UNIVERSITÉ DE SHERBROOKE

Faculté de génie

Département de génie électrique et de génie informatique

L'ARCHITECTURE ACTEUR CRITIQUE POUR LE CONTRÔLE DES SYSTÈMES

Thèse de doctorat es sciences appliquées

Spécialité : génie électrique

Sidi Mohamed OULD MOHAMED EL MUSTAPHA

Sherbrooke (Québec), Canada Mai 2003

Your file Votre référence
ISBN: 0-612-86723-4
Our file Notre référence
ISBN: 0-612-86723-4

à mes parents pour tous leurs sacrifices
à ma très chères sœur Mariem
à mes chers frères
... et à toi!

RÉSUMÉ

Les recherches menées dans cette thèse portent sur la conception d'agents intelligents autonomes pour le contrôle des systèmes en particulier et pour la résolution des problèmes complexes d'optimisation en général. Les approches actuelles dans ce domaine sont divisées principalement en deux grandes catégories ayant en commun de s'inspirer des principes de l'intelligence humaine pour concevoir des solutions intelligentes. La première catégorie vise à concevoir des agents assez intelligents pour extraire des connaissances à partir de l'expérience d'un superviseur externe. Ces agents visent essentiellement à reproduire la compétence d'un expert humain. Il s'agit des systèmes d'apprentissage supervisé. La deuxième catégorie vise plutôt à concevoir des agents capables d'apprendre, de façon autonome, de leurs propres expériences. Il s'agit des systèmes d'apprentissage par renforcement. Le travail présenté dans cette thèse portent sur la combinaison des deux approches afin de concevoir des agents assez flexibles pour assimiler toute l'expertise humaine disponible à priori, mais aussi assez généraux pour apprendre de façon autonome à accomplir des tâches complexes même en l'absence de toute aide externe. Nous avons conçu une nouvelle famille d'agents intelligents, désignés par l'acronyme FNAC signifiant *Fuzzy Neural Actor Critic*. La mise en œuvre de chaque agent FNAC utilise la logique floue, pour faciliter l'incorporation des informations disponibles à priori et la validation des solutions apprises, et les réseaux de neurones artificiels pour leur capacité d'apprentissage. Nous avons validé les performances de l'agent FNAC sur les problèmes du pendule inverse et de la conduite d'un vélo. Les résultats démontrent la pertinence et la validité des contributions proposées

REMERCIEMENTS

Je remercie mon directeur de recherche, M. Gérard Lachiver, pour m'avoir donné la latitude nécessaire afin d'aborder une problématique nouvelle ainsi que pour tout son support tout le long de cette thèse. Ses précieuses directives scientifiques ont été pour moi le meilleur des guides. J'ai particulièrement apprécié son respect pour l'autonomie de ses étudiants. Par dessus tout, M. Lachiver fait partie de ces rares personnes qui ont le don de vous redonner le goût de continuer même dans les moments les plus difficiles. Ce fut un grand plaisir de travailler avec lui.

Je tiens sincèrement à remercier Monsieur Adrien Leroux d'Avoir accepté la présidence de mon jury.

Mes plus vifs remerciements vont à Madame Doina Precup, Professeur à l'université McGill, qui a pris le temps d'examiner mon travail avec attention et rigueur. Ses commentaires ont grandement contribué à enrichir et à clarifier la modeste contribution de cette thèse.

Au delà de ces commentaires très pertinents en tant qu'examinateur de cette thèse, je suis reconnaissant à Monsieur Frodulad Kabanza, Professeur à l'université de Sherbrooke, pour toutes les discussions enrichissantes que nous avons eu aussi bien sur le thème de l'apprentissage par renforcement que sur le thème de la planification et de l'intelligence artificielle en général.

Je remercie très sincèrement Monsieur Ruben Gonzalez, Professeur à l'université de Sherbrooke, d'avoir accepté d'examiner ce travail et surtout d'assumer le rôle du rapporteur.

Merci aussi à tout ceux qui de près ou de loin ont contribué à la réalisation de ce travail.

TABLES DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

LEXIQUE

Acteur : élément adaptatif utilisé pour implémenter et améliorer une stratégie de contrôle.

Agent : une entité physique ou virtuelle capable d'agir dans un environnement, d'observer même de manière partielle son environnement, et qui peut éventuellement évoluer.

Agent glouton : un agent qui explore uniquement sur la base de la fonction d'évaluation.

