présenter les spécificités du codage du

[26] [Pareto, 1896]

[27] nous renvoyons le lecteur à [Coello, 2003] et [Deb, 2004] pour approfondir cette question.

[28] [Zitzler & Thiele, 1998, Eckart Zitzler & Thiele, 2001, Deb et al., 2002, Knowles & Corne., 1999, David W. Corne & Oates, 2001, Veldhuizen & Lamont, 2000, Zydallis et al., 2001]

[29] [Schaffer & Grefenstette, 1985, Fonseca & Fleming, 1993, Srinivas & Deb, 1994, Horn et al., 1994, Erickson et al., 2001]

[30] [Horn et al., 1994]

[31] [Deb, 2001]

génome des individus et des opérateurs de recombinaison que nous leur appliquons.

Codage

Comme nous l'avons indiqué précédemment, le nombre d'agents impliqués dans les problèmes que nous abordons pouvant être grand, plutôt que de rassembler le codage d'une fonction de déplacement pour chaque agent dans un même génome, nous faisons le choix de ne coder qu'une fonction de déplacement unique dans le génome et d'attribuer cette fonction identiquement à tous les agents. Une alternative pourrait consister faire évoluer des fonctions de déplacements distincts pour les différents agents, mais se poseraient alors des problèmes de co-évolution réputés difficiles auxquels nous n'avons pas souhaité nous confronter dans le cadre de la thèse de F. Flacher.

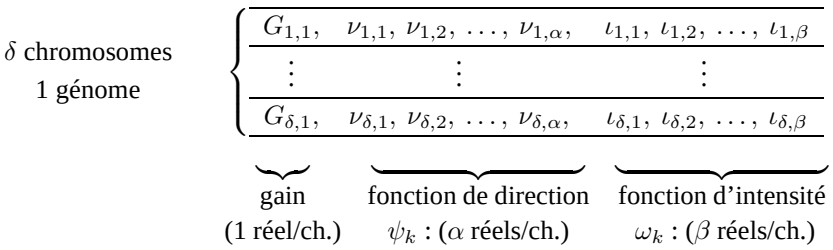


FIG. 6.4 – **Description d'un génome géré par GACS**

Le génome code une fonction de déplacement pour un agent. Comme nous l'avons vu précédemment, cette fonction de déplacement se décompose sous la forme d'un ensemble de δ forces. Le génome est donc constitué de δ chromosomes indépendants. Chaque chromosome représente avec $(1 + \alpha + \beta)$ valeurs réelles le gain, la fonction de direction et la fonction d'intensité permettant de déterminer une force, comme il apparaît sur la figure 6.4.

Tous les réels qui constituent un génome appartiennent à l'ensemble [0, 1]. Toutes les manipulations de ces réels par les opérateurs génétiques se font vers ce même ensemble afin de garantir la généricité des opérateurs de recombinaison. Lors de la transformation du génome en un contrôleur, ces paramètres génétiques sont traduits vers l'intervalle adéquat, qui est généralement propre à une simulation particulière.

Opérateurs génétiques

GACS fait appel à deux types d'opérateurs génétiques : la mutation et le croisement. La mutation, en opérant par légères modifications aléatoires, permet d'effectuer une exploration locale dans le voisinage d'une solution. Le croisement échange le matériel génétique entre deux solutions et permet d'explorer l'espace de recherche intermédiaire entre différentes solutions. Ces deux opérateurs sont appliqués avec des probabilités p_M et p_C qui diffèrent selon les expériences.

La mutation ajoute à chaque réel d'un génome une valeur aléatoire issue d'une loi normale. Cet ajout est effectué pour chaque valeur avec la probabilité p_M qui dépend de la taille du génome, de sorte que la probabilité de mutation globale d'au moins un réel dans tout le génome soit égale à 1.

Le croisement, appliqué avec une probabilité p_C, consiste à choisir aléatoirement n_{C3} chromosomes parmi les $(n_{C1}+n_{C2})$ chromosomes des parents. Ce choix d'échanger des chromosomes entiers entre individus est justifié par le fait que chaque chromosome représente intégralement l'influence d'un point d'intérêt, donc d'une combinaison de points pertinents, sur l'agent. Par conséquent, il s'agit d'un « bloc de construction [32] » pertinent pour notre mécanisme d'évolution.

[32] *building block*

6.5 Validation expérimentale

6.5.1 Un premier démonstrateur

L'environnement de simulation qui a servi de première plate-forme expérimentale pour évaluer GACS apparaît sur la figure 6.5. Il s'agit d'un problème dans lequel un groupe d'agents, baptisés « bergers », doit conduire un troupeau d'entités baptisées « canards » vers une zone précise au sein d'une arène circulaire. Cette simulation est inspirée des travaux de R. Vaughan [33], dans lequel un robot réel unique devait diriger de la même façon un troupeau de canards domestiques bien vivants.

Dans toutes les simulations décrites ici, il y a six canards et leur vitesse maximale est identique à celle des bergers.

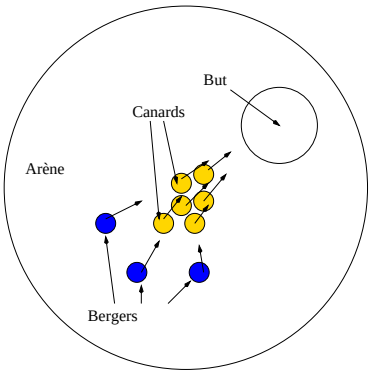


FIG. 6.5 – **L'arène, les canards et les bergers**

Le comportement des canards résulte de la combinaison de trois tendances :
- ils évitent de trop se rapprocher des murs qui bordent l'arène ;
- ils cherchent à se rapprocher des autres canards présents dans les limites de leur champ visuel ;
- ils cherchent à fuir les bergers présents dans les limites de leur champ visuel.

Le but des bergers est atteint aussitôt que tous les canards sont dans la zone désignée comme but.

Le comportement des canards est réglé de telle façon qu'il est extrêmement difficile à un berger isolé de conduire seul le troupeau vers le but.

Protocole expérimental

Dans cette application simple, l'objectif des agents peut s'exprimer simplement sous la forme d'un seul critère. Ces travaux ont été menés sans l'extension de GACS permettant le traitement de problèmes multi-critères. L'algorithme génétique employé est l'un des plus courants : il consiste à conserver à chaque génération les meilleurs individus correspondant à 20% de la génération précédente et d'engendrer par mutation et croisement de ces individus les 80% manquants pour compléter la nouvelle génération. Cet algorithme est lancé sous les conditions expérimentales suivantes :
- $N_P = 100$ individus dans la population ;
- $N_F = 5$ individus par fenêtre de voisinage ;
- $T_M = 500$ pas de temps alloué au maximum pour résoudre la tâche ;

[33] [Vaughan et al., 1998]

– $N_E = 10$ situations initiales pour chaque évaluation d'un contrôleur ;
– $P_c = 40\%$ de chance de sélectionner un vecteur de réels codant un point d'intérêt pour le croisement ;
– $P_m = 5\%$ de chance d'effectuer une mutation sur un des réels du génome.

La fonction de *fitness*, qui évalue la qualité d'un contrôleur après simulation, est donnée par le temps maximum T_M alloué aux bergers pour la résolution de la tâche, auquel on soustrait le temps moyen de résolution mis par les bergers, testés sur un nombre N_E d'essais. On a donc, si T_R est la fonction qui associe le temps de résolution de la tâche dans les conditions initiales P_i :

$$Fitness(C) = T_M - \frac{\Sigma_{i=1}^{N_E} T_R(C, P_I)}{N_E}$$

Résultats

Nous avons fait fonctionner le processus évolutif durant près de 900 générations. En fait, dès la $70^{ième}$ génération, on obtient un contrôleur tout à fait satisfaisant, puisqu'il mène les canards au but sans échec à partir de 1000 situations initiales différentes avec une moyenne de 410.08 pas de temps, soit à peine moins vite que des contrôleurs à base de systèmes de classeurs obtenus à l'issue d'une fastidieuse mise au point manuelle [34]. Cependant, à partir de la $92^{ième}$ génération, on obtient des individus généralement plus rapides, mais qui sont globalement moins robustes. Cette tendance illustre un phénomène bien connu en apprentissage, la nécessité de gérer le compromis entre généralisation et spécialisation. Dans notre cas, l'algorithme génétique explore l'espace de recherche en ne testant chacun des individus que pour 10 conditions initiales différentes. Le fait qu'on obtienne des contrôleurs aussi robustes sur 1000 essais prouve que notre formalisme possède de bonnes propriétés de généralisation. Malgré cela, à terme, la sélection des individus conduit l'algorithme génétique à privilégier des contrôleurs très performants sur les 10 situations initiales qui servent à son évaluation au détriment des autres situations, ce qui se traduit par une légère sur-spécialisation.

Cette expérience a permis de valider l'approche sur un problème simple. En particulier, la comparaison avec une résolution fondée sur des systèmes de classeurs a permis de mettre en évidence les nombreux bénéfices de cette approche vis-à-vis de tentatives pour appliquer des classeurs opérant sur une représentation discrète à un environnement continu.

En outre, l'analyse d'un contrôleur particulier a montré que GACS était capable de trouver une solution élégante, simple et efficace au problème du rassemblement des canards, faisant intervenir un phénomène de régulation de la tendance des canards à se disperser quand un berger s'approche trop du troupeau [35].

Fort de ce résultat, nous nous sommes donc tourné vers une application plus complexe, plus proche du domaine d'application donnant lieu au financement de F. Flacher et faisant intervenir une évaluation de performance fondamentalement multi-critères.

6.5.2 Un problème plus difficile

La seconde expérience traitée avec GACS est inspirée de celle développée dans les travaux de T. Balch et M. Hybinette [36] sur le maintien de formation. Dans cette étude, T. Balch s'appuie sur la théorie des schémas pour traduire son expertise sous la forme d'un paramétrage particulier d'un ensemble prédéfini de schémas. Nous avons choisi de reproduire cette expérience car GACS s'inspire de cette théorie et s'efforce de la généraliser pour automatiser la mise au point des mécanismes de coordination spatiale.

[34] [Flacher & Sigaud, 2003]
[35] [Flacher & Sigaud, 2002]
[36] [Balch & Hybinette, 2000]

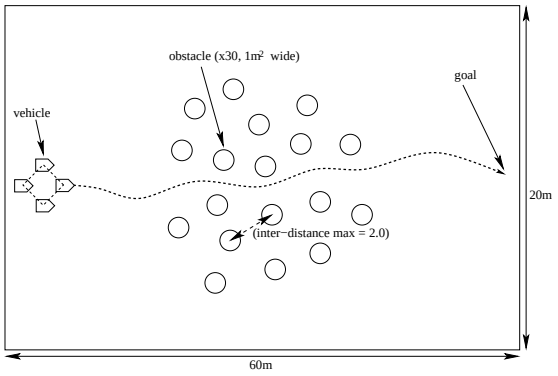


FIG. 6.6 – **Expérience de maintien en formation ; un groupe d'agents doit atteindre un but tout en restant en formation et en évitant les obstacles.**

Originellement, cette expérience a été développée par T. Balch sur la plate-forme *Team Bots*[37] dans le but de mettre au point des architectures de contrôle pour des véhicules militaires terrestres sans conducteurs. Notre simulation, reprenant leur spécification, implique quatre véhicules holonomes traversant une zone encombrée d'une vingtaine d'obstacles, comme indiqué sur la figure 6.6, selon une formation qui contraint les véhicules à se déplacer en losange en suivant le véhicule de tête. La zone de manœuvre est un rectangle de 20m x 60m. Trente obstacles de $1m^2$ de surface sont répartis autour du centre de la zone de façon à empêcher que le groupe de véhicules puisse traverser la zone encombrée sans rompre la formation[38].

Protocole expérimental

Pour évaluer le comportement collectif de maintien de formation dans un champ d'obstacles, nous avons recours à 3 critères : la qualité du maintien de formation, de l'évitement d'obstacles et de l'atteinte d'une zone but.

1. Une récompense itérative est déterminée à chaque instant à partir de l'erreur de positionnement du groupe de véhicules vis-à-vis de la formation idéale. La moyenne de cette récompense, sur la durée nécessaire à l'atteinte de la zone cible, donne l'évaluation du maintien de formation. Plus formellement, si t_M est le temps alloué pour la résolution de la tâche globale et si r_t est la fonction de récompense qui, à un groupe d'agent donné, associe une valeur positive ou nulle, on a :

$$f_1(a) = \frac{1}{t_M} \sum_{t=1}^{t_M} r[e_f(a)],\ a \in \mathcal{A} \tag{6.4}$$

avec

$$e_f(a) = \sum_{k=1}^{4} |\overrightarrow{A_k P_k}|,\ r(x) = \begin{cases} 1.0 & \text{if } x < 0.05 \\ \frac{0.5-x}{0.45} & \text{if } x \in [0.05, 0.5] \\ 0.0 & \text{if } x > 0.5 \end{cases} \tag{6.5}$$

où P_1^t, P_2^t, P_3^t et P_4^t sont les positions idéales dans la formation à l'instant t.

2. On compte le nombre de collisions entre un véhicule et un obstacle qui se produit durant le déplacement. La différence entre ce nombre de collisions et un nombre maximum de collisions autorisé donne l'évaluation

[37] disponible à l'adresse : `http ://www.cs.cmu.edu/coral/minnow`.

[38] La distance minimale entre chaque obstacle est de deux mètres.

de l'évitement d'obstacles. Plus précisément, si c_M est le nombre maximum de collisions autorisés et si q est la fonction qui associe, à un groupe d'agents donné, son nombre de collisions avec les obstacles de l'environnement, on pose :

$$f_2(a) = \frac{c_M - q(a)}{c_M},\ a \in \mathcal{A} \tag{6.6}$$

3. À la fin du temps alloué pour la résolution de la tâche, on détermine une récompense à partir de la distance moyenne de tous les véhicules à la zone cible. Cette récompense est l'évaluation de l'atteinte de la cible. Plus précisément, si t_M est le temps alloué pour la résolution, si C est la cible, e_g la fonction qui associe à un groupe donné d'agents l'erreur de positionnement cumulée vis-à-vis de la cible et r la fonction de récompense définie dans l'équation (6.4), on pose :

$$f_3(a) = r[e_g(a)],\ e_g(a) = \sum_{k=1}^{4} |\overrightarrow{A_k C}|, a \in \mathcal{A} \tag{6.7}$$

Contrôleur à base de schémas

Afin d'effectuer une étude comparative de son modèle avec celui de T. Balch [39], F. Flacher a reproduit le contrôleur développé pour cette expérience dans le cadre de la théorie des schémas. Ce contrôleur est composé de six schémas : **éviter-les-obstacles**, **éviter-les-robots**, **aller-vers-le-but**, **maintenir-la-formation**, **aller-au-centre** et **bruiter**[40]. À chaque instant, chaque schéma génère un vecteur représentant un composant du futur mouvement d'un véhicule. Chaque schéma est multiplié par un gain indiquant son importance relative vis-à-vis des autres schémas. Les six vecteurs résultants sont sommés pour obtenir la direction et l'intensité du mouvement.

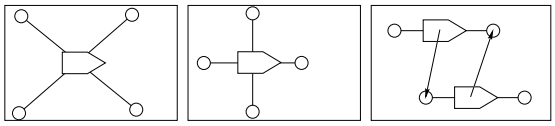


FIG. 6.7 – **Du point de vue d'un agent, un certain nombre de sites d'attachement (les cercles) sont associés à tous les autres agents. Ces sites sont les points visés par le schéma maintenir-la-formation.**

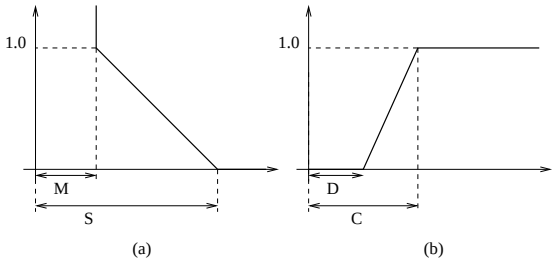


FIG. 6.8 – **Fonctions d'intensité utilisées par les schémas éviter-les-obstacles et éviter-les-robots (a), et les schémas aller-vers-le-but, aller-au-centre et maintenir-la-formation (b).**

Les six vecteurs de sortie des schémas sont déterminés à partir de deux fonctions d'intensité (ou de magnitude) qui renvoient l'intensité du vecteur selon la distance de l'agent au point visé par le schéma. Ces fonctions sont présentées sur la figure 6.8. La première fonction (a) est décroissante et paramétrée par les réels M et S. Associée à un gain négatif, cette fonction permet de représenter une répulsion de plus en plus forte si l'agent s'éloigne de l'objet concerné, un obstacle ou un autre agent selon le cas. La seconde fonction (b) est croissante et paramétrée par les réels C et D. Associée à un gain positif, cette fonction permet de représenter une attraction de plus en plus forte

[39] [Balch & Hybinette, 2000]

[40] ***avoid-static-obstacle***, ***avoid-robot***, ***move-to-goal***, ***maintain-formation***, ***move-to-unit-center*** et ***noise***.

si l'agent s'éloigne de l'objet concerné, le but, le centre de la formation ou la position idéale dans la formation selon le cas. Cette position idéale est le *site d'attachement* le plus proche de l'agent. Les sites d'attachements introduits par T. Balch sont construits à partir de la position des autres robots, comme illustré sur la figure 6.7. Le vecteur de sortie du schéma restant, **bruiter**, est construit à partir d'une direction aléatoire, et d'une intensité égale à 1. Il est paramétré par le nombre P d'itérations entre deux changements de direction.

Évolution des contrôleurs

Nous avons également soumis ce problème à deux séries d'évolutions pour obtenir deux autres contrôleurs de coordination spatiale : un contrôleur $S_{e.balch}$ évolué à partir de la solution originale de T. Balch, et un contrôleur S_{gacs} évolué via GACS.

Pour le premier contrôleur $S_{e.balch}$, l'évolution a réglé les valeurs du gain et des paramètres $\{M, S\} \cup \{D, C\}$ de chaque schéma. Les 16 réels correspondant ont été optimisés via 5 évolutions, lancées avec une population de $n_P = 500$ individus, une probabilité de mutation $p_M = 10\%$ et pas de croisement. Pour le second contrôleur S_{gacs}, l'évolution a réglé les valeurs des gains et des paramètres des fonctions de direction et d'intensité. Les 102 paramètres correspondant ont été optimisés via 5 évolutions, lancées avec une population de $n_P = 200$ individus, et des probabilités de mutation $p_M = 10\%$ et de croisement $p_C = 40\%$.

Dans les deux cas, le temps maximum alloué à une simulation est $t_M = 500$ pas de temps, et chaque évaluation d'un contrôleur est réalisée à partir de $n_E = 25$ essais. La moyenne des critères est calculée sur ces différents essais pour fournir les évaluations des contrôleurs à l'algorithme génétique. Dans la section suivante, nous présentons une comparaison entre la solution implémentée d'après l'article de T. Balch, $S_{m.balch}$, la meilleure solution évoluée à partir de la solution originale de T. Balch, $S_{e.balch}$ et la meilleure solution obtenue via GACS, S_{gacs}.

Résultats

Les résultats obtenus avec $S_{m.balch}$ sont conformes à ceux présentés dans l'article original. Cette conformité est toutefois essentiellement qualitative puisque T. Balch n'a procédé qu'à une seule mesure quantitative des performances de son modèle : celui de la robustesse au changement du nombre d'agents de traversée du champs d'obstacle. La solution $S_{m.balch}$ se révèle robuste au changement d'échelle sur cette mesure, puisque les performances restent proche de la valeur moyenne lorsqu'on augmente le nombre d'agents impliqués de 1 à 8. La variation observée des performances dans ce cas est d'ailleurs comparable à celle observée par T. Balch (de l'ordre de 10%).

Le tableau 6.1 rapporte les performances moyennes des différentes solutions testées. On constate que la solution $S_{m.balch}$ est dominée au sens de Pareto par la solution S_{gacs}, puisqu'elle obtient de moins bonnes performances en évitement d'obstacles et en maintien de formation et des performances équivalentes sur l'atteinte du but. On constate également que $S_{e.balch}$ et S_{gacs} sont équivalentes au sens de Pareto, bien que $S_{e.balch}$ ne domine pas $S_{m.balch}$.

6.6 Discussion

Vis-à-vis de nos travaux antérieurs sur les systèmes de classeurs, notre nouvelle approche partage avec celle de T. Balch l'intérêt des propriétés de robustesse inhérentes à l'usage des champs de potentiel. Comme nous l'avons montré dans nos travaux précédents [41], notre formalisme génère des solutions robustes aux perturbations sur les

[41] [Flacher & Sigaud, 2003]

solutions	évitement d'obstacle	maintien en formation	atteinte du but
$S_{m.balch}$	90.7 (12.6)	54.7 (11.1)	90.9 (14.3)
$S_{e.balch}$	99.9 (0.5)	58.0 (12.8)	89.8 (14.3)
S_{gacs}	92.8 (17.8)	58.6 (12.5)	96.1 (3.5)

a : relation avec $S_{m.balch}$ (erreur* estimée)			
S_{gacs}	$> (0.1\%)$	$> (2.0\%)$	$\sim (10.0\%)$
$S_{e.balch}$	$> (0.1\%)$	$> (5.0\%)$	$< (5.0\%)$

b : relation avec $S_{e.balch}$ (erreur * estimée)			
S_{gacs}	$< (1.0\%)$	$\sim (10.\%)$	$> (5.0\%)$

* *erreur sur l'hypothèse de différence des deux séries de données, estimée via le test statistique de Mann & Whitney*

TAB. 6.1 – **Performances en % des différentes solutions sur les 3 critères. Les résultats sont moyennés sur 100 essais (l'écart type est indiqué entre parenthèses). Les parties a et b du tableau donnent les relations entre les performances des solutions, validées de manière statistique. L'erreur statistique sur l'hypothèse de différence entre les données est indiquée entre parenthèses. Quand cette erreur est supérieure à 10.0% nous avons considéré que les deux performances étaient statistiquement équivalentes.**

conditions initiales. L'existence d'un attracteur au sein du système dynamique défini par les fonctions d'attraction/répulsion permet de justifier le maintien de la formation malgré la présence de perturbations. En effet, un tel attracteur maintient le système dans un état d'équilibre stable vers lequel le système sera naturellement ramené s'il est perturbé. Ce phénomène, que l'on peut qualifier d'« *homéostasie spatiale* », est une propriété d'auto-organisation typique des champs de potentiel ou de force.

Si le modèle s'avère efficace et robuste pour résoudre des problèmes de coordination spatiale une fois tous les paramètres réglés, il reste à expliquer la capacité de GACS à trouver efficacement le bon jeu de paramètres pour un problème donné. De façon générale, la capacité d'une approche évolutionniste à trouver une bonne solution à un problème donné dépend à la fois de la taille de l'espace de recherche parcouru par l'algorithme, de l'efficacité de l'algorithme lui-même et de la répartition des solutions satisfaisantes au sein de cet espace. L'espace de recherche parcouru par GACS est indiscutablement plus vaste que celui parcouru par l'algorithme d'optimisation de paramètres employé pour trouver $S_{e.balch}$. Le fait que nous obtenions des solutions aussi bonnes que $S_{e.balch}$ indique que notre approche compense ce handicap par un meilleur algorithme de recherche et une meilleure structuration de l'espace des solutions. Cette structuration est apportée dans notre formalisme de deux manières.

• Premièrement, la fonction de déplacement d'un agent est obtenue à partir de δ forces indépendantes qui représentent chacune une interaction particulière avec l'environnement. La structure de la fonction de déplacement est respectée dans le paramétrage puisqu'un génome est décomposé en δ chromosomes codant chacun pour une force. Ainsi, si une interaction particulière se révèle intéressante pour la résolution du problème, elle peut être isolée en tant que bloc de construction dans notre paramétrage pour être exploitée par l'algorithme d'évolution.

• Deuxièmement, chaque fonction de direction ψ_k d'une force F_k est construite sous la forme d'un barycentre. Celui-ci permet de mettre en évidence une relation spatiale pertinente pour la résolution du problème posé. Notre choix d'introduire des barycentres dans notre formalisme induit indiscutablement une certaine redondance, puisque l'on pourrait atteindre le même résultat avec une combinaison de forces orientées individuellement vers des ob-

jets pertinents. Néanmoins, ce choix facilite le travail de l'algorithme génétique en lui permettant d'identifier des blocs de construction élémentaires qui se propagent ensuite efficacement au sein de la population. En outre, lorsque l'usage de barycentres n'est pas pertinent pour un problème donné, l'algorithme a toujours la possibilité de sélectionner un barycentre qui ne s'attache qu'à un seul point pertinent, ce qui revient à avoir une force directement dirigée vers ce point.

La prise en compte de barycentres apporte de la souplesse à notre modèle vis-à-vis de celui de T. Balch. En effet, un point d'intérêt peut se déplacer relativement à chacun des objets pertinents auquel il est relié, alors que le déplacement des sites d'attachement de T. Balch est identique à celui des agents auxquels ils sont reliés. Lorsque les différents objets pertinents sont animés de trajectoires distinctes, notre solution est sensiblement plus riche et semble donner de meilleurs résultats. La composante d'auto-organisation de notre système dispose donc d'un degré de liberté supplémentaire par rapport à celle du système de T. Balch.

Par ailleurs, dans le formalisme de T. Balch, le concepteur est contraint de spécifier *a priori* le positionnement des sites d'attachement relativement aux agents, alors que, dans notre cadre, ce positionnement relatif est le résultat des sélections opérées par l'algorithme génétique. Ainsi, notre nouvelle approche permet de limiter au maximum l'apport de connaissances externes dans le processus de découverte d'une solution.

Enfin, notre modèle s'applique, comme nous l'avons vu, à la résolution de tâches plus complexes ou diverses que le seul maintien de formation.

6.7 Travaux futurs

Une conséquence importante du recours à des méthodes de sélection multi-critères est que, au lieu de produire un individu unique qui peut être considéré comme incarnant la meilleure solution trouvée à un problème donné, ces méthodes produisent une population de solutions qui ne se dominent pas mutuellement. La taille de cette population peut éventuellement être très importante, si bien que choisir à quel(s) individu(s) on va s'intéresser particulièrement est un problème en soi. Pour faire face à ce problème, F. Flacher entreprend à présent des travaux méthodologiques visant à automatiser la sélection de quelques solutions représentatives, en ajoutant a posteriori aux critères sur lesquels les individus ont été sélectionnés de nouveaux critères, comme la robustesse des performances ou la simplicité d'expression du contrôleur.

Ces travaux ont d'ores et déjà permis de révéler quelques points significatifs.

– F. Flacher a vérifié que, en lançant de nombreux algorithmes génétiques en parallèle puis en fusionnant les populations des meilleurs individus de chaque évolution, le nouvel ensemble d'individus non dominés résultant contient des individus issus de la plupart des évolutions. Cela implique que les évolutions convergent de façon stable vers le même ensemble d'individus non dominés, qui est sans doute extrêmement proche du front de Pareto intrinsèque au problème.
– Sur un problème applicatif concret, chercher à obtenir des solutions qui couvrent le plus exhaustivement possible le front de Pareto n'est pas toujours pertinent. En effet, il arrive que l'on ne s'intéresse pas aux solutions extrêmes qui privilégient un seul critère aux dépens de tous les autres. Par exemple, un groupe d'aéronefs qui reste parfaitement en formation mais n'évite aucun obstacle ne constitue pas une solution satisfaisante, bien qu'un contrôleur réalisant ce compromis appartienne au front de Pareto. Pourtant, focaliser la recherche de l'algorithme génétique a priori, en éliminant au fil des générations les individus extrêmes, n'est pas un bon choix. Une étude de relation de filiation montre qu'un très bon individu a souvent des parents provenant de zones du front de Pareto très distinctes de sa position à lui.

La relation entre la simplicité d'un contrôleur, qui se mesure au nombre de paramètres qu'il met en jeu, et son

interprétabilité ne va pas de soi, même s'il est clair qu'un contrôleur simple a plus de chances d'être interprétable qu'un contrôleur doté de davantage de paramètres. L'interprétabilité reste une des propriétés cruciales recherchées par F. Flacher, car c'est elle qui permet de suggérer des solutions originales qu'un ingénieur pourra ensuite chercher à adapter ou améliorer pour d'autres contextes.

Une deuxième voie de recherche actuellement suivie par F. Flacher consiste à s'efforcer d'intégrer son formalisme dans le cadre théorique proposé par K. Zeghal [42]. Cela suppose une adaptation significative, afin que toutes les forces manipulées par GACS dérivent d'un potentiel, comme c'est le cas dans les travaux de K. Zeghal. Les recherches de F. Flacher devraient y gagner en validité théorique et en généralité. Par la suite, il restera à confronter son modèle à celui que propose C. Reynolds, qui a mis au point de nombreux schémas de navigation efficaces dans une perspective beaucoup plus empirique.

6.8 Conclusion

Dans ce chapitre, nous avons présenté un modèle de coordination spatiale en suivant deux lignes directrices. D'une part, nous avons généralisé le modèle issu de la théorie des schémas de R. Arkin en introduisant un paramétrage complet des fonctions d'attraction et de répulsions ainsi que des points auxquels ces fonctions sont attachées. D'autre part, nous avons proposé un modèle qui facilite l'exploration de l'espace de recherche représenté. Nous avons adjoint à notre paramétrage une structuration de l'information et un algorithme évolutionniste multi-critères qui ont fait la preuve de leur efficacité.

Dans le cadre d'une comparaison expérimentale que nous n'avons pas détaillée ici [43], F. Flacher a montré que les mécanismes adaptatifs des systèmes de classeurs s'appliquent difficilement à un problème de coordination spatiale trop complexe, ce qui produit des performances insatisfaisantes, tandis que la mise au point des règles à la main peut être d'une complexité rapidement rédhibitoire. L'approche alternative proposée ici consiste à confier à des algorithmes évolutionnistes le rôle de définir les interactions locales entre les agents et à s'appuyer sur des phénomènes d'auto-organisation pour coordonner ces interactions locales. Nos résultats expérimentaux ont montré que notre démarche produit des résultats satisfaisants

Le trait le plus intéressant de notre modèle provient de ce que la coordination spatiale des agents émerge des interactions locales sans que le concepteur ait à spécifier directement la forme que doit prendre cette coordination spatiale. De plus, les contrôleurs obtenus de cette façon se montrent plus robustes aux perturbations que les contrôleurs écrits sous forme de systèmes de classeurs. Ces conclusions montrent les avantages méthodologiques très concrets que l'on peut retirer de l'adjonction de propriétés d'auto-organisation à un modèle sélectionniste classique.

La contrepartie de ces propriétés, et notamment de l'implication minimale du concepteur dans le processus de mise au point des comportements, est une moindre interprétabilité des contrôleurs obtenus. Comme nous venons de l'indiquer, des travaux sont en cours pour faciliter la recherche, au sein d'une population de contrôleurs satisfaisants, de ceux dont le fonctionnement a le plus de chances d'être interprétable. Nous espérons que le recours aux outils et concepts de la théorie des systèmes dynamiques permettra de faciliter cette compréhension et favorisera l'intervention éventuelle des concepteurs lorsqu'elle sera nécessaire.

En définitive, nous avons démontré la faisabilité de notre approche puisque nous parvenons à générer automatiquement des contrôleurs de coordination spatiale dont les performances sont comparables à celles de contrôleurs mis au point manuellement par des experts du domaine et optimisés par un algorithme évolutionniste.

[42] [Zeghal, 1994]
[43] [Flacher & Sigaud, 2003]

Cependant, il faut reconnaître que les algorithmes génétiques sur lesquels reposent les capacités adaptatives de GACS présentent de nombreux points faibles :

- ils supposent le calcul explicite d'une fonction de *fitness* qui peut être délicate à exprimer ;
- ils sont lents à converger ;
- ils ne permettent pas l'exploitation opportuniste de sources de récompenses intermédiaires ou la réadaptation rapide lorsque les sources de récompense changent.

Aux chapitres 8 et 9, nous exposerons donc nos efforts pour reprendre la même problématique dans un cadre impliquant des mécanismes d'apprentissage par renforcement. Cela nous conduira notamment à nous interroger sur l'opportunité de revenir au cadre formel des processus de décisions markoviens, étendus au cadre dans lequel les états et les actions sont continus.

La seconde limitation dont souffrent les travaux présentés dans ce chapitre provient de ce que la coordination des agents se limite à une composante strictement spatiale, alors que le cadre applicatif auquel nous nous intéressons est susceptible d'impliquer la coordination d'actions de toutes natures entre les agents. Là aussi, nous nous tournerons au chapitre 9 vers des extensions du cadre de processus de décision markovien destinées à traiter le cas où plusieurs agents doivent coordonner leurs actions.

Auparavant, nous allons revenir au chapitre suivant sur une limitation des travaux présentés ici, liée au fait que le réglage de tous les paramètres est réalisé à l'avance pour chaque individu et ne change plus en cours d'expérience. En effet, nous allons ajouter une propriété d'évolution dynamique de ce réglage pour traiter des problèmes de sélection de l'action.

Chapitre 7

Champs de forces et dynamique interne

« Vous n'avez jamais sniffé du Frisk ? Après, t'as l'impression d'avoir Val Thorens dans les narines ! »

T. Gourdin, le 26/11/2003

Dans ce chapitre, nous allons nous intéresser à la résolution de problèmes qui font intervenir des interactions entre la dynamique du métabolisme des agents et celle de leurs interactions avec leur environnement.

En pratique, il s'agit de traiter une classe de problèmes non markoviens, dans laquelle le caractère non stationnaire des interactions entre l'agent et son environnement est dû à la présence de variables internes. Nous présentons ici des travaux préliminaires, qui reposent sur une extension du formalisme proposé par F. Flacher au cas où l'agent est doté de variables internes qui évoluent continûment. Le travail, présenté ici assez succinctement, est resté sans suite pour l'instant, alors qu'il nous semble qu'il répond à une problématique importante pour les applications que nous visons. Nous espérons donc trouver à l'avenir le contexte adéquat pour prolonger cette réflexion dans une perspective plus large.

7.1 Introduction

Au chapitre précédent, nous nous sommes intéressé à une classes de problèmes dont le caractère non stationnaire provenait de la présence de plusieurs agents. Nous avons montré comment notre approche à base de champs de force fournissait des solutions satisfaisantes à ces problèmes.

Dans ce chapitre, nous allons mettre en évidence les modifications qu'il faut apporter au formalisme précédent pour traiter une autre classe de problèmes, dans laquelle le caractère non stationnaire provient cette fois de la présence de variables internes dont la valeur évolue spontanément et continûment au cours du temps.

Alors que, dans le cadre précédent, le problème posé à un algorithme adaptatif consistait généralement à trouver un comportement qui maximise une performance définie de l'extérieur, la prise en compte de variables internes permet de se poser un problème sensiblement différent. On cherche cette fois à trouver un comportement qui permet de maintenir certaines variables internes, dites « variables essentielles », entre des bornes qui définissent un domaine de viabilité. On passe donc d'un problème d'optimisation d'un ou plusieurs critères externes à un problème de satisfaction de contraintes internes.

Ce déplacement thématique nous rapproche du problème effectivement résolu par les êtres vivants, connu sous le nom de problème de sélection de l'action. C'est donc dans le domaine de la modélisation de systèmes de sélection

de l'action que nous allons rechercher un exemple de cadre expérimental.

Le problème de la sélection de l'action consiste pour un animal, un robot ou un agent logiciel à choisir à chaque instant, dans un répertoire d'actions possibles, l'action qui maximise ses chances de survie à long terme. Ces actions entrent généralement en conflit – par exemple, on ne peut pas manger et dormir en même temps – et leur satisfaction optimale nécessite un ordonnancement adéquat.

Les recherches sur la sélection de l'action sont issues de travaux en éthologie [1]. Tous les modèles classiques supposent l'existence d'un répertoire de *comportements élémentaires* [2] qui sont organisés soit hiérarchiquement [3], soit selon un graphe d'activations et d'inhibitions réciproques [4].

L'existence d'un répertoire de comportements élémentaires permet de ramener l'évolution continue d'un agent au sein d'un environnement complexe à des transitions discrètes entre des « états » abstraits bien séparés.

Lors de chaque transition, le problème de la sélection de l'action consiste à se reposer la question de choisir le comportement élémentaire suivant compte tenu de l'état abstrait dans lequel se situe l'agent. Un tel cadre rend bien compte de la démarche d'un éthologue qui cherche à élaborer un modèle du comportement d'un animal qu'il étudie. Une première phase d'observation consiste à établir le répertoire des comportements élémentaires de l'animal et à se donner une description pertinente des variables d'état qui interviennent dans la sélection de l'action, puis une seconde phase consiste à proposer un modèle qui rende compte du comportement observé.

La limite de cette approche est que, une fois la grille de lecture abstraite mise en place, elle ne permet pas de rendre compte de façon détaillée des variations du déroulement continu des comportements élémentaires.

Pour s'opposer à cette perspective, A. Seth [5] a montré que l'on pouvait construire un modèle de sélection de l'action qui ne passe pas par l'identification d'un répertoire de comportements élémentaires, mais qui agisse au niveau des interactions continues entre l'agent et son environnement en se basant sur des boucles sensori-motrices de très bas niveau paramétrées par des variables internes. Le travail présenté dans ce chapitre partage avec celui d'A. Seth [6] l'objectif d'obtenir par un algorithme génétique un système de sélection de l'action qui fonctionne dans un univers spatialement continu sans définir *a priori* un répertoire de comportements élémentaires ni un mécanisme explicite de sélection parmi ces comportements.

Notre contribution au problème de la sélection de l'action s'inscrit directement dans la lignée de celle d'A. Seth. Cependant, alors qu'il se focalise sur une liste de propriétés typiques de la sélection de l'action que possède son système, notre propos est de mettre en évidence la nécessité de disposer d'une dynamique interne pour résoudre efficacement de tels problèmes.

Pour arriver à ce résultat, nous commençons par décrire un problème élémentaire à la section 7.2. Notre stratégie pour résoudre ce problème consiste à étendre le formalisme de F. Flacher en introduisant des variables susceptibles de réaliser le type de dynamique que nous recherchons, puis à étendre l'algorithme génétique précédent pour qu'il engendre les associations pertinentes entre les variables introduites et les fonctions d'attraction et de répulsion du modèle précédent.

[1] [Lorenz, 1950, Tinbergen, 1951, Baerends, 1976]
[2] [Matarić, 1994a]
[3] [Tinbergen, 1951, Brooks, 1986, Tyrrell, 1993]
[4] [Maes, 1990, Decugis & Ferber, 1998]
[5] [Seth, 1998]
[6] [Seth, 1998]

7.2 Un environnement de simulation

Afin de représenter un problème élémentaire de sélection de l'action, nous avons développé un environnement de simulation dans lequel nous pouvons faire évoluer un mouton dans un environnement contenant un point d'eau, une zone couverte d'herbe et une bergerie, comme il apparaît sur la figure 7.1, page 145. Par ailleurs, nous associons à notre agent des variables essentielles (la faim, la soif et le sommeil) qui fournissent un modèle simpliste de métabolisme de l'agent et influencent sa durée de vie. Le problème de ce mouton consiste à se comporter de façon à maximiser sa durée de vie en régulant au mieux sa faim, sa soif et son sommeil.

Variable	Satisfaction	Sinon	Seuil	Condition de satisfaction
Faim	-0.03	0.01	1.8	Etre dans l'herbe
Soif	-0.2	0.02	1.0	Etre au point d'eau
Sommeil	-0.02	0.01	1.5	Etre dans la bergerie

TAB. 7.1 – **Evolution des variables essentielles. Par exemple, la faim diminue de 0.03 par unité de temps lorsque le mouton se trouve dans l'herbe ; sinon, elle augmente de 0.01.**

Le tableau 7.1 donne les lois d'évolution des variables essentielles du mouton. Nous représentons la dynamique de variation de ces variables en associant à chacune d'elles deux états définissant la variation de la variable lorsque l'agent se trouve dans ces états. Les transitions entre les états sont assurées par des événements relatifs au comportement et à la situation de l'agent. Par exemple, lorsqu'un agent se trouve près d'un point d'eau et boit, la variable « soif » passe d'un état dans lequel la soif augmentait lentement à un état dans lequel elle diminue rapidement, jusqu'à ce que l'agent quitte le point d'eau.

À chacune des variables essentielles sont associées des seuils définissant une zone de confort. L'agent dispose d'une vitalité initiale fixée arbitrairement à 10. À chaque fois qu'une variable essentielle dépasse son seuil, la vitalité est diminuée de l'amplitude du dépassement divisée par 100. Afin de borner la durée de vie de l'agent, on ajoute une variable essentielle qui ne fait qu'augmenter, ce qui réduit la vitalité une fois le seuil de cette variable dépassé. Un calcul simple montre que la durée de vie de l'agent ne peut excéder 4000 pas de temps.

7.3 Extension du formalisme

Notre approche de ce problème consiste à étendre le formalisme proposé par F. Flacher que nous avons présenté au chapitre précédent, de façon à gérer la présence de variables essentielles. Dans les contrôleurs engendrés par GACS, tous les paramètres qui interviennent dans la génération des forces appliquées à l'agent sont des constantes, si bien que les variations de l'action de l'agent qui en résulte à un instant donné ne dépendent que de ses perceptions à ce seul instant. Sur le problème de sélection de l'action défini à la section 7.2, un contrôleur de ce type ne peut pas donner de résultats satisfaisants. En effet, on peut montrer qu'un agent dirigé par un tel contrôleur dans un environnement où tous les objets pertinents sont statiques ne peut que converger vers un point de l'environnement d'où il ne bouge plus ou diverger à l'infini. Or de tels comportements ne permettent pas de satisfaire à la fois la faim, la soif et le sommeil.