Critique : élément adaptatif utilisé pour estimer et implémenter la fonction d'évaluation associée à une stratégie de contrôle.

Épisode : nombre d'essais nécessaires à un agent pour réussir sa mission

Essai : temps écoulé avant que un agent ne se trouve dans un état terminal.

Exploitation : Mode de fonctionnement où un agent suit strictement sa stratégie de contrôle.

Exploration : Mode de fonctionnement où un agent explore des actions différentes de celles calculées par sa stratégie de contrôle.

Fonction de renforcement : une application de l'espace d'états dans l'espace des nombres réels qui à chaque état associe un signal scalaire. Elle critique les décisions de l'agent dans le court terme. Son principal objectif est de définir le but de l'agent.

Fonction d'évaluation : une application de l'espace d'états dans l'espace des nombres réels qui à chaque état associe la somme cumulative des renforcements espérés par l'agent à long terme. Elle critique les décisions de l'agent dans le long terme. Son principal objectif est de définir une relation d'ordre sur l'espace d'états.

Stratégie de contrôle : une application de l'espace d'états dans l'espace des actions. Elle définit le comportement de l'agent.

LISTE DES ACRONYMES

AC : Actor Critic

AET : Accumulating Eligibility Traces

FNAC : Fuzzy Neural Actor Critic

MC : Monte-Carlo

PD : Programmation Dynamique

PDM : Processus De décision de Markov

PI : Policy Iteration

QL : Q-Learning

RET : Replacing Eligibility Traces

RL : Reinforcement Learning

RNA : Réseau de Neurones Artificiel

SIF : Système d'Inférence Flou

TD : Temporal Difference

TS : Takagi-Sugueno

VI : Value Iteration

1 INTRODUCTION

La révolution industrielle et technologique nous a permis de confier la majorité des tâches physiques aux machines. Aujourd'hui, en pleine ère cybernétique, nous confions aux ordinateurs de plus en plus de tâches décisionnelles jadis effectuées par les humains telles que la gestion du trafic routier dans les grands centres urbains, la gestion de portefeuilles boursiers, le routage des données dans les réseaux de communication, ou encore de jouer aux échecs ou au backgammon. Ces tâches ont en commun de poser de difficiles problèmes de prise de décisions séquentielles :

- L'accomplissement de telles tâches ne se fait pas par une seule décision, mais plutôt par une série de décisions.
- Le résultat d'une décision peut dépendre de facteurs environnementaux au-delà du contrôle de l'ordinateur.
- L'objectif ultime, mesuré en terme de congestion des routes, rendement d'un portefeuille ou la victoire dans un jeux, dépend de façon complexe de plusieurs décisions et de leurs résultats aléatoires.

Pour de tels problèmes, on ne dispose pas de stratégies de prise de décisions optimales et il est souvent d'usage de recourir à des heuristiques basées sur une expertise humaine pour leur trouver des solutions acceptables. Les systèmes experts matérialisent cette idée. Malheureusement, cette expertise humaine est souvent vague, incomplète, voire même indisponible, et de plus, il est difficile de la coder avec précision sous forme logicielle. Une approche issue de l'intelligence artificielle, aux adeptes de plus en plus nombreux, propose la méthodologie suivante :

Un algorithme de prise de décision peut apprendre de façon autonome des stratégies de contrôle efficaces pour des tâches séquentielles simplement en simulant ces tâches et en formant des statistiques sur les décisions qui ont tendance à conduire à de bonnes performances et sur celles qui ne l'ont pas (Boyan, 1998).

L'apprentissage par renforcement, auquel cette thèse contribue, concrétise et définit un cadre d'étude pour cette méthodologie. Il s'agit de programmer un ou plusieurs agents pour accomplir des tâches complexes dans un environnement inconnu au moyen d'une évaluation par pénalité/récompense sans avoir besoin de spécifier comment la tâche doit être remplie. La seule information disponible sur la qualité des actions effectuées est la suivante : si une suite de comportements débouche sur une situation positive (négative), le système qui a produit ces actions est récompensé (puni). À partir de cette seule information, la procédure d'apprentissage vise à améliorer le choix des actions afin de maximiser les récompenses, d'où le terme d'apprentissage par renforcement. Ce serait par exemple le cas d'un joueur qui n'apprendrait qu'en fonction des parties gagnées ou perdues qu'il aurait jouées.

Bien qu'il n'y ait pas de réel consensus sur la définition d'un agent, on sous-entendra, dans toute cette thèse, par le terme agent une entité physique ou virtuelle ayant les propriétés suivantes :

- Il est capable d'agir dans un environnement;
- Il est capable d'observer, au moins de manière partielle, son environnement;
- Il ne dispose que d'une représentation partielle de cet environnement (et éventuellement aucune);
- Il peut éventuellement évoluer.

1.1 Motivation

Une des caractéristiques d'un système intelligent est sa capacité d'apprendre à améliorer ses performances en évaluant de façon autonome et à long terme la qualité de ses interactions avec son milieu environnant. Cette propriété trouve ses racines en psychologie béhavioriste, en particulier de l'étude de l'apprentissage chez les animaux où l'on observe une association entre un objectif à atteindre et un simple signal scalaire perçu comme une récompense ou une punition. Lorsque l'apprenant réalise une bonne action, il reçoit une récompense; tandis que lorsqu'il se trompe, il reçoit une punition. L'apprentissage par renforcement repose donc sur une association entre un objectif à atteindre et un simple signal scalaire, appelé signal de renforcement.

Contrairement à l'apprentissage supervisé, mieux connu, qui vise à modéliser une relation d'entrée/sortie donnée, l'apprentissage par renforcement cherche à faire émerger des comportements intelligents permettant d'atteindre un objectif par interaction de type essai/erreur entre un agent et un environnement dynamique. Dans ce type d'apprentissage, un agent analyse en permanence les conséquences de ses actions, en ayant tendance à reproduire préférentiellement celles qui, dans les mêmes circonstances (états), ont tendance à maximiser la somme des renforcements reçus pendant un épisode d'apprentissage (par exemple une partie de jeux). Par conséquent, l'agent acquiert de la connaissance à partir de sa propre expérience et non de celle d'un instructeur comme c'est le cas dans l'apprentissage supervisé.

En réalité, les deux approches sont complémentaires. Tandis que l'apprentissage par renforcement cherche à calculer des solutions (stratégies de contrôle ou systèmes de prise de décision) optimales, l'apprentissage supervisé cherche, quant à lui, à implémenter des solutions déjà existantes de façon optimale. D'ailleurs, la majorité des algorithmes d'apprentissage par renforcement commencent d'abord par formuler le problème à résoudre sous la forme d'un problème d'apprentissage supervisé en calculant de façon incrémentale des solutions intermédiaires, et en utilisant ensuite les techniques de ce dernier, mieux connues et mieux développées, pour coder ces solutions. D'autre part, les solutions obtenues par le biais de l'apprentissage supervisé sont souvent incomplètes, car pour des

problèmes de tailles réalistes, il est pratiquement impossible de disposer de manière exhaustive de tous les exemples (entrées, sorties). Par contre, en l'absence de connaissances préalables, les solutions obtenues par le biais de l'apprentissage par renforcement nécessitent souvent un temps d'apprentissage élevé vu la pauvreté du signal de renforcement. On peut donc imaginer un schéma d'apprentissage où l'on utilise l'apprentissage par renforcement pour compléter les connaissances d'un expert dans les états ignorés ou oubliés par ce dernier (Berenji, 1992).

L'application de l'apprentissage par renforcement pour le contrôle intelligent des systèmes est motivée par le succès du contrôle adaptatif, mais aussi et surtout par ses limitations. En effet, le contrôle adaptatif permet de calculer des stratégies de contrôle optimales pour des systèmes dynamiques pour lesquels on ne dispose pas de modèles mathématiques fiables et compacts. Dans ce type de contrôle, on cherche à minimiser une fonction de coût en ajustant les paramètres d'un contrôleur. Dans la majorité des cas, la fonction de coût est une fonction quadratique mesurant la déviation du système par rapport à une trajectoire désirée et l'énergie dépensée par l'agent pour la suivre. Le problème de contrôle est alors réduit à un simple problème d'optimisation et l'intelligence de l'agent est souvent réduite à la détection des changements des conditions environnementales. Mais contrairement à l'apprentissage par renforcement, l'agent n'a ni le pouvoir d'explorer dans une direction autre que celle fixée par le concepteur, ni celui de comparer des décisions sur une base individuelle. Ces deux caractéristiques sont pourtant l'essence même de notre intelligence sinon du moins de notre capacité d'apprentissage. D'un autre côté, bien que la connaissance complète d'un modèle dynamique du système ne soit pas requise dans le contrôle adaptatif, la structure de ce dernier doit être déterminée au préalable (ordre du système, linéaire ou non, etc.), et complétée par une estimation paramétrique du modèle. Or, pour des applications aussi communes que la gestion du trafic routier ou le routage dans les réseaux de communication, il est souvent impossible de trouver la structure d'un modèle mathématique compact, de taille raisonnable et analytiquement traitable. Par conséquent, de telles applications, sans être nécessairement difficiles ou complexes, ne peuvent pas être traitées directement par les méthodes du contrôle adaptatif, qui malgré ses mérites indéniables, n'a guère réussi à s'étendre réellement au-delà des applications du contrôle conventionnel. Par contre, la flexibilité de la fonction de renforcement, exclusivement