Pour remédier à ce problème, il faut introduire une dynamique interne dans le contrôleur qui définit le comportement de l'agent, en intégrant des variables dans la construction des forces qui l'influencent.

Les variables essentielles associées à l'agent sont dotées d'une dynamique propre influencée indirectement par le comportement de l'agent. Par exemple, la soif est une variable qui augmente quel que soit le comportement de l'agent, sauf bien sûr lorsqu'il boit. Telles qu'elles sont définies, ces variables essentielles ne font pas partie de

la solution comportementale de l'agent pour survivre dans son environnement, mais du problème qui consiste à assurer sa survie. Le but de l'agent est de survivre le plus longtemps possible en adoptant un comportement qui lui permette de réguler la dynamique de ses variables essentielles de telle façon qu'elles ne dépassent pas les seuils qui réduisent sa vitalité.

Dans le cas de l'application que nous avons définie, certains points pertinents dans l'environnement de l'agent sont directement liés à la satisfaction des besoins élémentaires représentés par les variables essentielles de l'agent. Par exemple, le point d'eau est d'autant plus pertinent pour l'agent qu'il a plus soif. Par conséquent, il apparaît judicieux de relier le poids associé aux différentes forces d'attraction et de répulsion aux variables essentielles de l'agent.

Afin que l'agent puisse décider de son action en fonction de la valeur de ses variables essentielles, nous ajoutons donc au formalisme de GACS la possibilité que les gains G_i évoluent en fonction de ces valeurs.

Dans le cas que nous avons exposé dans l'étude présentée ici, la relation que nous établissons entre la valeur des gains et celle des variables essentielles est triviale ; la valeur de certains gains est directement égale à la valeur de certaines variables essentielles. À noter que nous pourrions souhaiter introduire des fonctions plus complexes que la fonction identité entre ces deux éléments. Par ailleurs, le choix optimal des associations entre certaines variables et certains gains est confié au même algorithme génétique qui règle déjà les autres paramètres des forces d'attraction / répulsion qui définissent les contrôleurs. En fin de compte, l'extension du formalisme de GACS s'avère très limitée.

À la place du gain G_i, un chromosome contient alors les données suivantes :

- un booléen B_i, qui indique si le coefficient de l'attracteur est donné par un gain réel constant ou par une variable,
- V_i, le numéro de la variable essentielle qui sera choisie si B_i vaut vrai,
- G_i, le coefficient réel codant le gain de l'attracteur si B_i vaut faux.

7.4 Étude expérimentale

7.4.1 Résultats

L'algorithme génétique qui permet de faire évoluer les champs de potentiel a été lancé de nombreuses fois avec des populations initiales aléatoires composées de 100 individus, chaque individu étant évalué à partir de 10 situations initiales.

Dans tous les essais que nous avons réalisés avec des populations initiales différentes, les contrôleurs des meilleurs individus obtenus sur 1000 générations survivent entre 2000 et 2700 pas de temps. Ils résolvent donc correctement le problème qui leur est posé, même s'ils n'atteignent pas la performance optimale.

Ces contrôleurs sont tous différents, mais, qualitativement, le comportement observé est similaire. Nous avons choisi d'étudier de plus près deux individus dont le contrôleur est significatif. Les trajectoires produites par ces deux individus au sein de l'environnement de simulation apparaissent sur la figure 7.1.

Les moutons se rendent alternativement au point d'eau, dans la zone herbeuse et à la bergerie, ce qui n'est jamais le cas avec des contrôleurs dont tous les gains sont des constanttes. Dans la section qui suit, nous présentons les deux contrôleurs correspondants et nous analysons leur fonctionnement.

7.4.2 Fonctionnement de deux contrôleurs dynamiques

Le contrôleur C_{2315} permet au mouton de survivre 2315 pas de temps. Sur la figure 7.2 apparaissent l'ensemble des fonctions d'attraction/répulsion qui le constituent ainsi que les gains et les points d'intérêt associés. On constate

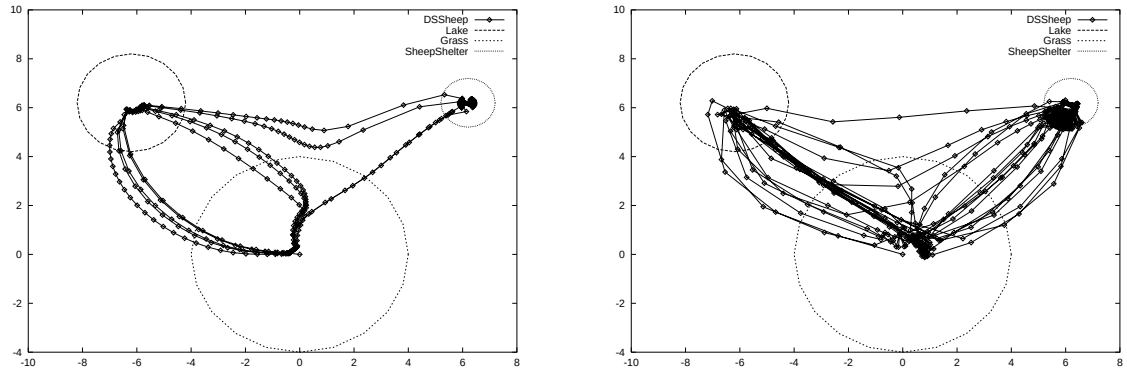


FIG. 7.1 – **Environnement de simulation et trajectoires de deux agents dirigés par les contrôleurs C_{2315} et C_{2631}.**

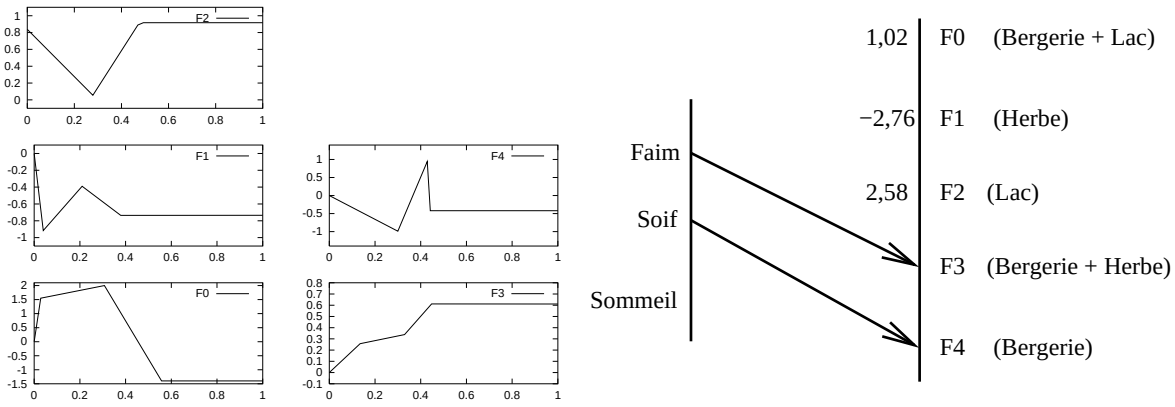


FIG. 7.2 – **Fonctions d'attraction/répulsion et structure du contrôleur C_{2315}**

que ce contrôleur est composé de deux parties :

- une partie statique constituée des fonctions F0 à F2 auxquelles sont associés des gains constants, qui produit un champ de potentiel statique,
- une partie dynamique (F3 et F4) associée à la faim et la soif, qui varient au cours du temps.

Les deux principales surprises qu'il nous réserve sont le fait que la variable essentielle correspondant au sommeil n'est pas prise en compte et le fait que la soif est associée à la bergerie et non au point d'eau.

Afin de comprendre qualitativement le fonctionnement de ce contrôleur, nous faisons apparaître sur la figure 7.3 les champs de potentiel engendrés par ses différentes composantes. En observant attentivement ces figures, nous pouvons proposer une explication qualitative du comportement de l'agent qui découpe ce comportement en phases successives, en ayant à l'esprit le fait que les phases en question n'ont d'existence que dans l'explication et non dans le fonctionnement effectif du système.

Comme il apparaît sur la figure 7.3a, si l'agent est soumis exclusivement au champ statique, il tend à se rendre soit vers la zone herbeuse, auquel cas il se dirige ensuite vers le point d'eau, soit directement au point d'eau. Au début d'une simulation, les variables essentielles du mouton sont initialisées à 0. Le champ auquel il est soumis correspond donc au champ statique et il se rend au point d'eau, éventuellement via la zone herbeuse. Sa soif devient rapidement négative, si bien qu'il est attiré par la bergerie. En effet, il apparaît que le champ engendré par la fonction

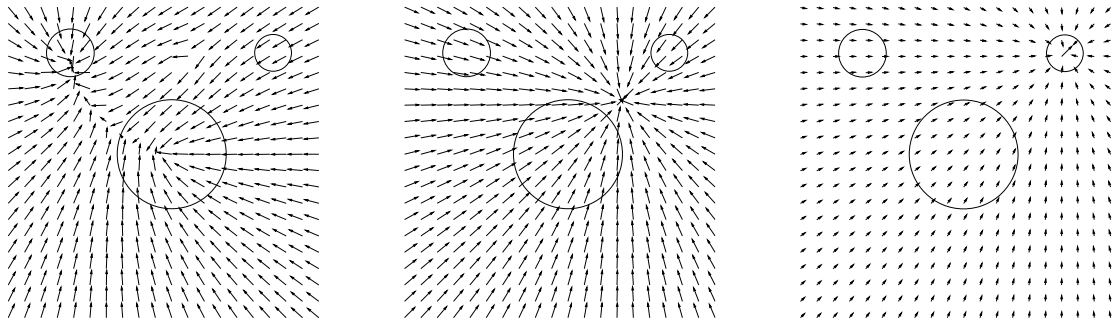


FIG. 7.3 – **Champs engendrés par le contrôleur : a) partie statique ; b) partie associée à la faim ; c) partie engendrée par une soif négative**

associée à la soif, lorsque celle-ci est négative, est orienté vers la bergerie (figure 7.3c). Le mouton reste dans la bergerie jusqu'à ce que la soif s'annule. Grâce au champ statique, il tend alors à retourner vers la zone herbeuse. Le champ associé à la faim fait apparaître un attracteur situé à mi-chemin entre la bergerie et la zone herbeuse (figure 7.3b). Il est donc crucial que le champ statique ramène le mouton à la zone herbeuse avant que le champ associé à la faim ne le piège dans l'attracteur associé. Une fois dans la zone herbeuse, sa faim diminue et il se rend à nouveau vers le point d'eau.

Un tel comportement peut être vu comme résultant du déclenchement de comportements élémentaires successifs qui consistent à aller boire, aller dormir et aller manger. De façon qualitative, on peut dire que le mouton va dormir non pas quand il a sommeil, mais quand il n'a ni faim, ni soif.

Là où définir des comportements élémentaires nous aurait permis de fixer de façon stricte l'enchaînement de tels comportements, il apparaît que le comportement d'un agent dirigé par notre système résulte d'un subtil équilibre entre la dynamique d'évolution des variables essentielles et le positionnement de l'agent dans son environnement.

Il apparaît donc que l'on peut rendre compte du fonctionnement d'un contrôleur en décrivant de façon qualitative « l'algorithme » qu'il réalise, alors même que son fonctionnement est dynamique et continu. Toutefois, il faut reconnaître que certains contrôleurs efficaces s'avèrent plus difficiles à analyser que C_{2315}. Ainsi, un contrôleur plus performant, C_{2631}, fait apparaître 17 fonctions d'attraction/répulsion. Nous donnons un aperçu des champs de potentiel que ce contrôleur engendre sur la figure 7.4.

La compréhension du contrôleur C_{2631} est rendue plus complexe par le fait que les centres des champs d'attraction et de répulsion de la figure 7.4 ne font pas apparaître des bassins d'attraction dirigeant clairement l'agent vers l'eau, l'herbe ou la bergerie. Pour comprendre comment le contrôleur C_{2631} permet à l'agent de survivre, il faut superposer ces différents champs pour un grand nombre de combinaisons des valeurs des variables essentielles de l'agent. Cela signifie que la position de l'agent dans son environnement à un instant donné et la valeur des variables essentielles de cet agent au même instant dépendent étroitement l'un de l'autre.

On peut assimiler la dynamique sensori-motrice de l'agent au comportement qu'il adopterait s'il était dirigé par un champ de vecteurs statique. En effet, c'est le comportement qu'il adopte sans dynamique interne, donc lorsque son comportement est dirigé uniquement par ses perceptions immédiates. Par ailleurs, la dynamique interne de l'agent est la dynamique qui résulte des variations de ses variables essentielles. On peut alors re-interpréter la difficulté à comprendre le contrôleur C_{2631} en énonçant que le couplage entre la dynamique sensori-motrice et la dynamique interne de l'agent doit être plus étroit pour le contrôleur C_{2631} que pour le contrôleur C_{2315}.

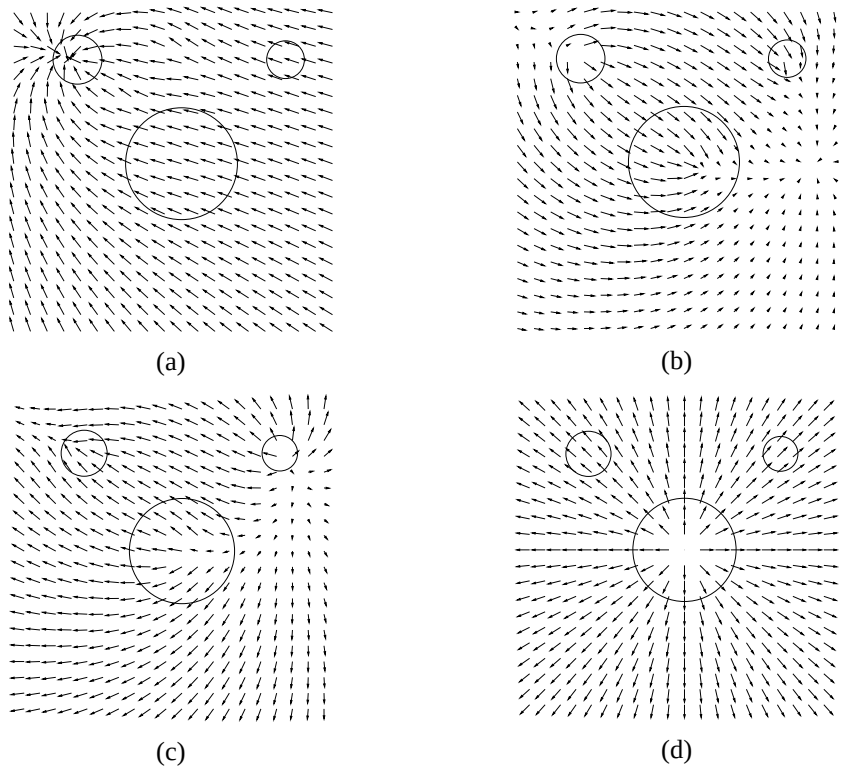


FIG. 7.4 – **Usage des variables essentielles pour C_{2631} et champs engendrés par le contrôleur C_{2631} : a) champ statique ; b) faim ; c) soif ; d) sommeil**

7.5 Discussion

L'originalité de notre approche, comme celle de Seth, vis-à-vis de modèles plus classiques de la sélection de l'action (cf. [7]), est qu'elle nous permet de ne pas définir a priori un répertoire de comportements élémentaires entre lesquels l'agent doit choisir. Comme nous l'avons montré précédemment [8], un avantage majeur de cette approche est sa simplicité de mise en œuvre. Définir un répertoire d'actions élémentaires suppose un effort de conception préalable à la réalisation effective, puis un effort de mise au point de ces comportements élémentaires dont notre approche permet de faire l'économie. Le recours à des comportements élémentaires présuppose en outre de faire appel à des mécanismes qui assurent le séquencement correct de l'activation de ces comportements. Avec notre méthode basée sur les champs de potentiel, un tel séquencement est tout simplement sans objet. L'ensemble des attractions et des répulsions est actif à tout instant, c'est simplement l'intensité de ces forces qui est modulée en fonction du contexte.

L'explication du fonctionnement du contrôleur C_{2315} met en évidence la capacité de notre approche à trouver des solutions efficaces et originales à un problème de déplacement dans un espace. Là où une décomposition fonctionnelle classique aurait suggéré d'associer la soif au point d'eau, la faim à la zone herbeuse et le sommeil à la bergerie, la solution exhibée par le contrôleur C_{2315} fait l'économie de la variable liée au sommeil en associant la soif au point d'eau et en associant au sommeil un état défini par « ni faim, ni soif ». Par ailleurs, il faut signaler que toutes nos tentatives pour coder à la main la solution intuitive issue de la décomposition fonctionnelle se sont

[7] [Tyrrell, 1993]
[8] [Flacher & Sigaud, 2003]

avérées infructueuses. L'explication ci-dessus ne vaut que pour le contrôleur C_{2315}, mais d'autres contrôleurs font apparaître des solutions analogues, dans lesquelles c'est souvent le sommeil qui n'est pas utilisé.

La diversité des solutions que nous avons obtenues sur différents problèmes de déplacement permet de mettre en évidence la robustesse des méthodes à base de champs de potentiel face à de tels problèmes. Grâce à cette robustesse, l'algorithme génétique parvient invariablement à trouver des solutions originales, efficaces et parfois inattendues, là où une conception manuelle s'est par ailleurs avérée difficile. La diversité de ces solutions nous semble à mettre au crédit de la possibilité de ne pas définir des comportements élémentaires qui résulte de l'approche dynamique, dans la mesure où les éléments de conception a priori qu'imposerait le recours à un répertoire de comportements élémentaires aurait pour effet de restreindre l'espace des solutions possibles.

Enfin, si nous avons montré à la section 7.4.2 qu'il était possible de fournir une explication qualitative du comportement de certains agents sans mobiliser les outils mathématiques sophistiqués dédiés à l'analyse générale des systèmes dynamiques, cette explication reste limitée. L'observation attentive de l'évolution des paramètres des simulations fait apparaître de nombreux phénomènes spécifiques dont l'explication ci-dessus ne rend pas compte. Notamment, le recours aux outils mathématiques de la théorie des systèmes dynamiques deviendra indispensable pour analyser plus précisément le couplage entre la dynamique sensori-motrice et la dynamique interne des agents. En particulier, identifier une mesure de l'intensité du couplage entre ces deux dynamiques nous semble nécessaire pour mieux comprendre le fonctionnement des différents contrôleurs.

7.6 Conclusion

Comme nous l'avons dit en introduction, le travail sur la sélection de l'action en présence de variables essentielles que nous avons présenté dans ce chapitre constitue le point de départ d'une thématique que nous jugeons importante, mais pour laquelle nous n'avons pas déployé de moyens de recherche spécifiques à l'heure actuelle.

Nous avons mentionné à la section 7.3 le fait que, dans le cadre du formalisme présenté ici, une famille de contrôleurs statiques échoue nécessairement si les points auxquels s'intéresse l'agent sont immobiles. Au contraire, si nous ajoutons des variables internes à cette famille de contrôleurs, alors un algorithme génétique est capable de sélectionner une solution satisfaisante. D'une façon plus générale, le recours à des variables internes semble indispensable pour résoudre efficacement tous les problèmes non stationnaires, que ce caractère non stationnaire provienne de l'existence de variables internes ou de la présence d'autres agents dans l'environnement.

Parmi les questions qu'il nous semble nécessaire d'aborder, il faut mentionner la prise en compte des interactions entre la perception et l'évolution des variables essentielles. Ainsi, la perception d'un aliment savoureux peut faire augmenter brusquement l'envie de manger d'un agent, alors que la perception d'un prédateur a un effet certain sur son degré d'anxiété.

Par ailleurs, la façon dont nous avons constitué une dynamique interne dans les travaux présentés ici est encore très limitée. En effet, la dynamique interne du contrôleur est calquée directement sur celle des variables essentielles du métabolisme, qui sont définies de l'extérieur par les automates qui nous servent à représenter l'évolution de la faim, de la soif ou du sommeil. La dynamique interne de nos contrôleurs ne dispose donc pas de toutes les propriétés des dynamiques auto-engendrées dont E. Daucé [9] se fait l'avocat.

Il faudra à l'avenir s'efforcer de remplacer la fonction identité qui relie les variables internes aux variables essentielles par des fonctions plus complexes susceptibles d'exhiber une dynamique auto-engendrée.

Par ailleurs, les travaux que nous avons présentés dans ce chapitre et le précédent ouvrent la voie à des recherches visant à étudier les méthodes à base de champs de force, à l'aide des outils conceptuels fournis par la

[9] [Daucé, 2002]

théorie des systèmes dynamiques. Nous espérons trouver à l'avenir l'occasion de nous engager plus franchement dans cette voie que nous n'avons fait qu'effleurer et qui devrait nous permettre de rationaliser l'usage que nous faisons des algorithmes génétiques afin de gagner en efficacité. En particulier, sur le plan méthodologique, il faudra se doter des outils – notamment graphiques – permettant à un concepteur de comprendre pourquoi et comment le contrôleur fonctionne.

Au chapitre suivant, nous présentons plutôt un premier effort de recherche visant à intégrer dans un cadre commun une partie des travaux présentés dans les chapitres précédents.

Chapitre 8

Apprentissage par population

« Ca m'a donné des idées de réfléchir ! »

B. Girard, le 10/07/2000

Dans les chapitres précédents, nous avons présenté des systèmes qui reposent exclusivement sur un algorithme génétique pour traiter chacun une classe de problèmes particuliers. D'un côté, ATNoSFERES s'appuie sur un mécanisme d'expression génétique à base de pile pour trouver des automates de contrôle efficaces dans le cadre de problèmes non markoviens. De l'autre, GACS s'appuie sur un algorithme génétique multi-critères pour engendrer des champs de force réalisant la coordination spatiale d'un ensemble d'agents dans des univers continus. Ces deux systèmes ont fait la preuve de leur capacité à trouver des solutions efficaces, mais la principale faiblesse des algorithmes génétiques sur lesquels ils reposent est une lenteur de convergence qui peut être rédhibitoire face à des problèmes de grandes dimensions.

Dans ce chapitre, nous allons examiner une solution générique à ce problème de lenteur. Cette solution consiste à remplacer l'algorithme génétique par un mécanisme issu d'une famille que nous avons choisie d'appeler « apprentissage par population » et qui s'inspire directement des architectures de style Michigan inventée pour les systèmes de classeurs. Nous commençons par prendre du recul pour présenter notre modèle générique au sein de mécanismes d'apprentissage par population et indiquer les travaux apparentés. Ensuite, nous nous efforçons de mettre en évidence les principales caractéristiques de notre modèle vis-à-vis des algorithmes évolutionnistes et des algorithmes d'apprentissage par renforcement. Enfin, nous indiquons comment il se décline dans le cadre des problèmes abordés respectivement par ATNoSFERES et GACS. Cela nous conduit à mettre en évidence quelques faiblesses de l'approche qu'il faudra dépasser par la suite.

8.1 Évolution et apprentissage

Les deux principaux mécanismes adaptatifs sur lesquels nous nous sommes appuyé jusqu'à présent dans ce document sont les algorithmes évolutionnistes et l'apprentissage par renforcement. Les premiers requièrent pour fonctionner :

- que l'on définisse une bonne fonction de *fitness*, apte à représenter fidèlement le problème que l'on cherche à résoudre et à guider l'évolution de la population en direction des meilleures solutions ;
- que l'on définisse un bon codage, capable de représenter une famille de solutions efficacement, de façon à faciliter l'exploration de cette famille à la recherche des solutions les plus satisfaisantes ;

– que l'on règle divers paramètres pour éviter une convergence prématurée ou au contraire une convergence trop lente de la population.

Si ces conditions sont remplies avec succès, les algorithmes évolutionnistes constituent une famille d'algorithmes d'optimisation très génériques et très robustes, dans la mesure où ils permettent d'obtenir de « bonnes » solutions sans faire trop d'hypothèses spécifiques sur le problème auquel on les applique. La contrepartie de cette généricité est une certaine lenteur de convergence, que nous avons déplorée dans le cadre de comparaisons de performance entre ATNoSFERES et les systèmes de classeurs, et qui reste préoccupante dans le cas de GACS.

De leur côté, les méthodes d'apprentissage par renforcement constituent une famille d'algorithmes sans doute moins génériques et moins robustes, car un certain nombre d'hypothèses plus fortes sont requises pour qu'ils s'appliquent. En revanche, en extrayant une information plus riche et plus structurée des interactions entre des agents et leur environnement, ces méthodes convergent généralement plus vite et sont dotées de propriétés supplémentaires, comme la capacité à modifier un comportement en cours d'exécution lorsque les sources de récompenses changent. Il y a donc tout lieu de chercher à combiner les deux classes de mécanismes à la recherche d'une solution qui présente les avantages des uns et des autres, et les inconvénients d'aucun.

Dans ce chapitre, nous allons nous intéresser à des classes de combinaisons particulières dans lesquelles le contrôle d'un agent est assuré par une population de solutions partielles.

8.2 Contrôle par population et adaptation

Il existe de nombreuses approches adaptatives à base d'évolution ou d'apprentissage, voire d'une combinaison des deux, dans lesquelles le contrôle d'un agent est assurés par une entité unique. C'est le cas, par exemple, pour les réseaux de neurones formels. Vis-à-vis de ces approches, les systèmes de classeurs se caractérisent par le fait que le contrôle de l'agent est assuré par un ensemble d'individus qui sont candidats à s'échanger le contrôle de l'agent les uns aux autres dans des circonstances bien définies, si bien que le contrôle est distribué sur une population.

Parmi les systèmes de classeurs, il existe diverses approches qui se distinguent par la façon dont elles combinent des mécanismes d'apprentissage par renforcement et des mécanismes d'évolution.

– Les approches de type Pittsburgh reposent exclusivement sur l'évolution. Dans cette approche, un algorithme génétique s'applique sur une population de plusieurs ensembles de classeurs pour sélectionner la population de classeurs qui résout bien le problème auquel elle est confrontée. L'ensemble des classeurs est donc considéré comme un tout indissociable du point de vue de l'évolution et de l'adaptation. Ce modèle ne propose donc aucune amélioration par rapport aux architectures d'ATNoSFERES et GACS dont nous avons critiqué la lenteur.
– Certains systèmes de classeurs à anticipation comme YACS et MACS reposent exclusivement sur l'apprentissage par renforcement.
– Comme nous l'avons déjà expliqué au chapitre 3, les approches de type Michigan se distinguent des approches de type Pittsburgh par le fait que les individus en compétition sur lesquels opère la sélection sont des classeurs et non plus des ensembles de classeurs. La capacité de chaque classeur à prendre le contrôle de l'agent est un paramètre dont la valeur est réglée par un mécanisme d'apprentissage par renforcement.
– L'évolution lamarckienne [1] est une forme de combinaison dans laquelle un individu obtenu par évolution réalise un apprentissage au cours de sa vie dont l'effet est propagé directement dans son génome, et c'est ce génome modifié qui est transmis à d'éventuels descendants. L'un des premiers systèmes de classeurs à

[1] [Yoshii et al., 1995]

faire appel à ce mécanisme, est SAMUEL [2]. SAMUEL relève de l'approche Pittsburgh – l'unité de sélection de l'algorithme génétique est une population de classeurs et non un classeur – mais il inclut des mécanismes d'apprentissage. En effet, la force des classeurs individuels est ajustée par apprentissage, et les opérateurs de modification des classeurs réalisent des heuristiques qui s'appuient directement sur l'expérience des agents. Au lieu d'appliquer simplement mutation et croisement, SAMUEL utilise aussi des opérateurs de généralisation, spécialisation et couverture qui s'apparentent à ceux que nous avons mis au point pour YACS et MACS. Toutes ces modifications sur la force et le contenu des classeurs, qui ont lieu en cours d'expérience, sont transmises à la génération suivante, d'où le caractère lamarckien.

– L'effet Baldwin [3] est le résultat d'une autre forme de combinaison, dans laquelle un individu obtenu par évolution réalise un apprentissage au cours de sa vie, mais l'effet de cet apprentissage n'est pas propagé dans le génome. L'interaction tient au fait que, la *fitness* étant mesurée sur des individus qui apprennent, ce sont les individus qui ont bien appris qui ont le plus de chance de transmettre leur génome à la génération future. À notre connaissance, le mécanisme n'a jamais été mis en œuvre dans le cadre de système de classeurs, ni dans aucun autre type de système d'apprentissage mettant en œuvre une population. Pour faire la différence entre une architecture abstraite de type Michigan et un système susceptible de bénéficier de l'effet Baldwin, il faut distinguer au sein de chaque individu entre un génome qui reste fixe durant les évolutions de l'agent et une entité de contrôle qui peut être modifiée par apprentissage dans le même laps de temps. Dans le cadre des systèmes de classeurs, la transformation d'un génome en instance de contrôle et réciproquement est triviale, si bien qu'il n'est pas certain que se priver des bénéfices de l'apprentissage dans les génomes de la génération suivante soit intéressant. Faute de travaux auxquels faire référence sur ce point, nous ne développerons pas plus loin cette possibilité dans ce document.
– Il faut aussi mentionner l'« approche parisienne » de P. Collet et ses collaborateurs [4]. Ces auteurs partent du même constat que nous d'une inefficacité relative des algorithmes évolutionnistes et de la volonté de se rapprocher d'une approche de type Michigan pour établir leur proposition. Ils disposent eux aussi d'une population d'individus qui fournissent une solution partielle à un problème mais, contrairement à ce qui se passe dans une approche Michigan, la *fitness* de ces individus n'est pas calculée à l'aide d'un algorithme d'apprentissage par renforcement. En effet, des traits spécifiques de leur problème leur permettent de définir une fonction d'évaluation locale *ad hoc*, très dépendante du problème. L'approche ne peut donc se généraliser qu'à des problèmes qui disposent du même type de propriété sur la possibilité d'une évaluation locale des individus.

Notre ambition dans ce chapitre est de discuter des avantages et des inconvénients de ces différentes formes de combinaison, mais nous souhaitons mener cette discussion dans un cadre aussi général que possible, qui fasse abstraction du formalisme de représentation spécifique des systèmes de classeurs. Pour ce faire, nous commençons par définir un modèle générique commun, que nous appelons « apprentissage par population », et qui doit nous permettre de discuter des différentes formes d'interaction entre évolution et apprentissage quand le contrôle d'un agent est confié à une population d'entités.

[2] [Grefenstette, 1991]
[3] [Hinton & Nowlan, 1987]
[4] [Collet et al., 2000]

8.3 Une famille générique de modèles

La spécification du mécanisme que nous appelons « apprentissage par population » consiste à généraliser l'idée d'une combinaison entre algorithme génétique et apprentissage par renforcement en faisant abstraction du formalisme de représentation des classeurs. Plutôt que des classeurs, un mécanisme d'apprentissage par population fait simplement appel à une population d'individus instanciés avec divers formalismes – automates, champs de forces, classeurs, ... – pour réaliser des systèmes dédiés à des classes de problèmes spécifiques. Dans ce qui suit, nous allons donner des spécifications de plus en plus précises de mécanismes d'apprentissage par population en nous abstenant de faire des hypothèses sur le formalisme de représentation des individus. Faire apparaître les niveaux d'abstraction successifs permet de faire ressortir les principaux concepts en jeu, mais aussi de définir les couches les plus génériques d'un futur logiciel regroupant les principales techniques qui relèvent de l'apprentissage par population.

8.3.1 Une racine commune

Nous commençons par une spécification très abstraite qui s'applique à l'ensemble des combinaisons que nous avons envisagées dans la section précédente. Les propriétés que nous retenons au niveau le plus global sont les suivantes :

- le contrôle d'un agent est assuré par une population d'individus, de taille fixe ou variable, bornée ou non ;
- chaque individu spécifie un contexte bien défini dans lequel il est susceptible de prendre le contrôle de l'agent ;
- à chaque pas de temps, une action est déterminée soit en donnant le contrôle à un individu unique, soit en considérant des groupes d'individus qui proposent la même action ;
- si plusieurs individus ou groupes d'individus proposant la même action sont en compétition pour la prise de contrôle de l'agent, un choix se fait sur la base d'une comparaison entre des évaluations locales de chaque individu ou groupe d'individus.

À ce stade on peut déjà distinguer les approches de type Pittsburgh, dans lesquelles il n'y a ni mécanisme d'apprentissage ni évolution séparée des individus, mais une évolution globale de la population, de toutes les autres approches, dans lesquelles il y a des mécanismes de compétition interne à la population. À présent, si nous voulons comparer les différentes formes de combinaison dans lesquelles les individus entrent en compétition, nous devons ajouter une contrainte sur les mécanismes d'évaluation et sur la description des individus. Pour rester très général, nous décidons de doter les individus de deux mesures distinctes, appelées « qualité estimée » et « *fitness* ». Nous noterons Q_i la qualité estimée d'un individu I_i et F_i sa *fitness*. Les deux sections qui suivent donnent les spécifications abstraites respectives de systèmes dans lesquels les individus évoluent et des systèmes dans lesquels les individus apprennent, les deux catégories n'étant évidemment pas incompatibles.

8.3.2 Evolution des individus

Pour caractériser les mécanismes dans lesquels les individus peuvent entrer en compétition au sein de la population, on peut ajouter les spécifications suivantes :

- la population évolue au cours du temps. Certains individus sont supprimés, d'autres sont créés. Cette évolution peut être le fait d'opérateurs génétiques, mais aussi d'opérateurs plus spécialisés tels que l'opérateur de couverture que l'on trouve dans la plupart des systèmes de classeurs, ou bien des opérateurs de généralisation ou spécialisation, par exemple.

– chaque individu I_i est évalué localement sur la base de sa contribution au comportement global de l'agent ;
– chaque individu I_i tient à jour une *fitness* F_i qui reflète sa capacité à survivre dans le cadre d'une compétition avec les autres individus.

8.3.3 Apprentissage des individus

Nous avons indiqué que tous les individus I_i étaient dotés d'une qualité estimée Q_i qui reflète leur capacité à prendre le contrôle de l'agent à un instant donné. On peut noter que dans le cas d'une approche de type Pittsburgh, ce qu'on appelle « qualité estimée » est fixé par le génome et évolue au gré des mutations successives. Dans le cas de l'approche parisienne, cette qualité est calculée par une évaluation locale *ad hoc*. Dans tous les autres cas, la qualité estimée est mise à jour par un mécanisme d'apprentissage par renforcement. Pour couvrir tous ces derniers cas, nous pouvons donc ajouter les spécifications suivantes :

– l'agent reçoit des signaux de renforcement à un niveau global [5] ;
– chaque individu I_i tient à jour une qualité Q_i qui exprime sa propension à prendre le contrôle et reflète la récompense que l'agent est susceptible d'obtenir sur le long terme si c'est effectivement I_i qui prend le contrôle de l'agent. Pour qu'un mécanisme générique d'apprentissage par population dispose effectivement de toutes les propriétés d'un système d'apprentissage par renforcement, le calcul de la qualité estimée des membres de la population doit s'effectuer de telle façon que cette estimation converge vers la qualité optimale au sens de Bellman. Nous allons voir que cette propriété n'est pas toujours facile à assurer pour différentes instances de systèmes d'apprentissage par population.

8.3.4 Familles d'opérateurs

Une troisième source de distinctions entre différentes formes de combinaisons d'apprentissage et d'évolution tient à la nature des opérateurs employés pour la modification des individus. En effet, on peut distinguer :

– les opérateurs génétiques de mutation et de croisement, employés dans toutes les approches impliquant une évolution génétique de la population, ce qui exclut YACS et MACS ;
– les opérateurs de couverture, qui existent dans la plupart des systèmes qui relèvent de l'approche Michigan ainsi que dans SAMUEL ;
– des opérateurs plus spécialisés tels que la généralisation et la spécialisation, que l'on retrouve dans SAMUEL, dans ALECSYS [8], dans ACS, YACS et MACS, ainsi que d'autres opérateurs spécifiques dans CSF-C [9], CSF-C2 [10], etc.

Un certain nombre de ces opérateurs sont spécifiques des systèmes de classeurs et seront difficiles à transposer dans un cadre d'apprentissage par population reposant sur un formalisme plus général.

8.3.5 Interactions

Les niveaux de spécification successifs que nous venons de poser permettent à présent de faire ressortir les mécanismes d'interaction entre évolution et apprentissage.

[5] Rien ne nous impose à ce stade de choisir entre les problèmes « à pas de temps unique » [6], dans lesquels les décisions successives de l'agent sont indépendantes les unes des autres et les problèmes « à pas de temps multiples » [7], dans lesquels l'agent doit réaliser une séquence d'actions qui font évoluer son état. Cependant, compte tenu des objectifs de notre programme de recherche, le second cadre nous intéresse plus que le premier.

[8] [Dorigo, 1993]

[9] [Robertson & Riolo, 1988]

[10] [Riolo, 1991]

– Dans le cas des approches Pittsburgh et de l'approche parisienne, il n'y a pas de mécanisme d'apprentissage, donc pas d'interaction.
– Dans le cas de SAMUEL, il n'y a pas de compétition entre les classeurs, par contre on trouve un apprentissage de la qualité estimée et des opérateurs spécifiques qui font évoluer la population de classeurs en fonction de l'expérience de l'agent.
– Dans le cas des approches Michigan, l'examen de la littérature sur les systèmes de classeurs nous conduit à envisager deux modes d'interaction entre la *fitness* F_i et la qualité estimée Q_i. Dans ZCS, Q_i est utilisé à la fois pour décider si un classeur mérite d'être conservé au sein de la population et pour décider quel est le classeur qui prend le contrôle dans une situation donnée. Donc $F_i = Q_i$ à tout instant. Dans XCS, en revanche, Q_i n'est utilisé que pour le choix de l'action. La *fitness* est égale à la capacité du classeur à prédire correctement sa récompense sur le long terme. Nous distinguerons donc deux types de mécanismes d'apprentissage par population relevant de l'approche Michigan, ceux dont la sélection est basée sur la qualité estimée, et ceux dont la sélection est basée sur la précision de la prédiction de la qualité.

Il ressort de cette première énumération que l'essentiel des interactions intéressantes entre évolution et apprentissage sont à rechercher du côté de l'approche Michigan. Ce sont donc essentiellement ces approches, débarrassées du formalisme des classeurs, dont nous allons discuter les avantages et inconvénients dans la suite. Nous retenons toutefois l'idée que certains opérateurs spécifiques peuvent donner à l'évolution un caractère lamarckien qui débouche sur des propriétés intéressantes, et que de tels opérateurs ne sont pas incompatibles avec les modèles issus de l'approche Michigan.

8.3.6 Travaux apparentés

Nous ne sommes pas les seuls à tenter d'abstraire les mécanismes généraux d'interaction entre apprentissage et évolution que l'on trouve dans les systèmes de classeurs vis-à-vis du formalisme des classeurs. P.-L. Lanzi a déjà proposé de remplacer les classeurs par des *S-expressions*, sortes d'atomes logiques liant une variable à une valeur, ce qui permet d'obtenir un système manipulant des règles de premier ordre [11]. De même, les membres de l'équipe de L. Bull [12] ont proposé de remplacer des classeurs par des réseaux de neurones. Cela leur permet de bénéficier des capacités de généralisation et de résistance au bruit de ces réseaux, notamment pour les appliquer dans un contexte robotique [13]. De son côté, O. Heguy propose dans sa thèse [14] une généralisation des systèmes de classeurs qui consiste à remplacer l'expression du génome d'un classeur sous la forme d'une chaîne de bits par une forme plus générale pouvant inclure des bits, des entiers, des réels ou des intervalles. En conséquence, son système dispose d'un formalisme capable d'exprimer simultanément la plupart des extensions connues des systèmes de classeurs standards. Cependant, aucune de ces tentatives successives n'a débouché sur le degré d'abstraction que nous nous efforçons d'atteindre dans ce chapitre.

[11] [Lanzi, 1999, Lanzi & Perrucci, 1999]
[12] [Bull & O'Hara, 2002]
[13] [Hurst & Bull, 2004]
[14] [Heguy, 2003]

8.4 Discussion des modèles

8.4.1 Réflexion sur l'approche Michigan

Afin de bien comprendre ce que la combinaison entre apprentissage et évolution réalisée par une architecture de type Michigan apporte vis-à-vis d'un algorithme évolutionniste et vis-à-vis d'un algorithme d'apprentissage par renforcement, il est intéressant de comparer, d'une part, un système de classeurs sans généralisation à un système d'apprentissage par renforcement tabulaire standard et, d'autre part, ce même système de classeurs simplifié à une architecture de type Pittsburgh reposant exclusivement sur un algorithme génétique.

Dans le cadre de la première comparaison, les choses sont relativement simples. Un système d'apprentissage par renforcement tabulaire construit une table exhaustive associant une qualité à tous les couples (situation, action) possibles dans l'environnement. Il s'efforce de faire converger ces qualités de façon à ce que choisir dans toute situation l'action de meilleure qualité permette d'adopter un comportement globalement optimal. Vis-à-vis de ce fonctionnement, un système de classeurs sans capacité de généralisation peut être vu comme une heuristique consistant à ne développer qu'une partie de la table, de façon à rechercher la politique optimale dans cette partie exclusivement. En effet, chaque classeur est l'équivalent d'une entrée dans la table, la qualité qui lui est associée converge vers la qualité optimale comme indiqué précédemment et l'algorithme génétique fait évoluer la population des entrées de la table prise en compte, de façon à couvrir au mieux la politique optimale. Si la taille maximale de la population est inférieure à la taille de la table, alors il faut choisir les entrées les plus pertinentes.