destinée à spécifier les objectifs de l'agent sans se soucier de comment les atteindre, et l'absence de toute contrainte sur le modèle du système, font de l'apprentissage par renforcement une alternative très prometteuse et une extension de la théorie du contrôle optimale pour adresser des problématiques jadis réservées à l'intelligence artificielle.

On trouve des applications de l'apprentissage par renforcement dans plusieurs disciplines. Tesauro l'a utilisé avec un succès retentissant pour concevoir un logiciel de backgammon, appelé TD-Gammon, qui a gagné plusieurs championnats contre des experts humains de renommée internationale (Tesauro, 1992). Plus impressionnant encore, TD-Gammon atteint ce niveau de performance en ayant comme seule connaissance initiale les règles du jeu. Crites et Barto (Crites, 1996) ont proposé d'utiliser l'apprentissage par renforcement pour minimiser le temps d'attente d'un système de 4 ascenseurs desservants un immeuble de 10 étages. La solution qu'ils ont obtenue, quoiqu'elle ne soit pas plus performante que les heuristiques habituellement utilisées dans l'industrie pour résoudre ce problème, a le mérite d'être apprise automatiquement sans recours à aucune expertise humaine. Tong et Brown (Tong, 2002) ont formulé le problème du contrôle d'admission d'appels et leur routage dans les réseaux multimédia comme un processus dynamique semi-markovien et ils ont, par la suite, utilisé une version hiérarchique de l'algorithme Q-learning pour résoudre le problème. La stratégie de contrôle ainsi apprise a donné de meilleurs résultats que les heuristiques habituellement utilisées pour ce problème. D'autres applications de l'apprentissage par renforcement sont rapportées continuellement et couvrent des domaines aussi divers que la planification des tâches manufacturières (Zhang, 1996), l'allocation dynamique de canaux pour maximiser la capacité des systèmes de téléphonie cellulaire (Singh, 1997), ou encore la gestion du trafic dans les réseaux de communications de haute vitesse (Atlasis, 2002).

L'intérêt que nous portons, dans cette thèse, à l'apprentissage par renforcement est motivé par les possibilités qu'il offre pour réduire le fossé entre les théories du contrôle optimal et celles de l'intelligence artificielle. Nous proposons donc de l'étudier en tant que méthode hybride de conception de systèmes de contrôle autonomes. On s'intéresse spécialement aux applications de contrôle dans lesquelles l'espace d'actions est typiquement continu. Ce contrôle consiste généralement à amener un processus dans un état donné, à le maintenir

dans une zone prédéfinie ou bien encore à éviter certaines zones indésirables ou trop dangereuses.

Comme toutes les méthodes d'apprentissage par interaction de type essai/erreur, l'apprentissage par renforcement souffre d'un problème de vitesse de convergence. Cette thèse contribue à l'amélioration de cet aspect.

1.2 Contribution de la thèse

Cette thèse est consacrée principalement à une famille de méthodes d'apprentissage par renforcement connue sous le nom de *méthodes acteur critique*. On s'y intéresse également aux systèmes neuro-flous qu'on utilisera pour l'implémentation de ces méthodes. Dans ce cadre, nous proposons un ensemble de nouveaux concepts visant à analyser et à améliorer les performances et à combler certaines des limitations des principales méthodes acteur critique existantes. Par la suite, nous introduisons l'agent FNAC (Fuzzy Neural Actor-Critic) dont les principales caractéristiques sont :

- Aucun modèle de l'environnement n'est nécessaire;
- Le raisonnement de l'agent est lisible;
- S'il existe des connaissances a priori, il peut aisément les intégrer; accélérant ainsi le processus d'apprentissage;
- Très bonne capacité de généralisation.
- L'espace d'actions est continu;
- Flexibilité : permet le contrôle multivariable et/ou hiérarchique;
- Meilleures performances comparativement aux architectures RL existantes.