De façon annexe à notre discussion, on voit au passage qu'ajouter un mécanisme de généralisation permet d'assurer une meilleure couverture de la table à moindre frais, puisqu'un même classeur comportant des # permet de couvrir plusieurs entrées simultanément. Le problème résolu par un système de classeurs consiste alors à trouver les bonnes généralisations pour couvrir au mieux une partie pertinente de la table, compte tenu de la contrainte sur la taille de la population. C'est cette vision que propose P.-L. Lanzi [15] lorsqu'il énonce qu'un système de classeurs est la combinaison d'un système d'apprentissage par renforcement avec une capacité de généralisation. Toutefois, les interactions entre la gestion de la population et l'apprentissage ne s'arrêtent pas là. En effet, dès lors qu'un classeur possède au moins un #, il couvre au moins deux entrées de la table. Par conséquent, appliquer un algorithme d'apprentissage sur ce classeur revient à mettre à jour la qualité d'au moins deux entrées de la table d'un coup, ce qui permet aussi éventuellement d'accélérer la convergence de l'apprentissage. Pour que l'apprentissage soit effectivement accéléré, il faut que la mise à jour soit la bonne pour toutes les entrées, ce qui implique que toutes les entrées de la table partagent la même qualité du point de vue de la politique optimale. L'apprentissage est donc accéléré sous réserve que les classeurs portent les bonnes généralisations, si bien qu'apparaît une interaction forte entre la problématique de la généralisation et celle de l'apprentissage.

Pour ce qui est de la comparaison entre un système de classeurs sans généralisation et un algorithme évolutionniste pur, les choses sont plus complexes car il y a différents critères de sélection possibles pour l'algorithme génétique dans les systèmes de classeurs, qui conduisent à différentes formes de combinaison avec l'apprentissage par renforcement. Néanmoins, dans toutes les variantes connues, la *fitness* F_i d'un classeur est liée directement ou indirectement à la qualité estimée Q_i calculée par un algorithme d'apprentissage par renforcement. Si l'on considère l'apprentissage par population en tant qu'approche évolutionniste, l'apport des mécanismes d'apprentissage a donc trait à la définition de la fonction d'évaluation sur laquelle opère la sélection.

Dans les approches de type Michigan, chaque individu dispose de sa fonction d'évaluation partielle en fonction des récompenses reçues au cours du temps par la population globale. Le rôle des algorithmes d'apprentissage par renforcement est de diffuser la récompense globale au sein de la population en fonction de la part que chacun a pris

[15] [Lanzi, 2002]

dans l'obtention de ces récompenses globales au cours du déroulement de l'action. Grâce au mécanisme de diffusion de récompenses, l'approche de Michigan peut donc être vue comme dotant l'algorithme évolutionniste d'une fonction d'évaluation très spécifique, particulièrement adaptée aux problèmes faisant apparaître des phénomènes de récompense retardée consécutive à des séquences d'actions.

L'exemple de l'approche parisienne évoqué plus haut [16] montre qu'une évaluation locale reposant sur des mécanismes d'apprentissage par renforcement n'est pas la seule solution. On peut en effet s'abstenir de toute interaction avec un algorithme d'apprentissage, mais on sort alors du cadre des architectures de type Michigan.

Pour évaluer l'apport de l'approche Michigan par rapport à un algorithme évolutionniste simple, la discussion que nous venons de mener permet de faire ressortir les points suivants.

• Tout d'abord, l'approche reste fondée sur un algorithme évolutionniste dont la spécificité principale est de reposer sur une fonction d'évaluation très particulière. Comme dans tout algorithme évolutionniste, le moteur de l'amélioration de la performance reste la proposition de nouveaux individus obtenus par recombinaison aveugle d'individus existants. Si chaque individu est défini par un grand ensemble de paramètres, alors la convergence vers des individus performants peut être très lente, compte tenu du caractère aléatoire des améliorations. De ce point de vue, un algorithme reposant sur des opérateurs lamarckiens tels qu'en propose SAMUEL, par exemple, apparaît plus efficace. En effet, c'est un apprentissage qui guide la modification des paramètres et les nouveaux individus sont créés avec des paramètres dont la valeur a déjà été affinée par cet apprentissage. Par contre, il peut se produire des phénomènes de convergence prématurée.

• Dans le cadre d'une approche de type Michigan sans opérateur lamarckien, l'évolution de chaque individu reste lente, si le nombre de paramètres à faire évoluer sur cet individu est élevé. Toutefois, chaque individu ne représentant qu'une partie de la solution au problème global résolu par l'architecture, son nombre de paramètres reste généralement faible vis-à-vis du nombre de paramètres total qu'il faudrait faire évoluer pour obtenir une solution complète dans le cadre d'une approche de type Pittsburgh. Dans le cadre d'une approche de type Michigan, on fait évoluer un ensemble de solutions partielles en parallèle, ce qui permet de casser la combinatoire que devrait affronter une évolution globale.

• Dans le cas des architectures de type Michigan, les individus se partagent le contrôle de l'agent, chacun prenant le contrôle dans des circonstances bien définies. Par conséquent, la contribution de chaque individu doit être évaluée en fonction du comportement global de l'agent. Toute la difficulté consiste alors à faire en sorte que l'adaptation locale des individus conduise à une amélioration de la performance globale. Pour que l'approche Michigan soit meilleure à coup sûr qu'un algorithme génétique, reste donc à faire en sorte que l'évaluation locale de chaque individu soit correctement distribuée en fonction du comportement global et des récompenses reçues.

Lorsqu'il s'agit de résoudre un problème de comportement dans lequel l'hypothèse de Markov est vérifiée, le problème des interactions entre la performance locale des individus et la performance globale du comportement de l'agent se résout simplement. En effet, nous avons vu au chapitre 2 que, sous réserve que le problème soit markovien, les algorithmes de programmation dynamique et d'apprentissage par renforcement reposent sur le principe selon lequel choisir localement l'action de qualité optimale pour chaque état permet d'obtenir le comportement globalement optimal.

Dans les architectures de type Michigan, on s'appuie donc sur ce principe en attribuant à chaque individu une qualité locale qui se déduit de la valeur du comportement global grâce à un algorithme d'apprentissage par renforcement.

[16] [Collet et al., 2000]

Malheureusement, ce mécanisme est à double tranchant. En effet, si la qualité estimée a eu le temps de converger vers une valeur proche de la qualité optimale, alors le fait de disposer d'une qualité par classeur fournit à l'algorithme génétique une information précise sur sa contribution à la qualité du comportement global de l'agent. Dans ce cas, l'algorithme a toutes les chances de converger efficacement. Cependant, on voit bien que, ces qualités étant estimées, si l'estimation est mauvaise faute d'avoir convergé suffisamment tôt, l'algorithme sera mal guidé et peut donc nuire à la progression en supprimant des bons individus, au lieu de la favoriser en supprimant les mauvais individus. Vis-à-vis d'un algorithme génétique qui s'appuie exclusivement sur une signal d'évaluation externe qui n'est jamais erroné, on voit que cette solution n'est meilleure qu'à condition de bien maîtriser la gestion des interactions entre l'apprentissage et l'évolution, ce qui ne va pas de soi. Par conséquent, l'architecture d'ensemble s'avère fragile.

Il se dégage de cette discussion préalable l'idée que, s'il y a de bonnes raisons d'attendre du recours à un mécanisme d'apprentissage par population une augmentation de performance vis-à-vis d'un algorithme évolutionniste, cette amélioration est assortie d'un certain nombre de conditions qui ne sont pas encore élucidées sur le plan théorique, et qui demandent donc à être examinées par des moyens empiriques.

Aux sections 8.5 et 8.6, nous proposerons deux instanciations du modèle générique d'apprentissage par population destinées à tester en pratique la validité de notre approche. Le lecteur verra que nous ne disposons pas encore de résultats sur ces questions.

8.4.2 Apprentissage par population et problèmes non markoviens

Nous venons de mettre en évidence le caractère crucial du respect de l'hypothèse de Markov pour le bon fonctionnement des mécanismes d'apprentissage par population. En effet, si cette hypothèse n'est pas vérifiée, les qualités estimées ont de bonnes chances de ne pas converger, ce qui peut ruiner à la fois l'apprentissage et l'évolution. Cette architecture doit donc être aménagée dans le cadre des problèmes qui ne vérifient pas l'hypothèse de Markov.

Au chapitre 5, nous avons comparé deux approches vis-à-vis de ce problème :

– d'une part, nous avons présenté quelques architectures de type Michigan dédiées à la résolution de problèmes non markoviens [17] ;
– d'autre part, nous avons proposé avec ATNoSFERES une approche de type Pittsburgh dédiée à cette même classe de problèmes.

Que ce soit dans le cas de l'approche Michigan ou dans celui de l'approche Pittsburgh, une condition du succès est que le formalisme de représentation du contrôleur des agents prenne en compte la présence d'états cachés. Par contre, dans une architecture de style Pittsburgh telle qu'ATNoSFERES, on peut appliquer un algorithme évolutionniste à celui qui fonctionnerait dans le cas markovien, tandis que dans les approches de style Michigan, il faut aussi adapter les algorithmes d'apprentissage. Ce point illustre une fois de plus la plus grande généricité et la plus grande robustesse des algorithmes évolutionnistes vis-à-vis des algorithmes d'apprentissage.

Chacune des deux approches, l'approche Pittsburgh et l'approche Michigan, dispose donc de ses forces et ses faiblesses. Comme n'ont pas manqué de le signaler S. Wilson et D. Goldberg [18], il semble donc intéressant de mettre au point une architecture qui combine les forces des approches Pittsburgh et Michigan tout en étant exempte de leurs faiblesses, et la résolution de problèmes non markoviens est un cadre privilégié pour de telles tentatives.

L'architecture de SAMUEL peut être vue comme un premier pas dans ce sens. J. Grefenstette, son auteur, proclame en effet que « SAMUEL représente une intégration des principales approches génétiques de l'apprentissage,

[17] [Cliff & Ross, 1994, Stolzmann, 1999, Tomlinson & Bull, 2000b, Tomlinson & Bull, 2000a, Lanzi & Wilson, 2000]
[18] [Wilson & Goldberg, 1989]

à savoir l'approche Michigan et l'approche Pittsburgh » [19]. Toutefois, comme nous l'avons exposé précédemment, SAMUEL repose en fait sur un algorithme génétique qui dispose de quelques opérateurs d'apprentissage lamarckien, ce qui diffère nettement de ce que réalise une architecture de type Michigan.

Les systèmes de classeurs corporatifs constituent d'autres tentatives plus conformes à l'esprit de ce que proposent S. Wilson et D. Goldberg. Dans ces approches, en effet, une corporation est un sous-ensemble de la population de classeurs au sein duquel les individus ont un destin commun et n'entrent pas en compétition. Cependant, nous avons vu au chapitre 5 que ce mécanisme de corporation est intrinsèquement limité parce qu'il ne peut exprimer que des chaînes linéaires de classeurs, ce qui peut être irrémédiablement sous optimal dans le cas de certains problèmes non markoviens.

Pour aller plus loin, il faut trouver le moyen de lier le destin de plusieurs individus de telle façon que le sous-ensemble qu'ils constituent soit organisé de façon plus riche qu'une simple chaîne linéaire. C'est exactement la solution que nous allons mettre en œuvre avec FACS, système dans lequel l'équivalent de plusieurs classeurs sont reliés entre eux pour former un automate partiel.

Les mécanismes qui opèrent au sein de la population d'automates partiels sont des mécanismes de type Michigan. Néanmoins, à force de lier des automates partiels entre eux, la population peut converger vers un automate unique qui assure le contrôle exhaustif de l'agent, auquel cas on se retrouve dans le cadre d'une approche de type Pittsburgh.

Le système FACS que nous allons présenter dans la section suivante constitue donc à notre avis une combinaison particulièrement prometteuse de l'approche Michigan et de l'approche Pittsburgh, combinaison dans laquelle les individus, qui sont des automates, peuvent être vus eux-mêmes comme des populations d'objets plus simples de type classeur, ce qui ouvre la voie à une approche hiérarchique que nous ne développerons pas dans le cadre de ce document.

8.5 Présentation de FACS

8.5.1 Introduction

Au chapitre 5, nous avons décrit ATNoSFERES, une architecture fondée sur un algorithme génétique à base de pile et capable d'engendrer des automates à états finis pour commander un agent dans un environnement non markovien. Dans le cadre d'une comparaison avec les systèmes de classeurs, nous avons montré que le formalisme des automates construits par ATNoSFERES était suffisamment riche pour exprimer l'équivalent du chaînage de classeurs aussi bien que la gestion explicite d'un état interne. En conséquence, les performances d'ATNoSFERES sont meilleures que celles des systèmes qui reposent sur ces deux types de mécanismes dans les environnements où l'autre type est plus adapté. En revanche, ne disposant pas des mécanismes d'apprentissage par population qui font la force des systèmes de classeurs, mais reposant au contraire sur une évolution aveugle, ATNoSFERES est beaucoup plus lent que ses concurrents à atteindre une performance équivalente [20].

Dans cette section, nous allons décrire de façon détaillée le système FACS, qui résulte de la transformation d'ATNoSFERES en un modèle conforme à notre modèle générique d'apprentissage par population.

La première modification à apporter provient de ce qu'ATNoSFERES élaborait un automate unique qui dirigeait l'agent en toute situation, alors que le modèle d'apprentissage par population fait appel à un ensemble d'individus qui se partagent le contrôle de l'agent en prenant le contrôle successivement dans les circonstances où ils peuvent

[19] [Grefenstette et al., 1990]

[20] voir la section 5.5, page 118.

être choisis. FACS va donc reposer sur une population d'automates.

Nous avons fait le choix de doter chaque automate d'un nœud « Start » d'où part un unique arc, dont la partie `[Condition]` doit être vérifiée pour que l'automate soit candidat à prendre le contrôle. En outre, conformément à notre modèle d'apprentissage par population, nous associons à chaque automate une enchère qui reflète sa propension à prendre le contrôle et une *fitness* dont la valeur conditionne sa survie au sein de la population.

Dans les sections qui suivent, nous allons décrire successivement le mécanisme de calcul des enchères et de partage du contrôle entre les automates, puis le mécanisme de calcul de *fitness* et de gestion de l'évolution de la population.

8.5.2 Prise de contrôle et calcul de l'enchère

Dans le cas des systèmes de classeurs standards, un classeur est un individu qui prend le contrôle dans un état donné, réalise une action, puis relâche aussitôt le contrôle dans l'état suivant. Dans le cas du critère de récompense dévalué sur un horizon infini, la présence d'un facteur de dévaluation γ garantit que la fonction de valeur associée à chaque état du problème est fonction de la valeur des récompenses associées à certains états du problème, et décroît en fonction de la distance à ces récompenses. Il en est de même pour la fonction de qualité associée aux couples (état, action).

En clair, pour que la qualité estimée converge vers la qualité optimale au sens de Bellman, il suffit que la qualité estimée du classeur soit égale à γ fois la qualité estimée du classeur qui prendra le contrôle au pas de temps suivant, ou bien à γ fois la récompense obtenue si une source de récompense est atteinte au pas de temps suivant. C'est sur ce principe que repose l'algorithme *Bucket Brigade* [21] dont l'équivalence avec l'algorithme du *Q-learning* a été démontrée par M. Dorigo et H. Bersini [22]. Donc si un individu I_i prend le contrôle à une distance supérieure ou égale à n pas d'une récompense R, sa qualité estimée doit converger vers $Q_i = \gamma^n R$.

Dans notre modèle d'apprentissage par population instancié par des automates, nous voulons assurer la même propriété via un mécanisme identique de propagation des qualités estimées. Nous faisons appel pour cela à un mécanisme à base d'enchères inspiré de celui proposé par R. Sun [23].

Toutefois, une difficulté supplémentaire est induite par le fait qu'un automate peut réaliser un nombre indéterminé d'actions et non une seule entre le moment où il prend le contrôle et le moment où il la relâche. La solution que nous proposons pour faire en sorte que la qualité estimée de chaque automate converge vers la qualité optimale au sens de Bellman en dépit de cette difficulté est la suivante.

Dans notre modèle, la compétition entre les automates pour prendre le contrôle a lieu à chaque pas de temps. Le système identifie tous les automates dont la partie `[Condition]` du premier arc est compatible avec la situation courante. Il ajoute à cet ensemble l'automate courant si celui-ci possède un arc dont la partie `[Condition]` du premier arc est compatible avec la situation courante à partir du nœud courant. L'ensemble résultant est appelé *match-set*, par analogie avec l'ensemble équivalent dans XCS [24]. Le système fait alors la somme des enchères de tous les automates qui proposent la même action. Lorsqu'il fonctionne en mode de pure exploitation, il choisit l'action pour laquelle cette somme est la plus élevée. Sinon, il peut utiliser l'une des nombreuses méthodes disponibles pour gérer le compromis entre exploration et exploitation. Notre choix s'est porté sur une méthode dite « ϵ-*greedy* » qui consiste à choisir avec une probabilité très faible ϵ un automate quelconque plutôt que l'automate de plus forte qualité

[21] [Holland, 1985]
[22] [Dorigo & Bersini, 1994]
[23] [Sun & Sessions, 2000]
[24] voir la section 3.1.3, page 55.

L'enchère E_i d'un automate A_i reflète la qualité estimée Q_i que nous avons définie dans le cadre de notre modèle d'apprentissage par population. Le mécanisme d'évolution des enchères est conçu de telle façon que l'enchère E_i converge vers la qualité optimale du problème pour l'état dans lequel l'automate prend le contrôle, ce qui correspond à la récompense attendue sur le long terme si l'agent suit une politique optimale après avoir réalisé l'action spécifiée par l'arc courant de l'automate sélectionné. Par exemple, si le facteur de dévaluation γ vaut 0.9, si la récompense finale vaut 1 et si l'automate prend le contrôle deux pas de temps avant d'atteindre cette récompense finale, on veut que son enchère converge vers $0.9 \times 0.9 \times 1 = 0.81$. Une façon de remplir cette exigence est d'appliquer à l'enchère un algorithme de différence temporelle [25].

Pour cela, nous associons à chaque automate P qui prend le contrôle son crédit local c_P, correspondant à la somme des récompenses amorties qu'il reçoit entre le moment t_i où il prend le contrôle et le moment t_f où il le rend [26].

$$c_P = \sum_{t=t_i}^{t=t_f} \gamma^{t-t_i}.r_t + \gamma^{t_f-t_i}.B_Q \tag{8.1}$$

où r_t est la récompense externe immédiate reçue à l'instant t et B_Q est l'enchère de l'automate qui prend le contrôle après P. Effectuer ce calcul suppose que, au moment où il prend le contrôle, chaque automate initialise un compteur qui mesure le nombre de pas de temps écoulés entre le moment où il prend le contrôle et le moment où il reçoit une récompense.

Ensuite, lorsque l'automate rend le contrôle, nous mettons à jour son enchère conformément à l'algorithme de prédiction de la différence temporelle [27] :

$$B_P(T+1) = B_P(T) + \beta.(c_P(T+1) - B_P(T)) \tag{8.2}$$

où β est le taux d'apprentissage [28]. Nous calculons alors l'erreur entre l'enchère précédente et le crédit local reçu :

$$\epsilon_B(T+1) = |c_P(T+1) - B_P(T)| \tag{8.3}$$

Pour déterminer la moyenne de l'erreur d'enchère de chaque automate sur le long terme, nous utilisons à nouveau l'algorithme de prédiction de la différence temporelle :

$$\epsilon_B(T+1) = \epsilon_B(T) + \beta'(|c_P(T+1) - B_P(T)| - \epsilon_B(T)) \tag{8.4}$$

Cette erreur nous servira de mesure de *fitness* pour appliquer un algorithme d'évolution par sélection.

L'enchère estimée de l'automate qui détient déjà le contrôle au pas de temps t_c est :

$$B_P(t_c) = \frac{B_P(t_i)}{\gamma^{t_c-t_i}} \tag{8.5}$$

où t_i est le pas de temps auquel il a pris le contrôle de l'agent.

Ce mécanisme doit garantir que, si les automates qui spécifient la politique optimale existent dans la population, ils prendront peu à peu le dessus sur les autres et le comportement de l'agent convergera vers un comportement optimal, comme ce serait le cas avec n'importe quel algorithme d'apprentissage par renforcement dans le cas markovien.

[25] [Sutton & Barto, 1998]

[26] Nous notons T et $T+1$ les pas successifs de prise de contrôle par les divers automates et t et $t+1$ les pas de temps élémentaires définissant la dynamique de l'environnement.

[27] [Sutton & Barto, 1998]

[28] *learning rate*

L'algorithme d'évolution par sélection devra donc garantir que les automates qui spécifient les couples (situation, action) optimaux ou les séquences optimales de tels couples dans le cas non markovien apparaissent bien dans la population.

8.5.3 L'algorithme d'évolution par sélection dans FACS

Initialisation et couverture

Dans la mesure où XCS est reconnu comme le système de classeurs de style Michigan le plus efficace à l'heure actuelle, il nous est apparu efficace de nous inspirer autant que possible du fonctionnement de ce système pour mettre au point l'algorithme d'évolution par sélection qui opère dans FACS. Cette source d'inspiration nous a conduit à faire les choix qui suivent.

La première partie de l'algorithme concerne l'initialisation de la population grâce à un opérateur de couverture.

1. L'algorithme commence avec une population initiale vide. C'est l'opérateur de couverture qui se charge de remplir la population avec N automates dont les arcs initiaux sont compatibles avec la situation courante. Ce procédé évite de créer une grande quantité d'automates dont l'arc initial ne sera jamais compatible avec aucune situation.
2. Le comportement de l'opérateur de couverture est le suivant. À chaque nouveau pas de temps, FACS construit le *match-set* correspondant à l'ensemble des automates qui peuvent être activés dans la situation courante. Si, dans cet ensemble, il existe certaines actions qui ne sont proposées par aucun automate, l'opérateur de couverture crée un ensemble de N automates dont l'arc initial est compatible avec la situation courante et qui propose l'action absente. La méthode employée consiste à engendrer des génomes aléatoirement, à en déduire des automates et à ne conserver que ceux qui sont compatibles avec la situation courante, puis à forcer le gène portant l'action de l'arc initial pour qu'il représente l'action manquante. Cette méthode inefficace est rendue nécessaire par la difficulté à calculer a priori un génome donnant un automate doté des propriétés attendues. Il apparaît que l'opérateur de couverture est beaucoup moins naturel dans le contexte de FACS que dans celui des systèmes de classeurs. Nous discuterons ce point à la section 8.5.4.
3. Les automates créés par l'opérateur de couverture sont initialisés avec une enchère qui vaut R_{max}, la plus grande récompense que l'on puisse rencontrer dans l'environnement, et une erreur nulle. Cette initialisation favorise l'exploration en poussant l'algorithme de prise de contrôle à favoriser les individus qui viennent d'être créés.

À ce stade, si l'agent explore tout l'environnement, on dispose pour tout couple (situation, action) de au moins N automates qui proposent cette action dans cette situation. Dans le cas d'un problème non markovien, rien ne garantit que les automates proposant les séquences d'actions optimales fassent partie de cette population. Par contre, si le problème est markovien, cette population doit suffire et l'apprentissage doit converger.

Dans le cas non markovien, il faut faire évoluer la population d'automates afin que les automates optimaux finissent par y figurer. Nous spécifions donc dans la section suivante un algorithme génétique qui permet de faire évoluer la population d'automates.

Évolution génétique

Là encore, nous suivons d'aussi près que possible les spécifications d'XCS, qui ont prouvé leur efficacité.

– La sélection génétique opère au sein de l'*action-set* courant. On appelle « *action-set* » courant l'ensemble des individus qui proposent l'action qui a été choisie par le système pour être appliquée dans l'environnement

de l'agent. Dès lors que l'*action-set* courant contient assez d'individus, ce qui arrive rapidement grâce à l'opérateur de couverture, l'algorithme élimine de la population les deux individus de l'*action-set* qui ont la *fitness* la plus faible. Il sélectionne ensuite deux individus parmi ceux qui ont les meilleures *fitness* à l'aide d'un opérateur de type « roue de la fortune[29] », et il engendre deux descendants de ces individus en appliquant des opérateurs classiques de mutation et de crossover. Les deux descendants sont ajoutés dans la population globale. Leur enchère initiale est la moyenne des enchères de leurs parents et leur erreur de prédiction initiale est nulle.

– Pour chaque individu, on compte le nombre de fois où il a été impliqué dans un *action-set* depuis sa création. L'opération de brassage génétique est lancée à chaque fois que la moyenne de ces nombres sur l'*action-set* courant est supérieur à un seuil, qui est un autre paramètre du système.

8.5.4 Discussion

Il sera nécessaire de comparer les performances d'un système tel que FACS d'une part à celles d'ATNoSFERES, et, d'autre part, à celles des systèmes de classeurs concurrents. En attendant des comparaisons empiriques, nous pouvons déjà faire ressortir le fait que, dans le cas des systèmes de classeurs, le passage du code génétique à l'architecture de contrôle est immédiat, chaque codon spécifiant un « *token* » des parties condition ou action des classeurs. Dans le cas de FACS, au contraire, ce passage est moins immédiat. En conséquence, l'écriture d'un opérateur de couverture est beaucoup moins naturel dans le cadre de FACS que dans celui des systèmes de classeurs.

Pour ce qui est de l'évaluation des performances, le système FACS que nous venons de décrire a donné lieu à une réalisation informatique très préliminaire, mais il est encore trop tôt pour présenter des résultats empiriques convaincants. Des premiers essais mettent en évidence le fait que les degrés de liberté, dans les détails de la spécification d'un système d'apprentissage par population, sont nombreux et les interactions entre mécanismes sont subtils, si bien qu'il est difficile de trouver une bonne combinaison qui donne rapidement de bonnes performances. Nous restons convaincu qu'une bonne combinaison existe, mais il faudra sans doute de nombreux travaux empiriques pour la trouver. Cela semble indiquer que le niveau d'abstraction auquel nous nous plaçons pour spécifier notre mécanisme d'apprentissage par population est trop élevé pour servir de guide méthodologique clair en vue de la construction d'un système performant. La conduite d'une réflexion sur ce thème trouve sa place dans notre programme de recherche à court et moyen terme.

Par ailleurs, il semble possible de généraliser l'approche hiérarchique mise en œuvre à deux niveaux dans FACS à des architectures dotées de davantage de niveaux, ce que nous comptons explorer à plus long terme.

8.6 Apprentissage de champs de forces

8.6.1 Procédé de transformation du modèle

Au chapitre 6, nous avons décrit GACS, une architecture fondée sur un algorithme génétique spécialisé dans la réalisation des problèmes multi-critères et capable d'engendrer des contrôleurs à base de champs de forces pour assurer la coordination spatiale d'un groupe d'agents dans un environnement continu. De même que nous avons modifié ATNoSFERES pour y introduire un mécanisme d'apprentissage par population, nous pouvons appliquer une transformation similaire à GACS.

La première opération à réaliser pour appliquer le modèle d'apprentissage par population à GACS consiste à transformer un contrôleur unique, qui propose un déplacement pour toute situation de l'agent, en une population

[29] *roulette wheel*

de contrôleurs partiels qui ne proposent un déplacement que dans un domaine restreint de l'espace des situations possibles pour l'agent.

En l'état actuel de GACS, le contrôleur de chaque agent est constitué d'un ensemble de forces d'attraction et/ou de répulsion, associées à des points d'intérêts, qui sont eux-mêmes des barycentres de points pertinents dans l'environnement de l'agent. Chaque force indique l'intensité avec laquelle l'agent est attiré ou repoussé par un point d'intérêt, pour toute distance entre l'agent et le point d'intérêt correspondant. L'intensité de la force est donnée par une fonction linéaire par morceaux [30].

Un procédé immédiat pour transformer un tel contrôleur universel en un ensemble de contrôleurs partiels consiste à commencer par voir un contrôleur engendré par GACS comme une collection de contrôleurs partiels dont les domaines de validité forment une partition de l'espace des distances aux points pertinents qui entourent l'agent.

En effet, si un contrôleur engendré par GACS définit k fonctions d'attraction/répulsion $\{F_1, ..., F_k\}$ qui sont linéaires par morceaux et si les morceaux de la fonction F_i sont définis par β_i intervalles $[x^i_n, x^i_{n+1}[$, pour n compris entre 0 et $\beta_i - 1$, alors on peut voir ce contrôleur comme une collection de $\prod_{i\in\{1,...,k\}} \beta_i$ contrôleurs partiels, chaque contrôleur proposant sur un domaine défini par une combinaison particulière de k intervalles une force globale résultant de la somme de k fonctions d'attraction/répulsion linéaires définies sur les segments correspondant aux morceaux appartenant au domaine de ce contrôleur.

Si l'on suppose fixé a priori l'ensemble des points d'intérêts auxquels est assujetti le contrôleur global de l'agent, un contrôleur partiel s'exprime sous la forme de deux éléments, comme il apparaît sur la figure 8.1 :

- un domaine de validité défini par autant d'intervalles qu'il y a de points d'intérêt ;
- une fonction de déplacement donnée par la combinaison de segments de fonctions d'attraction/répulsion linéaires définies sur ces intervalles.

La structure d'un tel contrôleur s'apparente à celle d'un classeur dont la partie `[Condition]` est constituée d'intervalles sur chacune des distances associées à chaque point d'intérêt et dont la partie `[Action]` propose un calcul effectif de force de déplacement en fonction de la distance de l'agent à chacun de ses points d'intérêt.

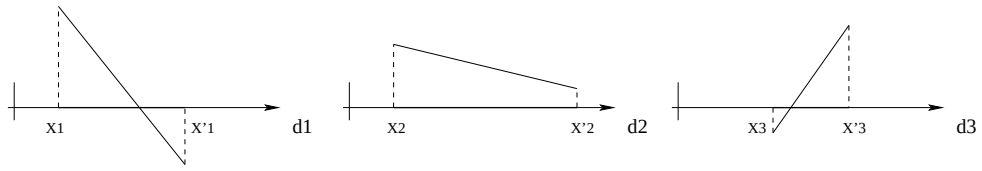


FIG. 8.1 – **Groupe de forces d'attraction/répulsion. Contrôleurs partiels vus comme des classeurs. La partie condition des contrôleurs est vérifiée si les distances $d1$, $d2$ et $d3$ aux points d'intérêt $P1$, $P2$ et $P3$ sont respectivement comprises dans les intervalles $[x_1, x'_1[$, $[x_2, x'_2[$ et $[x_3, x'_3[$. Pour une position donnée de l'agent, le déplacement proposé par le contrôleur se calcule comme dans le cadre de GACS, ce qui suppose d'associer à chaque point d'intérêt un gain qui est soit le même pour tous les contrôleurs, soit qui prend une valeur différente pour chaque contrôleur partiel.**

Comme il apparaît dans la légende de la figure 8.1, pour calculer l'action exactement comme cela est fait dans le cadre de GACS, il faut en outre associer à chaque point d'intérêt un gain qui intervient dans la combinaison linéaire des forces engendrées par chaque point d'intérêt. Ces gains peuvent soit être communs à tous les contrôleurs, soit prendre des valeurs spécifiques pour chaque contrôleur partiel. La seconde solution est plus souple mais elle implique une recherche parmi un nombre de paramètres plus grand que la première.

Une fois cette première étape de spécification franchie, on dispose d'un formalisme de représentation pour les

[30] voir la section 6.4.2, page 126.

individus qui se partagent la commande d'un agent. Dès lors, l'approche générale que nous appelons « apprentissage par population » peut s'appliquer directement. En effet, il suffit d'associer à chaque individu une mesure de qualité dont la valeur évolue conformément à un algorithme d'apprentissage par renforcement et une mesure de *fitness* qui peut être soit directement la qualité de l'individu si l'on choisit une approche de type ZCS, soit la précision d'une prédiction sur cette qualité si l'on choisit une approche de type XCS.

La principale différence avec l'approche proposée par GACS est que les individus évolueront indépendamment les uns des autres et non plus comme un tout, si bien que la collection de tous les individus ne formera certainement plus une partition de l'espace des distances aux points d'intérêt.

Avant de spécifier les algorithmes d'évolution et d'apprentissage permettant d'instancier l'apprentissage par population avec des champs de force, il convient de prendre du recul pour évaluer la pertinence d'une telle approche dans ce cadre précis.

8.6.2 Discussion

Si nous lui appliquons les algorithmes d'évolution caractéristiques de XCS, le système dont nous venons de présenter la spécification serait assez proche du système de classeurs XCSR [31] dont la partie `[Condition]` est vérifiée lorsque les variables sur lesquelles elle porte sont comprises dans les intervalles qu'elle spécifie. En effet, notre système et XCSR travaillent tous les deux sur un espace d'état continu découpé en intervalles.

La principale différence provient de ce que XCSR propose des actions discrètes, choisies parmi un répertoire limité et dont la valeur est fixée pour tout le domaine dans lequel un classeur est éligible, tandis que, dans notre système, l'action est continue et peut prendre des valeurs différentes au sein du même domaine. Plus précisément, dans le formalisme que nous proposons, l'action est calculée par une combinaison linéaire de valeurs prises par des fonction linéaires au point défini par l'état précis de l'agent, représenté par l'ensemble des distances à ses points d'intérêt. Ainsi, pour un problème faisant intervenir k points d'intérêt, l'action est définie par $4k + 1$ paramètres si le gain est partagé et $5k$ si chaque individu a son propre gain.

Par conséquent, dans le cadre d'une approche évolutionniste, là où XCSR doit choisir une action parmi n, notre système doit choisir une fonction définissant l'action en réglant entre $4k$ et $5k$ paramètres.

Or nous avons montré dans la section 8.4 que, si l'apprentissage par population fournit des fonctions d'évaluation capables de propager une récompense globale sur les différents individus de la population, il ne dispose d'aucun mécanisme spécifique susceptible de traiter efficacement le cas où le nombre de paramètres à régler au sein d'un individu est important.

Un système de classeurs tel que XCSR n'ayant jamais été appliqué à un problème de grande taille et faisant preuve de performances modestes sur des problèmes de taille plus restreinte, il est peu probable qu'un système reposant sur un mécanisme d'apprentissage par population comparable à celui du système XCSR soit capable de traiter efficacement le problème beaucoup plus complexe auquel nous envisageons de l'appliquer.

La situation générale est alors la suivante.

– Vis-à-vis d'un système de classeurs travaillant dans le domaine continu tel que XCSR, l'approche que nous avons défendue ici présente l'intérêt de proposer une action continue, qui varie en fonction de l'état précis de l'agent, ce qui limite les risques de problèmes de discontinuité aux bornes des intervalles spécifiés par les classeurs. En revanche, elle présente l'inconvénient d'imposer le réglage d'un plus grand nombre de paramètres, ce qui conduit à douter de sa capacité à converger en un temps raisonnable.
– Vis-à-vis de l'approche de type Pittsburgh telle qu'elle est mise en œuvre dans GACS, notre approche peut

[31] [Wilson, 2000]

se montrer plus rapide et plus efficace, mais elle risque de s'avérer moins robuste et difficile à régler, comme nous l'avons indiqué dans la section 8.4.

– En outre, il ne faut pas oublier que, pour disposer d'un algorithme doté de capacités équivalentes à celles de GACS, il faut ajouter un mécanisme externe capable de sélectionner des points d'intérêt et d'y associer des gains, si ces derniers ne sont pas associés à chaque contrôleur individuel. Cette partie peut être dévolue à un algorithme génétique externe, comme c'était déjà le cas dans le cadre de GACS, mais cela rajoute un second niveau d'évolution à gérer, donc de nouveaux problèmes de temps de convergence.
– Enfin, on peut reprocher à cette approche de ne pas bénéficier du gain en vitesse de convergence que procure l'apprentissage d'un modèle du monde. En effet, nous avons montré au chapitre 3 que l'apprentissage par renforcement direct est beaucoup plus lent que l'apprentissage par renforcement indirect, qui profite d'un modèle des transitions pour accélérer la convergence des fonctions de valeur ou de qualité.

La voie que nous avons proposée mériterait sans doute d'être explorée davantage pour voir où se situent les limites de ses capacités de convergence. Cela étant, compte tenu de la fragilité supposée de la gestion des interactions entre apprentissage et évolution et de la lenteur probable de la convergence vers une bonne solution, T. Gourdin, à qui nous l'avons proposée à titre de point de départ pour sa thèse, a décidé de ne pas suivre directement cette voie, pour lui opposer une autre solution dotée de qualités supplémentaires, que nous décrirons brièvement dans le prochain chapitre.

8.7 Conclusion

Notre proposition d'une approche adaptative générique mêlant apprentissage et évolution, que nous appelons « apprentissage par population », joue un rôle important dans notre programme de recherche. En effet, elle nous offre une première perspective d'intégration de plusieurs problématiques d'adaptation, auxquelles nous nous confrontons dans le cadre de la rédaction de ce mémoire. La réflexion que nous avons menée devrait déboucher sur la construction d'un logiciel permettant l'intégration de différentes approches adaptatives sur une base générique commune.

Il n'en reste pas moins que cet axe de recherche en est encore à un stade très préliminaire. D'une part, l'étude conceptuelle des différentes combinaisons que nous avons menées à la section 8.4 mérite certainement d'être étayée par un travail de formalisation permettant des comparaisons à un niveau plus théorique. D'autre part, nous n'avons pas encore mené les travaux de réalisation informatique et de validation empirique qui sont indispensables pour attester de l'utilité pratique de notre approche appliquée à d'autres entités que les classeurs.

En attendant de pousser plus loin la réflexion théorique et les travaux empiriques dans cette direction, nous allons présenter au chapitre suivant les deux dernières problématiques que nous abordons dans ce mémoire, à savoir l'apprentissage par renforcement continu et l'apprentissage de la coordination dans un cadre multi-agents.

Chapitre 9

Apprentissage de la coordination

> « De toutes façons, je ne fais pas beaucoup confiance à mon cerveau. »
>
> T. Gourdin, le 03/06/2003

Sous un même titre, « apprentissage de la coordination », ce chapitre rassemble deux problématiques complémentaires, toutes les deux réputées difficiles, et qui constituent la partie la plus prospective de notre programme de recherche.

Tout d'abord, pour traiter les problèmes de coordination spatiale sur lesquels nous nous sommes déjà étendu au chapitre 6, T. Gourdin se place résolument dans le cadre de l'apprentissage par renforcement en environnement continu. Sa recherche est destinée à assurer la navigation d'un agent dans son environnement en prenant en compte les déplacements de toutes les autres entités pertinentes autour de lui. Pour présenter ce travail, nous commençons par exposer rapidement ce que devient le formalisme des processus de décision markoviens dans le cadre continu et nous mettons en évidence les difficultés importantes que posent les problématiques associées. Puis nous situons au sein de la littérature du domaine l'approche choisie par T. Gourdin pour dépasser les problèmes exposés au chapitre précédent. Nous présentons alors l'état de notre réflexion sur ce thème.

Pour assurer à un plus haut niveau la coordination des déplacements de différents agents dirigés par les mécanismes de navigation de T. Gourdin, nous abordons dans un deuxième temps une problématique nouvelle, à savoir la coordination des actions d'un groupe d'agents pour des problèmes qui ne relèvent pas exclusivement de la navigation. Là aussi, nous commençons par un état de l'art succinct et une mise en évidence des difficultés que posent ces problèmes. Puis nous indiquons les principales pistes sur lesquelles T. Degris, qui commence une thèse sur ce sujet, s'est engagé.

9.1 Apprentissage par renforcement continu

9.1.1 Difficulté du cadre continu

L'apprentissage par renforcement en environnement à état, action et temps continus est clairement formalisé dans la thèse de R. Munos [1]. La thèse de R. Coulom [2] apporte sur le même domaine un regard complémentaire,

[1] [Munos, 1997]
[2] [Coulom, 2002]

mais plus appliqué. La présentation qui suit s'appuie sur une synthèse proposée par R. Munos [3].

Dans le domaine continu en espace, en action et en temps, on note $x(t) \in X$ l'état du système, sachant que l'espace d'état X est un sous-ensemble compact de $\mathbb{R}^d$, où d est le nombre de dimension de l'espace. On note $u(t)$ l'action du système et l'équation au premier ordre de la dynamique du système est de la forme :

$$\frac{dx(t)}{dt} = f(x(t), u(t))$$

où f est une fonction quelconque. Plutôt que de considérer un horizon infini, on s'intéresse au cas où le système réalise des évolutions entre le temps $t = 0$ et le temps $t = \tau$. Le problème du contrôle optimal d'un tel système, consiste à trouver l'action optimale $u^*(t)$ qui maximise le terme

$$J(x(t), u(t)) = \int_0^\tau \gamma^t r(x(t), u(t))dt + \gamma^\tau r_b(x(\tau))$$

où $r(x(t), u(t))$ est la récompense reçue au temps t et $r_b(x)$ la récompense obtenue en fin de trajectoire. La fonction J est l'équivalent continu de la somme pondérée des récompenses que l'on considère dans le cas discret.

Pour résoudre un tel problème, il faut adapter la méthode de la programmation dynamique, qui consiste à introduire une fonction de valeur V pour tout état. Alors que les solutions au problème de la programmation dynamique dans le cas discret vérifient l'équation de Bellmann, dans le cas continu elles vérifient l'équation de Hamilton-Jacobi-Bellmann :

$$V(x) \ln \gamma + \max_{u \in U} [\Delta V(x).f(x, u) + r(x, u)] = 0 \tag{9.1}$$

où $\Delta V(x)$ est le gradient de la fonction de valeur en x. On montre alors que la politique optimale s'exprime de la façon suivante :

$$\pi(x) \in \arg\max_{u \in U} [\Delta V(x).f(x, u) + r(x, u)] = 0 \tag{9.2}$$

Les ordinateurs ne sont pas adaptés pour traiter directement des variables continues et encore moins un temps continu. Ils représentent les nombres réels avec une précision limitée et découpent l'écoulement du temps en intervalles dont la brièveté maximale dépend de la fréquence d'horloge de l'ordinateur. Au mieux, on peut approximer la résolution des équations continues ci-dessus en échantillonnant le temps et les valeurs des variables d'état et d'action. Si l'on procède ainsi, apprendre directement un modèle de la fonction de valeur ou de la fonction de qualité reste difficile à réaliser en pratique, car le nombre potentiel d'états et d'actions du système pour lesquels il faut stocker une valeur ou une qualité est énorme. Si l'on fait abstraction des limites de représentation de l'ordinateur, ce stockage suppose en théorie une table de dimension infinie.

Face à cette situation, il faut se doter d'une représentation plus compacte de l'espace d'états et/ou d'actions. Il existe pour cela deux grandes classes de solutions. La première consiste à se doter d'une famille de fonctions paramétrables sur l'espace d'états et/ou d'actions, et à approximer la fonction de valeur ou qualité en réalisant une combinaison de ces fonctions et en leur associant les bons paramètres. L'intérêt de cette méthode est que le nombre de paramètres à régler peut être petit devant la dimension de l'espace considéré. Les supports les plus couramment utilisés dans cette perspective sont les réseaux de neurones [4], mais on peut aussi utiliser les fonctions à base radiale [5], par exemple.

L'autre solution consiste à discrétiser l'espace continu en y découpant un ensemble de domaines. Cette méthode est un cas particulier de la précédente, dans lequel les fonctions d'approximation sont définies sur des supports

[3] [Munos & Moore, 2001]

[4] [Coulom, 2002]

[5] *Radial Basis Functions*, ou RBF [6]

restreints à des domaines particuliers. Plutôt que d'attribuer une valeur unique à chaque domaine, on peut aussi combiner les deux méthodes en associant des fonctions d'approximation à des domaines spécifiques, le cas de la valeur unique pour chaque domaine se ramenant au cas particulier où la fonction d'approximation est une constante sur le domaine en question.

Parmi les méthodes de discrétisation, on peut distinguer celles qui proposent un découpage régulier de l'espace d'états, telle l'architecture CMAC, inspirée d'un modèle du cervelet [7], et celles qui proposent un découpage en éléments de taille variable, l'idéal étant l'approche dite « adaptative » dans laquelle la taille des éléments s'adapte aux besoins de la qualité de l'approximation. Dans cette seconde catégorie, R. Munos propose une méthode de discrétisation efficace fondée sur la triangulation de Kuhn [8].

Quelle que soit la méthode de discrétisation, le nombre d'éléments simples qu'elle engendre croît de façon exponentielle en fonction du nombre de dimensions de l'espace d'états et/ou d'actions.

Dans tous les cas, que l'on ait recours à des fonctions d'approximation ou à une discrétisation, il apparaît un compromis à gérer entre la qualité de l'approximation que l'on veut atteindre et la taille de la représentation que l'on est prêt à manipuler.

Obtenir un bon modèle de la fonction de valeur ou de qualité dans le cadre continu est donc déjà très difficile lorsqu'on s'intéresse exclusivement à un problème de planification dans lequel le modèle des transitions du système et les sources de récompenses sont connus à l'avance. Tous ces problèmes ne font que prendre une acuité supplémentaire dans le cadre plus difficile de l'apprentissage, dans lequel transitions et récompenses doivent être découvertes par l'agent. En particulier, trouver une bonne discrétisation adaptative, qui donne davantage d'informations sur la fonction de valeur là où cette fonction accuse les variations les plus critiques, est beaucoup plus difficile lorsqu'on ne dispose pas a priori des informations qui permettent de calculer précisément cette fonction.

9.1.2 Solution envisagée

Face à la difficulté bien identifiée du cadre général de l'apprentissage par renforcement en environnement continu, l'approche méthodologique choisie par T. Gourdin consiste d'une part à se doter d'une architecture générique lui permettant de tester différentes classes de solutions pour les différents sous-problèmes auxquels il s'attaque et, d'autre part, à restreindre la portée de ses travaux au cas particulier de la navigation continue avec un modèle à base de champs de forces, ce qui lui permet de mettre en œuvre des solutions spécifiques.

L'architecture choisie par T. Gourdin est inspirée des architectures acteur-critique [9]. Ces architectures nous intéressent parce qu'elles impliquent très naturellement un découplage important entre la construction d'un modèle de la fonction de valeur par le critique et la recherche d'une politique optimale pour l'acteur.

Toutefois, il existe une différence importante entre l'architecture acteur-critique classique et l'architecture proposée par T. Gourdin. Dans l'architecture acteur-critique, le critique envoie à l'acteur un signal d'écart entre la récompense attendue et la récompense effectivement obtenue. L'acteur modifie alors la qualité qu'il attribue aux différentes actions possibles dans le contexte courant, ce qui peut le conduire par la suite à proposer une autre action dans le même contexte. Cette amélioration est donc strictement conforme au cadre de l'apprentissage par renforcement non supervisé. Dans la proposition de T. Gourdin que nous allons détailler, au contraire, le critique associe une qualité aux différentes actions possibles. Lorsque l'acteur propose une action qui n'est pas celle que le critique aurait choisie, le critique lui transmet cette action préférée afin que l'acteur s'applique des opérateurs qui tendent à rapprocher son action de l'action idéale. L'interaction entre le critique et l'acteur prend donc la forme

[7] [Albus, 1975]
[8] [Munos & Moore, 2001]
[9] voir la section 2, page 27.

d'un apprentissage supervisé, même si le critique lui-même réalise un apprentissage par renforcement. Nous parlerons donc de superviseur plutôt que de critique, mais il faut garder en tête le fait que le superviseur lui-même apprend par renforcement si bien que l'architecture d'ensemble ne réalise pas un apprentissage supervisé. Nous allons maintenant présenter les différents éléments de cette architecture avant de la discuter davantage.

9.1.3 Un formalisme pour l'acteur

S'il n'a pas retenu l'approche de l'apprentissage par population, T. Gourdin reste attaché à l'objectif général qui consiste à introduire des mécanismes d'apprentissage par renforcement dans la mise au point de contrôleurs de coordination spatiale tels qu'ils sont engendrés par la plate-forme GACS, afin d'accélérer la découverte de ces contrôleurs. Pour ce faire, il identifie la partie acteur de son architecture à un contrôleur tel que GACS est susceptible d'en engendrer [10].

Ainsi, son système disposera des principaux atouts dont est doté GACS pour la coordination spatiale, à savoir un formalisme souple et général qui s'adapte à un problème donné en optimisant la valeur de ses paramètres. La force du système que conçoit T. Gourdin doit provenir de ce que les paramètres évolueront de façon dirigée par des heuristiques liées aux mécanismes d'apprentissage, au lieu d'être soumis aux mécanismes d'essais et erreurs aveugles d'un algorithme évolutionniste.

Ce choix implique que l'état du système est défini par la position relative d'un ensemble de points pertinents vis-à-vis de l'agent. Cette position relative est représentée en coordonnées polaires, en trois dimensions. Il faut donc être conscient du fait que, si un contrôleur fait appel à k points pertinents, l'espace d'état qui est représenté sur la figure 9.1 est constitué de $3k$ dimensions. Il sera donc crucial d'être parcimonieux dans la définition des points pertinents, sachant que cette définition est effectuée à la main.

Par ailleurs, le formalisme de F. Flacher fait appel à une notion de point d'intérêt, un point d'intérêt étant défini comme un barycentre de points pertinents. Nous avons dit à la section 6.6, page 135, que cette spécificité introduisait de la redondance dans le formalisme, mais que cette redondance était favorable à la découverte par l'algorithme génétique de contrôleurs efficaces, car elle permet d'isoler des « blocs de construction » pertinents. Dans la mesure où nous n'aurons plus recours à un algorithme génétique, il faudra choisir sur la base d'études expérimentales entre continuer à utiliser ces barycentres ou bien associer des fonctions d'attraction et de répulsion directement aux points pertinents.

9.1.4 Une ébauche de superviseur

Le problème que doit résoudre le superviseur consiste à construire un modèle de la fonction de qualité pour les actions d'un acteur qui agit de façon continue dans un environnement continu. Se concentrent donc ici toutes les difficultés liées à la discrétisation que nous avons évoquées à la section 9.1.1.

Nous avons dit que, face à ces difficultés, il existe de nombreuses solutions qui relèvent soit de l'approximation, soit de la discrétisation. Au sein de ces solutions potentielles, celle qu'a retenu T. Gourdin fait partie des méthodes d'approximation dites « à grain grossier » [11] reposant sur une interpolation entre des points répartis dans l'espace d'état, en fonction de l'expérience précédente de l'agent.

Les travaux dont T. Gourdin s'inspire ont été menés par J.-C. Santamaria [12]. Il s'agit d'une adaptation de

[10] voir le chapitre 6.
[11] *coarse-coded approximators*
[12] [Santamaria et al., 1997]

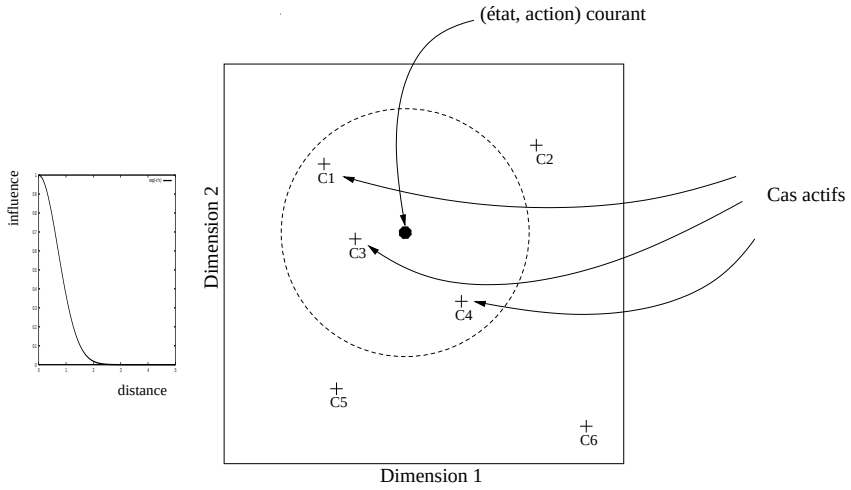


FIG. 9.1 – **Illustration du fonctionnement du superviseur sur un exemple à deux dimensions. Les points C1 à C6 désignent des états correspondant à des expériences précédentes de l'agent. Le superviseur stocke des informations sur la qualité des actions associées à ces points. Il n'ajoute de nouveaux points que si la distance au point le plus proche est supérieure à un seuil. Lorsque l'agent se trouve dans l'état courant et réalise l'action courante, les informations stockées aux points les plus proches (ici, C1, C3 et C4) sont mobilisées pour identifier l'action proposée par le superviseur. La contribution de chaque point à ce calcul dépend de sa distance à l'état courant, selon la fonction à noyau apparaissant sur la partie gauche de la figure. Le même calcul d'influence est effectué entre l'action courante et les actions stockées aux différents points qui participent au calcul.**

l'apprentissage à partir de cas [13] de C. Atkeson [14] au problème de l'apprentissage de la fonction de qualité. L'idée générale consiste à sauvegarder dans l'espace $S \times A$ des points correspondant aux couples (s, a) rencontrés par l'acteur. On estime ensuite la fonction de qualité en tout point de l'espace à partir des points déjà enregistrés et de fonctions à noyau qui modulent l'influence de chaque point en fonction de sa distance au point où se situe l'agent (voir la figure 9.1). L'apport de nouveaux points à partir desquels on estime la fonction de qualité est régulé par une contrainte globale sur la densité des points.

Dans le cas des algorithmes dénommés « à partir d'instances [15] », les points sont stockés dans un espace $S \times A$, et on stocke la qualité de l'action en question, tandis que dans le cas des algorithmes « à partir de cas [16] », ils sont stockés dans l'espace engendré par les états seulement et on stocke en chacun de ces points une valeur de qualité pour un ensemble fini d'actions couvrant l'espace.

Par ailleurs, afin d'estimer correctement la fonction de qualité aux points enregistrés, il faut un mécanisme de propagation de la qualité qui permette de résoudre les problèmes de récompense retardée. Pour assurer cette propagation, J.-C. Santamaria choisit de faire appel au mécanisme des traces d'éligibilité que nous avons décrit à la section 2.4.4, page 43. Ce mécanisme permet en effet de propager les qualités sur plusieurs pas de temps consécutifs, sans pour autant stocker explicitement la séquence des transitions effectuées par l'acteur, ce qui s'avère économique.

Il est trop tôt, à ce stade du travail de T. Gourdin, pour donner une description plus détaillée du fonctionnement des algorithmes qu'il met au point. Notons toutefois que, malgré les qualités certaines des algorithmes proposés par J.-C. Santamaria, un certain nombre de points posent encore problème :

[13] *case-based learning*
[14] [Atkeson, 1991]
[15] [McCallum et al., 1995]
[16] [Ram & Santamaria, 1997]

– Tout d'abord, si l'acteur est représenté avec le formalisme de GACS, l'état de cet acteur est défini dans l'espace engendré par les distances aux points d'intérêt auxquels le contrôleur de l'agent est sensible. Ces points d'intérêt peuvent être nombreux, donc le nombre de dimensions définissant l'espace d'états peut être grand. Estimer correctement la fonction de qualité dans un espace vaste sera certainement difficile.
– Ensuite, la solution « à partir de cas » de J.-C. Santamaria impose de stocker une valeur de qualité pour chaque action en chaque point. Stocker une qualité par action suppose de disposer d'un répertoire d'actions discrètes. Or nous travaillons dans un espace d'actions continu. Il faut donc soit discrétiser la représentation des actions dans le superviseur, en prenant en compte le fait que le nombre d'actions représentées doit être faible pour limiter des problèmes d'encombrement mémoire, soit renoncer à l'approche « à partir de cas » au profit de l'approche « à partir d'instances », dans laquelle la prise en compte d'une action continue apparaît plus naturelle. J.-C. Santamaria avance des résultats qui plaident en faveur des approches « à partir de cas » mais, dans le problème qui lui sert de référence, l'action se prête bien à une discrétisation simple. À l'inverse, D. Smart et L. Kaelbling [17] se font les avocats d'une approche « à partir d'instances », toujours dans un cadre où l'action se discrétise naturellement.

  La solution envisagée par T. Gourdin pour traiter le cas de l'action continue consistera à généraliser pour l'action l'approche adoptée pour l'état. Son algorithme stockera en chaque point les différentes actions entreprises avec leur qualité, en veillant à ce qu'un seuil de densité d'actions ne soit pas franchi, et la contribution de ces actions dans le calcul de la qualité de l'action courante dépendra des distances entre actions au travers d'une fonction à noyau.

  Nos travaux sur la question en sont encore à un stade trop préliminaire pour que l'ensemble des questions qu'il soulève puissent être tranchées.

9.1.5 Adaptation de l'acteur par la correction du superviseur

Nous avons expliqué que le superviseur impose à l'acteur un apprentissage de type supervisé. L'action proposée par l'acteur est corrigée au vu de celle proposée par le superviseur selon le mécanisme suivant. À chaque pas de temps, l'action réalisée par l'acteur s'exprime sous la forme d'un vecteur qui donne simultanément la direction et l'intensité du déplacement. De même, le superviseur propose à chaque pas de temps le vecteur qui correspond selon lui au déplacement optimal qu'aurait dû effectuer l'agent dans cette situation. Corriger l'acteur consiste donc à modifier la direction et l'intensité du déplacement en fonction de l'écart constaté avec la proposition du superviseur. Pour cela, T. Gourdin propose de faire appel à un ensemble d'opérateurs spécifiques au formalisme utilisé pour l'acteur.

Comme nous l'avons indiqué à la section 9.1.3, le déplacement de l'agent résulte de la combinaison linéaire d'un ensemble de forces élémentaires associées à des points d'intérêt. Pour modifier le déplacement global en fonction des indications du superviseur, il faut donc répartir la modification sur l'ensemble des forces élémentaires qui le définissent. T. Gourdin propose que cette répartition se fasse en fonction de la contribution de chacune des forces élémentaires au déplacement global. La modification appliquée à chaque force élémentaire est donc une portion de la modification globale, portion qui dépend de l'intensité relative de la force élémentaire. Reste alors à spécifier les opérateurs de modification des forces élémentaires.

La première famille d'opérateurs est destinée à prendre en charge une modification de la direction de la force. La seule façon de modifier la direction d'une force élémentaire consiste à déplacer le point d'intérêt auquel elle est rattachée. Chaque point d'intérêt étant défini comme le barycentre d'un certain nombre de points pertinents, il y a

[17] [Smart & Kaelbling, 2000, Smart & Kaelbling, 2002]

trois façons de déplacer un point d'intérêt, chacune de ces façons définissant un opérateur potentiel :

- on peut modifier les coefficients associés à chacun des points pertinents pris en compte dans le calcul du barycentre ;
- on peut ajouter un nouveau point pertinent à ceux qui sont pris en compte dans le calcul du barycentre ;
- on peut supprimer l'un de ces points pertinents.

Dans un premier temps, afin de valider l'algorithme progressivement, il semble raisonnable de considérer que la définition des points pertinents auxquels doit être rattaché chacun des opérateurs est fournie par un concepteur humain. On se contentera donc d'appliquer le premier opérateur pour réaliser une modification de direction.

La seconde famille d'opérateurs est destinée à prendre en charge une modification de l'intensité de la force. Cette fois, les opérateurs peuvent agir sur la définition des fonctions de sensibilité ou sur le gain qui leur est appliqué. Il apparaît alors quatre opérateurs possibles.

- ajouter un sommet à la fonction de sensibilité ;
- supprimer un sommet de la fonction de sensibilité ;
- déplacer un sommet de la fonction de sensibilité ;
- modifier le gain associé à la fonction de sensibilité.

Il faut noter que modifier le gain revient à déplacer l'ensemble des sommets d'une fonction en une seule opération.

À chaque pas de temps, on dispose donc de l'action choisie par l'acteur, de l'action préférée par le superviseur, et d'un ensemble d'opérateurs destinés à modifier la première de façon à ce qu'elle se rapproche de la seconde. L'ensemble des opérateurs est largement redondant, il existe en fait une infinité de façons de modifier l'acteur pour que l'action qu'il propose coïncide avec un vecteur défini. Il reste donc à définir une politique d'application des opérateurs qui ait le mérite à la fois de faire converger l'acteur vers un comportement spécifié par le superviseur et de limiter autant que possible la complexité du jeu de paramètres qui définit l'acteur. Ces recherches en sont pour l'instant à un stade préliminaire, nous n'avons pas encore de réponse toute faite à cette dernière question, ni de résultats empiriques nous permettant de statuer sur la faisabilité de l'approche que nous venons de présenter.

9.1.6 Une validation progressive

La section 9.1.4 a permis de souligner les difficultés nombreuses que posent la mise au point d'un superviseur dans notre cadre. Face à cette situation, nous envisageons de suivre la démarche de validation progressive suivante :

- Dans un premier temps, nous demandons au joueur humain de contrôler lui-même l'agent dans son environnement, afin de vérifier la capacité de l'algorithme de réglage des paramètres de l'acteur à converger vers un acteur satisfaisant, s'il dispose d'un superviseur supposé parfait. On peut aussi donner des trajectoires pré-enregistrées ou fusionner de telles trajectoires, dans l'esprit de ce qu'a réalisé C. Thurau pour le jeu *Quake* [18].
- Cette phase permet en outre de réaliser un amorçage efficace du superviseur, en lui imposant un apprentissage sur des trajectoires qui conduisent à la récompense, au lieu de réaliser initialement des trajectoires aléatoires qui seront longues à trouver une première source de récompense.
- Ensuite, une fois la partie correspondant au réglage de l'acteur validée et le superviseur amorcé, il sera temps de se passer de l'intervention humaine pour observer la capacité de l'architecture à résoudre le problème de navigation de façon autonome.

[18] [Thurau et al., 2004]

9.1.7 De nombreux points en suspens

Au-delà de l'ébauche de solutions que nous venons de décrire, de nombreux points qu'il sera nécessaire de traiter dans le cadre de la thèse de T. Gourdin n'ont pas encore été abordés. En particulier, nous retenons les points suivants :

- spécifier un mécanisme de navigation pour des simulations complexes en raisonnant uniquement à partir de points pertinents ne va pas de soi. La représentation d'obstacles tels que des murs sous forme de points, par exemple, pose problème. Il faudra intégrer ce type de questions relatives aux applications dans le cadre de la réflexion générale.
- dans le cadre du travail de F. Flacher, les points d'intérêt auxquels les forces sont attachées sont définis comme des barycentres de points pertinents grâce à un algorithme génétique. Nous n'avons pas à l'heure actuelle de solution plus satisfaisante au problème de la détermination de ces points d'intérêt dans le cadre du travail de T. Gourdin.
- par ailleurs, F. Flacher exprime les problèmes de coordination spatiale qu'il résout comme des problèmes d'optimisation multi-critères. La dimension multi-critères n'a pas été intégrée pour l'instant dans la réflexion de T. Gourdin, elle devra l'être à terme, ce qui soulève un certain nombre de difficultés théoriques supplémentaires.

Ce plan de travail en est encore à une phase préliminaire, de nombreuses questions théoriques, techniques et méthodologiques devront être résolues pour le mener à bien en pratique, si bien que nous ne pouvons nous avancer au-delà de ce point dans le cadre de ce document.

9.2 Coordination d'actions collectives

9.2.1 Extension du cadre markovien : les jeux de Markov

Le cadre des processus de décision markoviens permet de rendre compte des interactions d'un agent avec son environnement. En effet, il permet de définir l'évolution de l'état du système composé de l'agent et de son environnement en fonction des actions de l'agent. Il permet en outre d'associer des récompenses au comportement de l'agent. Toutefois, pour représenter un environnement dans lequel évoluent plusieurs agents, il faut étendre ce cadre. D'une part, l'état du système composé de l'environnement et des agents évolue en fonction des actions simultanées de chacun des agents. D'autre part, la récompense que reçoit chacun des agents peut dépendre de l'état du système ainsi que des actions de tous les agents. Une extension satisfaisante de ce cadre est proposée en théorie des jeux sous le nom de « jeux de Markov » ou « jeux stochastiques » [19].

En théorie des jeux, une classe de problèmes élémentaires appelés « jeux en forme normale » [20] est constituée de processus dans lesquels il y a un seul état et un ensemble d'agents dont la récompense individuelle dépend de l'action entreprise par chacun des autres agents. La table 9.1 représente un exemple particulier dans lequel il n'y a que deux agents qui peuvent faire chacun deux actions. On peut généraliser immédiatement à des problèmes dans lesquels n agents peuvent faire k actions. On considère que les agents choisissent leur action simultanément et on appelle « action jointe » le vecteur d'actions associant une action à chaque agent. Par ailleurs, on appelle « stratégie » d'un agent la distribution de probabilité selon laquelle il joue les différentes actions à sa disposition pour un état donné.

[19] [Littman, 1994, Shapley, 1953]
[20] *normal form games*

	a_{21} (avouer)	a_{22} (nier)
a_{11} (avouer)	(-5,-5)	(-10,0)
a_{12} (nier)	(0,-10)	(-1,-1)

TAB. 9.1 – **Un exemple de jeu à un état et deux agents dotés de deux actions. L'agent 1 peut faire les actions a_{11} ou a_{12}, l'agent 2 peut faire les actions a_{21} ou a_{22}. Dans la table, on fait apparaître les récompenses reçues respectivement par l'agent 1 et l'agent 2 selon les actions choisies par l'un et l'autre. La table présentée ici correspond au « dilemme du prisonnier », problème très étudié en psychologie sociale et en théorie des jeux.**

Pour définir un jeu de Markov, il faut considérer un processus constitué d'un ensemble d'états, dans lequel chaque état dispose d'une table de récompense en fonction des actions de chacun des agents du type de celle de la table 9.1. En outre, il faut définir une fonction de transition qui indique, pour chaque état, dans quel état successeur se retrouvera le système en fonction de l'ensemble des actions entreprises par chacun des agents. Cette information peut à nouveau être représentée sous la forme d'une table, du type de celle qui apparaît sur la table 9.2 dans un cas à deux agents dotés de deux actions.

	a_{21}	a_{22}
a_{11}	S_1	S_3
a_{12}	S_3	S_2

TAB. 9.2 – **Table de transition entre états à partir de l'état S_0. Dans cet exemple, l'état atteint (S_3) est le même selon que l'agent 1 réalise a_{11} et l'agent 2 a_{22} ou bien que l'agent 1 réalise a_{12} et l'agent 2 a_{21}.**

La version que nous avons décrite ci-dessus est déterministe. On étend trivialement à un cadre stochastique en associant à chaque état des distributions de probabilité de récompense et des distributions de probabilité d'atteindre les différents états possibles pour chaque ensemble d'actions choisies par les agents.

Formellement, on décrit un jeu de Markov par la donnée des éléments suivants :

- un ensemble fini α de n agents ;
- un ensemble fini S d'états discrets notés s ;
- un ensemble fini A d'actions individuelles discrètes notées a_i pour l'agent i [21] ;
- une fonction de transition $T : S \times A \rightarrow \Pi(S)$ où $\Pi(S)$ est l'ensemble des distributions de probabilité sur S. Une distribution de probabilité particulière $p(S)$ indique pour tout couple $(s^t, < a_1, ..., a_n >)$ la probabilité pour que les agents se retrouvent dans chacun des états s^{t+1} possibles quand ils font les actions $< a_1, ..., a_n >$ dans l'état s^t ;
- une fonction de renforcement $R_i : S \times A \rightarrow \mathbb{R}$ qui indique sous forme d'un nombre réel pour tout agent i et pour tout triplet $(s^t, < a_1, ..., a_n >, s^{t+1})$, le signal de renforcement (punition ou récompense) que reçoivent les agents quand ils font les actions $< a_1, ..., a_n >$ dans l'état s^t et se retrouvent dans l'état s^{t+1} .

Le cadre des jeux de Markov est strictement descriptif et présente l'avantage d'être très général. En particulier, il ne préjuge en rien de l'organisation de la décision parmi les agents ni du processus de décision de chaque agent particulier. Pour rendre compte de la diversité des organisations possibles du processus de décision au sein d'un jeu de Markov, il convient de distinguer d'une part des décideurs, qui choisissent l'action d'un certain nombre d'agents et, d'autre part, les agents eux-mêmes, qui exécutent l'action choisie par le décideur qui les dirige. Dans le cadre des

[21] Pour être tout à fait général, il faudrait considérer des ensembles d'actions possibles différents pour chaque agent et pour chaque état. L'ensemble A défini ici peut être vu comme la réunion de tous ces ensembles d'actions.

processus de décision markoviens, le rôle d'un décideur consiste à choisir les actions qui permettent de maximiser la récompense reçue par le groupe d'agents dont il a la charge. Une fois cette distinction introduite, l'éventail de possibilités suivant apparaît :

- il peut y avoir un décideur unique qui dirige la totalité des agents. Dans ce cas, le rôle du décideur est de coordonner les actions des agents de façon à maximiser une fonction des récompenses individuelles reçues. Ce cadre général est qualifié de centralisé ;
- à l'autre extrême, chaque agent peut disposer de son décideur individuel, auquel cas le rôle du décideur individuel est de maximiser la récompense individuelle de cet agent sans se soucier des récompenses des autres agents ;
- entre les deux extrêmes, on peut avoir toutes les formes d'organisation possibles, dans lesquelles un nombre variable de décideurs dirigent un nombre variable d'agents. Un cas qui correspond à de nombreuses applications est le cas dans lequel il y a deux décideurs qui coopèrent ou au contraire s'affrontent alors qu'ils dirigent chacun un nombre quelconque d'agents.

9.2.2 Classes de jeux et notions d'optimalité

	a_{21}	a_{22}
a_{11}	$(1,1)$	$(0,0)$
a_{12}	$(-3,-3)$	$(2,2)$

	a_{21}	a_{22}
a_{11}	1	0
a_{12}	-3	2

TAB. 9.3 – **Exemple de jeu coopératif. La table de droite est une forme condensée de la table de gauche, les récompenses étant les mêmes pour tous les agents.**

Dès lors qu'il y a plus d'un décideur, la présence de cadre coopératif ou au contraire conflictuel entre divers décideurs dépend moins de l'organisation du processus de décision que de la nature des fonctions de récompenses des différents agents. En effet, les recherches en théorie des jeux ont permis de mettre en évidence différentes classes de problèmes dans lesquelles la relation entre les agent se décline différemment.

Tout d'abord, si tous les agents reçoivent la même récompense dans tous les états, alors leurs décideurs ont intérêt à coopérer pour choisir collectivement l'ensemble d'actions qui leur procure à tous la récompense la plus élevée. Un exemple de jeu coopératif apparaît dans la table 9.3.

	a_{21}	a_{22}
a_{11}	(5,-5)	(10,-10)
a_{12}	(-10,10)	(-1,1)

TAB. 9.4 – **Exemple de jeu à somme nulle.**

Au contraire, si les récompenses de certains agents sont faibles quand celles d'autres agents sont élevées et réciproquement, alors leurs décideurs sont en conflit. Les jeux conflictuels les plus étudiés dans un cadre théorique sont les jeux à somme nulle, dans lesquels, pour toute action, la somme des gains des agents qui gagnent est égale à la somme des pertes des agents qui perdent. Un exemple de jeu à somme nulle apparaît dans la table 9.4.

Distinguer ces différentes sous-classes de problèmes est important en théorie des jeux car certains algorithmes simples ne permettent aux agents de trouver un comportement optimal que dans certaines sous-classes. Comme le

montre très bien K. Verbeeck [22], dans le cas général où les récompenses associées aux différentes actions jointes sont quelconques, la notion d'optimalité elle-même ne va pas de soi. En effet, on se trouve face à un problème multi-critères dans lequel la récompense associée à la stratégie de chacun des agents est l'un des critères que l'on cherche à optimiser, si bien que deux solutions peuvent ne pas être comparables.

Deux notions relatives à l'optimalité s'avèrent tout de même bien fondées, à savoir les notions d'équilibre de Nash et d'ensemble d'actions Pareto-optimal.

Un équilibre de Nash est une situation dans laquelle tout agent qui tenterait de changer de politique unilatéralement verrait sa récompense réduite par ce changement. Pour sortir d'un tel équilibre à la recherche d'un équilibre meilleur, il faut donc que plusieurs agents changent de politique simultanément.

Par ailleurs, on appelle « ensemble d'actions Pareto-optimal » un ensemble d'actions tel que si au moins l'un des joueurs modifie sa stratégie, personne n'y gagne et au moins un des joueurs (pas nécessairement le même) recevra une récompense moindre.

Il est important de signaler qu'un ensemble d'actions Pareto-optimal n'est pas nécessairement un équilibre de Nash, comme l'illustre le dilemme du prisonnier illustré à la table 9.1. En effet, l'ensemble d'actions Pareto-optimal est $< a_{12}, a_{22} >$ dans lequel chaque agent ne perd qu'un point, mais chacun des agents peut améliorer sa stratégie en trahissant l'autre, à condition que l'autre ne le trahisse pas simultanément.

La plupart des travaux théoriques s'efforcent de trouver des algorithmes de recherche de stratégie pour les agents confrontés à des jeux en forme normale ou des jeux de Markov dont on peut prouver formellement qu'ils convergent à coup sûr vers un équilibre de Nash ou vers un ensemble d'actions Pareto-optimal, et cela dans le cadre soit des jeux coopératifs, soit des jeux à somme nulle, soit dans d'autres cadres dont le cas général. En règle générale, les preuves de convergence de ces algorithmes sont assorties d'hypothèses selon lesquelles l'agent a accès soit aux actions des autres agents, soit à leur récompense, afin d'estimer au mieux les stratégies jouées par les autres agents.

C'est le cas par exemple des « jeux fictifs » de G. Brown [23] dont la convergence vers un équilibre de Nash a été montré pour différentes classes de jeux en forme normale, alors qu'elle a été infirmée pour le cas général. Dans le cas des jeux fictifs, on suppose que l'agent collecte des statistiques sur les actions des autres agents pour se faire un modèle de leur stratégie.

Le cadre des jeux fictifs a été étendu aux jeux de Markov quelconque par les algorithmes successifs *Nash Q-learning* [24], *BR Q-learning* [25] et *NSCP Q-learning* [26], qui font l'hypothèse supplémentaire que les agents découvrent les transitions et récompenses du problème auquel ils sont confrontés.

Parallèlement aux jeux fictifs, l'algorithme des « jeux adaptatifs » de H. Young [27] introduit une forme d'exploration qui permet de converger vers le meilleur équilibre de Nash plutôt que vers le premier équilibre rencontré. O. Gies [28] a proposé une extension des « jeux adaptatifs » similaire à *BR Q-learning* et *NSCP Q-learning*, qui bénéficie de la même propriété dans le cadre des jeux de Markov avec apprentissage des fonctions de transition et de récompense.

Dans la mesure où la plupart des algorithmes ne disposent pas de preuve de convergence dans le cas général, nous allons nous concentrer dans la suite sur les travaux qui se sont développés pour des classes de sous-problèmes spécifique et, en particulier, les jeux purement coopératifs.

[22] [Verbeeck, 2004]
[23] [Brown, 1951]
[24] [Hu & Wellman, 2003]
[25] [Hu, 2003]
[26] [Weinberg & S., 2004]
[27] [Young, 1998]
[28] [Gies, mise]

9.2.3 Processus de décision markoviens multi-agents

L'extension la plus simple du formalisme des processus de décision markoviens au cadre dans lequel il y a plusieurs agents est sans doute celle proposée par C. Boutilier [29] avec C. Claus [30] sous le nom de processus de décision markovien multi-agents [31].

La principale simplification qui définit le cadre des processus de décision markoviens multi-agents par rapport au cadre général décrit à la section précédente provient de ce que l'on considère une récompense unique, résultant du comportement de tous les agents. On se ramène alors très simplement au cadre standard des processus de décision markoviens en considérant une action unique correspondant à l'action jointe, définie comme le vecteur des actions individuelles effectuées par chacun des agents. On a alors une fonction de transition classique qui donne la distribution de probabilité d'atteindre un état suivant, en fonction de l'état courant et de l'action jointe réalisée par l'ensemble des agents et une fonction de récompense unique associée aux transitions du système global.

Tous les algorithmes de planification et d'apprentissage qui s'appliquent dans le cadre standard des processus de décision markoviens s'appliquent directement au cadre des processus de décision markoviens multi-agents. En effet, adopter ce formalisme revient à considérer l'ensemble des agents comme un agent unique, plus complexe, dont les actions sont définies par le vecteur des actions des agents individuels.

Comme nous l'avons indiqué précédemment, la fonction de récompense étant unique, les agents ont intérêt à coopérer. En effet, ils doivent trouver collectivement les séquences d'actions qui conduisent à la récompense la plus élevée, cette récompense étant distribuée uniformément à tous les agents.

Résolution centralisée

Le cadre des processus de décision markoviens multi-agents est encore un cadre strictement descriptif qui n'implique pas directement le choix d'une forme particulière d'organisation de la décision. Cependant, pour bénéficier directement des propriétés inhérentes au cadre standard des processus de décision markoviens, la forme d'organisation la plus naturelle consiste à considérer un unique décideur qui choisit simultanément l'action de tous les agents.

Si cette solution présente l'avantage d'être simple à mettre en œuvre, elle souffre en pratique de trois inconvénients inhérents au caractère centralisé de la solution :

- Déterminer une politique consiste à imposer simultanément une action à chacun des agents, à partir d'un point de vue unique dans lequel l'état global du système, donc de tous les agents est connu. Le premier défaut est que l'on est contraint de supposer que le décideur central dispose d'un accès parfait aux agents pour leur imposer leur action, ce qui ne va pas de soi dans le cadre d'applications robotiques ou distribuées physiquement sur des ordinateurs distants. Dans le cas de certaines simulations, si l'on s'autorise un accès exhaustif aux structures de données représentant l'environnement et tous les agents, ce point est moins problématique. Malheureusement, nous ne pouvons nous appuyer sur une telle hypothèse simplificatrice sans perdre considérablement en généralité.
- Le deuxième défaut de cette approche est qu'elle repose sur une hypothèse très difficile à réaliser en pratique. En effet, on peut avoir recours à une solution centralisée dans l'espoir d'appliquer directement des algorithmes tirés du cadre des processus de décision markoviens. Or, pour rester dans le cadre markovien, il faut supposer que le décideur central dispose d'une information telle que l'hypothèse de Markov est vérifiée, donc qu'il a accès à une connaissance complète de l'état du système. Ce point est encore plus difficile à vérifier

[29] [Boutilier, 1996, Boutilier, 1999]
[30] [Claus & Boutilier, 1998]
[31] *Multi-agent Markov Decision Process*, ou MMDP

dans un cadre multi-agents que dans le cadre d'un agent unique. Nous verrons qu'il existe des extensions au cadre multi-agents des processus de décision markoviens partiellement observables.

– Le troisième défaut est que le nombre d'actions potentielles du décideur central, s'il y a n agents qui peuvent faire k actions individuelles est k^n, donc il y a une explosion combinatoire de l'espace des actions qui rend cette approche inutilisable dès que le nombre d'agents devient grand.

Résolution décentralisée et coordination

L'alternative au cadre centralisé consiste à considérer des agents indépendants qui disposent chacun de leur propre décideur. Ce second cadre est exempt des difficultés précédentes, mais il en fait apparaître de nouvelles. La principale difficulté est que les agents doivent se coordonner pour trouver collectivement l'ensemble d'actions optimales dans chacun des états qu'ils parcourent, ce qui ne va pas de soi.

Pour illustrer l'existence de problèmes de coordination, C. Boutilier propose un exemple simple dans lequel il y a deux agents et, pour un certain état, deux actions jointes optimales $< a, a >$ et $< b, b >$. Si les deux agents ne se coordonnent pas, chacun des autres agents n'a aucun moyen de savoir si l'autre va choisir l'action a ou l'action b. Les décisions étant simultanées, ils risquent de réaliser $< a, b >$ ou $< b, a >$, ce qui n'est pas optimal. C. Boutilier [32] donne une définition formelle générale de ce type de situations. Il propose en outre une liste de mécanismes potentiels pour remédier à ces problèmes dans un cadre décentralisé.

9.2.4 Les processus de décision markoviens partiellement observables décentralisés

Nous avons dit que le formalisme des processus de décision markoviens multi-agents se prêtait plus naturellement à une approche de type centralisée qu'à une approche décentralisée. Cela provient surtout du fait que l'état du système global est supposé parfaitement connu de tous les agents à tout instant, hypothèse difficilement tenable dans un cadre décentralisé.

Pour revenir à la fois sur le caractère centralisé et sur le caractère d'observabilité totale des processus de décision markoviens multi-agents, D. Bernstein [33] propose le cadre des processus de décision markoviens partiellement observables décentralisés, ou DEC-POMDP [34].

L'idée générale tient en deux points. D'une part, on conserve le caractère coopératif des agents, qui s'exprime comme pour les processus de décision markoviens multi-agents par le fait qu'il y a une fonction de récompense unique et commune à tous les agents. D'autre part, on introduit le fait que chaque agent ne dispose que d'une observation partielle sur l'état du système global, comme c'est le cas dans le cadre des processus de décision markoviens partiellement observables.

Formellement, un DEC-POMDP se décrit donc par la donnée de :

– un ensemble fini α de n agents ;
– un ensemble fini S d'états discrets notés s ;
– un ensemble fini A d'actions discrètes notées $a_{ii\in\alpha}$;
– une fonction de transition $T : S \times A \rightarrow \Pi(S)$ où $\Pi(S)$ est l'ensemble des distributions de probabilité sur S. On note $P(s^t, \{a_i\}_{i\in\alpha}, s^{t+1})$ la probabilité de transition de l'état s^t à l'état s^{t+1} quand les agents font les actions $< a_1, ..., a_n >$;

[32] [Boutilier, 1996, Boutilier, 1999]
[33] [Bernstein et al., 2002]
[34] *Decentralized POMDP*

– une unique fonction de renforcement $R : S \times A \rightarrow \mathbb{R}$ qui indique le signal de renforcement (punition ou récompense) que reçoivent tous les agents. $R(s^t, \{a_i\}_{i\in\alpha}, s^{t+1})$ représente la récompense reçue par les agents quand ils passent de l'état s^t à l'état s^{t+1} en faisant l'action jointe $< a_1, ..., a_n >$;
– un ensemble fini Ω d'observations notées $\omega_{ii\in\alpha}$;
– une fonction d'observation $O : S \times A \rightarrow \Pi(\Omega)$ qui indique globalement la probabilité pour les agents observent respectivement $< o_1, ..., o_n >$ quand ils passent de l'état s^t à l'état s^{t+1} en faisant les actions $< a_1, ..., a_n >$;

Comme le fait remarquer D. Bernstein, un DEC-POMDP dans lequel il y a un seul agent est un POMDP. Par ailleurs, il définit un DEC-MDP comme un DEC-POMDP particulier dans lequel il existe une bijection de l'espace des observations des agents vers celui des états, ce qui signifie que l'observation jointe de tous les agents suffit à connaître l'état du système global.