Les principales contributions de ce travail sont liées à l'accélération du processus d'apprentissage ainsi qu'à la qualité de la solution apprise. En ce qui concerne les

performances d'apprentissage, nous avons mis en évidence l'existence d'un problème d'inconsistance entre les processus d'évaluation et d'amélioration des stratégies de contrôle. Nous avons par la suite proposé une solution pour atténuer ce problème. Nous avons également proposé une nouvelle méthode permettant l'usage de traces d'éligibilité pour l'apprentissage de l'acteur ainsi qu'une nouvelle sorte de traces d'éligibilité substitutives particulièrement conçues pour l'usage avec des approximateurs neuro-flous mais assez générales pour être utilisées avec n'importe quel type d'approximateur linéaire. Notre approche intuitive d'équilibre entre l'exploitation et l'exploration est aussi originale et se distingue par la simplicité de sa mise en œuvre. Finalement, mentionnons que nous avons effectué un ensemble d'études empiriques qui nous ont permis de répondre concrètement à des questions de mise en œuvre pratique de l'agent FNAC tels que, par exemple, l'initialisation des paramètres du critique.

En ce qui concerne la qualité de la stratégie de contrôle apprise, nous avons d'une part montré que sa globalité sur l'espace d'état peut être considérablement améliorée en utilisant des heuristiques d'exploration tel que l'utilisation de conditions initiales aléatoires, l'utilisation d'une stratégie de contrôle non discriminante, i.e. où chaque action à une chance (non nulle) d'être exécutée par l'agent. L'analyse de la loi de contrôle apprise nous a également confirmé que l'exploration stochastique, bien qu'elle puisse ralentir l'apprentissage, permet également d'améliorer la robustesse de la solution apprise. D'autre part, nous avons montré qu'on peut lisser la loi de contrôle apprise par apprentissage progressive. L'agent apprend progressivement en résolvant des versions simplifiées, à la difficulté croissante, du problème jusqu'à la résolution du problème complet.

Finalement, nous avons étendu et démontré la validité de la méthode acteur-critique pour la résolution des problèmes de contrôle multivariable où l'espace d'actions est continu. Dans ce cadre, nous avons proposé deux méthodes d'exploration stochastique et avons associé à chacune une méthode d'affectation des crédits adéquate.

La mise en œuvre logicielle de l'agent FNAC est réalisée par une plate-forme développée dans l'environnement Visual C++, que nous avons écrite spécialement pour les besoins de

cette thèse. Une version Matlab simplifiée a été également développée pour la visualisation des résultats, ainsi que pour la validation des stratégies de contrôle apprises.

1.3 Plan de la thèse

Cette thèse comporte cinq chapitres. Le premier chapitre introduit le sujet de la thèse et les motivations de notre intérêt pour ce sujet. On y trouve aussi le plan de la thèse.

Le chapitre 2 dresse l'état de l'art de la théorie de l'apprentissage par renforcement. Après avoir formulé le problème sous la forme d'un processus de décision de Markov (PDM) fini, nous présenterons et analyserons les principaux algorithmes de l'apprentissage par renforcement. À la lumière de cette analyse, nous validerons le choix de l'architecture d'apprentissage que nous avons adoptée pour cette thèse, en l'occurrence l'architecture *Acteur-Critique* (AC).

Le chapitre 3 commence par un examen détaillé de l'architecture AC; ce qui nous permettra d'en relever les principales faiblesses. Nous présenterons ensuite l'agent FNAC, résultat d'un ensemble de propositions visant à palier ces faiblesses. Finalement, nous validerons les performances de l'agent FNAC en l'appliquant au problème bien connu du pendule inverse.

Au chapitre 4, on procédera à une généralisation de l'architecture FNAC pour des applications aux systèmes multivariables continus. La validation de cette architecture sera démontrée par un système de contrôle d'équilibre et de poursuite d'une trajectoire d'une bicyclette.

La conclusion reprend la contribution de la thèse au problème de l'apprentissage par renforcement et discute de ses perspectives futures.

2 L'APPRENTISSAGE PAR RENFORCEMENT : ÉTAT DE L'ART

2.1 Introduction

L'étude de l'apprentissage par renforcement (RL) date des premiers travaux sur la cybernétique où il a été utilisé tant dans le domaine des statistiques, que de la psychologie, des neurosciences ou de l'informatique. Il s'agit du problème auquel fait face un agent qui doit apprendre à se comporter adéquatement à partir d'interactions de type essai/erreur avec un environnement dynamique qu'on suppose au moins partiellement observable. Il existe deux grandes approches pour résoudre ce type de problèmes (Kaelbling, 1996). La première approche consiste à chercher, dans l'espace des comportements, un état qui soit optimal par rapport à un critère d'évaluation des trajectoires de l'agent dans l'environnement. Cette approche, d'inspiration naturelle, est du ressort des algorithmes génétiques (Schmidhuber, 1996). L'idée sous-jacente aux algorithmes génétiques est d'utiliser une population de solutions potentielles et de croiser les individus afin d'obtenir de meilleurs candidats. Cette technique repose sur trois opérations qui sont la sélection (des meilleurs individus) grâce à l'évaluation, le croisement (de couples d'individus sélectionnés) et la mutation qui permet de modifier aléatoirement les valeurs des chromosomes pour la prochaine génération. La seconde approche pour résoudre un problème de type RL est basée sur la combinaison de deux domaines. D'une part, elle utilise l'apprentissage supervisé qui permet d'identifier une fonction paramétrique pour modéliser, dans une structure compacte, le comportement de l'agent dans un environnement quelconque. Cette technique nécessite l'accès à un ensemble de couples {entrée, sortie} qui échantillonne la fonction à estimer. L'accès à cette information n'étant pas toujours possible, l'approche RL invoque également la programmation dynamique (PD) utilisée traditionnellement pour résoudre des problèmes d'optimisation et de contrôle. Mais du fait de la difficulté de sa mise en œuvre, la PD est limitée par la taille et la complexité des problèmes traités. En plus,

la PD nécessite la connaissance d'un modèle du système étudié. La combinaison des deux domaines permet, d'une part, de générer les couples {entrée, sortie}, nécessaires pour l'entraînement des algorithmes d'apprentissage supervisé, au moyen de la PD et d'autre part de surmonter les difficultés de mise en œuvre des algorithmes de la PD au moyen de l'apprentissage supervisé. Dans tout ce qui suit, on s'intéressera uniquement à cette approche qu'on appellera l'approche RL.

2.2 Le modèle RL standard

Pour mieux comprendre le défi posé par les problèmes RL, considérons l'exemple d'un agent qui doit apprendre à faire du vélo. L'agent reçoit une pénalité, un renforcement négatif, à chaque fois qu'il tombe. À son premier essai, l'agent prend une série d'actions qui le conduisent à l'état "inclinaison de 15° à droite". Il a alors le choix entre deux actions possibles : se pencher vers la gauche ou vers la droite. Il choisit de se pencher vers la gauche, tombe immédiatement, et reçoit une pénalité. Par conséquent, l'agent apprend qu'il faut dorénavant éviter de se pencher vers la gauche dans l'état "inclinaison de 15° à droite". À l'essai suivant, l'agent prend une série d'actions qui le conduisent à nouveau dans l'état "inclinaison de 15° à droite". Il choisit alors de se pencher vers la droite, tombe immédiatement, et reçoit une pénalité. L'agent apprend alors que cet état, synonyme d'échec quoiqu'il entreprend, est à éviter ainsi que toute action y conduisant. Tenace, l'agent entreprend un nouvel essai et il se trouve dans l'état "inclinaison de 13° à droite". L'agent choisit de se pencher vers la droite; ce qui le conduit à l'état "inclinaison de 15° à droite" où quoi qu'il entreprenne, il tombera et recevra une pénalité. Il apprend alors à éviter toute action le conduisant à cet état désespéré. Après un nombre suffisamment grand d'interactions de type essai/erreur avec son environnement, l'agent aurait appris de façon autonome à favoriser les actions utiles au maintien de son équilibre et à défavoriser celles le conduisant ou le rapprochant de l'échec.

Cet exemple met en évidence la première particularité des agents RL : ils apprennent de leurs propres expériences contrairement à la plupart des autres méthodes d'apprentissage où l'agent cherche à mimer le comportement d'un expert humain en se basant sur l'expérience

de ce dernier. Une autre particularité intéressante des agents RL est leur capacité d'explorer des décisions différentes de celles dictées par leur propre système de prise de décision et d'enrichir ce dernier en fonction du résultat de l'exploration. Par conséquent, il faut définir un modèle d'interaction entre l'agent et son environnement.