La différence entre un processus de décision markovien multi-agents et un processus de décision markovien décentralisé se résume au fait que dans un processus de décision markovien multi-agents, l'organe de décision dispose d'un accès direct à l'état du système alors que, dans un processus de décision markovien décentralisé, l'agrégation de l'ensemble des observations de tous les agents permet de déterminer l'état du système sans ambiguïté. Cette agrégation peut être réalisée par un décideur central ou bien par chacun des agents individuels par le biais d'une communication.

D. Bernstein propose alors le schéma d'inclusion qui apparaît à la figure 9.2 pour les différents cadres théoriques issus des processus de décision markoviens.

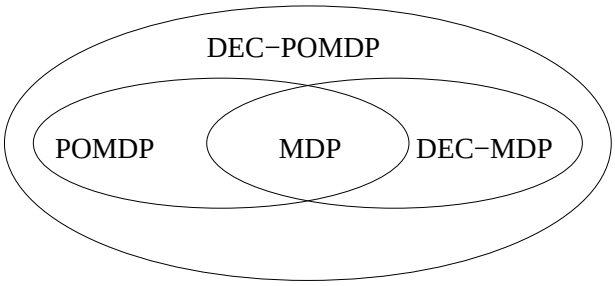


FIG. 9.2 – **Inclusions des cadres théoriques entre MDP et DEC-POMDP**

Dans ce cadre, rien n'empêche d'avoir recours à une décision centralisée qui tire parti de l'ensemble des observations de chacun des agents pour déterminer une politique jointe, mais une telle approche est moins naturelle dans ce cadre que dans celui des processus de décision markoviens multi-agents. On a plutôt recours à des mécanismes de communication qui permettent à chaque agent de choisir son action à partir d'une information partielle ou totale sur l'observation, l'action et/ou l'état interne des autres agents.

Dès lors, se pose le problème de déterminer quoi communiquer et à quel moment, ce que posent très clairement les auteurs du cadre COM-MTDP [35]. De son côté, C. Goldman propose un cadre dans lequel les actions de communication sont dotées d'un coût qu'il faut s'efforcer de minimiser. De façon plus pragmatique, D. Szer propose un mécanisme de restriction de la communication à des échanges d'information sur la récompense obtenue par les agents lorsque celle-ci est plus élevée que prévu, qui permet de conserver des propriétés de convergence des algorithmes de programmation dynamique [36].

[35] *Communicative Multiagent Team Decision Problem* [Pynadath & Tambe, 2002][Nair et al., 2003]
[36] [Szer & Charpillet, 2004]

Par ailleurs, il faut avoir à l'esprit que des résultats théoriques de complexité [37] montrent que, dès lors que le nombre d'agents considérés dépasse deux ou trois, il n'y a pas d'espoir de faire converger un algorithme de programmation dynamique standard et encore moins un algorithme d'apprentissage dans le cadre théorique des DEC-POMDPs, ni même des DEC-MDPs si l'on n'a pas recours à des heuristiques pour réduire la taille des problèmes. D. Bernstein démontre en effet que la recherche d'une politique optimale dans un tel cadre est un problème *NEXP-hard*, ce qui signifie que le temps de calcul augmente au mieux en fonction exponentielle du nombre d'agents.

Pour résoudre ce type de problèmes, de très nombreux travaux ont recours à trois types de solutions :

- avoir recours à des hypothèses supplémentaires restreignant encore le cadre ;
- chercher des heuristiques permettant d'atteindre une performance satisfaisante mais non optimale ;
- abandonner le cadre formel pour apporter des solutions *ad hoc* qui ne valent que pour certaines applications particulières.

Ainsi, A. Beynier se place dans le premier cadre lorsqu'elle choisit de travailler dans un cadre complètement observable auquel elle ajoute des contraintes temporelles sur l'ordre d'exécution des tâches par les agents [38]. Elle se sert alors de ces contraintes pour trouver un ordonnancement optimal.

I. Chadès [39], de son côté, choisit la seconde solution lorsqu'elle propose un algorithme dans lequel les agents modifient leur politique tour à tour, ce qui permet de converger vers un équilibre de Nash qui n'est pas forcément le meilleur équilibre atteignable. C'est aussi le cas d'O. Buffet [40], qui combine la prise en compte de plusieurs motivations et de comportements de haut niveau reposant sur des actions plus élémentaires.

Pour sa part, P. Garcia [41] se situe entre la seconde et la troisième solution lorsqu'il propose d'introduire de la connaissance spécifique d'un problème sous la forme de programmes permettant d'exécuter des macro-actions sans sortir du cadre des modèles de Markov.

Le travail de T. Degris, qui nous allons présenter à la section suivante, devrait reposer sur une combinaison de la première et de la troisième solution.

9.3 Coordination d'objectifs de déplacement

La thèse de T. Degris est financée par la DGA. Elle a pour objet la coordination d'un groupe de micro-drones opérant à l'intérieur de bâtiments. Faute de disposer d'un ensemble de micro-drones prêts à l'emploi, le premier objectif de T. Degris consiste à identifier les techniques susceptibles de répondre à un besoin opérationnel encore assez vague puis à valider ces techniques en simulation.

Dans ce contexte, le cadre théorique choisi par T. Degris est conçu pour être complémentaire de celui de T. Gourdin. Les travaux de ces deux doctorants devront à terme s'intégrer dans une architecture commune susceptible de répondre aux problèmes applicatifs visés dans le cadre de ce mémoire. Alors que T. Gourdin cherche à doter des agents d'une capacité de navigation individuelle qui leur permet de réagir à la situation courante dans laquelle ils sont plongés, T. Degris doit proposer un mécanisme d'attribution de buts aux agents d'une même équipe, de façon à coordonner leur action dans l'espace et le temps en présence d'une équipe adverse. L'idée générale est que le système de T. Degris fixe les buts des agents de façon à prendre en compte des considérations collectives et celui de T. Gourdin réalise la navigation qui permet à chaque agent d'atteindre ces buts.

[37] [Bernstein et al., 2002]
[38] [Beynier et al., 2004]
[39] [Chadès et al., 2002]
[40] [Buffet et al., mise]
[41] [Garcia, mise]

Compte tenu de la difficulté théorique et computationnelle des problèmes de coordination multi-agents, ce découpage présente pour T. Degris de nombreux avantages :

- le choix de buts étant découplé de l'exécution de l'action, il est possible de réaliser ce choix avant la réalisation de la mission sans contrainte temporelle forte, entre deux exécutions de la mission des agents ;
- le choix des buts peut être accompli à un niveau relativement abstrait vis-à-vis du contexte précis dans lequel opèrent les agents, ce qui permet d'alléger la représentation des états sur laquelle le processus de décision doit agir ;
- dans le contexte d'une exécution découplée de l'exécution de la mission, rien ne s'oppose en première approximation à l'idée de réaliser le choix des buts de manière centralisée, ce qui élimine de nombreux problèmes de coordination.

Par ailleurs, un tel découpage soulève un certain nombre de problèmes. En particulier, dans la mesure où le mécanisme de navigation de T. Gourdin est susceptible de réagir à des observations dont le modèle de T. Degris ne dispose pas, l'action de déplacement attendue échouera certainement sans qu'il soit possible d'imputer une cause fiable à cet échec. Le système de T. Degris sera donc contraint de considérer les actions comme dotées de probabilités de réussite sans avoir un accès à un modèle sous-jacent aux probabilités observées.

9.3.1 Un cadre formel abstrait

Sachant que le mécanisme de navigation proposé par T. Gourdin gère les détails du positionnement dans un univers continu, T. Degris propose de modéliser abstraitement l'environnement dans lequel évoluent les agents sous la forme d'un graphe de lieux reliés entre eux par des arcs. Dès lors, une définition minimale de l'état du système est donnée par la position de chacun des agents sur l'ensemble des nœuds du graphe. Par ailleurs, l'action de chaque agent peut être représentée sommairement par un déplacement de l'agent d'un nœud à un autre nœud dans le graphe. Les agents étant dotés d'une mission, on peut représenter abstraitement la réussite de cette mission par une fonction de renforcement qui associe des récompenses ou des punitions à certains états du système, c'est-à-dire à l'atteinte de certains nœuds dans le graphe par certains agents. Si la récompense est unique pour tous les agents, alors le cadre abstrait que nous venons de définir correspond exactement au cadre des processus de décision markoviens multi-agents. Si chaque agent reçoit une récompense qui lui est propre, alors on se situe dans le cadre plus général des jeux de Markov. Dans tous les cas, les mécanismes fondamentaux de la programmation dynamique ou de l'apprentissage par renforcement peuvent s'appliquer dans un tel cadre. Il suffit d'associer à chaque état du système ainsi défini une valeur – ou à chaque couple (état, action) une qualité – et d'attendre de ces valeurs ou qualités qu'elles convergent pour que les agents puissent rechercher les états de plus grande valeur.

Toujours dans ce cadre abstrait, on peut représenter l'état du système par un vecteur indiquant pour chaque agent dans quel nœud du graphe il se situe. En guise d'alternative, on pourrait aussi choisir de représenter le même état par un vecteur indiquant pour chaque nœud quels agents il contient, mais cette représentation a toutes les chances d'être moins compacte. Une fois ce cadre général posé, nous allons voir tous les aménagements qu'il faut y apporter pour le rendre utilisable en pratique.

9.3.2 Limitations du cadre théorique

Si le cadre que nous venons de définir permet en théorie d'attribuer une valeur à chaque répartition des agents au sein du graphe, donc de trouver une politique optimale de positionnement des agents sur les différents nœuds, cette solution s'avère difficilement réalisable en pratique. Les sources de difficulté sont les suivantes :

– Tout d'abord, si l'on dispose d'un ensemble de n agents dans un graphe de k nœuds, la représentation met en jeu k^n états possibles. Dans le cadre des jeux vidéo, les graphes de points de navigation comprennent souvent quelques centaines de nœuds, et le nombre d'agents par équipe peut être de l'ordre de la dizaine. $100^{10} = 10^{20}$ états n'est pas gérable avec les ordinateurs actuels.
– Ensuite, la technique que nous proposons suppose un calcul centralisé. Nous avons dit qu'il était parfaitement envisageable de réaliser ce calcul centralisé entre deux exécutions de missions, mais se pose le problème de l'exécution par les agents de la politique correspondante.

Une solution simple consiste à embarquer au sein de chacun des agents la totalité de la table des valeurs ou des qualités calculée de façon centralisée, afin qu'il puisse choisir en toute situation la meilleure action. Néanmoins, pour qu'il puisse faire ce choix, il faut qu'il ait accès à tout instant aux informations correspondant à la représentation d'états, donc qu'il connaisse au moins la position de tous les autres agents, ce qui ne va pas de soi dans certains cadres applicatifs. Deux alternatives se dessinent :

– soit on donne à chaque agent le moyen de communiquer à tout instant avec un décideur central qui dispose donc d'une vision globale et choisit la meilleure action jointe pour chacun des agents. On a donc recours à une solution centralisée qui repose sur la communication ;
– soit on modifie la représentation proposée précédemment, afin qu'elle comporte des informations moins difficiles à collecter pour chacun des agents individuels.

Enfin, la représentation que nous avons proposée ne prend pas en compte la présence d'une éventuelle équipe adverse, qui a pourtant une incidence considérable sur l'évolution de l'état du système et sur la valeur des récompenses qui peuvent être reçues par les différents agents. La solution théorique propre à ce problème consisterait à étendre la représentation d'états, en prenant en compte la position de chacun des agents adverses. Cette solution est difficilement envisageable pour deux raisons. D'une part, elle augmente d'autant la taille de l'espace d'états, dont nous avons déjà dit qu'il était trop vaste. D'autre part, elle fait sortir du cadre coopératif dans lequel tous les agents de la même équipe reçoivent la même récompense, pour se retrouver dans le cadre conflictuel des jeux de Markov à somme nulle, qui donne lieu à des solutions plus complexes à mettre en oeuvre.

9.3.3 Eléments de solution pratique

L'obstacle central sur lequel reposent toutes les autres difficultés que nous avons énumérées à la section précédente a trait à la taille de la représentation. Compte tenu des avancées théoriques que nous avons décrites lors des premiers chapitres de ce document, il nous semble aller de soi qu'une voie de résolution de ce problème de taille de la représentation est à rechercher du côté des représentations factorisées. Au lieu de représenter les points de navigation, donc les nœuds d'un graphe, par des numéros, il semble légitime d'essayer d'associer à chacun de ces points de navigation des vecteurs d'observables suffisamment riches pour permettre de les distinguer les uns des autres, et de les situer les uns par rapport aux autres. On peut alors espérer trouver des généralisations afin de bâtir une représentation plus compacte que l'énumération exhaustive de toutes les combinaisons de position des agents à tous les nœuds possibles. Les travaux de F. Teichtel [42], qui combinent un cadre factorisé avec des opérateurs d'agrégation en régions, nous semblent constituer une piste à suivre.

Un axe central du travail de T. Degris consistera à appliquer les techniques tirées du cadre des processus de décisions markoviens factorisés, pour traiter les problèmes de coordination auxquels il est confronté. Dès lors que l'on abandonne une représentation monolithique de l'état de l'agent par le lieu où il se trouve au profit d'une représentation sous la forme d'un vecteur d'observables, apparaît la possibilité d'intégrer dans la représentation de

[42] [Teichteil-Koenigsbuch & Fabiani, 2004]

nombreuses informations supplémentaires qui sont certainement pertinentes dans le choix des objectifs des agents, tel que l'état de leurs variables internes, par exemple. En revanche, plus on ajoute d'informations nouvelles de ce type, plus la taille de l'espace représenté va croître à nouveau. Il apparaît donc un compromis à gérer entre la taille de la représentation, l'exhaustivité des informations fournies aux processus de décision, sachant qu'il faut aussi prendre en compte la capacité des agents à se communiquer ou non les uns aux autres les informations nécessaires à l'élaboration de cette représentation d'états.

Du fait de la présence de ce compromis, il apparaît clair que les processus de décision vont devoir fonctionner dans un contexte où leur observation de l'état du système global est très partielle. Les informations manquantes se traduiront par une évolution de l'état représenté qui sera perçue comme stochastique. Les agents devront donc choisir leurs objectifs au mieux en collectant des statistiques sur ce qui peut se passer dans le déroulement de leur mission, compte tenu de l'information partielle dont ils disposent. Parmi les informations qu'il semble utile d'ajouter dans la représentation d'état fournie à chaque agent pour assurer une bonne coordination, figure l'objectif visé par chacun des agents. Alors que les algorithmes de programmation dynamique permettent d'atteindre un objectif de façon optimale sans représenter cet objectif explicitement, T. Degris propose d'intégrer dans la description de l'état une représentation explicite de l'objectif de chacun des agents, ce qui présente plusieurs avantages.

- Tout d'abord, connaître son objectif à long terme permet d'anticiper le comportement d'un agent de façon relativement fiable sans disposer de beaucoup d'autres informations sur son état courant. Disposer de cette information devrait permettre aux agents de se coordonner, sans avoir à communiquer à chaque pas de temps une quantité d'information importante sur leur état courant. Il est crucial, par contre, qu'ils s'informent les uns les autres de tout changement dans les objectifs visés.
- Ensuite, la prise en compte explicite des objectifs devrait permettre de rendre plus efficace les algorithmes de programmation dynamique visant à attribuer une valeur à chacun des états possibles du système. En effet, le processus de programmation dynamique standard propage des valeurs entre états voisins de façon exhaustive, ce qui peut le conduire à considérer des états qui ne seront jamais visités, car ils ne présentent aucun intérêt. Connaissant les objectifs de chacun des agents, on peut déterminer à partir de n'importe quel état vers quel nœud ils vont se diriger à l'aide d'un algorithme de planification tel que A^* [43], par exemple. Cette information permet d'orienter un processus de programmation dynamique de façon à considérer prioritairement les états les plus susceptibles d'être visités, comme le font les algorithmes de balayage prioritaire [44].

Ces deux avantages se payent par le fait qu'ils débouchent sur une augmentation significative de la taille de la représentation, si les objectifs possibles pour chacun des agents sont nombreux. En outre, cette solution ne peut pas être étendue au cas où la représentation prend en compte le comportement de l'adversaire, car l'objectif de chacun des agents adverses ne peut pas être considéré comme connu de façon sûre.

Une autre voie d'amélioration concerne la prise en compte du caractère stochastique de la représentation que nous allons construire. Les processus de programmation dynamique sur lesquels nous allons nous appuyer considèrent a priori des probabilités uniformes de manifestation des effets de la présence d'information manquante. Ainsi, si l'on décide de ne pas modéliser la présence et le comportement d'une équipe adverse, une approche stochastique considérera a priori une distribution de probabilité uniforme pour que les adversaires aient tous les comportements que l'on est susceptible de leur attribuer. Afin d'affiner le processus de décision en présence de cette information cachée, il apparaît essentiel de procéder à un apprentissage qui collecte des statistiques, pour attribuer aux phénomènes perçus comme stochastiques des probabilités de manifestation conformes à ce qui se passe effectivement. C'est de cette façon que le choix d'une approche stochastique est susceptible de se compor-

[43] [Hart et al., 1968, Hart et al., 1972]

[44] voir la section 3.3.1, page 59.

ter au mieux en présence de processus sous-jacents non modélisés, qui entraînent un biais dans l'évolution de la dynamique de l'état observé du système.

Ce qui est vrai pour la présence des agents adverses l'est aussi de toutes les autres sources d'information manquante. C'est encore vrai en ce qui concerne la capacité des agents à se rendre ou non d'un point de navigation à un autre point de navigation, en fonction des objectifs qui leur sont assignés. En effet, si la représentation manipulée par T. Degris donne une vision abstraite des déplacements des agents de lieu en lieu, ces déplacements sont réalisés concrètement par le procédé de navigation fourni par T. Gourdin en fonction de la situation concrète à laquelle les agents sont confrontés, si bien que peuvent apparaître des régularités dans le fait que telle transition d'un nœud à un autre est réussie avec telle probabilité dans telle situation, et avec telle autre probabilité dans telle autre situation. Au lieu de considérer des probabilités de transition uniforme entre les états, on pourra collecter des statistiques pour affiner ces probabilités au fil de l'expérience des agents.

En fin de compte, pour résoudre le problème auquel il est confronté, T. Degris dispose d'un cadre théorique clairement posé mais inutilisable en pratique. Il dispose par ailleurs d'éléments d'une approche visant à trouver des solutions concrètes. Cette approche reposant sur le choix d'une représentation factorisée, il apparaît qu'un facteur crucial du succès de la démarche sur le plan applicatif concerne le choix des observables qui seront introduits dans la représentation d'états. Il sera nécessaire d'acquérir une expérience empirique des observables qui s'avèrent les plus pertinentes, en fonction des différentes applications auxquelles peut être confronté le système. En outre, il semble utile de réfléchir dès maintenant à la mise au point d'un système capable de choisir des observables pertinentes dans un ensemble plus vaste que celui qu'il est susceptible de manipuler dans sa représentation d'état. Ce point fait partie de nos axes de réflexion pour un plus long terme.

9.4 Conclusion

La thèse de T. Gourdin comme celle de T. Degris constituent nos premières tentatives pour aborder de front des problèmes d'apprentissage par renforcement dans le cadre de simulations vraiment complexes. Dans un cas comme dans l'autre, nous avons montré qu'il existait un cadre théorique bien identifié, mais donnant lieu à des algorithmes fragiles vis-à-vis des problèmes de grande taille. Nous avons aussi présenté une démarche pragmatique visant à traiter ces problèmes, quitte à perdre en généralité vis-à-vis du cadre théorique. Les systèmes informatiques correspondant aux travaux de T. Gourdin et T. Degris étant en cours de réalisation, il est trop tôt pour tirer des conclusions sur la validité empirique de notre démarche.

Toutefois, puisque nous atteignons le point où il faut dresser un bilan des voies de recherche présentées dans ce mémoire, nous allons revenir dans le chapitre de conclusion sur le fait qu'il y a sans doute encore beaucoup d'enseignements à tirer des dernières avancées théoriques que connaît le domaine actuellement, afin de faire évoluer notre démarche empirique dans une direction qui lui fasse perdre aussi peu que possible en généralité.

La force des travaux théoriquement fondés est de fournir des solutions valables pour des domaines d'application très divers, même si la mise en oeuvre des algorithmes résultants nécessite des aménagements spécifiques pour telle ou telle application, lorsque celles-ci sont complexes. C'est à cette interface entre les résultats théoriques récents et les applications complexes que l'essentiel de notre travail de recherche devrait se situer dans les prochaines années.

Chapitre 10

Convergences

« Maintenant, je vois beaucoup mieux ce que je voulais dire. »

O. Sigaud, le 20/02/2002

Dans l'introduction de ce document, nous avons expliqué pourquoi des simulations réalistes de grande envergure telles qu'en proposent par exemple les jeux vidéo nous semblent fournir un bon cadre pour l'étude des mécanismes adaptatifs auxquels s'intéresse l'approche animat. Après avoir exposé sur le plan théorique les techniques adaptatives qui sont au cœur de notre projet de recherche, il est temps de nous demander dans quelle mesure ces techniques peuvent répondre aux problèmes applicatifs que nous souhaitons affronter. Nous allons donc consacrer ce dernier chapitre à décrire les moyens que nous nous donnons pour déployer nos travaux dans le cadre de simulations complexes. En particulier, nous commençons par décrire sommairement la plate-forme logicielle sur laquelle nous allons nous appuyer pour réaliser nos expériences, puis nous exposons quelques-uns des travaux entrepris dans notre entourage dans le domaine des jeux vidéo. Afin de doter notre programme de recherche d'un objectif concret susceptible de fédérer les différents axes de recherche sur lesquels nous nous sommes engagé, nous présentons un scénario de jeu vidéo destiné à nous servir de problème de référence.

Ce retour vers les applications est aussi destiné à nous servir de support pour tirer le bilan des travaux que nous avons présentés dans ce mémoire et des perspectives que nous avons dégagées. Nous allons nous efforcer de montrer qu'un effort d'intégration supplémentaire est nécessaire sur le plan théorique pour repousser les limites que nous nous attendons à rencontrer sur le plan applicatif.

Pour terminer, nous remettons en perspective notre contribution dans le cadre des recherches menées autour de l'approche animat, en insistant sur la nécessité d'un dialogue approfondi avec les travaux dont les modèles s'inspirent davantage du fonctionnement des systèmes vivants.

10.1 Une plate-forme expérimentale

Si, comme nous l'avons dit en introduction, travailler dans le cadre de simulations complexes est a priori moins coûteux en temps et en argent que d'utiliser des robots, l'effort technique à fournir n'est tout de même pas négligeable. Il est donc intéressant d'investir dans la réalisation d'une plate-forme expérimentale générique disponible « une fois pour toutes », ce qui permet par la suite de se concentrer sur les problématiques théoriques sans être trop retardé par les difficultés de mise en œuvre. C'est dans cet esprit que T. Degris a conçu la plate-forme générique SATOM, qui nous permettra à terme de connecter des modules de comportement adaptatif écrits dans divers langages

de programmation à des agents évoluant dans différentes simulations, qu'il s'agisse de jeux vidéo ou d'univers simulés autres que des jeux, voire d'autres types d'environnements logiciels.

SATOM est une architecture à base de composants qui permet à différents composants orientés objets de dialoguer ensemble sans avoir besoin d'être recompilés. Au contraire de CORBA qui impose que la communication entre deux composants passe par le réseau même si les composants tournent sur la même machine, SATOM est capable de choisir selon le contexte entre une communication directe et une communication à travers le réseau. En outre, au contraire de COM, l'architecture de composants de *Microsoft*, SATOM peut tourner sur différentes familles de systèmes d'exploitation. Par ailleurs, SATOM permet de faire dialoguer des composants écrits dans des langages de programmation différents, tels que C++ et JAVA. Le dialogue entre les composants s'organise en associant à chaque composant une interface qui définit l'ensemble des méthodes que l'on est susceptible d'appeler au sein de ce composant. En faisant en sorte que divers composants offrent exactement la même interface, on obtient une propriété de généricité : un troisième composant pourra dialoguer avec l'un ou l'autre des composants exactement de la même façon.

Ainsi, une interface commune a été définie pour piloter des bots dans les moteurs de jeu *Unreal Tournament* et *Counter Strike*, si bien qu'un composant adaptatif qui pilote un bot au travers des méthodes fournies par cette interface peut s'appliquer à l'un ou l'autre jeu indifféremment. En outre, cette même interface est associée à un troisième composant qui encapsule le moteur de simulation « *Crystal Space* » qui fournit un environnement de simulation proche des jeux en trois dimensions, mais plus ouvert et entièrement programmable, puisque l'on dispose de son code source. Cet ajout permet de disposer d'un cadre expérimental plus malléable que les tests réalisés directement dans les univers de jeux. Par ailleurs, il fournit à T. Degris les moyens de développer une simulation spécifique de micro-drones se déplaçant dans des bâtiments, pour couvrir l'application imposée par le contexte de sa thèse.

10.2 Réalisations préliminaires

En marge du programme de recherche que nous avons élaboré dans ce document, un certain nombre de travaux visant des applications dans les jeux vidéo ont été réalisés dans notre entourage immédiat. Un étudiant du LIP6, S. Manier, est parvenu à développer les comportements d'une équipe de « bots » pour le jeu *Enemy Territory* [1] de façon assez satisfaisante, sans les doter toutefois de capacités adaptatives. La thèse de G. Robert [2] porte sur la mise au point d'une architecture de contrôle pour les bots, MHiCS, qui repose sur des systèmes de classeurs hiérarchiques. Il l'applique au contrôle des bots du jeu *Team Fortress*, après avoir travaillé dans le cadre du développement du jeu de rôle *Ryzom*. Nous n'avons pris aucune part à l'encadrement de ces deux réalisations, mais nous espérons en retirer des indications utiles sur l'application de techniques adaptatives au domaine des jeux vidéo.

Notre investissement à l'égard du travail de V. Labbé est un peu plus direct. Dans le cadre de son stage de DEA, il a conçu un mécanisme d'anticipation de trajectoires cycliques qui permet à un agent soit d'en intercepter un autre, soit au contraire d'éviter d'être intercepté, par exemple pour éviter d'être vu par un garde qui fait sa ronde. Ce travail a fait l'objet d'une publication commune [3], mais n'a pas encore été validé dans le cadre d'un jeu vidéo du commerce.

Enfin, nous avons encadré le stage de DEA de N. Schneider, qui portait sur la navigation d'un bot dans le cadre du jeu *Counter Strike*. Dans ce jeu, les bots ne perçoivent pas directement la position des murs qui les entourent.

[1] ©*ID Software*
[2] [Robert, 2002]
[3] [Labbé & Sigaud, 2004]

Ils disposent simplement d'un mécanisme de lancer de rayon qui les renseigne sur la distance à l'obstacle le plus proche, dans une direction donnée et à une hauteur donnée. Pour qu'ils puissent naviguer efficacement malgré ce lourd handicap, on leur associe un ensemble de points de navigation [4] disposés aux emplacements pertinents de l'environnement et reliés entre eux par un graphe. Les bots n'ont alors plus qu'à rechercher les plus courts chemins parmi les arcs de ce graphe à l'aide d'un algorithme de type A^* [5] pour naviguer efficacement.

Cette solution a fait ses preuves, mais d'une part elle impose au concepteur d'un bot de poser un à un tous les points de navigation pertinents pour toutes les cartes dans lesquelles il veut que son bot puisse se déplacer et, d'autre part, elle ne permet pas de rendre compte des comportements d'exploration d'un joueur confronté à une zone qu'il ne connaît pas. Le travail de N. Schneider élimine ces deux défauts à la fois. En s'inspirant des recherches de J.-Y. Donnart [6], il a conçu un algorithme qui permet à un bot de naviguer en direction d'un but en contournant les obstacles qui se trouvent sur sa route, détectés avec un lancer de rayon. En outre, le bot pose lui-même des points de navigation au cours de son déplacement exploratoire, si bien que, s'il repasse dans une région connue, il naviguera beaucoup plus efficacement. Ce travail demande à être consolidé et validé sur une plus grande échelle avant d'être publié dans une revue scientifique et mis à disposition de l'immense communauté de joueurs de *Counter Strike*. Les doctorants et stagiaires de l'AnimatLab dont le sujet a trait aux applications dans les jeux vidéo s'emploient à cette consolidation et intégreront le composant de navigation résultant dans leur architecture.

10.3 Un objectif fédérateur

Les voies de recherche présentées au fil des chapitres de ce document constituent un ensemble de réponses partielles à des problèmes partiels. Pour favoriser la convergence progressive de ces voies de recherches vers un cadre scientifique et technique unifié, il nous a semblé utile de réfléchir aussi à un cadre applicatif fédérateur permettant de les intégrer dans une plate-forme commune. Dans cette section, nous présentons un scénario de jeu vidéo dans lesquels nous aimerions que des bots construits à l'aide de techniques adaptatives que nous avons présentées soient capables de se comporter de façon satisfaisante.

Considérons la situation décrite sur la figure 10.1. Un groupe de terroristes détient une arme bactériologique dans la pièce A. Les terroristes s'attendent à être pris d'assaut et sont puissamment armés. Un groupe d'intervention doit récupérer l'arme bactériologique. Pour cela, ils peuvent élaborer le plan suivant :

- deux membres du groupe, puissamment armés et lourdement protégés, attaquent de front les terroristes par l'accès nord de la pièce A. Leur rôle est de faire diversion en occupant autant de terroristes que possible à l'aide d'un feu nourri.
- deux autres membres du groupe profitent de cette diversion pour se faufiler vers l'accès arrière de la pièce.
- l'un doit rejoindre le local technique et y rester cantonné pour couper la lumière au moment opportun. Il est armé d'un pistolet silencieux pour éliminer tout terroriste qui se trouverait sur son chemin sans être détecté.
- l'autre est équipé de lunettes à infrarouge et doit profiter de l'effet de surprise provoqué par l'extinction de la lumière pour atteindre la pièce A par le conduit d'aération, récupérer l'arme bactériologique et rejoindre son camp le plus rapidement possible.
- celui qui a coupé la lumière doit le rejoindre pour protéger sa fuite au cas où il se ferait détecter.

Dans ce scénario, les sources d'incertitude sont nombreuses :

- une partie du bâtiment est mal connue, ce qui est représenté par des « ? » sur la figure 10.1.

[4] *waypoints*
[5] [Hart et al., 1968, Hart et al., 1972]
[6] [Donnart & Meyer, 1996, Donnart, 1998]

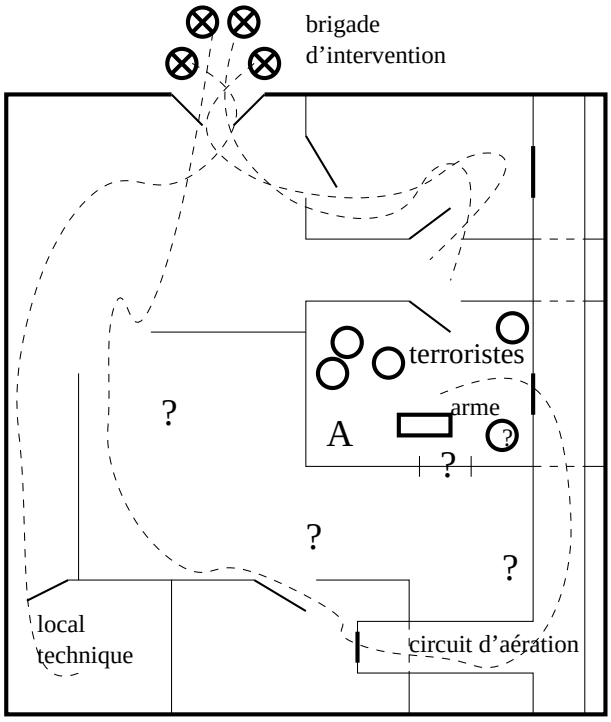


FIG. 10.1 – **Carte définissant un scénario de référence. Voir le texte pour une explication.**

– en particulier, les agents de la brigade d'intervention ne savent pas de façon sûre s'il y a une seconde porte à l'arrière de la pièce A.
– le nombre d'agents terroristes est incertain à une unité près.
– surtout, la stratégie des terroristes n'est pas connue : vont-ils se laisser prendre par la diversion ? Vont-ils déplacer l'arme bactériologique ? Ont-ils placé des hommes dans la zone mal connue ?

Ces sources d'incertitude impliquent que les agents soient capables de s'adapter individuellement et collectivement à la situation qu'ils vont rencontrer sur le terrain une fois engagés dans le feu de l'action. Ce scénario ne constitue évidemment qu'un exemple autour duquel il est possible de broder un nombre infini de situations.

Nous allons à présent nous efforcer de montrer comment la gestion de ce type de scénario implique les différentes techniques adaptatives que nous avons présentées dans le cadre de ce mémoire.

10.4 Adéquation des techniques adaptatives

Il apparaît clairement que mettre au point des agents susceptibles de se comporter intelligemment dans le cadre d'un tel scénario suppose de résoudre l'ensemble des problématiques que nous avons listées dans l'introduction de ce document, en énonçant notre programme de recherche, page 22, que nous reproduisons ici pour mémoire :

– la capacité à évoluer intelligemment dans des univers de grande taille dans lesquels les perceptions et les actions possibles sont nombreuses ;
– la capacité à adapter son comportement dans le cadre d'environnements continus, dans lesquels la perception et l'action peuvent être continues ;

- la capacité à se localiser, à s'orienter, à naviguer dans son environnement en l'absence de référentiel absolu et d'informations externes sur sa position ;
- la capacité à gérer l'incertitude sur des informations importantes, à cause des limites de la perception et de la connaissance des agents ;
- la capacité à faire face à un environnement dynamique dans lequel divers agents évoluent simultanément et sont susceptibles de modifier l'environnement et leurs propres stratégies au cours du temps ;
- la capacité à anticiper les comportements des agents pilotés par les êtres humains, et des autres agents logiciels ;
- la capacité à se coordonner à la fois dans l'espace, le temps et le choix des actions, afin d'atteindre des objectifs collectifs.

Mettons à présent en face de ces problématiques les éléments de réponse que nous avons développés dans les chapitres précédents :

- Pour apprendre dans des univers continus, les travaux de T. Gourdin décrits au chapitre 9, qui s'appuient sur ceux de F. Flacher décrits au chapitre 6, apportent une réponse limitée au problème spécifique de la navigation. Il faut généraliser cette approche à des actions autres que les déplacements. Ces travaux, augmentés peut-être des solutions *ad hoc* développées par N. Schneider, répondent en outre au second point qui concerne la localisation, l'orientation et la navigation.
- La dimension stochastique des processus de décision markoviens est adaptée par nature à la gestion des problèmes d'incertitude les plus simples. Face aux informations manquantes, par contre, il faut avoir recours à des extensions telles que celles que nous avons évoquées aux chapitres 5 et 8, tout en ayant conscience des limitations intrinsèques de ces extensions.
- Pour faire face à la diversité et à la richesse des perceptions et des actions, la capacité de généralisation des systèmes de classeurs décrits au chapitre 3, ou les propriétés de compacité des processus de décision markoviens factorisés décrits au chapitre 4, doivent jouer un rôle crucial. Reste à mieux réfléchir à l'intégration de ces travaux dans un cadre plus général proposant des réponses aux autres problématiques.
- Pour faire face au caractère dynamique ou non stationnaire des univers qui nous intéressent, nous comptons de façon encore assez vague sur les propriétés adaptatives des techniques que nous avons développées pour traiter les problèmes non markoviens, tout en ayant conscience des problèmes difficiles auxquels nous allons nous heurter. Il faut reconnaître que nous avons travaillé essentiellement pour l'instant sur le problème des perceptions ambiguës, qui est plus facile.
- Pour ce qui est de la capacité à anticiper le comportement des autres agents, les systèmes de classeurs à anticipation ou les réseaux bayésiens dynamiques apportent une réponse probablement trop générale pour être applicable de façon efficace à l'ensemble des problèmes. Il faudra peut-être se tourner vers des travaux plus spécifiques de l'anticipation du comportement des joueurs humains, tels que la thèse de C. Meyer [7] ou les travaux de J. Laird [8].
- La dernière capacité, centrée sur la coordination, s'appuiera à la fois les travaux relatifs à la coordination spatiale de F. Flacher et T. Gourdin, et sur le travail de T. Degris, qui se concentre plus spécifiquement sur le problème du choix des objectifs dans un cadre collectif. En particulier, le scénario que nous avons décrit fait ressortir trois contraintes qui pèsent sur le travail que devra réaliser T. Degris.

  • il faut être capable d'élaborer un plan avant de s'engager dans l'action, en présence d'informations incertaines et d'agents susceptibles de manifester des comportements dynamiques imprévus ;

[7] [Meyer, 1999]
[8] [Laird, 2000]

• il faut être capable d'affecter différentes missions à différents agents en fonction de leurs capacités, induites par leur équipement dans le scénario que nous avons décrit ;

• il faut être capable d'assurer la coordination du comportement des agents une fois l'action engagée, ce qui suppose de réadapter le plan au fil de l'évolution de la situation.
Ces trois points sont tous les trois difficiles. Rien ne permet d'affirmer qu'il sera possible de les traiter simultanément de façon satisfaisante dans le cadre d'une thèse unique. S'il faut choisir, c'est certainement le troisième point qui relève le plus directement du cadre des comportements adaptatifs.

10.5 Prise de recul

Utiliser des techniques adaptatives pour régler le comportement de créatures autonomes dans le cadre de jeux vidéos n'implique pas que les politiques suivies par ces créatures atteignent effectivement un optimum.

En effet, jouer contre un adversaire qui se comporte toujours « parfaitement » peut-être lassant pour un joueur humain. Plutôt que des outils d'optimisation, les techniques adaptatives doivent donc être vues comme un moyen privilégié pour renouveler en permanence l'intérêt du jeu en faisant en sorte que le comportement des créatures autonomes évolue en réponse à celui du joueur, si bien qu'une stratégie qui était gagnante lors d'une partie ne l'est plus forcément à la partie suivante.

Les phénomènes d'explosion combinatoire qui se manifestent dans le cadre de ces simulations complexes sont certainement des obstacles à la recherche d'une stratégie optimale, mais il n'est pas certain qu'ils s'opposent à l'application des techniques que nous avons proposées si nous considérons l'objectif moins ambitieux d'un renouvellement des comportements des créatures pilotées par l'ordinateur.

Reste que l'on attend tout de même de l'adaptation du comportement qu'elle donne l'illusion de l'intelligence, ce qui suppose d'atteindre des niveaux de performance acceptables et de s'y maintenir en dépit des changements de stratégies des joueurs. Par ailleurs, cet argument qui vaut dans le cadre des jeux vidéo ne s'applique pas forcément pour d'autres simulations complexes où la recherche d'un comportement optimal reste nécessaire.

Quoi qu'il en soit, il ne nous semble donc pas possible aujourd'hui d'apporter une réponse théorique définitive aux problématiques impliquées par cet objectif, que l'on considère l'optimisation ou un critère de satisfaction moins difficile. Nous allons tirer un bilan des réponses scientifiques que nous proposons face à ces difficultés, tout en ayant conscience de la nécessité d'avoir recours à des solutions éventuellement *ad hoc* pour résoudre les problèmes pour lesquels ces réponses ne suffiront pas.

10.6 Bilan et perspectives scientifiques

Suite au tour d'horizon que nous avons réalisé dans le cadre de ce mémoire, il apparaît clairement que les difficultés centrales que nous allons rencontrer avec les applications que nous visons proviennent du caractère continu et du caractère hautement non stationnaire de ces applications. En effet, dans la plupart des jeux vidéo auxquels nous nous intéressons, toute les sources de variabilité du problème sont présentes :

– les personnages sont dotés de variables internes qui évoluent au cours du temps, tels que leur nombre de points de vie, leur équipement, etc. ;
– ils agissent dans un cadre multi-agents où le comportement des agents pilotés par des joueurs humains peut être hautement imprévisible ;

– les sources de récompense peuvent varier au cours du temps selon l'évolution des scénarios auxquels ils sont confrontés.

Nous avons indiqué qu'il existait des techniques pour aborder ces différentes difficultés, mais toutes ces techniques sont confrontées à des problèmes d'explosion combinatoire et ne fonctionnent en pratique que sur des problèmes de petites dimensions. Or les problèmes que nous visons ont toutes les chances d'être de grande taille. Donc il faudra non seulement trouver le moyen d'intégrer ces techniques au sein d'une architecture générale, mais aussi trouver le moyen de les rendre moins fragiles vis-à-vis de l'augmentation de la taille des problèmes.

Au moment de clore ce mémoire, nous voulons exprimer la conviction que l'approche reposant sur les processus de décision markoviens factorisés est destinée à jouer un rôle central dans notre programme de recherche, tant face au problème consistant à intégrer les différentes techniques que face aux problèmes liés à la taille des espaces définis par les applications. En effet, cette approche permet de traiter des problèmes de grande taille en bâtissant des modèles exponentiellement plus compacts que les modèles tabulaires et elle permet de tirer parti de la construction de ces modèles pour accélérer l'apprentissage et la propagation des valeurs. En outre, des travaux ont déjà été entrepris pour l'appliquer dans des cadres multi-agents [9]. Cependant, il faut avoir conscience du fait que bâtir un modèle des transitions dans un cadre non stationnaire est beaucoup plus difficile que dans le cadre stationnaire, pour lequel les algorithmes de référence ont été conçus.

D'autres pistes joueront sans doute aussi un rôle important dans notre réflexion.

– Comme le souligne de façon très pertinente Z. Guessoum [10], dès lors que le nombre d'agents devient grand, il n'est pas raisonnable de se doter d'une représentation dans laquelle le comportement de chaque agent est pris en compte individuellement. Des opérateurs d'agrégation doivent être mis en œuvre pour abstraire des comportements de groupe, moins coûteux à décrire à partir d'ensembles de comportement individuels.
– Par ailleurs, face à des situations complexes, tenter de casser la complexité, en faisant apparaître des niveaux de hiérarchie dans la représentation, est une idée qui a largement fait ses preuves. Bien que nous en ayons évoqué la possibilité au chapitre 8, nous avons consacré peu d'attention jusqu'ici aux approches hiérarchiques, car elles requièrent souvent une part d'intervention de concepteurs humains qui va à l'encontre de l'idéal d'autonomie et d'adaptabilité que nous avons défendu dans le cadre de ce mémoire. Pour aborder cette problématique, des travaux comme ceux de J.-Y. Donnart [11] sont destinés à nous servir de source d'inspiration.
– Enfin, compte tenu de la complexité des applications que nous souhaitons traiter, il ne va pas de soi qu'un cadre théorique unifié soit à même de proposer des solutions à l'ensemble des problèmes qui vont se poser. Face à cette situation, une approche légitime consiste à mettre en œuvre les techniques d'ingénieur non adaptatives qui ont fait leurs preuves sur les problèmes en question, pour combler les lacunes des techniques adaptatives. Cependant, une autre approche, plus conforme à nos objectifs fondamentaux, consiste à considérer que les systèmes biologiques ont évolué de telle façon qu'ils disposent, grâce à diverses structures cérébrales, de toutes les fonctionnalités dont nous pourrions avoir besoin, si bien que se tourner vers ces structures cérébrales, à la recherche de solutions plus conformes aux modèles biologiques, est sans doute une bonne idée.

[9] [Guestrin et al., 2001, Guestrin et al., 2002a]
[10] [Guessoum, 2003]
[11] [Donnart, 1998]

10.7 Retour vers l'intelligence des êtres vivants

Dans les derniers chapitres de ce mémoire, nous avons décrit la partie la plus prospective des travaux de recherche que nous avons entrepris. En les rédigeant, nous nous sommes efforcé de ne pas masquer nos doutes quant au degré de maturité et d'efficacité des techniques que nous proposons pour traiter les problèmes difficiles que nous abordons. Le cadre général des processus de décision markoviens s'est avéré adapté pour la résolution de problèmes simples dans des univers discrets, il ne va pas de soi que ses extensions au cadre continu, au cadre multi-agents ou au cadre factorisé constituent une réponse adéquate à ces problématiques plus difficiles. Partout, les problèmes combinatoires limitent l'applicabilité des techniques, même si l'on s'efforce de retarder leur explosion par tous les moyens.

Face à ces doutes, le remède universel que préconise l'approche animat consiste à se tourner à nouveau vers les mécanismes qui opèrent au sein des êtres vivants. S'il s'avère que les techniques que nous avons mises en avant tout au long de ce mémoire ne permettent pas de répondre à tous les problèmes que nous avons posés, il doit être clair que les êtres vivants, eux, sont capables de les résoudre, si bien que l'on doit trouver, en examinant de plus près leur fonctionnement, les solutions qui nous auront fait défaut jusque-là.

Nous avons dit en introduction de ce mémoire que l'approche animat se donnait deux objectifs simultanés :

- d'une part, comprendre l'intelligence des êtres vivants, ce qui se traduit concrètement par la mise au point de modèles proches de la biologie, que nous qualifions dans notre jargon de « bio-inspirés » ;
- d'autre part, mettre au point des systèmes adaptatifs fonctionnels pour résoudre des problèmes applicatifs précis, dans une démarche qui relève de la modélisation mathématique et de l'ingénierie.

Un projet scientifique tel que le projet PSIKHARPAX, mené par nos collègues de l'AnimatLab, relève de la première approche. Il vise précisément à examiner comment des mécanismes directement extraits du fonctionnement du vivant peuvent contribuer à résoudre une partie des problèmes que nous avons soulevés. La structure de contrôle du robot PSIKHARPAX est constituée de réseaux de neurones formels, dont l'architecture générale suit d'aussi près que possible ce que l'avancée fulgurante des connaissances en neurophysiologie nous apprend sur l'architecture neuronale du cerveau du rat. Cette architecture dote le robot de capacité de navigation et de localisation en environnement continu, comme elle le dote de la capacité à rejoindre des emplacements spécifiques pour satisfaire des objectifs liés à sa survie. Elle incorpore en outre des capacités d'apprentissage, inspirées de la physiologie, qui permettent au modèle de s'adapter aux caractéristiques d'un environnement particulier.

Il y a donc matière à approfondir sensiblement les interactions entre la dimension bio-inspirée et la dimension plus mathématique de l'approche animat. En guise de conclusion de ce mémoire, nous voudrions faire ressortir les éléments suivants, qui nous occuperont certainement bien au-delà du programme de recherche limité que nous avons énoncé dans les chapitres précédents.

- Tout d'abord, il y a matière à mener des comparaisons empiriques des propriétés respectives des systèmes mis au point dans le cadre de ces deux approches. Un système dont le modèle est tiré de la biologie se comporte-t-il mieux qu'un système reposant sur des modèles formels face à certaines catégories de problèmes ? L'un des deux consomme-t-il moins de ressources en mémoire et/ou en temps de calcul ? Lequel fournit les solutions les plus efficaces ? Les plus intelligibles ? Les plus faciles à adapter à de nouveaux environnements ? Les plus robustes face à une augmentation significative de la complexité des problèmes posés ? Toutes ces comparaisons n'ont pas été faites et méritent de l'être. Cela peut être fait dans un cadre spécifique au sein de l'AnimatLab, car nous disposons de différentes architectures de contrôle qui ne demandent qu'à être comparées dans un cadre expérimental unique. Cependant, au delà de ces comparaisons ponctuelles, il serait intéressant de déterminer les enseignements plus généraux que l'on peut tirer d'une confrontation des deux approches.

– En effet, il faut avoir conscience du fait que les modèles bio-inspirés qui sont élaborés dans le cadre de projets comme PSIKHARPAX restent des modèles, c'est-à-dire des objets informatiques abstraits que l'on peut étudier sur le plan de leurs propriétés mathématiques. Dans le cas qui nous intéresse, ces modèles reposent essentiellement sur des réseaux de neurones formels, qui sont vus dans le cadre théorique de l'apprentissage par renforcement comme une classe d'approximateurs, dont un certain nombre de propriétés mathématiques commencent à être bien comprises. Il nous semble donc que l'approche formelle des techniques adaptatives peut fournir aux recherches fondées sur l'observation des structures neuronales présentes chez les êtres vivants des outils théoriques pour mieux comprendre le fonctionnement de ces réseaux biologiques. En retour, s'il s'avère que les réseaux biologiques sont dotés de capacités adaptatives que les techniques formelles n'ont pas pu obtenir par leur axe de développement propre, comprendre dans le détail ce qui procure à ces réseaux de telles propriétés doit aider à modifier les cadres formels, pour que les techniques qui en découlent fassent les progrès correspondants.

Une réflexion de ce type est en voie d'être menée à l'AnimatLab, autour des architectures acteur-critique sur lesquelles travaille M. Khamassi [12] dans le cadre du projet PSIKHARPAX. Notre sentiment est que ce dialogue n'en est qu'à son début et que l'approche de la modélisation suivie dans notre laboratoire a tout à gagner à renforcer et élargir ce dialogue à l'ensemble des propriétés des systèmes adaptatifs.

[12] [Khamassi et al., 2004]

Annexe A

Curriculum Vitæ

A.1 État civil

Adresse	:	4 rue André Chénier
	:	92130 Issy-les-Moulineaux
tél dom.	:	01.47.36.74.38
tél prof.	:	01.44.27.88.53
courriel	:	Olivier.Sigaud@lip6.fr

Né le 3 août 1968, marié, un enfant, libéré des obligations militaires

A.2 Formation et Diplômes

A.2.1 Formation scientifique

2004	:	Thèse d'habilitation à diriger des recherches de l'Université PARIS 6
Spécialité	:	Informatique
Titre	:	« Comportements adaptatifs pour les agents dans des
	:	environnements informatiques complexes »
Soutenance	:	le 10 décembre 2004
Rapporteurs	:	François Charpillet (DR LORIA, Nancy)
	:	Jean-Paul Haton (Prof. Univ. Henri Poincaré (Nancy))
	:	Stewart Wilson (Prof. UIUC (Urbana-Champaign, Illinois))
Examinateurs	:	Jacques Malenfant (Prof. LIP6)
	:	Jean-Arcady Meyer (DR1 CNRS, LIP6)
	:	Patrice Perny (Prof. LIP6)
	:	Marc Schoenauer (DR INRIA-Future, Orsay)

93-96	:	Thèse de doctorat de l'Université PARIS 11
Spécialité	:	Informatique (Intelligence Artificielle)
Titre	:	« Apprentissage : de la commande aux comportements »
Labo. d'acceuil	:	Etablissement Technique Central de l'Armement (DGA)
Soutenance	:	le 5 juin 1996, Mention Très Honorable
Directeur	:	Dominique Luzeaux
Rapporteurs	:	Jacques Dubucs, Jean Sallantin
Examinateurs	:	Philippe Gaussier, Jean-Arcady Meyer, Bertrand Zavidovique

1992 : DEA d'intelligence artificielle, Institut Blaise Pascal PARIS 6 (Mention Bien)
Spécialités : Apprentissage (J.-G. Ganascia) et I.A. Distribuée (J. Ferber)
1991 : Ingénieur ISEN (Institut Supérieur d'Electronique du Nord) option informatique
1986 : Baccalauréat C, matières scientifiques

A.2.2 Formation générale

96-02	:	Thèse de doctorat de l'Université PARIS I
Spécialité	:	Philosophie
Titre	:	« Automatisme et Subjectivité : l'anticipation au cœur de l'expérience »
Soutenance	:	le 10 décembre 2002, Mention Très Honorable et félicitations du jury
Directeur	:	Jacques Dubucs
Rapporteurs	:	John Stewart, Bernard Victorri
Examinateurs	:	Jacques Droulez, Jean Petitot

1994-1995 : DEA d'histoire et philo. des sciences, PARIS I (Mention Très Bien)
Directeur du mémoire : Jacques Dubucs
1993 : Licence de philosophie, PARIS I Sorbonne
1988 : Cambridge First Certificate, Grade B
1985 : Baccalauréat A1 (Mention Bien)

A.2.3 Expérience professionnelle

Depuis 9/2001 : maître de conférences en informatique à l'Université PARIS 6
9/95 → 9/01 : ingénieur de recherche chez Dassault Aviation
1995 → 2001 : Vacataire dans l'enseignement supérieur

Note

Chez Dassault Aviation, j'étais chargé à partir de 1998 au sein du Département des Etudes Scientifiques Amont de la veille technologique dans le domaine des comportements adaptatifs. Cette situation professionnelle particulière m'a permis d'acquérir une bonne vue d'ensemble du domaine et une expérience solide en gestion de projet et en encadrement. Je considère donc qu'il s'agit d'une expérience très positive pour la suite de ma carrière. Par contre, elle a pesé négativement sur mon activité de publication, peu encouragée dans mon cadre de travail, même si je me suis battu pour disposer d'un dossier me permettant de rejoindre une carrière académique par la suite.

A.3 Enseignement

Mon nombre total d'heures est d'environ 1100 heures d'équivalent TD, en comptant mon service 2004-2005 en cours.

A.3.1 A l'Université PARIS 6

Depuis septembre 2001, j'effectue un service complet à l'Université Pierre et Marie Curie (PARIS 6). Mes heures se répartissent comme suit (les volumes sont convertis en équivalent TD) :

En 2004-2005 (prévisionnel)

Thème	Public	Volume	Effectif	Nature
Simulation de systèmes complexes	M2	6h	31	Cours
Approche animat	M2	5h	30 ?	Cours
Ingéniérie du Logiciel	M1	40h	33	TD + TME
Programmation par Objets	L3	8h + 72h	220 ?	Cours+TD+TME
Projet d'initiation	L2	36h	20 ?	TD+TME
Prise en charge de 3 projets	M1	12h	6	Encadrement

En 2003-2004

Thème	Public	Volume	Effectif	Nature
Modèles de Langages à Objets	Maîtrise	88h	51	TD+ TME
Ingéniérie du Logiciel	Maîtrise	88h	45	TD + TME
Cours JAVA option Info.	DEUG MIAS 2	30h	28	Cours+TD
Prise en charge de 2 projets	Maîtrise	12h	10	Encadrement

En 2002-2003

Modèles de Langages à Objets	Maîtrise	44h	30	TD + TME
Ingéniérie du Logiciel	Maîtrise	88h	40	TD + TME
Cours JAVA option Info.	DEUG MIAS 2	30h	28	Cours+TD
Prise en charge de 6 projets	Maîtrise	30h	10	Encadrement

En 2001-2002

Programmation Objets Distribuée	Maîtrise	36h	25	TD + Projet
Génie logiciel	Maîtrise	72h	31	TD + TME
Machine et représentation	DEUG MIAS 2	30h	18	TD+TP
Système et Internet	DEUG MIAS 2	38h	18	TD+TP
Cours JAVA option Info.	DEUG MIAS 2	30h	32	Cours+TD

A.3.2 A l'ENSTA

Année	Thème	Public	Volume	Effectif	Nature
2004-2005	JAVA et UML	ENSTA 2	42h	30	cours+TD+Projet
2003-2004	JAVA et UML	ENSTA 2	42h	33	cours+TD+Projet
2002-2003	JAVA et UML	ENSTA 2	42h	32	cours+TD+Projet
2001-2002	JAVA et UML	ENSTA 2	42h	35	cours+TD+Projet
2000-2001	JAVA et UML	ENSTA 2	21h	65	cours+TD+Projet
1999-2000	C++ et UML	ENSTA 3	24h	31	cours+TD+Projet
1999-2000	C++ et UML	ENSTA 2	21h	138	cours+TD+Projet

Total : 234 heures

Depuis 1998, je suis **professeur responsable** du cours « Génie Logiciel et Programmation Orientée Objets » à l'ENSTA (Ecole Nationale Supérieure des Techniques Avancées). Cette charge comprend l'élaboration du cours, des TDs et d'un projet associé, d'un examen et d'une épreuve de rattrapage, la rédaction des polycopiés de cours, la prestation des cours, l'encadrement d'un groupe de TD et des projets, ainsi que le recrutement et la coordination de l'ensemble des chargés de TD.

Chaque groupe de TD (environ 15 élèves ingénieurs) constitue une équipe de projet, la dimension collective du développement en équipe étant au cœur de l'enseignement méthodologique.

Documents pédagogiques

Site web associé au cours : **http ://animatlab.lip6.fr/Sigaud/teach/in204**

• « *Introduction à la modélisation orientée objets avec UML* », support du cours « Génie logiciel et programmation orientée objets » de l'ENSTA, 2004, 49 pages.

• « *Introduction à la conduite de projets logiciels* », support du cours « Génie logiciel et programmation orientée objets » de l'ENSTA, 2004, 26 pages.

• « *Introduction à la programmation orientée objets en* JAVA », support du cours « Génie logiciel et programmation orientée objet » de l'ENSTA, 2004, 82 pages.

A.3.3 Divers

Année	Thème	Public	Volume	Effectif	Nature
2000-2001	JAVA	DESS	28h	11	cours+TD
1998-1999	UML	Officiers ENSTA	4h	~ 15	cours
1994-1995	Syst. Expl.	ENSTA 2	14h	~ 15	TD,TP
1997-1998	Lang. Nat.	FIIFO 3	7h	~ 15	Projet

Total : 53 heures

A.3.4 Synthèse

Mon activité d'enseignement porte principalement sur la Programmation Orientée Objets et le Génie Logiciel. Le rythme soutenu auquel évoluent ces domaines à l'heure actuelle impose de maintenir un niveau de compétence

élevé, ce qui implique une relative spécialisation. Ceci étant, je suis capable d'enseigner à un niveau moindre la plupart des sous-disciplines de base de l'informatique, comme une partie de mon service le démontre.

L'organisation des enseignements au LIP6 ne m'a guère permis jusqu'à présent d'enseigner dans les domaines relatifs à ma spécialité de recherche devant des étudiants de DEA (ou M2). D'une part, cette situation est en train d'évoluer sous l'effet de la réforme LMD et, d'autre part, j'ai partiellement compensé cet état de fait en profitant de mes enseignements en programmation orientée objets et génie logiciel pour faire réaliser aux étudiants des programmes illustrant mon domaine de prédilection.

A.4 Encadrement

Toutes mes activités d'encadrement jusqu'en 2001 compris ont été menées chez Dassault Aviation. Mon arrivée au LIP6 coïncide avec la mise en place d'un encadrement tournée vers le développement du projet KODAMAT dont j'ai la charge.

A.4.1 Thèses

Année	Nom	Type	Thème
2003-2006	Thierry Gourdin	Thèse MNER	Apprentissage de la coordination spatiale
2003-2006	Thomas Degris	Thèse DGA	Apprentissage de stratégies collectives
2002-2005	Fabien Flacher	Thèse CIFRE	Comportements collectifs adaptatifs
1999-2002	Pierre Gérard	Thèse CIFRE	Systèmes de classeurs à anticipation

Pierre Gérard est à présent maître de conférences au LIPN, à Villetaneuse. Les autres doctorants sont encore en thèse.

A.4.2 DEA

Année	Nom	Type	Thème
2004	Cédric Lutz	DEA IARFA	Planification dans les FMDPs
2004	Julien Villers	DEA IARFA	Systèmes de classeurs pour les jeux vidéo
2003	Nicolas Schneider	DEA IARFA	Navigation autonome pour les jeux vidéo
2002	Th. Gourdin	DEA IARFA	Sélection de l'Action
2001	Fabien Flacher	DEA IARFA	Comportements collectifs adaptatifs
1999	Chr. Lannic	DEA + Ing.	Modélisation du vol

J'ai co-encadré Cédric Lutz avec Pierre-Henri Wuillemin, Maître de Conférences au LIP6

A.4.3 Autre

Année	Nom	Type	Thème
2002-2003	Thomas Degris	Stage de recherche	Modèles de cellules de direction de la tête
2000	Fabien Flacher	Maîtrise	Algorithmes génétiques
1998-1999	Rémi Munos	PostDoc CMU	Apprentissage par renforcement

En outre, depuis 1992, j'ai encadré 12 stages d'élèves ingénieurs (ENSTA, Ecole Navale, FIIFO, ENAC) au fil de mes affectations successives.

A cette liste officielle s'ajoutent mes interactions avec Samuel Landau, qui sont à classer entre des rubriques « encadrement informel » et « collaboration ».

A.5 Projet de recherche

A.5.1 Problématique scientifique

Mon activité de recherche porte sur la mise au point de mécanismes adaptatifs informatiques dont les modèles sont tirés de l'observation des être vivants.

Les deux principales classes de méthodes adaptatives auxquelles je m'intéresse sont les algorithmes évolutionnistes et l'apprentissage par renforcement. La première classe regroupe des méthodes d'optimisation qui ont montré en pratique leur robustesse et leur généricité, mais dont l'étude théorique reste encore largement à mener. La seconde classe concerne des méthodes beaucoup plus clairement formalisées dans le cadre mathématique des processus de décision markoviens et donne lieu à des algorithmes dont la convergence est prouvée dans des conditions relativement restrictives.

Ma réflexion théorique se concentre sur deux axes principaux.

• D'une part, je m'intéresse à l'intégration des mécanismes évolutionnistes et des mécanismes d'apprentissage par renforcement au sein d'un cadre général, que je nomme « apprentissage par population », destiné à combiner les forces de ces différentes approches. L'exemple paradigmatique d'une telle combinaison est fourni par les systèmes de classeurs, qui sont au cœur de ma spécialité.

• D'autre part, je m'efforce d'introduire des techniques d'apprentissage par renforcement dans le cadre formel des réseaux bayésiens dynamiques, qui permettent de bâtir des représentations compactes des problèmes en jouant sur la structure de dépendance entre les variables qui définissent les états du problème.

Ces deux axes de recherches sont destinés à aborder avec des méthodes adaptatives spécialisées des domaines d'application qui sont hors de portée des méthodes classiques qui relèvent de ce domaine.

A.5.2 Problématique appliquée

Dans ce contexte scientifique général, je m'intéresse plus spécifiquement à la mise au point de comportements adaptatifs pour des agents plongés dans des simulations réalistes de grande envergure tels que les jeux vidéos multijoueurs en trois dimensions ou les simulations militaires. La complexité de ces simulations impose de traiter des difficultés mal résolues par des algorithmes adaptatifs existants, qu'elles soient liées à la taille des problèmes, à leur caractère continu, à leur caractère non stationnaire, ou aux informations incomplètes fournies aux agents. Les travaux que je mène en compagnie de trois doctorants et quelques stagiaires dans le cadre du projet KODAMAT visent à repousser les limites induites par l'existence de ces difficultés.

Le domaine d'application dans lequel ces recherches trouvent leur cadre le plus naturel est celui des jeux vidéo. En effet, les logiciels de jeu fournissent des plates-formes de test à la fois complexes, dynamiques, peuplées de nombreux agents, et dans lesquelles l'atteinte d'objectifs peut être quantifiée précisément. Lorsque ces logiciels sont des jeux en trois dimension et en temps réel, ils proposent une alternative intéressante à la robotique mobile pour l'expérimentation des techniques adaptatives.

Réciproquement, la mise au point de comportements adaptatifs pour les agents autonomes de ces simulations répond à un besoin industriel qui va grandissant au fur et à mesure que les ordinateurs deviennent plus puissants et les utilisateurs plus exigeants.

A.5.3 Travaux réalisés, apports et visibilité

Le premier domaine auquel je me suis intéressé pour attaquer des problèmes de grande taille dans le cadre de l'apprentissage par renforcement est celui des systèmes de classeurs. La thèse de P. Gérard, sur les systèmes de classeurs à anticipation, nous a permis de mettre au point le système MACS, qui propose l'algorithme le plus performant dans ce domaine restreint. Ce travail nous a valu deux publications de journaux internationaux et me permet de jouer un rôle actif au niveau international dans la communauté scientifique qui s'intéresse aux systèmes de classeurs. Par ailleurs, P. Gérard a profité de son excellent dossier de recherche pour obtenir un poste de Maître de Conférences au LIPN. Enfin, ces travaux, complétés par d'autres travaux décrits dans la suite de ce document, me donnent une visibilité nationale importante dans la communauté de recherche sur l'apprentissage par renforcement, la planification et la résolution de problèmes multi-agents dans le cadre mathématique des processus de décision markoviens. J'ai notamment organisé un atelier réunissant une trentaine de chercheurs sur ce thème en 2004, et je suis éditeur en collaboration avec F. Charpillet, F. Garcia et P. Perny, d'un numéro spécial du journal national RIA qui fait suite à cet atelier.

Mon deuxième axe de recherche concerne le traitement des problèmes de décision dans lesquels une mémoire du passé récent est indispensable pour prendre les meilleures décisions. En établissant un rapport entre le système évolutionniste ATNoSFERES conçu par S. Landau dans le cadre de sa thèse et les systèmes de classeurs, j'ai orienté les travaux de S. Landau vers une évaluation comparative de son approche et de celles de différents systèmes de classeurs qui a débouché sur une publication dans « *Information Sciences* ». Cette publication met en évidence le caractère plus général du formalisme manipulé par ATNoSFERES vis-à-vis de celui des systèmes de classeurs. Ce travail se poursuit dans le cadre de ma réflexion visant à intégrer apprentissage et évolution par la mise au point du système FACS, dont les résultats ne sont pas encore publiés.

Mon troisième axe de recherche concerne la mise au point de mécanismes de coordination spatiale dans des univers continus habités par plusieurs agents. La thèse de F. Flacher, financée par Dassault Aviation, a pour objectif la coordination spatiale d'aéronefs dans le cadre de simulations militaires. Le système résultant, GACS, découvre automatiquement des champs de force d'attraction et de répulsion qui assurent la coordination des déplacements d'un ensemble d'agents au sein de leur environnement. Les paramètres de ces champs de force sont optimisés par un algorithme génétique multi-critères simple qui permet à GACS de traiter des applications très diverses. Ce travail en cours a fait l'objet de publications en conférences internationales et en journal national, des articles à destination de journaux internationaux étant en préparation.

A.5.4 Travaux en cours et perspectives

Les trois axes de recherche précédents me servent de base pour construire un système ambitieux capable de régler des comportements adaptatifs dans des univers continus, de grande taille et rendus non stationnaires par la présence de nombreux agents.

Le cœur de mon effort dans ce sens consiste à transposer les recherches précédentes sur les systèmes de classeurs à anticipation dans le cadre mathématique des réseaux bayésiens dynamiques. Ce cadre mieux formalisé et plus général que celui des systèmes de classeurs est en pleine expansion au niveau international dans la vaste communauté qui s'intéresse aux modèles de Markov, avec le développement des recherches sur les processus de décision markoviens factorisés. Ma contribution, en cours de concrétisation suite au stage de DEA de C. Lutz, va consister à introduire les techniques d'apprentissage incrémental tirées des systèmes de classeurs dans ce domaine où les questions relatives à l'apprentissage ont rarement été abordées pour l'instant. Cet effort devrait me permettre de rejoindre une communauté de chercheurs dont la visibilité internationale est très importante. Il devrait surtout me

donner les moyens de traiter des problèmes de grande taille dans un cadre mieux formalisé et avec des algorithmes plus économes que précédemment.

Examiner les systèmes de classeurs au travers de la grille de lecture formelle fournie par les réseaux bayésiens dynamiques permet de comprendre le recours aux algorithmes génétiques dans les systèmes de classeurs comme une heuristique destinée à résoudre un problème d'optimisation de la taille d'une représentation, dont la résolution formelle est soumise à une explosion combinatoire.

Sur la base de ce constat, je suis conduit à examiner la combinaison d'algorithmes génétiques et d'apprentissage par renforcement qui caractérise les systèmes de classeurs dans un cadre plus général que je nomme « apprentissage par population » et dont les propriétés formelles restent largement à explorer.

Par ailleurs, cette réflexion m'a conduit à spécifier deux nouvelles instances du mécanisme d'apprentissage par population, à savoir le système FACS, évoqué à la section précédente, et une proposition pour introduire de l'apprentissage par renforcement dans le cadre du travail de F. Flacher, proposition qui n'a pas été concrétisée pour l'instant.

Reprendre la problématique de la coordination spatiale en y introduisant des mécanismes d'apprentissage par renforcement est l'objet de la thèse de T. Gourdin. Ce travail doit déboucher sur la mise au point automatique de mécanismes de navigation collective dans des environnements continus. La thèse de T. Degris, complémentaire, traite de l'apprentissage de la coopération multi-agents. Cette thèse est financée par la DGA et se donne pour objectif la coordination de micro-drones lors de missions réalisées dans des bâtiments.

Ces deux thèses s'appuient sur mes travaux relatifs à la résolution de problèmes de décision markoviens factorisés. Si les objectifs initiaux qui les animent sont atteints, nous devrions disposer à terme d'une plate-forme capable de traiter des problèmes de grande taille, dans des environnements continus et exigeants des mécanismes de coopération entre les agents, ce qui est exactement le cadre imposé par l'objectif applicatif de produire des agents autonomes et adaptatifs pour les jeux vidéos et d'autres catégories de simulations complexes.

A.6 Collaborations scientifiques, projets contractuels

- Responsable du projet KODAMAT à l'AnimatLab
  Objet : mise au point de comportements adaptatifs pour les créatures autonomes dans les jeux vidéo.
  Envergure actuelle : 3 doctorants, quelques stagiaires

- Participation au projet PSIKHARPAX, financement CNRS (programme ROBEA), 2001-2003.
  Objet : construction d'un robot dont l'organe de commande est directement inspiré de la biologie du rat.
  Partenaires : LPPA (Collège de France), LIRMM (Montpellier), sociétés BEV et Wany SA.

- Participation au projet SMAARTS, financement LIP6, 2004-2005,
  Objet : apprentissage de stratégies collectives pour les jeux de stratégie temps-réel.
  Partenaires : Vincent Corruble et Pierre-Henri Wuillemin (LIP6).

- Participation annuelle au groupe de travail « Processus de Décision Markoviens et Intelligence Artificielle », financement partiel du GDR I3. Organisation d'ateliers et de numéros spéciaux (JEDAI, RIA).

- Gestion des relations avec Dassault Aviation pour le pôle « Intelligence Artificielle » du LIP6.

A.7 Animation scientifique

A.7.1 Organisation de colloques et séminaires

- En mai 2004, co-organisation avec P. Perny (LIP6) et F. Garcia (INRA) de PDMIA'04 au LIP6.
- En 2004, co-organisateur avec M. Butz et S. Swarup de la seconde édition du Workshop « Anticipatory Behavior in Adaptive Learning Systems » à Los Angeles.
- En 2002, co-organisateur avec M. Butz et P. Gérard du Workshop « Anticipatory Behavior in Adaptive Learning Systems » à Edinburgh.
- Depuis 2003, co-organisateur (avec Vincent Corruble) du séminaire « IA et jeux vidéos » du LIP6.

A.7.2 Edition d'ouvrages et numéro spéciaux

- Fin 2004, Co-éditeur d'un numéro spécial dans RIA (Revue d'Intelligence Artificielle) suite à PDMIA'04, avec P. Perny (LIP6), F.Charpillet (INRIA) et F. Garcia (INRA).
- En 2003, co-éditeur avec M. Butz et P. Gérard de l'ouvrage « Anticipatory Behavior in Adaptive Learning Systems », Springer-Verlag, version augmentée des actes du colloque ci-dessous.

A.7.3 Jury de thèses

- Décembre 2004, thèse de M. Métivier, « Méthodes évolutionnaires et apprentissage : apprentissage et imitation dans le cadre des systèmes de classeurs », LIAP5, Paris 5.
- Septembre 2004, thèse de K. Verbeeck, « *Coordinated Exploration in Multi-Agent Reinforcement Learning* », VUB, Bruxelles.

A.7.4 Participation à des commissions

- Membre externe (suppléant) de la commission de spécialiste de l'Université Henri Poincaré (Nancy) en section 27.
- Membre (suppléant) du conseil scientifique du LIP6.

A.7.5 Membre de comités de programme et relecteur

Numéros spéciaux de revues

- En 2003, « Journal of Evolutionary Computation » sur les systèmes de classeurs ;
- En 2001, « Extraction des connaissances et Apprentissage » (Hermès Science), sur « Apprentissage et évolution : théories et applications » ;
- En 2001, « Technique et Science Informatique » (Hermès Science), sur la vie artificielle.

Colloques internationaux

- From Animals to Animats (SAB) de 2000 à 2004 (bi-annuel),

- Genetic and Evolutionary Computation Conference (GECCO) de 2001 à 2004 (annuel),

– International Workshop on Learning Classifier Systems (IWLCS) de 2002 à 2004 (annuel).

Autres

J'ai été relecteur d'articles à diverses occasions pour « Journal of Evolutionary Computation », « IEEE Transactions on Evolutionary Computation », « Revue d'Intelligence Artificielle » et « Technique et Science Informatique ».

A.8 Publications

Synthèse

Revues internationales	4 (+1)
Revues nationales	2
Revues électroniques	1
Edition d'ouvrage	1
Conférences internationales	23
Conférences nationales	4
Chapitres	3
Séminaires et ateliers	6

Articles de revues internationales avec comité de lecture

1. « *Combining Latent Learning and Dynamic Programming in MACS* », Pierre Gérard, Jean-Arcady Meyer et Olivier Sigaud, 2005, *European Journal of Operational Research*, vol. 160, 614-637, Elsevier.
2. « *A Comparison between ATNoSFERES and Learning Classifier Systems on non-Markov problems* » Samuel Landau et Olivier Sigaud, *Information Sciences*, acceptée, 36 p. (sortie prévue début 2005).
3. « *Rapid response of head direction cells to reorienting visual cues : A computational model.* » Thomas Degris, Olivier Sigaud, Sid Wiener et Angelo Arleo, 2004, Neurocomputing, vol. 58-60C, 675-682, Elsevier.
4. « *YACS : a New Learning Classifier System Using Anticipation* », Pierre Gérard, Wolfgang Stolzmann et Olivier Sigaud, 2002, *Journal of Soft Computing, special issue on Learning Classifier Systems*, 6 (3-4), 216-228, Springer-Verlag.

Articles de revues internationales avec comité de lecture (en philosophie)

1. Analyse critique (recension) de « *Self-Awareness and Alterity* » (Dan Zahavi, Evanston, Northwestern University Press, 1999), Olivier Sigaud et Natalie Depraz, 2000, *Alter*.

Articles de revues nationales avec comité de lecture

1. « Coordination spatiale émergente par champs de potentiel », Fabien Flacher et Olivier Sigaud, 2003, Numéro spécial de la revue « Technique et Science Informatiques » sur la Vie Artificielle, J.-A. Meyer et A. Guillot éditeurs, Hermès, 22(2), 171-195
2. « Généralisation et Apprentissage Latent dans les Systèmes de Classeurs », Pierre Gérard et Olivier Sigaud, 2001, Numéro spécial de la revue « Extraction des connaissances et apprentissage » sur le thème : Apprentissage automatique et évolution artificielle, 3(1), 87-114, Hermès

Articles de revues électronique nationales avec comité de lecture

1. « Apprentissage par renforcement indirect dans les systèmes de classeurs » Olivier Sigaud et Pierre Gérard, 2004, JEDAI.

Edition d'ouvrages

1. LNCS 2684 : « *Anticipatory Behavior in Adaptive Learning Systems* », Martin Butz, Olivier Sigaud, et Pierre Gérard, 2003, 301 p., Springer-Verlag.

Conférences internationales avec comité de lecture

Conférences majeures (>100 participants)

1. « *Improving MACS thanks to a comparison with 2TBNs* » Olivier Sigaud, Thierry Gourdin et Pierre-Henri Wuillemin, Fifth annual Genetic and Evolutionary Computation Conference (GECCO'04), Seattle, WA, 810-823, Springer Verlag.
2. « *BASC, a Bottom-up Approach to automated design of Spatial Coordination* » Fabien Flacher et Olivier Sigaud, 2004, Proceedings of the Eighth International Conference on Simulation of Adaptive Behavior (SAB-2004), Los Angeles, CA, 435-444, MIT Press.
3. « *Rapid response of head direction cells to reorienting visual cues : A computational model* », Thomas Degris, Nicolas Brunel, Olivier Sigaud et Angelo Arleo, 2004, In De Schutter (Ed). Proceedings of The Annual Computational Neuroscience Meeting, Elsevier.
4. « *Designing Efficient Exploration with MACS : Modules and Function Approximation* », Pierre Gérard et Olivier Sigaud, 2003, Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'03), 1882-1893, Springer-Verlag.
5. « *Spatial Coordination through Social Potential Fields and Genetic Algorithms* », Fabien Flacher et Olivier Sigaud, 2002, Proceedings of the Seventh International Conference on Simulation of Adaptive Behavior (SAB-2002), Edinburgh, Ecosse (poster), MIT Press.
6. « *A comparison between ATNoSFERES and XCSM* », Samuel Landau, Sébastien Picault, Pierre Gérard et Olivier Sigaud, 2002, Fourth annual Genetic and Evolutionary Computation Conference (GECCO-2002) New York, NY, 926-933
7. « *Adding a Generalization Mechanism to YACS* », Pierre Gérard et Olivier Sigaud, 2001 Third annual Genetic and Evolutionary Computation Conference (GECCO-2001), San Francisco, CA, 951-957.
8. « *Model-Based Partioning for Control Learning* », Olivier Sigaud, 1998, Vienne, Autriche, Fourteenth European Meeting in Cybernetics and Systems Research (EMCSR-98), Edited by R. Trappl, 675-680.
9. « *Control and learning : from local actions to global behaviors* », Olivier Sigaud et Dominique Luzeaux, 1995, Third European Control Conference (ECC-95), Rome, Italie, 3172-3177.
10. « *Learning hierarchical controllers for situated agents* », Olivier Sigaud, 1995, Namur, Belgique, Fourteenth International Congress of Cybernetics (ICC-95).

Workshops avec actes Springer-Verlag (type LNCS, LNAI)

1. « *A Michigan style architecture for learning finite state controllers : a first step* » Olivier Sigaud et Samuel Landau, 2004, In Stolzmann et al. (Eds). Proceedings of the Seventh International Workshop of Learning Classifier Systems (IWLCS-04) (à paraître en tant que volume LNAI, Springer-Verlag.)
2. « *An Experimental Comparison between ATNoSFERES and ACS.* », Samuel Landau, Olivier Sigaud, Sébastien Picault et Pierre Gérard, 2003, In Stolzmann et al. (Eds). Proceedings of the Sixth International Workshop of Learning Classifier Systems (IWLCS-03). (à paraître en tant que volume LNAI, Springer-Verlag.)
3. « *Further Comparison between ATNoSFERES and XCSM* ». Samuel Landau, Sébastien Picault, Olivier Sigaud et Pierre Gérard, 2002, LNAI2661 : Learning Classifier Systems. Proceedings of the Fifth International Workshop on Learning Classifier Systems (IWLCS-02), Grenade, Espagne, 99-117, Springer-Verlag.
4. « *YACS : Combining Dynamic Programming with Generalization in Classifier Systems* », Pierre Gérard et Olivier Sigaud, 2000, LNAI 1996 : Advances in Learning Classifier Systems. Proceedings of the Third International Workshop on Learning Classifier Systems (IWLCS-00), Paris, France 53-69, Springer-Verlag.
5. « *Using Classifier Systems as Adaptive Expert Systems for Control* », Olivier Sigaud et Pierre Gérard, 2000, LNAI 1996 : Advances in Learning Classifier Systems. Proceedings of the Third International Workshop on Learning Classifier Systems (IWLCS-00), Paris, France, 138-157, Springer-Verlag.
6. « *Being reactive by exchanging roles : an experimental study* » Olivier Sigaud et Pierre Gérard, 2001, LNAI 2103 : Balancing reactivity and Social Deliberation in Multiagent Systems, Berlin, Allemagne, 150-172, Springer-Verlag.

Conférences internationales avec sélection sur résumé

1. « *Anticipation of Periodic Movements in Real Time 3D Environments* », Vincent Labbe et Olivier Sigaud, 2004, Workshop à SAB'04, « *Anticipatory Behavior in Adaptive Learning Systems* » (ABiALS'04).
2. « *Using XCS to build Adaptive Agents* » Zahia Guessoum, Lilia Rejeb et Olivier Sigaud, 2004, Proceedings of the Fourth Symposium on Adaptive Agents and Multi-Agent Systems (AAMAS-4), AISB convention, Leeds, 101-106.
3. « *Using roles in multiagent adaptive simulations* », Olivier Sigaud, 2000, Wuerzburg, Allemagne, Summer School on « *Learning : the acquisition of behavioral competence* ».
4. « *Why using anticipation to solve POMDPs ?* », Olivier Sigaud, 1999, Lugano, Suisse, Fourth European Workshop in Reinforcement Learning (EWRL-4).
5. « *From Situated Action to Noetical Consciousness : the Role of Anticipation* », Olivier Sigaud, 1997, **best paper award**, First International Conference on Computing Anticipatory Systems (CASYS-97), Liège, Belgique, Edited by Daniel M. Dubois, 127-137, AIP Conference proceedings 437.
6. « *Anticipation as a key of Cognitive Development* », Olivier Sigaud, 1997, Frieburg in Breisgau, Allemagne, Fifteenth Congress of the European Society for the Study of Cognitive Systems (ESSCS-97).
7. « *Learning : from control to behavioral development* », Olivier Sigaud, 1996, S^t-Maximin, France, Fourteenth Congress of the European Society for the Study of Cognitive Systems (ESSCS-96).

Chapitres d'ouvrages

1. « *Anticipatory Behavior : Exploiting Knowledge about the Future to Improve Current Behavior* » Martin Butz, Olivier Sigaud, et Pierre Gérard, 2003, Dans LNCS 2684 : « *Anticipatory Behavior in Adaptive Learning Systems* », 1-10, Springer-Verlag.
2. « *Internal Models and Anticipations in Adaptive Learning Systems* » Martin Butz, Olivier Sigaud, et Pierre Gérard, 2003, Dans LNCS 2684 : « *Anticipatory Behavior in Adaptive Learning Systems* », 87-110 Springer-Verlag.
3. « *Vers une approche dynamique de la sélection de l'action* », Olivier Sigaud et Fabien Flacher, 2002, dans « Approche dynamique de la cognition artificielle », A. Guillot et E. Daucé éditeurs, 163-178, Hermès.

Conférences nationales

1. « *Contribution au problème de la sélection de l'action en environnement partiellement observable* », Olivier Sigaud et Pierre Gérard, 1999, Première Conférence d'Intelligence Artificielle Située (IAS-99), Paris, France, 129-146.
2. « *L'apprentissage de catégories sensori-motrices* », Olivier Sigaud, 1997, Journées Françaises d'Apprentissage (JFA-97), Roscoff, France, 149-161.
3. « *De la commande réactive à la catégorisation* », Olivier Sigaud, 1996, Troisièmes Rencontres des jeunes chercheurs en Intelligence Artificielle
(JCIA-96), Nantes, France, 205-212.
4. « *Un algorithme de partitionnement pour l'apprentissage de la commande* », Olivier Sigaud, 1996, Deuxièmes Rencontres des jeunes chercheurs en Sciences Cognitives (JCSC-96), Giens, France (poster).

Séminaires et ateliers sans actes publiés

1. En 2003, présentation au séminaire Complexité de l'IHSPST : « Système cognitifs et systèmes dynamiques : contourner la complexité ? »
2. « Une comparaison entre ATNoSFERES et XCSM », Sébastien Picault, Olivier Sigaud, Samuel Landau et Pierre Gérard, 2002, PDMIA, Nancy.
3. « Méthodes d'apprentissage par renforcement pour les POMDPs », Olivier Sigaud et Pierre Gérard, 2001, PDMIA, Toulouse.
4. Le 7 janvier 2000, au séminaire RESA (LIP6) : « De l'adaptatif au multi-agents : le cas d'une meute de robots bergers »
5. Le 21 avril 2000, au séminaire RESA (LIP6) : « Adaptatif et multi-agents : un programme de recherche épistémologiquement fondé »
6. En juin 2000, présentation aux journées scientifiques de Dassault Aviation, devant 300 personnes, d'un exposé sur la modélisation des comportements adaptatifs.
7. En 1995, présentation de mon travail de doctorat au séminaire ANIMAT organisé par Jean-Arcady Meyer (alors à l'ENS Ulm).

Bibliographie

[Adobe Systems, 1985] Adobe Systems (1985). *The PostScript Language Reference Manual*. Addison-Wesley, Reading MA.

[Aha & Molineaux, 2004] Aha, D. W. & Molineaux, M. (2004). Integrating Learning in Interactive Gaming Simulators. Edité dans *Challenges of Game AI : Proceedings of the AAAI'04 Workshop*, San Jose, CA. AAAI, AAAI Press.

[Albus, 1975] Albus, J. S. (1975). A new approach to manipulator control : The cerebellar model articulation controller (CMAC). *Journal of Dynamic Systems, Measurement and Control*, 97(3) :220–227.

[Arbib, 1981] Arbib, M. (1981). Perceptual structure and distributed motor control. Edité dans Brooks, R., editeur, *Handbook of psychology – The nervous system II*, pages 1441–1465.

[Arbib & House, 1985] Arbib, M. & House, D. (1985). Depth and detours : An essay on visually guided behavior. Coins tr 85-20, University of Massachussetts, Amherst, MA.

[Arkin, 1989] Arkin, R. C. (1989). Motor Schema-Based Mobile Robot Navigation. *The International Journal of Robotics Research*, 8(4) :92–112.

[Arkin, 1992] Arkin, R. C. (1992). Behavior-based Robot Navigation for Extended Domains. *Adaptive Behavior*, 2(1) :201–225.

[Arnborg et al., 1987] Arnborg, S., Corneil, D. G., & Proskurowski, A. (1987). Complexity of finding embeddings in a k-tree. *SIAM Journal of Algebraic and Discrete Methods*, 8(2) :277–284.

[Atkeson, 1991] Atkeson, C. G. (1991). Memory-based control learning. Edité dans *Proceedings of the 1991 American Control Conference*, New York, NY.

[Baerends, 1976] Baerends, G. P. (1976). The functional organization of behaviour. *Animal Behaviour*, 24 :726–738.

[Balch & Arkin, 1995] Balch, T. & Arkin, R. C. (1995). Motor Schema-Based Formation Control for Multiagent Robot Teams. Edité dans *Proceedings of the First International Conference on Multiagent Systems*, pages 10–16.

[Balch & Arkin, 1998] Balch, T. & Arkin, R. C. (1998). Behavior-based Formation Control for Multi-robot Teams. *IEEE Transactions on Robotics and Automation*, 14(6) :1–15.

[Balch & Hybinette, 2000] Balch, T. & Hybinette, M. (2000). Social Potentials for Scalable Multi-Robots Formations. Edité dans *IEEE International Conference on Robotics and Automation (ICRA-2000)*.

[Banzhaf et al., 1999] Banzhaf, W., Daida, J., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M., & E., S. R., editeurs (1999). *Proceedings of the 1999 Genetic and Evolutionary Computation Conference Workshop Program*. Morgan Kaufmann.

[Barraquand et al., 1992] Barraquand, J., Langlois, B., & Latombe, J.-C. (1992). Potential Fields for Robot Motion Planning. *IEEE Transactions on Systems, Man and Cybernetics*, 22(2) :224–241.

[Barto, 1995] Barto, A. G. (1995). Adaptive Critics and the Basal Ganglia. Edité dans *Models of Information Processing in the Basal Ganglia*, pages 215–232. MIT Press, Cambridge, MA.

[Barto et al., 1983] Barto, A. G., Sutton, R. S., & Anderson, C. W. (1983). Neuron-like adaptive elements that can solve difficult learning control problems. *IEEE Transaction on Systems, Man, and Cybernetics*, 13(5) :834–846.

[Becker & Geiger, 2001] Becker, A. & Geiger, D. (2001). A sufficiently fast algorithm for finding close to optimal clique trees. *Artificial Intelligence*, 125(1-2) :3–17.

[Becker, 1970] Becker, J. D. (1970). *An information processing model of intermediate cognition*. Thèse de doctorat, Artificial Intelligence Project, Standford University.

[Bellman, 1957] Bellman, R. E. (1957). *Dynamic Programming*. Princeton University Press, Princeton, NJ.

[Bernardó et al., 2001] Bernardó, E., X., L., & M., G. J. (2001). XCS and GALE : a comparative study of two Learning Classifer Systems with six other learning algorithms on classification tasks. Edité dans Lanzi, P.-L., Stolzmann, W., & Wilson, S. W., editeurs, *Proceedings of the fourth international workshop on Learning Classifer Systems*.

[Bernstein et al., 2002] Bernstein, D. S., Givan, R., Immerman, N., & Zilberstein, S. (2002). The complexity of decentralized control of markov decision processes. *Mathematics of Operations Research*, 27(4) :819–840.

[Berridge & Robinson, 1998] Berridge, K. C. & Robinson, T. E. (1998). What is the role of dopamine in reward : hedonic impact, reward learning , or incentive salience. *Brain Research Reviews*, 28 :309–369.

[Bertele & Brioschi, 1972] Bertele, U. & Brioschi, F. (1972). *Nonserial Dynamic Programming*. Academic Press, New York, NY.

[Bertsekas, 1995] Bertsekas, D. P. (1995). *Dynamic Programming and Optimal Control*. Athena Scientific, Belmont, MA.

[Bertsekas & Tsitsiklis, 1996] Bertsekas, D. P. & Tsitsiklis, J. (1996). *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA.

[Beynier et al., 2004] Beynier, A., Mouaddib, A., & Washington, R. (2004). An optimal execution of autonomous agents with plans under time and resource uncertainty. Edité dans *AAMAS Workshop on Game Theoretic and Decision Theoretic Agents*.

[Booker et al., 1989] Booker, L., Goldberg, D. E., & Holland, J. H. (1989). Classifier Systems and Genetic Algorithms. *Artificial Intelligence*, 40(1-3) :235–282.

[Booker, 2000] Booker, L. B. (2000). Do we really need to estimate rule utilities in classifier systems ? Edité dans Lanzi, P.-L., Stolzmann, W., & Wilson, S. W., editeurs, *Learning Classifier Systems. From Foundations to Applications*, volume 1813 of *Lecture Notes in Artificial Intelligence*, pages 125–142, Berlin. Springer-Verlag.

[Boutilier, 1996] Boutilier, C. (1996). Planning, learning and coordination in multiagent decision processes. Edité dans *Sixth Conference on Theoretical Aspects of Rationality and Knowledge (TARK-96)*, pages 195–210, Amsterdam.

[Boutilier, 1999] Boutilier, C. (1999). Sequential optimality and coordination in multiagent systems. Edité dans *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, pages 478–485, Stockholm.

[Boutilier et al., 1999] Boutilier, C., Dean, T., & Hanks, S. (1999). Decision-theoretic planning : Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 11 :1–94.

[Boutilier et al., 1995] Boutilier, C., Dearden, R., & Goldszmidt, M. (1995). Exploiting structure in policy construction. Edité dans *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 1104–1111, Montreal.

[Boutilier et al., 2000] Boutilier, C., Dearden, R., & Goldszmidt, M. (2000). Stochastic dynamic programming with factored representations. *Artificial Intelligence*, 121(1) :49–107.

[Boutilier & Goldszmidt, 1996] Boutilier, C. & Goldszmidt, M. (1996). The frame problem and bayesian network action representations. Edité dans *Proceedings of the Eleventh Biennial Canadian Conference on Artificial Intelligence (AI '96)*, pages 69–83, Toronto, CA.

[Brafman & Tennenholtz, 2001] Brafman, R. I. & Tennenholtz, M. (2001). R-max : a general polynomial time algorithm for near-optimal reinforcement learning. Edité dans *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, pages 953–958.

[Brooks, 1986] Brooks, R. A. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2 :14–23.

[Brooks, 1991] Brooks, R. A. (1991). Intelligence without reason. A.I. Memo 1293, Massachusetts Institute of Technology, Artificial Intelligence Laboratory.

[Brown, 1951] Brown, G. W. (1951). Iterative solution of games by fictitious play. Edité dans Koopmans, T. C., editeur, *Activity Analysis of Production and Allocation*, chapitre XXIV. Wiley, New York.

[Buffet et al., mise] Buffet, O., Dutech, A., & Charpillet, F. (2004, soumise). Etude de différentes combinaisons adaptatives de comportements au sein d'un agent. *Revue d'Intelligence Artificielle*.

[Buhmann, 2003] Buhmann, M. D. (2003). *Radial Basis Functions*. Cambridge University Press.

[Bull & O'Hara, 2002] Bull, L. & O'Hara, T. (2002). Accuracy-based neuro and neuro-fuzzy classifier systems. Edité dans Langdon, W. B., Cantu-Paz, E., Mathias, K., Roy, R., Davis, D., Poli, R., Balakrishnan, K., Honavar, V., Rudolph, G., Wegener, J., Bull, L., Potter, M. A., Schultz, A. C., Miller, J. F., Burke, E., & Jonoska, N., editeurs, *Proceeding of the Genetic and Evolutionary Computation Conference (GECCO2002)*, pages 905–911. Morgan Kaufmann.

[Butz, 2002] Butz, M. V. (2002). An algorithmic description of acs2. Dans [Lanzi et al., 2002a], pages 211–229.

[Butz & Goldberg, 2003] Butz, M. V. & Goldberg, D. E. (2003). Generalized state values in an anticipatory Learning Classifier System. Edité dans Butz, M. V., Sigaud, O., & Gérard, P., editeurs, *LNAI 2684 : Anticipatory Behavior in Adaptive Learning Systems*, pages 281–301. Springer-Verlag.

[Butz et al., 2000] Butz, M. V., Goldberg, D. E., & Stolzmann, W. (2000). Introducing a genetic generalization pressure to the Anticipatory Classifier Systems part I : Theoretical approach. Edité dans *Proceedings of the 2000 Genetic and Evolutionary Computation Conference (GECCO 2000)*, pages 34–41.

[Cantu-Paz et al., 2003] Cantu-Paz, E., Foster, J. A., Deb, K., O'Reilly, U.-M., Beyer, H.-G., Standish, R., Kendall, G., Wilson, S., Harman, M., Weneger, J., Dasgupta, D., Potter, M. A., Schultz, A. C., Dowsland, K. A., Jonoska, N., & Miller, J., editeurs (2003). *Proceedings of the 2003 Genetic and Evolutionary Computation Conference Workshop Program*, Chicago, IL. Springer Verlag.

[Chadès et al., 2002] Chadès, I., Scherrer, B., & Charpillet, F. (2002). A heurisitic approach for solving decentralized-pomdp : Assessment on the pursuit problem. Edité dans *Proceedings of the 17th ACM Symposium on Applied Computing (SAC 2002)*, Madrid.

[Chomsky, 1962] Chomsky, N. (1962). Context-free grammars and pushdown storage. Quarterly Progress Report 65, MIT Research Laboratory in Electronics.

[Chomsky & Miller, 1963] Chomsky, N. & Miller, G. (1963). Introduction to the formal analysis of natural languages. Edité dans Luce, Bush, & Galanter, editeurs, *Handbook of Mathematical Psychology 2*, pages 269–321. Wiley and Sons, New York.

[Chvatal, 1983] Chvatal, V. (1983). *Linear Programming*. Freeman.

[Clark, 1997] Clark, A. (1997). *Being There, Putting Brain, Body and World Together Again*. MIT Press, Cambridge, MA.

[Clark et al., 1992] Clark, R., Arkin, R. C., & Ram, A. (1992). Learning momentum : On-line performance enhancement for reactive systems. Edité dans *International Conference on Robotics and Automation*, pages 112–116.

[Claus & Boutilier, 1998] Claus, C. & Boutilier, C. (1998). The dynamics of reinforcement learning in cooperative multiagent systems. Edité dans *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, pages 746–752, Madison, WI.

[Cliff & Ross, 1994] Cliff, D. & Ross, S. (1994). Adding memory to ZCS. *Adaptive Behavior*, 3(2) :101–150.

[Coello, 2003] Coello, C. A. (2003). Multi-objective optimization with evolutionary computation. Edité dans Foster, J., editeur, *GECCO-03 Tutorial Program*, pages 359–383.

[Collet et al., 2000] Collet, P., Lutton, E., Raynal, F., & Schoenauer, M. (2000). Polar IFS+parisian genetic programming=efficient IFS inverse problem solving. *Genetic Programming and Evolvable Machines*, 1(4) :339–361.

[Coulom, 2002] Coulom, R. (2002). *Apprentissage par renforcement utilisant des réseaux de neurones, avec des applications au contrôle moteur*. Thèse de doctorat, Laboratoire Leibniz-IMAG, Grenoble.

[Darwiche & Goldszmit, 1994] Darwiche, A. & Goldszmit, M. (1994). Action networks : A framework for reasoning about action and change under uncertainty. Edité dans *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence (UAI-94)*, pages 136–144, Seattle, WA.

[Daucé, 2002] Daucé, E. (2002). Apprentissage et émergence de structures dynamiques dans les réseaux récurrents. Edité dans Guillot, A. & Daucé, E., editeurs, *Approche dynamique de la cognition artificielle (ce volume)*. Hermès.

[David W. Corne & Oates, 2001] David W. Corne, Nick R. Jerram, J. D. K. & Oates, M. J. (2001). Pesa-ii : Region-based selection in evolutionary multiobjective optimization. Edité dans Spector, L., Goodman, E., Wu, A., Langdon, W. B., Voigt, H.-M., Gen, M., Sen, S., Dorigo, M., Pezeshk, S., Garzon, M. H., & Burke, E., editeurs, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 283–290, San Francisco, CA. Morgan Kaufmann Publishers.

[Dayan & Sejnowksi, 1994] Dayan, P. & Sejnowksi, T. J. (1994). Td converge with probability 1. *Machine Learning*, 14 :295–301.

[De Jong, 1975] De Jong, K. A. (1975). *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. Thèse de doctorat, Dept. of Computer and Communication Sciences, University of Michigan.

[Dean & Kanazawa, 1989] Dean, T. & Kanazawa, K. (1989). A model for reasoning about persistence and causation. *Computational Intelligence*, 5(3) :142–150.

[Deb, 2001] Deb, K. (2001). *Multi-Objective Optimization using Evolutionary Algorithms*. Wiley.

[Deb, 2004] Deb, K. (2004). Evolutionary multi-objective optimization. Edité dans Poli, R., editeur, *GECCO-04 Tutorial Program*, pages 585–609.

[Deb et al., 2004] Deb, K., Poli, R., Banzhaf, W., Beyer, H.-G., Burke, E., Darwen, P., Dasgupta, D., Floreano, D., Foster, J. A., Harman, M., Holland, O., Lanzi, P.-L., Spector, L., Tettamanzi, A., Thierens, D., & Tyrrell, A., editeurs (2004). *Proceedings of the 2004 Genetic and Evolutionary Computation Conference Workshop Program*, Seattle, WA. Springer Verlag.

[Deb et al., 2002] Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm : Nsga–ii. *IEEE Transactions on Evolutionary Computation*, 6(2) :182–197.

[Decugis & Ferber, 1998] Decugis, V. & Ferber, J. (1998). An Extension of Maes' Action Selection Mechanism for Animats. Edité dans Pfeifer, R., Blumberg, B., Meyer, J.-A., & Wilson, S. W., editeurs, *From Animals to Animats 5 : Proceedings of the Fifth International Conference on Simulation of Adaptive Behavior*, pages 153–158, Cambridge, MA. MIT Press.

[DeLong, 2000] DeLong, M. R. (2000). The basal ganglia. Edité dans Kandel, E. R., Schwartz, J. H., & Jessel, J. M., editeurs, *Principal of Neural Science (Fourth Edition)*, pages 853–867. Prentice Hall, New Jersey.

[Dennett, 1990] Dennett, D. C. (1990). Cognitive wheels : the frame problem of AI. Edité dans Boden, M. A., editeur, *The Philosophy of Artificial Intelligence*, chapitre 7, pages 147–170. Oxford Readings in Philosophy.

[Doncieux, 2003] Doncieux, S. (2003). *Évolution de contrôleurs neuronaux pour animats volants : méthodologie et applications*. Thèse de doctorat, LIP6/AnimatLab, Université Pierre et Marie Curie, Paris, France.

[Donnart, 1998] Donnart, J.-Y. (1998). *Architecture cognitive et propriétés adaptatives d'un animat motivationnellement autonome*. Thèse de doctorat, Université Pierre et Marie Curie, Paris, France.

[Donnart & Meyer, 1996] Donnart, J.-Y. & Meyer, J.-A. (1996). Learning reactive and planning rules in a motivationally autonomous animat. *IEEE Transactions on Systems, Man, and Cybernetics - Part B : Cybernetics*, 3(26) :381–395.

[Dorigo, 1993] Dorigo, M. (1993). Genetic and non-genetic operators in ALECSYS. *Evolutionary Computation*, 1(2) :151–164.

[Dorigo & Bersini, 1994] Dorigo, M. & Bersini, H. (1994). A comparison of Q-Learning and Classifier Systems. Edité dans *From Animals to Animats 3*, pages 248–255.

[Drescher, 1991] Drescher, G. L. (1991). *Made-Up Minds : A constructivist approach to Artificial Intelligence*. MIT Press, Cambridge MA.

[Dreyfus, 1972] Dreyfus, H. (1972). *What computers can't do : a critique of artificial reason*. Harper and Row, New York.

[Dreyfus, 1996] Dreyfus, H. (1996). What computers still can't do. *Artificial Intelligence*.

[Eckart Zitzler & Thiele, 2001] Eckart Zitzler, M. L. & Thiele, L. (2001). Spea2 : Improving the strength pareto evolutionary algorithm. Edité dans Giannakoglou, K., Tsahalis, D., Periaux, J., Papailou, P., & Fogarty, T., editeurs, *EUROGEN 2001, Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, pages 12–21, Athens, Greece.

[Erickson et al., 2001] Erickson, M., Mayer, A., & Horn, J. (2001). The niched pareto genetic algorithm 2 applied to the design of groundwater remediation systems. Edité dans Zitzler, E., Deb, K.and Thiele, L., Coello, C. A., & Corne, D., editeurs, *First International Conference on Evolutionary Multi-Criterion Optimization*, pages 681–695. Springer-Verlag (Lecture Notes in Computer Science No. 1993).

[Flacher & Sigaud, 2002] Flacher, F. & Sigaud, O. (2002). Spatial coordination through social potential fields and genetic algorithms. Edité dans Hallam, B., Floreano, D., Meyer, J.-A., & Hallam, J., editeurs, *From Animals to Animats 7 : Proceedings of the Seventh International Conference on Simulation of Adaptive Behavior*, pages 389–390, Cambridge, MA. MIT Press.

[Flacher & Sigaud, 2003] Flacher, F. & Sigaud, O. (2003). Coordination spatiale émergente par champs de potentiel. *TSI*, numéro spécial sur la vie artificielle.

[Fogel et al., 1966] Fogel, L. J., Owens, A. J., & Walsh, M. J. (1966). *Artificial Intelligence through Simulated Evolution*. John Wiley & Sons.

[Fonseca & Fleming, 1993] Fonseca, C. M. & Fleming, P. J. (1993). Genetic algorithms for multiobjective optimization : Formulation, discussion and generalization. Edité dans *Genetic Algorithms : Proceedings of the Fifth International Conference*, San Mateo, CA.

[Friedman et al., 1998] Friedman, N., Murphy, K., & Russell, S. (1998). Learning the structure of dynamic probabilistic networks. Edité dans *Proceedings of UAI 98*.

[Garcia, mise] Garcia, P. (2004, soumise). Exploration guidée en apprentissage par renforcement. *Revue d'Intelligence Artificielle*.

[Gérard, 2001] Gérard, P. (2001). Generalisation and latent learning in learning classifier systems. Edité dans *Proceedings of the XIX EURO Summer Institute*, Toulouse, France.

[Gérard, 2002] Gérard, P. (2002). *Apprentissage latent et apprentissage par renforcement dans les systèmes de classeurs*. Thèse de doctorat de l'Université PARIS VI.

[Gérard et al., 2005] Gérard, P., Meyer, J.-A., & Sigaud, O. (2005). Combining latent learning with dynamic programming in MACS. *European Journal of Operational Research*, 160 :614–637.

[Gérard & Sigaud, 2001] Gérard, P. & Sigaud, O. (2001). Adding a Generalization Mechanism to YACS. Dans [Spector et al., 2001], pages 951–957.

[Gérard & Sigaud, 2003] Gérard, P. & Sigaud, O. (2003). Designing efficient exploration with MACS : Modules and function approximation. Edité dans *Proceedings of the Genetic and Evolutionary Computation Conference 2003 (GECCO'03)*, pages 1882–1893, Chicago, IL. Springer-Verlag.

[Gérard et al., 2002] Gérard, P., Stolzmann, W., & Sigaud, O. (2002). YACS : a new Learning Classifier System with Anticipation. *Journal of Soft Computing : Special Issue on Learning Classifier Systems*, 6(3-4) :216–228.

[Ghahramani, 1997] Ghahramani, Z. (1997). Learning dynamic bayesian networks. Edité dans Giles, C. L. & Gori, M., editeurs, *Adaptive Processing of Temporal Information*. LNAI, Springer-Verlag, Berlin.

[Gies, mise] Gies, O. (2004, soumise). Apprentissage de la coordination multi-agents : Q-learning par jeu adaptatif. *Revue d'Intelligence Artificielle*.

[Girard, 2002] Girard, B. (2002). *Elaboration d'une architecture de sélection de l'action inspirée des ganglions de la base chez le rat artificiel Psikharpax*. Rapport de pré-soutenance de thèse, AnimatLab, Laboratoire d'informatique de Paris 6.

[Goldberg, 1989] Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison Wesley, Reading, MA.

[Graybiel, 1995] Graybiel, A. M. (1995). Building action repertoires : Memory and learning functions of the basal ganglia. *Current Opinion in Neurobiology*, 5 :733–741.

[Grefenstette, 1991] Grefenstette, J. J. (1991). Lamarckian learning in multi-agent environments. Edité dans Belew, R. & Booker, L., editeurs, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 303–310, San Mateo, CA. Morgan Kaufmann.

[Grefenstette et al., 1990] Grefenstette, J. J., Ramsey, C. L., & Schultz, A. C. (1990). Learning sequential decision rules using simulation models and competition. *Machine Learning*, 5(4) :355–381.

[Gruau, 1994] Gruau, F. (1994). *Neural Network Synthesis Using Cellular Encoding and the Genetic Algorithm*. Thèse de doctorat, ENS Lyon – Université Lyon I.

[Guessoum, 2003] Guessoum, Z. (2003). *Modèles et architectures d'agents et de systèmes multi-agents adaptatifs*. Dossier d'habilitation de l'Université de Paris 6, Laboratoire d'Informatique de Paris 6.

[Guestrin et al., 2003a] Guestrin, C., Koller, D., Gearhart, C., & Kanodia, N. (2003a). Generalizing plans to new environments in relational mdps. Edité dans *International Joint Conference on Artificial Intelligence (IJCAI-03)*, Acapulco, Mexico.

[Guestrin et al., 2001] Guestrin, C., Koller, D., & Parr, R. (2001). Multiagent planning with factored mdps. Edité dans *Advances in Neural Information Processing Systems (NIPS)*, Vancouver, British Columbia, Canada.

[Guestrin et al., 2003b] Guestrin, C., Koller, D., Parr., R., & Venkataraman, S. (2003b). Efficient solution algorithms for factored mdps. *Journal of Artificial Intelligence Research*, 19 :399–468.

[Guestrin et al., 2002a] Guestrin, C., Lagoudakis, M., & Parr, R. (2002a). Coordinated reinforcement learning. Edité dans *ICML-2002 The Nineteenth International Conference on Machine Learning*, pages 227–234, Sydney, Australia.

[Guestrin et al., 2002b] Guestrin, C., Patrascu, R., & Schuurmans, D. (2002b). Algorithm-directed exploration for model-based reinforcement learning in factored mdps. Edité dans *ICML-2002 The Nineteenth International Conference on Machine Learning*, pages 235–242, Sydney, Australia.

[Guillot & Meyer, 2001] Guillot, A. & Meyer, J.-A. (2001). The animat contribution to cognitive systems research. *Journal of Cognitive Systems Research*, 2(2) :157–165.

[Hansen, 1998] Hansen, E. A. (1998). *Finite Memory Control of Partially Observable Systems*. Thèse de doctorat, University of Massachusetts, Amherst, MA.

[Harnad, 1990] Harnad, S. (1990). The symbol grounding problem. *Physica D*, 42 :335–346.

[Hart et al., 1968] Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimal cost paths. *IEEE transactions on SSC*, 4 :100–107.

[Hart et al., 1972] Hart, P. E., Nilsson, N. J., & Raphael, B. (1972). Correction to : a formal basis for the heuristic determination of minimal cost paths. *SIGART Newsletters*, 3 :28–29.

[Hasegawa et al., 1999] Hasegawa, Y., Ito, Y., & Fukuda, T. (1999). Behavior adaptation on behavior-based controller for brachiation robot. Edité dans Floreano, D., Nicoud, J.-D., & Mondada, F., editeurs, *Advances in Artificial Life*, pages 294–303, Lausanne, Switzerland. Springer.

[Heckerman et al., 1995] Heckerman, D., Geiger, D., & Chickering, D. M. (1995). Learning bayesian networks : The combination of knowledge and statistical data. *Machine Learning*, 20 :197–243.

[Heguy, 2003] Heguy, O. (2003). *Architecture comportementale pour l'émergence d'activités coopératives en environnement virtuel*. Thèse de doctorat, Université Paul Sabatier, Toulouse, France.

[Herbart, 1825] Herbart, J. F. (1825). *Psychologie als Wissenschaft neu gegründet auf Erfahrung, Metaphysik und Mathematik. Zweiter, analytischer Teil*. August Wilhem Unzer, Koenigsberg, Germany.

[Hinton & Nowlan, 1987] Hinton, G. E. & Nowlan, S. J. (1987). How learning can guide evolution. *Complex Systems*, 1 :495–502.

[Hoey et al., 1999] Hoey, J., St-Aubin, R., Hu, A., & Boutilier, C. (1999). SPUDD : Stochastic planning using decision diagrams. Edité dans *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 279–288. Morgan Kaufmann.

[Hoffmann, 1993] Hoffmann, J. (1993). *Vorhersage und Erkenntnis [Anticipation and Cognition]*. Hogrefe, Göttingen.

[Holland, 1975] Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems : An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. University of Michigan Press, Ann Arbor, MI.

[Holland, 1985] Holland, J. H. (1985). Properties of the Bucket Brigade Algorithm. Edité dans Grefenstette, J. J., editeur, *Proceedings of the 1st international Conference on Genetic Algorithms and their applications (ICGA85)*, pages 1–7. L. E. Associates.

[Holland et al., 1986] Holland, J. H., Holyoak, K. J., Nisbett, R. E., & Thagard, P. R. (1986). *Induction*. MIT Press.

[Holland & Reitman, 1978] Holland, J. H. & Reitman, J. S. (1978). Cognitive Systems based on Adaptive Algorithms. *Pattern Directed Inference Systems*, 7(2) :125–149.

[Holmes, 2002] Holmes, J. H. (2002). A new representation for assessing classifier performance in mining large databases. Edité dans Stolzmann, W., Lanzi, P.-L., & Wilson, S. W., editeurs, *IWLCS-02. Proceedings of the International Workshop on Learning Classifier Systems*, LNAI, Granada. Springer-Verlag.

[Horn et al., 1994] Horn, J., Nafpliotis, N., & Goldberg, D. E. (1994). A niched pareto genetic algorithm for multiobjective optimization. Edité dans *Proceedings of the First IEEE Conference on Evolutionary Computation*, volume 1, pages 82–87, Piscataway, NJ. IEEE Service Center.

[Houk et al., 1995] Houk, J. C., Davis, J. L., & Beiser, D. G. (1995). *Models of Information Processing in the Basal Ganglia*. The MIT Press.

[Howard, 1971] Howard, R. A. (1971). *Dynamic Probabilistic Systems*. Wiley.

[Hu, 2003] Hu, J. (2003). Best-response algorithm for multiagent reinforcement learning.

[Hu & Wellman, 2003] Hu, J. & Wellman, H. P. (2003). Nash q-learning for general-sum stochastic games. *Journal of Machine Learning Research*, 4 :1039–1069.

[Hurst & Bull, 2004] Hurst, J. & Bull, L. (2004). A neural learning classifier system with self-adaptive constructivism for mobile robot control. *Artificial Life*.

[IBM Corporation, 2003] IBM Corporation (2003). An architectural blueprint for autonomic computing. Rapport technique, IBM.

[Ikemoto & Panksepp, 1999] Ikemoto, S. & Panksepp, J. (1999). The role of nucleus accumbens dopamine in motivated behavior : a unifying interpretation with special reference to reward-seeking. *Brain Research Reviews*, 31 :6–41.

[Jakobi, 1997] Jakobi, N. (1997). Evolutionary robotics and the radical envelope of noise hypothesis. *Adaptive Behavior*, 6(2) :325–368.

[Jozefowiez, 2001] Jozefowiez, J. (2001). *Conditionnement opérant et Problèmes décisionnels de Markov*. Thèse de doctorat de l'Université Lille III, Lille.

[Kaelbling et al., 1998] Kaelbling, L. P., Littman, M. L., & Cassandra, A. R. (1998). Planning and Acting in Partially Observable Stochastic Domains. *Artificial Intelligence*, 101.

[Kalman, 1960] Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D) :35–45.

[Kearns & Koller, 1999] Kearns, M. & Koller, D. (1999). Efficient reinforcement learning in factored mdps. Edité dans *IJCAI-99*.

[Kearns et al., 2002] Kearns, M., Mansour, Y., & Ng, A. Y. (2002). A sparse sampling algorithm for near-optimal planning in large Markov decision processes. *Machine Learning*, 49 :193–208.

[Kearns & Singh, 1998] Kearns, M. & Singh, S. (1998). Near-optimal reinforcement learning in polynomial time. *Machine Learning*.

[Khamassi, 2003] Khamassi, M. (2003). *Un modèle d'apprentissage par renforcement dans une architecture de contrôle de la sélection de l'action chez le rat artificiel Psikharpax*. DEA de Sciences Cognitives de l'Université Paris VI.

[Khamassi et al., 2004] Khamassi, M., Lachèze, L., Girard, B., Guillot, A., & Berthoz, A. (submitted, 2004). Actor-critic models of reinforcement learning in the basal ganglia : From natural to artificial rats. *Adaptive Behavior*.

[Khatib, 1985] Khatib, O. (1985). Real time obstacle avoidance for manipulators and mobile robots. Edité dans *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 500–505.

[Kjærulff, 1993] Kjærulff, U. (1993). *Aspects of efficiency improvement in Bayesian networks*. Thèse de doctorat, Computer Science Aalborg University.

[Klopf, 1972] Klopf, H. A. (1972). Brain function and adaptive systems, a heterostatic theory. Rapport Technique Technical Report AFCRL-72-0164, Air Force Cambridge Research Laboratories.

[Klopf, 1975] Klopf, H. A. (1975). A comparison of natural and artificial intelligence. *SIGART newsletter*, 53 :11–13.

[Knowles & Corne., 1999] Knowles, J. D. & Corne., D. W. (1999). The pareto archived evolution strategy : A new baseline algorithm for multiobjective optimisation. Edité dans *1999 Congress on Evolutionary Computation*, pages 98–105, Washington, D.C. IEEE Service Center.

[Kodjabachian & Meyer, 1995] Kodjabachian, J. & Meyer, J.-A. (1995). Evolution and development of control architecture in animats. *Robotics and Autonomous Systems*, 16 :161–182.

[Kodjabachian & Meyer, 1998] Kodjabachian, J. & Meyer, J.-A. (1998). Evolution and Development of Neural Controllers for Locomotion, Gradient-Following, and Obstacle-Avoidance in Artificial Insects. *IEEE Transactions on Neural Networks*, 9 :796–812.

[Koller & Parr., 2000] Koller, D. & Parr., R. (2000). Policy iteration for factored mdps. Edité dans *Proceedings of the 16th Annual Conference on Uncertainty in AI (UAI)*, pages 326–334, Stanford, California.

[Korenz & Borenstein, 1991a] Korenz, Y. & Borenstein, J. (1991a). Potentiel field methods and their inherent limitations for mobile robot navigation. Edité dans *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 489–493.

[Korenz & Borenstein, 1991b] Korenz, Y. & Borenstein, J. (1991b). The vector field histogram – fast obstacle-avoidance for mobile robots. Edité dans *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 7, pages 278–288.

[Koza, 1992] Koza, J. R. (1992). *Genetic Programming : On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA.

[Krogh, 1984] Krogh, B. H. (1984). A generalized potentiel field approach to obstacle avoidance control. Edité dans *International Robotics Researh Conference*.

[Kuvayev & Sutton, 1996] Kuvayev, L. & Sutton, R. S. (1996). Model-based reinforcement learning with an approximate, learned model. Edité dans *Proceedings of the Ninth Yale Workshop on Adaptive and Learning Systems*, pages 101–105, New Haven, CT. Yale University Press.

[Labbé & Sigaud, 2004] Labbé, V. & Sigaud, O. (2004). Anticipation of periodic movements in real time 3d environments. Edité dans *Proceedings of the ABiALS 2004 Workshop*, Los Angeles, CA.

[Laird, 2000] Laird, J. E. (2000). It knows what you're going to do : adding anticipation to a quakebot. Edité dans AAAI & Spring Symphosium Series : Artificial Intellingence and Interactive Entertainment, editeurs, *AAAI Technical Report*.

[Landau, 2003] Landau, S. (2003). *Des AG vers la GA – Sélection Darwinienne et Systèmes Multi-Agents*. Phd thesis, Université Paris VI.

[Landau et al., 2002a] Landau, S., Doncieux, S., Drogoul, A., & Meyer, J.-A. (2002a). SFERES : un framework pour la conception de systèmes multi-agents adaptatifs. *Technique et Science Informatique*, 21(4) :427–446.

[Landau & Picault, 2001] Landau, S. & Picault, S. (2001). ATNoSFERES : a Model for Evolutive Agent Behaviors. Edité dans *Proceedings of the AISB'01 Symposium on Adaptive Agents and Multi-Agent Systems*.

[Landau & Picault, 2002] Landau, S. & Picault, S. (2002). Stack-Based Gene Expression. Rapport Technique LIP6 2002/011, LIP6, Paris.

[Landau et al., 2002b] Landau, S., Picault, S., Sigaud, O., & Gérard, P. (2002b). Further Comparison between ATNoSFERES and XCSM. Edité dans Stolzmann, W., Lanzi, P.-L., & Wilson, S. W., editeurs, *IWLCS-02. Proceedings of the Fifth International Workshop on Learning Classifier Systems*, LNAI, Granada. Springer-Verlag.

[Landau & Sigaud, 2004] Landau, S. & Sigaud, O. (2004). A Comparison between ATNoSFERES and LCSs on non-Markov problems (*accepted for publication*). *Information Sciences*.

[Langdon et al., 2002] Langdon, W. B., Cantu-Paz, E., Mathias, K., Roy, R., Davis, D., Poli, R., Balakrishnan, K., Honavar, V., Rudolph, G., Wegener, J., Bull, L., Potter, M. A., Schultz, A. C., Miller, J. F., Burke, E., & Jonoska, N., editeurs (2002). *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2002*, New York, NY. Morgan Kaufmann.

[Lanzi, 1998] Lanzi, P.-L. (1998). An analysis of the memory mechanism of XCSM. Edité dans Koza, J. R., Banzhaf, W., Chellapilla, K., Deb, K., Dorigo, M., Fogel, D. B., Garzon, M. H., Goldberg, D. E., Iba, H., & Riolo, R., editeurs, *Proceedings of the Third Genetic Programming Conference*, University of Wisconsin, Madison, Wisconsin, USA.

[Lanzi, 1999] Lanzi, P.-L. (1999). Extending the representation of classifier conditions part i : From binary to messy coding. Edité dans Banzhaf, W., Daida, J., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M., & E., S. R., editeurs, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 99)*, pages 337–344, Orlando, FL. Morgan Kaufmann.

[Lanzi, 2002] Lanzi, P.-L. (2002). Learning Classifier Systems from a Reinforcement Learning Perspective. *Journal of Soft Computing*, 6(3-4) :162–170.

[Lanzi & Perrucci, 1999] Lanzi, P.-L. & Perrucci, A. (1999). Extending the representation of classifier conditions part ii : From messy coding to s-expressions. Edité dans Banzhaf, W., Daida, J., Eiben, A. E., Garzon, M. H.,

Honavar, V., Jakiela, M., & E., S. R., editeurs, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 99)*, pages 345–352, Orlando, FL. Morgan Kaufmann.

[Lanzi & Riolo, 2000] Lanzi, P.-L. & Riolo, R. L. (2000). A roadmap to the last decade of Learning Classifier Systems research (from 1989 to 1999). Edité dans Lanzi, P.-L., Stolzmann, W., & Wilson, S. W., editeurs, *Learning Classifier Systems : from Foundations to Applications*, pages 33–62. Springer-Verlag, Heidelberg.

[Lanzi et al., 2000] Lanzi, P.-L., Stolzmann, W., & Wilson, S. W., editeurs (2000). *Learning Classifier Systems. From Foundations to Applications*, volume 1813 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, Berlin.

[Lanzi et al., 2001] Lanzi, P.-L., Stolzmann, W., & Wilson, S. W., editeurs (2001). *Advances in Learning Classifier Systems*, volume 1996 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, Berlin.

[Lanzi et al., 2002a] Lanzi, P.-L., Stolzmann, W., & Wilson, S. W., editeurs (2002a). *Advances in Learning Classifier Systems*, volume 2321 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, Berlin.

[Lanzi et al., 2002b] Lanzi, P.-L., Stolzmann, W., & Wilson, S. W., editeurs (2002b). *Proceedings of the International Workshop on Learning Classifier Systems (IWLCS2002)*, Granada.

[Lanzi & Wilson, 2000] Lanzi, P.-L. & Wilson, S. (2000). Toward Optimal Performance in Classifier Systems. *Evolutionary Computation Journal*, 8(4) :393–418.

[Laurent, 2002] Laurent, G. (2002). *Synthèse de comportements par apprentissages par renforcement parallèles*. Thèse de doctorat de l'universit''e de Franche-Comté.

[Lin, 1993] Lin, L.-J. (1993). Scaling up reinforcement learning for robot control. Edité dans *Proceedings of the Tenth International Conference on Machine Learning*, pages 182–189, Amherst, MA. Morgan Kaufmann.

[Lin & Mitchell, 1992] Lin, L.-J. & Mitchell, T. M. (1992). Memory approaches to reinforcement learning in non-markovian domains. Rapport Technique CMU-CS-92-138, Carnegie Mellon University, School of Computer Science.

[Lindenmayer, 1968] Lindenmayer, A. (1968). Mathematical Models for Cellular Interaction in Development, parts I and II. *Journal of Theoretical Biology*, 18.

[Littman, 1994] Littman, M. L. (1994). Markov games as a framework for multiagent reinforcement learning. Edité dans *Proceeding of the 11th International Conference in Machine Learning*, pages 157–163, New Brunswick, NJ. Morgan Kaufmann.

[Lorenz, 1950] Lorenz, K. (1950). The comparative method in studying innate behavior patterns. Edité dans *Society for Experimental Biology, Symposium no. 4, Physiological Mechanisms of Animal Behavior*, pages 221–268. NewYork : Academic Press.

[Luenberger, 1979] Luenberger, D. G. (1979). *Introduction to Dynamic Systems*. John Wiley and Sons.

[Luke & Spector, 1996] Luke, S. & Spector, L. (1996). Evolving Graphs and Networks with Edge Encoding : Preliminary Report. Edité dans Koza, J. R., editeur, *Late Breaking Papers at the Genetic Programming 1996 Conference*, pages 117–124, Stanford University, CA. Stanford Bookstore.

[Lutz, 2004] Lutz, C. (2004). *Designing Efficient Exploration with MACS : Modules and Function Approximation*. Thèse de doctorat, rapport de stage du DEA IARFA, LIP6.

[Maes, 1990] Maes, P., editeur (1990). *Designing Autonomous Agents : Theory and Practice from Biology to Engineering and Back*. MIT Press, Cambridge, MA.

[Matarić, 1994a] Matarić, M. J. (1994a). *Interaction and Intelligent Behavior*. Thèse de doctorat, MIT AI Mobot Lab.

[Matarić, 1994b] Matarić, M. J. (1994b). Rewards functions for accelerated learning. Edité dans Cohen, W. W. & Hirsch, H., editeurs, *Proceedings of the Eleventh International Conference on Machine Learning*, pages 181–189, San Francisco, CA. Morgan Kaufmann Publishers.

[Matarić, 1995] Matarić, M. J. (1995). Designing and Understanding Adaptive Group Behavior. *Adaptive Behavior*, 4(1) :51–80.

[McCallum, 1995] McCallum, R. A. (1995). *Reinforcement Learning with Selective Perception and Hidden State*. Thèse de doctorat, University of Rochester, Rochester, NY.

[McCallum et al., 1995] McCallum, R. A., Tesauro, G., Touretzky, D., & Leen, T. (1995). Instance-based state identification for reinforcement learning. *Advances in Neural Information Processing Systems*, 7 :377–384.

[McCarthy & Hayes, 1969] McCarthy, J. & Hayes, P. J. (1969). Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence*, 4 :463–502.

[Métivier & Lattaud, 2002] Métivier, M. & Lattaud, C. (2002). Anticipatory Classifier System using Behavioral Sequences in Non-Markov Environments. Edité dans Stolzmann, W., Lanzi, P.-L., & Wilson, S. W., editeurs, *IWLCS-02. Proceedings of the International Workshop on Learning Classifier Systems*, LNAI, pages 143–163, Granada. Springer-Verlag.

[Meuleau et al., 1999] Meuleau, N., Peshkin, L., Kim, K.-E., & Kaelbling, L. P. (1999). Learning finite-state controllers for partially observable environments. Edité dans *Fifteenth Conference on Uncertainty in Artificial Intelligence. AAAI*, pages 427–436.

[Meyer, 1999] Meyer, C. (1999). *Etude de mécanismes de modélisation pour l'apprentissage, l'adaptation et l'anticipation dans un système multi-agents hybride*. Thèse de doctorat, Université PARIS VI.

[Meyer, 1996a] Meyer, J.-A. (1996a). Artificial life and the animat approach to artificial intelligence. Edité dans Boden, M., editeur, *Artificial Intelligence*. Academic Press.

[Meyer, 1996b] Meyer, J.-A. (1996b). Pour une approche complémentaire de l'ia traditionnelle : le manifeste animat. *In Cognito*, 6(1-4).

[Meyer, 1998] Meyer, J.-A. (1998). The animat approach : Simulation of adaptive behavior in animals and robots. Edité dans Alexandre, F. & Kant, J.-D., editeurs, *Actes de la conférence Neurosciences Pour l'Ingénieur*, Nancy. Loria.

[Michie, 1961] Michie, D. (1961). Trial and error. *Science Survey*, 2 :129–145.

[Michie & Chambers, 1968] Michie, D. & Chambers, R. (1968). Boxes : an experiment in adaptive control. *Machine Intelligence*, 2 :137–152.

[Montague et al., 1996] Montague, P. R., Dayan, P., & Sejnowski, T. J. (1996). A framework for mesencephalic dopamine systems based on predictive hebbian learning. *Journal of Neuroscience*, 16(5) :1936–1947.

[Montana, 1995] Montana, D. J. (1995). Strongly Typed Genetic Programming. *Evolutionary Computation*, 3(2).

[Moore & Atkeson, 1993] Moore, A. W. & Atkeson, C. (1993). Prioritized sweeping : Reinforcement learning with less data and less real time. *Machine Learning*, 13 :103–130.

[Moore & Leach, 1970] Moore, C. H. & Leach, G. C. (1970). FORTH - A Language for Interactive Computing. internal publication, Mohasco Industries, Amsterdam NY.

[Mott, 1981] Mott, D. (1981). *Sensori-motor Learning in a Robot*. Thèse de doctorat, Department of Computer and Statistics, Queen Mary College, University of London.

[Munos, 1997] Munos, R. (1997). *Apprentissage par Renforcement, Etude du cas continu*. Thèse de doctorat, EHESS, Paris.

[Munos & Moore, 2001] Munos, R. & Moore, A. (2001). Variable resolution discretization in optimal control. *Machine Learning*, 49 :291–323.

[Nair et al., 2003] Nair, R., Pynadath, D., Yokoo, M., Tambe, M., & Marsella, S. (2003). Taming decentralized pomdps : Towards efficient policy computation for multiagent settings. Edité dans *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03)*.

[Nehmzow, 1992] Nehmzow, U. (1992). *Experiments in Competence Acquisition for Autonomous Mobile Robots*. Thèse de doctorat, University of Edinburgh.

[Newell, 1980] Newell, A. (1980). Physical symbol systems. *Cognitive Science*, 4 :135–183.

[Newell & Simon, 1972] Newell, A. & Simon, H. A. (1972). *Human Problem Solving*. Prentice-Hall.

[Nolfi & Floreano, 2000] Nolfi, S. & Floreano, D. (2000). *Evolutionary Robotics*. MIT Press, Cambridge, MA.

[Panatier et al., 2000] Panatier, C., Sanza, C., & Duthen, Y. (2000). Adaptive Entity thanks to Behavioral Prediction. Edité dans Meyer, J.-A., Wilson, S. W., Berthoz, A., Roitblat, H., & Floreano, D., editeurs, *From Animals to Animats 6 : Proceedings of the Sixth International Conference on Simulation of Adaptive Behavior (supplement book)*, pages 295–303, Cambridge, MA. MIT Press.

[Pareto, 1896] Pareto, V. (1896). *Cours d'Economie Politique*. Rouge, Lausanne.

[Partridge, 1982] Partridge, B. L. (1982). The Structure and Functions of Fish Schools. *Scientific American*, june :114–123.

[Pavlov, 1927] Pavlov, I. P. (1927). *Conditionning reflexes*. Oxford University Press, Oxford, UK.

[Pearce et al., 1992] Pearce, M., Arkin, R. C., & Ram, A. (1992). The Learning of Reactive Control Parameters through Genetic Algorithms. Edité dans *Proceedings 1992 International Conference on Intelligent Robotics and Systems (IROS)*, pages 130–137.

[Pearl, 1988] Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*. Morgan Kaufman, San Mateo, CA.

[Peng & Williams, 1992] Peng, J. & Williams, R. (1992). Efficient learning and planning within the DYNA framework. Edité dans Meyer, J.-A., Roitblat, H. L., & Wilson, S. W., editeurs, *Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, pages 281–290, Cambridge, MA. MIT Press.

[Peng & Williams, 1996] Peng, J. & Williams, R. J. (1996). Incremental Multi-Step Q-learning. *Machine Learning*, 22 :283–290.

[Pennartz et al., 1994] Pennartz, C. M. A., Groenewegen, H. J., & Lopes Da Silva, F. H. (1994). The nucleus accumbens as a complex of functionally distinct neuronal ensembles : an integration of behavioral, electrophysiological and automatic data. *Progress in Neurobiology*, 42 :719–761.

[Pennartz et al., 2000] Pennartz, C. M. A., McNaughton, B. L., & Mulder, A. B. (2000). The glutamate hypothesis of reinforcement learning. *Progress in Brain research*, 126 :231–253.

[Peret & Garcia, 2004] Peret, L. & Garcia, F. (2004). On-line search for solving mdps via heuristic sampling. Edité dans *Proceedings of the European Conference on Artificial Intelligence*.

[Piaggio et al., 2000] Piaggio, M., Sgorbissa, A., & Zaccaria, R. (2000). Micronavigation. Edité dans Meyer, J.-A., Wilson, S. W., Berthoz, A., Roitblat, H., & Floreano, D., editeurs, *From Animals to Animats 6 : proceedings of the Sixth International Conference on Simulation of Adaptative Behavior*, pages 209–218, Paris. MIT Press.

[Picault & Landau, 2001] Picault, S. & Landau, S. (2001). Ethogenetics and the Evolutionary Design of Agent Behaviors. Edité dans Callaos, N., Esquivel, S., & Burge, J., editeurs, *Proceedings of the 5th World Multi-Conference on Systemics, Cybernetics and Informatics (SCI'01)*, volume III, pages 528–533.

[Poli & Langdon, 1998] Poli, R. & Langdon, W. B. (1998). Schema Theory for Genetic Programming with One-point Crossover and Point Mutation. *Evolutionary Computation Journal*, 6(3) :231–252.

[Puterman, 1994] Puterman, M. L. (1994). *Markov Decision Problems*. Wiley, New York.

[Puterman & Shin, 1978] Puterman, M. L. & Shin, M. C. (1978). Modified policy iteration algorithms for discounted Markov Decision Problems. *Management Science*, 24 :1127–1137.

[Pynadath & Tambe, 2002] Pynadath, D. V. & Tambe, M. (2002). The communicative multiagent team decision problem : Analyzing teamwork theories and models. *Journal of Artificial Intelligence Research*, 16 :389–423.

[Rabiner & Juang, 1986] Rabiner, L. R. & Juang, B. H. (1986). An introduction to hidden markov models. *IEEE ASSP Magazine*, Jan. :4–16.

[Ram & Santamaria, 1997] Ram, A. & Santamaria, J.-C. (1997). Continuous case-based reasoning. *Artificial Intelligence*, 90(1-2) :25–77.

[Reed, 1992] Reed, B. (1992). Finding approximate separators and computing tree-width quickly. Edité dans *24th Annual Symposium on Theory of Computing, ACM*, pages 221–228.

[Rescorla & Wagner, 1972] Rescorla, R. A. & Wagner, A. R. (1972). A theory of pavlovian conditioning : variations in the effectiveness of reinforcement and nonreinforcement. Edité dans Black, A. H. & Prokazy, W. F., editeurs, *Classical Conditioning II*, pages 64–99. Appleton Century Croft, New York, NY.

[Reynolds, 1987] Reynolds, C. (1987). Flocks, Herds and Schools : a Distributed Behavioral Model. Edité dans *Computer Graphics*, volume 21, pages 25–34.

[Reynolds, 1994] Reynolds, C. (1994). Evolution of corridor following behavior in a noisy world. Edité dans *From Animals to Animats 3 : Proceedings of the Third International Conference on Simulation of Adaptive BehaviorSimulation of Adaptive Behavior*, volume 3, pages 402–410.

[R.I. Bahar et al., 1993] R.I. Bahar, E.A. Frohm, C.M. Gaona, G.D. Hachtel, E. Macii, A. Pardo, & F. Somenzi (1993). Algebraic Decision Diagrams and Their Applications. Edité dans *IEEE /ACM International Conference on CAD*, pages 188–191, Santa Clara, California. IEEE Computer Society Press.

[Riolo, 1991] Riolo, R. L. (1991). Lookahead planning and latent learning in a Classifier System. Edité dans Meyer, J.-A. & Wilson, S. W., editeurs, *From animals to animats : Proceedings of the First International Conference on Simulation of Adaptative Behavior*, pages 316–326. MIT Press.

[Robbins & Everitt, 1996] Robbins, T. W. & Everitt, B. J. (1996). Neurobehavioral mechanisms of reward and motivation. *Current opinion in neurobiology*, 6 :228–236.

[Robert, 2002] Robert, G. (2002). *Contribution des méthodologies animat et multi-agent à l'élaboration des jeux en ligne, persistants et massivement multi-utilisateurs*. Thèse de doctorat, Pré-soutenance de thèse au LIP6.

[Robertson & Riolo, 1988] Robertson, G. G. & Riolo, R. L. (1988). A tale of two classifier systems. *Machine Learning*, 3 :139–159.

[Rose, 1970] Rose, D. J. (1970). Triangulated graphs and the elimination process. *Journal of Mathematical Analysis and Its Applications*, 32 :597–609.

[Ryan & O'Neill, 1998] Ryan, C. & O'Neill, M. (1998). Grammatical evolution : A steady state approach. Edité dans Koza, J. R., editeur, *Late Breaking Papers at the Genetic Programming 1998 Conference*, University of Wisconsin, Madison, Wisconsin, USA. Stanford University Bookstore.

[Samuel, 1959] Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3 :221–229.

[Santamaria et al., 1997] Santamaria, J.-C., Sutton, R., & Ram, A. (1997). Experiments with reinforcement learning in problems with continuous state and action spaces. *Adaptive Behavior*, 6(2) :163–218.

[Schaffer & Grefenstette, 1985] Schaffer, J. D. & Grefenstette, J. (1985). Multiobjective learning via genetic algorithms. Edité dans *Proceedings of the 9th International Joint Conference on Artificial Intelligence (IJCAI-85)*, pages 593–595, Los Angeles, CA. AAAI.

[Schultz, 1998] Schultz, W. (1998). Predictive reward signal of dopamine neurons. *Journal of Neurophysiology*, 80 :1–127.

[Schultz et al., 1997] Schultz, W., Dayan, P., & Montague, P. R. (1997). A neural substrate of prediction and reward. *Science*, 275 :1593–1599.

[Schwefel, 1995] Schwefel, H.-P. (1995). *Evolution and Optimum Seeking*. John Wiley and Sons, Inc.

[Seth, 1998] Seth, A. K. (1998). Evolving Action Selection and Selective Attention Without Actions, Attention or Selection. Edité dans Pfeifer, R., Blumberg, B., Meyer, J.-A., & Wilson, S. W., editeurs, *From Animals to Animats 5 : Proceedings of the Fifth International Conference on Simulation of Adaptive Behavior*, pages 139–146, Cambridge, MA. MIT Press.

[Seward, 1949] Seward, J. P. (1949). An Experimental Analysis of Latent Learning. *Journal of Experimental Psychology*, 39 :177–186.

[Shapley, 1953] Shapley, L. S. (1953). Stochastic games. *Proceedings of the National Academy of Sciences*, 39 :327–332.

[Sigaud & Gérard, 2001] Sigaud, O. & Gérard, P. (2001). Being Reactive by Exchanging Roles : an Empirical Study. Edité dans Hannebauer, M., Wendler, J., & Pagello, E., editeurs, *LNAI 2103 : Balancing reactivity and Social Deliberation in Multiagent Systems*. Springer-Verlag.

[Simonin & Ferber, 2000] Simonin, O. & Ferber, J. (2000). Modeling Self Satisfaction and Altruism to Handle Action Selection in Reactive Cooperation. Edité dans Meyer, J.-A., Wilson, S. W., Berthoz, A., Roitblat, H., & Floreano, D., editeurs, *From Animals to Animats 6 : proceedings of the Sixth International Conference on Simulation of Adaptative Behavior (supplement book)*, pages 314–323, Paris. MIT Press.

[Skinner, 1938] Skinner, B. F. (1938). *The Behavior of Organisms*. Appleton-Century Crofts, Inc., New-York.

[Smart & Kaelbling, 2000] Smart, W. D. & Kaelbling, L. P. (2000). Practical reinforcement learning in continuous spaces. Edité dans *17th International Conference on Machine Learning*.

[Smart & Kaelbling, 2002] Smart, W. D. & Kaelbling, L. P. (2002). Effective reinforcement learning for mobile robots. Edité dans *Proceedings of the IEEE International Conference on Robotics and Automation*.

[Smith, 1994] Smith, R. E. (1994). Memory exploitation in learning classifier systems. *Evolutionary Computation*, 2(3) :199–220.

[Smith, 1980] Smith, S. F. (1980). *A Learning System Based on Genetic Algorithms*. Thèse de doctorat, Department of Computer Science, University of Pittsburg, Pittsburg, MA.

[Spector et al., 2001] Spector, L., D., G. E., Wu, A., Langdon, W. B., Voigt, H. M., & Gen, M., editeurs (2001). *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO01)*. Morgan Kaufmann.

[Srinivas & Deb, 1994] Srinivas, N. & Deb, K. (1994). Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3) :221–248.

[St-Aubin et al., 2000] St-Aubin, R., Hoey, J., & Boutilier, C. (2000). APRICODD : Approximate policy construction using decision diagrams. Edité dans *NIPS*, pages 1089–1095.

[Stolzmann, 1998] Stolzmann, W. (1998). Anticipatory Classifier Systems. Edité dans Koza, J., Banzhaf, W., Chellapilla, K., Deb, K., Dorigo, M., Fogel, D. B., Garzon, M. H., Goldberg, D. E., Iba, H., & Riolo, R., editeurs, *Genetic Programming*, pages 658–664. Morgan Kaufmann Publishers, Inc., San Francisco, CA.

[Stolzmann, 1999] Stolzmann, W. (1999). Latent Learning in khepera robots with Anticipatory Classifier Systems. Edité dans Wu, A. S., editeur, *Proceedings of the 1999 Genetic and Evolutionary Computation Conference Workshop Program*, pages 290–297.

[Stolzmann et al., 2002] Stolzmann, W., Lanzi, P.-L., & Wilson, S. W., editeurs (2002). *Proceedings of the International Workshop on Learning Classifier Systems (IWLCS2003)*, Chicago, IL.

[Sun & Peterson, 2000] Sun, R. & Peterson, T. (2000). Automatic partitioning for multi-agent reinforcement learning. Edité dans Meyer, J.-A., Wilson, S. W., Berthoz, A., Roitblat, H., & Floreano, D., editeurs, *From Animals to Animats 6 : Proceedings of the Sixth International Conference on Simulation of Adaptive Behavior*, Paris. MIT Press.

[Sun & Sessions, 2000] Sun, R. & Sessions, C. (2000). Multi-agent reinforcement learning with bidding for segmenting action sequences. Edité dans Meyer, J.-A., Wilson, S. W., Berthoz, A., Roitblat, H., & Floreano, D., editeurs, *From Animals to Animats 6 : Proceedings of the Sixth International Conference on Simulation of Adaptive Behavior*, pages 317–324, Paris. MIT Press.

[Sutton, 1988] Sutton, R. S. (1988). Learning to Predict by the Method of Temporal Differences. *Machine Learning*, 3 :9–44.

[Sutton, 1990] Sutton, R. S. (1990). Planning by incremental dynamic programming. Edité dans *Proceedings of the Eighth International Conference on Machine Learning*, pages 353–357, San Mateo, CA. Morgan Kaufmann.

[Sutton & Barto, 1981] Sutton, R. S. & Barto, A. G. (1981). Toward a modern theory of adaptive network : Expectation and prediction. *Psychological Review*, 88 :135–170.

[Sutton & Barto, 1998] Sutton, R. S. & Barto, A. G. (1998). *Reinforcement Learning : An Introduction*. MIT Press.

[Szer & Charpillet, 2004] Szer, D. & Charpillet, F. (2004). Coordination through mutual notification in cooperative multiagent reinforcement learning. Edité dans *AAMAS'2004 - Third International Joint Conference on Autonomous Agents and Multi Agent Systems*, New York, NY.

[Teichteil-Koenigsbuch & Fabiani, 2004] Teichteil-Koenigsbuch, F. & Fabiani, P. (2004). Processus décisionnel de markov décomposé et factorisé pour l'optimisation d'une stratégie d'exploration. Edité dans *Actes des journées PDMIA*, LIP6, Paris.

[Thorndike, 1911] Thorndike, E. L. (1911). *Animal Intelligence*. MacMillan Company, New York.

[Thurau et al., 2004] Thurau, C., Bauckage, C., & Sagerer, G. (2004). Learning Human-like Movement Behavior for Computer Games. Edité dans Schaal, S., Ijspeert, A., Billard, A., Vijayakumar, S., Hallam, J., & Meyer,

J.-A., editeurs, *From Animals to Animats 8 : Proceedings of the Eighth International Conference on Simulation of Adaptive Behavior*, pages 315–323, Cambridge, MA. MIT Press.

[Tinbergen, 1951] Tinbergen, N. (1951). *The Study of Instinct*. Clarendon Press, Atlanta, GA.

[Tolman, 1932] Tolman, E. C. (1932). *Purposive behavior in animals and men*. Appletown, New York.

[Tomlinson & Bull, 2000a] Tomlinson, A. & Bull, L. (2000a). CXCS. Edité dans Lanzi, P.-L., Stolzmann, W., & Wilson, S. W., editeurs, *Learning Classifier Systems : From Foundations to Applications*, pages 194–208. Springer-Verlag, Heidelberg.

[Tomlinson & Bull, 2000b] Tomlinson, A. & Bull, L. (2000b). A zeroth level corporate classifier system. Edité dans Lanzi, P.-L., Stolzmann, W., & Wilson, S. W., editeurs, *Learning Classifier Systems. From Foundations to Applications*, volume 1813 of *Lecture Notes in Artificial Intelligence*, pages 306–313, Berlin. Springer-Verlag.

[Tomlinson & Bull, 2002] Tomlinson, A. & Bull, L. (2002). An Accuracy-based Corporate Classifier System. *Journal of Soft Computing*, 6(3–4) :200–215.

[Tyrrell, 1993] Tyrrell, T. (1993). *Computational Mechanisms for Action Selection*. Thèse de doctorat, University of Edinburgh.

[Utgoff et al., 1997] Utgoff, P. E., Berkman, N. C., & Clouse, J. A. (1997). Decision tree induction based on efficient tree restructuring. *Machine Learning*, 29(1) :5–44.

[Van Parunak, 1997] Van Parunak, H. (1997). "Go to the Ant" : Engineering Principles from Natural Multi-Agent Systems. Edité dans *Annals of Operations Research*, pages 69–101.

[Varela et al., 1991] Varela, F. J., Thompson, A., & Rosch, E. (1991). *The Embodied Mind : Cognitive Science and Human Experience*. MIT Press, Cambridge, MA.

[Vaughan et al., 1998] Vaughan, R., Stumpter, N., Frost, A., & Cameron, S. (1998). Robot sheepdog project achieves automatic flock control. Edité dans Pfeifer, R., Blumberg, B., Meyer, J.-A., & Wilson, S. W., editeurs, *From Animals to Animats 5 : Proceedings of the Fifth International Conference on Simulation of Adaptive Behavior*, pages 489–493, Cambridge, MA. MIT Press.

[Veldhuizen & Lamont, 2000] Veldhuizen, D. A. V. & Lamont, G. B. (2000). Multiobjective evolutionary algorithms : Analyzing the state-of-the-art. *Evolutionary Computation*, 8(2) :125–147.

[Verbeeck, 2004] Verbeeck, K. (2004). *Coordinated Exploration in Multi-Agent Reinforcement Learning*. Thèse de doctorat, VUB CoMo Lab, Brussels, Belgium.

[Watkins, 1989] Watkins, C. J. C. H. (1989). *Learning with Delayed Rewards*. Thèse de doctorat, Psychology Department, University of Cambridge, England.

[Watkins & Dayan, 1992] Watkins, C. J. C. H. & Dayan, P. (1992). Q-learning. *Machine Learning*, 8 :279–292.

[Weinberg & S., 2004] Weinberg, M. & S., R. J. (2004). Best-response multiagent learning in non-stationary environments. Edité dans *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-04)*, Columbia University, New York City.

[Whitley et al., 2000] Whitley, L. D., Goldberg, D. E., Cantú-Paz, E., Spector, L., Parmee, I. C., & Beyer, H.-G., editeurs (2000). *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO00)*. Morgan Kaufmann.

[Widrow & Hoff, 1960] Widrow, B. & Hoff, M. (1960). Adaptive switching circuits. *Western Electronic Show and Convention*, pages 96–104.

[Wiener et al., 2002] Wiener, S. I., Tabuchi, E., & Mulder, A. B. (2002). Neural representations of goal-directed action in ventral caudate and nucleus accumbens. *Society of Neuroscience Abstracts*, 32 :477.10.

[Wiering & Schmidhuber, 1997] Wiering, M. & Schmidhuber, J. (1997). HQ-Learning. *Adaptive Behavior*, 6(2) :219–246.

[Wilson, 1985] Wilson, S. W. (1985). Knowledge Growth in an Artificial Animat. Edité dans Grefenstette, J. J., editeur, *Proceedings of the 1st international Conference on Genetic Algorithms and their applications (ICGA85)*, pages 16–23. L. E. Associates.

[Wilson, 1987] Wilson, S. W. (1987). Classifier Systems and the Animat Problem. *Machine Learning*, 2 :199–228.

[Wilson, 1991] Wilson, S. W. (1991). The Animat Path to AI. Edité dans Meyer, J.-A. & Wilson, S. W., editeurs, *From Animals to Animats 1 : Proceedings of the First International Conference on Simulation of Adaptive Behavior*, Cambridge, MA. Paris, France, MIT Press.

[Wilson, 1994] Wilson, S. W. (1994). ZCS, a Zeroth level Classifier System. *Evolutionary Computation*, 2(1) :1–18.

[Wilson, 1995] Wilson, S. W. (1995). Classifier Fitness Based on Accuracy. *Evolutionary Computation*, 3(2) :149–175.

[Wilson, 2000] Wilson, S. W. (2000). Get Real ! XCS with Continuous-valued Inputs. Edité dans Lanzi, P.-L., Stolzmann, W., & Wilson, S. W., editeurs, *LNAI 1813 : From Foundations to Applications*, pages 209–220. Springer Verlag.

[Wilson, 2001] Wilson, S. W. (2001). Function Approximation with a Classifier System. Edité dans Spector, L., D., G. E., Wu, A., Langdon, W. B., Voigt, H. M., & Gen, M., editeurs, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO01)*, pages 974–981. Morgan Kaufmann.

[Wilson & Goldberg, 1989] Wilson, S. W. & Goldberg, D. E. (1989). A critical review of Classifier Systems. Edité dans *Proceedings of the Third International Conference on Genetic Algorithms*, pages 244–255, Los Altos, California. Morgan Kaufmann.

[Witkowski, 1997] Witkowski, C. M. (1997). *Schemes for Learning and behaviour : A New Expectancy Model*. Thèse de doctorat, Department of Computer Science, University of London, England.

[Witten, 1977] Witten, I. H. (1977). An adaptive optimal controller for discrete-time markov environments. *Information and Control*, 34 :286–295.

[Yao, 1999] Yao, X. (1999). Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9) :1423–1447.

[Yoshii et al., 1995] Yoshii, S., Suzuki, K., & Kakazu, Y. (1995). Lamarckian ga with genetic supervision. Edité dans *ICEC'95*.

[Young, 1998] Young, H. P. (1998). *Individual Strategy and Social Structure : An Evolutionary Theory of Institutions*. Princeton University Press, Princeton, New Jersey.

[Zeghal, 1994] Zeghal, K. (1994). *Vers une théorie de la coordination d'action. Application à la navigation aérienne*. Thèse de doctorat, LAFORIA, Université Paris VI.

[Zitzler & Thiele, 1998] Zitzler, E. & Thiele, L. (1998). Multiobjective optimization using evolutionary algorithms–a comparative study. Edité dans Eiben, A. E., editeur, *Parallel Problem Solving from Nature V*, pages 292–301, Amsterdam. Springer-Verlag.

[Zydallis et al., 2001] Zydallis, J. B., Van Veldhuizen, D. A., & Lamont, G. B. (2001). A statistical comparison of multiobjective evolutionary algorithms including the momga-ii. Edité dans Zitzler, E., Deb, K.and Thiele, L.,

Coello, C. A., & Corne, D., editeurs, *First International Conference on Evolutionary Multi-Criterion Optimization*, pages 226–240. Springer-Verlag (Lecture Notes in Computer Science No. 1993).

Index

[Aha & Molineaux, 2004], 22
[Albus, 1975], 171
[Arbib, 1981], 123
[Arbib & House, 1985], 123
[Arkin, 1989], 19, 123, 124, 126
[Arkin, 1992], 124
[Arnborg et al., 1987], 92
[Atkeson, 1991], 173
[Baerends, 1976], 142
[R.I. Bahar et al., 1993], 86
[Balch & Hybinette, 2000], 124, 125, 132, 134
[Balch & Arkin, 1995], 124
[Balch & Arkin, 1998], 124
[Barraquand et al., 1992], 124
[Barto et al., 1983], 30, 44
[Barto, 1995], 45
[Becker & Geiger, 2001], 92
[Becker, 1970], 59
[Bellman, 1957], 35
[Bernardó et al., 2001], 52
[Bernstein et al., 2002], 181, 183
[Berridge & Robinson, 1998], 45
[Bertele & Brioschi, 1972], 92
[Bertsekas, 1995], 32, 38
[Bertsekas & Tsitsiklis, 1996], 30
[Beynier et al., 2004], 183
[Booker et al., 1989], 51
[Booker, 2000], 52
[Boutilier et al., 2000], 82, 85, 86
[Boutilier et al., 1995], 50, 80, 85
[Boutilier & Goldszmidt, 1996], 80
[Boutilier, 1996], 180, 181
[Boutilier et al., 1999], 50, 75, 80
[Boutilier, 1999], 180, 181
[Hasegawa et al., 1999], 19
[Brafman & Tennenholtz, 2001], 95
[Brooks, 1991], 17, 18
[Brooks, 1986], 142
[Brown, 1951], 179
[Buffet et al., mise], 183
[Bull & O'Hara, 2002], 156
[Butz et al., 2000], 62
[Butz, 2002], 62
[Butz & Goldberg, 2003], 75
[Chadès et al., 2002], 183
[Chomsky, 1962], 111
[Chomsky & Miller, 1963], 111
[Chvatal, 1983], 87
[Clark et al., 1992], 125
[Clark, 1997], 17, 18
[Claus & Boutilier, 1998], 180
[Cliff & Ross, 1994], 106, 159
[Coello, 2003], 128
[Collet et al., 2000], 153, 158
[David W. Corne & Oates, 2001], 129
[Coulom, 2002], 47, 169, 170
[Darwiche & Goldszmit, 1994], 82
[Daucé, 2002], 148
[Dayan & Sejnowksi, 1994], 42
[Dean & Kanazawa, 1989], 81
[Deb, 2001], 129
[Deb et al., 2002], 129
[Decugis & Ferber, 1998], 142
[De Jong, 1975], 110
[DeLong, 2000], 45
[Dennett, 1990], 17
[Doncieux, 2003], 110
[Donnart & Meyer, 1996], 191
[Donnart, 1998], 109, 191, 195
[Dorigo, 1993], 67, 155
[Dorigo & Bersini, 1994], 161
[Drescher, 1991], 59, 62

[Dreyfus, 1972], 17
[Dreyfus, 1996], 17
[Erickson et al., 2001], 129
[Flacher & Sigaud, 2002], 132
[Flacher & Sigaud, 2003], 132, 135, 138, 147
[Fogel et al., 1966], 110
[Fonseca & Fleming, 1993], 129
[Moore & Leach, 1970], 114
[Friedman et al., 1998], 94
[Whitley et al., 2000], 52
[Spector et al., 2001], 52
[Langdon et al., 2002], 52
[Cantu-Paz et al., 2003], 52
[Deb et al., 2004], 52
[Banzhaf et al., 1999], 52
[Gérard, 2001], 62
[Gérard & Sigaud, 2001], 62
[Gérard et al., 2002], 62, 64
[Gérard, 2002], 63, 76
[Gérard & Sigaud, 2003], 62
[Gérard et al., 2005], 54, 62, 72, 76
[Ghahramani, 1997], 94
[Gies, mise], 179
[Girard, 2002], 45
[Goldberg, 1989], 51, 110
[Graybiel, 1995], 45
[Grefenstette et al., 1990], 160
[Grefenstette, 1991], 153
[Gruau, 1994], 111
[Guessoum, 2003], 195
[Guestrin et al., 2001], 99, 195
[Guestrin et al., 2002a], 99, 195
[Guestrin et al., 2002b], 95
[Guestrin et al., 2003a], 99
[Guestrin et al., 2003b], 81, 92, 93
[Guillot & Meyer, 2001], 17
[Hansen, 1998], 109
[Harnad, 1990], 17
[Hart et al., 1968], 186, 191
[Hart et al., 1972], 186, 191
[Heckerman et al., 1995], 94
[Heguy, 2003], 156
[Herbart, 1825], 58
[Hinton & Nowlan, 1987], 153
[Hoey et al., 1999], 86
[Hoffmann, 1993], 62
[Holland, 1975], 51, 110
[Holland & Reitman, 1978], 51, 56, 61, 106
[Holland, 1985], 161
[Holland et al., 1986], 51
[Holmes, 2002], 52
[Horn et al., 1994], 129
[Houk et al., 1995], 45
[Howard, 1971], 37, 38, 85
[Hu & Wellman, 2003], 179
[Hu, 2003], 179
[Hurst & Bull, 2004], 156
[IBM Corporation, 2003], 20
[Ikemoto & Panksepp, 1999], 45
[Lanzi et al., 2001], 52
[Lanzi et al., 2002a], 52
[Lanzi et al., 2002b], 52
[Stolzmann et al., 2002], 52
[Lanzi et al., 2000], 52
[Jakobi, 1997], 19
[Jozefowiez, 2001], 18, 29, 30, 32, 40, 45
[Kaelbling et al., 1998], 105
[Kalman, 1960], 81
[Kearns et al., 2002], 96
[Kearns & Singh, 1998], 77, 94
[Kearns & Koller, 1999], 77, 81, 94
[Khamassi, 2003], 45
[Khamassi et al., 2004], 197
[Kjærulff, 1993], 92
[Klopf, 1972], 30
[Klopf, 1975], 30
[Knowles & Corne., 1999], 129
[Kodjabachian & Meyer, 1995], 111
[Kodjabachian & Meyer, 1998], 111
[Koller & Parr., 2000], 50, 81
[Korenz & Borenstein, 1991a], 123
[Korenz & Borenstein, 1991b], 123
[Koza, 1992], 110
[Krogh, 1984], 123
[Kuvayev & Sutton, 1996], 61
[Labbé & Sigaud, 2004], 190

[Laird, 2000], 193
[Landau & Picault, 2001], 110, 115
[Landau et al., 2002b], 112
[Landau et al., 2002a], 110
[Landau & Picault, 2002], 114
[Landau, 2003], 110
[Landau & Sigaud, 2004], 112, 118
[Lanzi & Riolo, 2000], 52
[Lanzi & Wilson, 2000], 106, 107, 159
[Lanzi, 2002], 54, 55, 157
[Lanzi, 1998], 106
[Lanzi, 1999], 156
[Lanzi & Perrucci, 1999], 156
[Laurent, 2002], 34
[Lin & Mitchell, 1992], 107
[Lin, 1993], 61
[Lindenmayer, 1968], 111
[Littman, 1994], 176
[Lorenz, 1950], 142
[Luenberger, 1979], 81
[Luke & Spector, 1996], 111
[Lutz, 2004], 87, 98
[Maes, 1990], 142
[Matarić, 1994a], 124, 142
[Matarić, 1994b], 125
[Matarić, 1995], 124
[McCallum, 1995], 107
[McCarthy & Hayes, 1969], 17
[McCallum et al., 1995], 173
[Métivier & Lattaud, 2002], 109
[Meuleau et al., 1999], 109
[Meyer, 1996a], 17
[Meyer, 1996b], 17
[Meyer, 1998], 17
[Meyer, 1999], 193
[Michie, 1961], 30
[Michie & Chambers, 1968], 30
[Montague et al., 1996], 45
[Montana, 1995], 110
[Moore & Atkeson, 1993], 61
[Mott, 1981], 59
[Munos & Moore, 2001], 170, 171
[Munos, 1997], 47, 169
[Nair et al., 2003], 182
[Nehmzow, 1992], 18
[Newell & Simon, 1972], 17
[Newell, 1980], 17
[Nolfi & Floreano, 2000], 19
[Panatier et al., 2000], 125
[Pareto, 1896], 129
[Partridge, 1982], 124
[Van Parunak, 1997], 122
[Pavlov, 1927], 28
[Pearce et al., 1992], 123
[Pearl, 1988], 81
[Peng & Williams, 1992], 61
[Peng & Williams, 1996], 43
[Pennartz et al., 2000], 45, 46
[Pennartz et al., 1994], 45
[Peret & Garcia, 2004], 96
[Garcia, mise], 183
[Piaggio et al., 2000], 123
[Picault & Landau, 2001], 110
[Poli & Langdon, 1998], 111
[Adobe Systems, 1985], 114
[Puterman & Shin, 1978], 38, 85
[Puterman, 1994], 88
[Pynadath & Tambe, 2002], 182
[Rabiner & Juang, 1986], 32, 81, 94
[Ram & Santamaria, 1997], 173
[Reed, 1992], 92
[Rescorla & Wagner, 1972], 30
[Reynolds, 1987], 124
[Reynolds, 1994], 125
[Riolo, 1991], 61, 108, 155
[Robbins & Everitt, 1996], 45
[Robert, 2002], 21, 190
[Robertson & Riolo, 1988], 106, 155
[Rose, 1970], 92
[Ryan & O'Neill, 1998], 114
[Samuel, 1959], 30
[Santamaria et al., 1997], 172
[Schaffer & Grefenstette, 1985], 129
[Schultz et al., 1997], 45
[Schultz, 1998], 45
[Schwefel, 1995], 110

[Seth, 1998], 142
[Seward, 1949], 58
[Shapley, 1953], 176
[Sigaud & Gérard, 2001], 50
[Simonin & Ferber, 2000], 125
[Skinner, 1938], 28, 29
[Smart & Kaelbling, 2000], 174
[Smart & Kaelbling, 2002], 174
[Smith, 1980], 51
[Smith, 1994], 106
[Srinivas & Deb, 1994], 129
[St-Aubin et al., 2000], 86
[Stolzmann, 1998], 61, 62
[Stolzmann, 1999], 108, 109, 159
[Sun & Peterson, 2000], 109
[Sun & Sessions, 2000], 109, 161
[Sutton & Barto, 1981], 30
[Sutton, 1988], 32, 41
[Sutton, 1990], 59, 61, 73, 76
[Sutton & Barto, 1998], 29, 30, 36, 37, 61, 73, 162
[Szer & Charpillet, 2004], 182
[Teichteil-Koenigsbuch & Fabiani, 2004], 185
[Thorndike, 1911], 28, 29
[Thurau et al., 2004], 175
[Tinbergen, 1951], 142
[Tolman, 1932], 28, 57
[Tomlinson & Bull, 2000a], 108, 109, 159
[Tomlinson & Bull, 2002], 109
[Tomlinson & Bull, 2000b], 108, 159
[Tyrrell, 1993], 142, 147
[Utgoff et al., 1997], 96
[Veldhuizen & Lamont, 2000], 129
[Varela et al., 1991], 17
[Vaughan et al., 1998], 125, 131
[Verbeeck, 2004], 179
[Watkins, 1989], 32
[Watkins & Dayan, 1992], 40, 42, 43
[Weinberg & S., 2004], 179
[Widrow & Hoff, 1960], 66, 67
[Wiener et al., 2002], 45
[Wiering & Schmidhuber, 1997], 109
[Wilson, 2000], 166
[Wilson, 2001], 54
[Wilson, 1985], 17
[Wilson, 1987], 17
[Wilson & Goldberg, 1989], 52, 106, 108, 159
[Wilson, 1991], 17
[Wilson, 1994], 52, 56, 106, 108
[Wilson, 1995], 52, 56, 61
[Witkowski, 1997], 59, 62
[Witten, 1977], 44
[Yao, 1999], 110
[Yoshii et al., 1995], 152
[Young, 1998], 179
[Zeghal, 1994], 124, 138
[Eckart Zitzler & Thiele, 2001], 129
[Zitzler & Thiele, 1998], 129
[Zydallis et al., 2001], 129
action-set, 163
fitness, 51

action jointe, 176
algorithmes génétiques, 51
anticipation, 58
apprentissage latent, 28, 57–59
apprentissage par population, 154
apprentissage par renforcement, 29, 32
apprentissage par renforcement continu, 169
apprentissage par renforcement indirect, 59
approche animat, 17
approche Michigan, 51
approche Pittsburgh, 51
architecture Dyna, 59
architectures acteur-critique, 44
ATNoSFERES, 110

balayage prioritaire, 61

champs de potentiel, 123
COM-MTDP, 182
comportements élémentaires, 142
coordination spatiale, 121

DEC-POMDP, 181

ensemble d'actions Pareto-optimal, 179
équilibre de Nash, 179
état interne, 103

FACS, 160
front de Pareto, 129

généralisation, 54
GACS, 125

hypothèse de Markov, 33

itération sur la politique, 37
itération sur la valeur, 38
ITI, 96

jeu de Markov, 177
jeux de Markov, 176

méthodes de Monte Carlo, 39
MACS, 63
match-set, 55
MMDP, 180
modèle interne, 59, 103

processus de décision markovien, 31
processus de décision markovien factorisé, 80
processus de décision markovien multi-agents, 180
processus de décision markovien partiellement observable, 101
programmation dynamique, 37
programmation neurodynamique, 30

Q-learning, 42

réseau bayésien dynamique, 81
réseaux bayésiens, 81

site d'attachement, 135
système de classeurs, 49, 52
systèmes de classeurs à anticipation, 62

théorie des schémas, 124
trace d'éligibilité, 43