Cet exemple illustre aussi les deux principes de base de l'apprentissage par renforcement. Premièrement, si une action prise dans un état donné provoque immédiatement une pénalité, l'agent doit apprendre à éviter cette action la prochaine fois que le même état est observé. Mais à lui seul, ce principe n'est pas suffisant pour corriger les actions prises plus tôt et qui ont conduit en partie à cet état risqué. Le second principe stipule que si dans un état donné toutes les actions possibles conduisent à l'échec, alors cet état doit être évité au même titre qu'un état d'échec. La combinaison de ces deux principes permettra à l'agent d'apprendre désormais de son expérience avant même de connaître l'échec.

Dans le premier principe, l'agent reçoit un signal de renforcement externe défini par le concepteur et traduisant son but. La valeur de ce signal externe est en général calculée ou estimée à partir de l'état observé par l'agent, mais elle échappe au contrôle de ce dernier, et dans la majorité des cas, il est incapable de la prédire. C'est la fonction de renforcement. Dans le second principe l'agent forme lui même un signal de renforcement interne afin d'évaluer l'utilité des différents états observés. C'est la fonction d'évaluation dont le but est de définir une relation d'ordre sur l'espace d'états.

De cet exemple ressortissent les 4 principaux éléments d'un problème RL que nous allons développer dans ce qui suit.

2.2.1 Modèle d'interaction agent environnement

Pour la formulation mathématique du problème RL, nous modéliserons le processus d'interaction de l'agent avec son environnement par un processus de décision de Markov (PDM) fini. À chaque instant discret, $t = 1,2,3,...$, l'agent se trouve dans un état $s_t \in S$, S étant l'ensemble des états possibles, où il doit choisir une action $a_t \in A(s_t)$, $A(s_t)$ étant

l'ensemble des actions que l'agent peut prendre dans l'état s_t. Sous l'effet de l'action a_t, l'environnement passe de l'état s_t vers un nouvel état s_{t+1} avec une probabilité de transition fixe $P(s_t, a_t, s_{t+1})$, et l'agent reçoit, en retour, un signal de renforcement r_{t+1} ainsi qu'une estimation du nouvel état s_{t+1} (Fig. 2-1). L'agent se trouve par la suite face au double défi de répartir le crédit r_{t+1} aux couples (états, actions) antérieures qui ont influencé sa valeur (problème de distribution des crédits), et de modifier sa stratégie de contrôle afin de maximiser la totalité des crédits reçus à long terme (problème d'amélioration de la stratégie de contrôle); ce qui lui permettrait d'atteindre son but ultime de façon optimale.

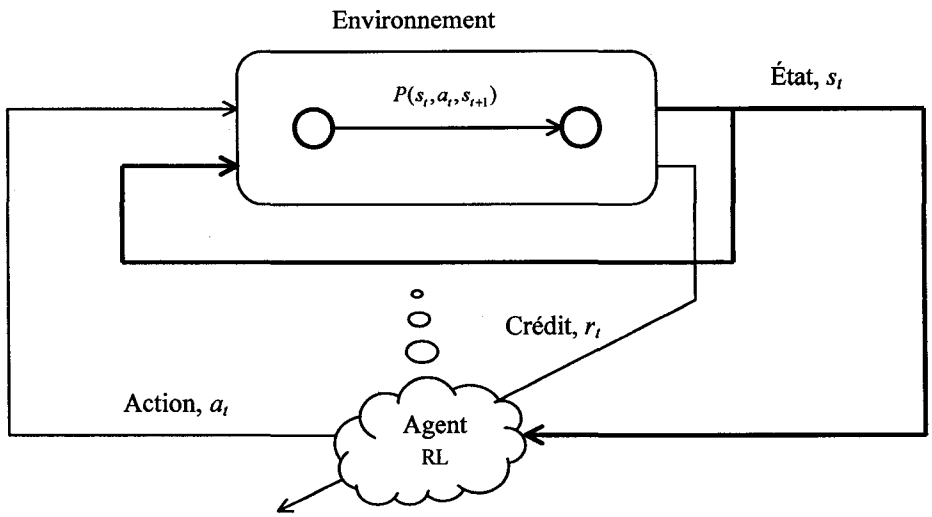


Figure 2-1 Modèle d'interaction agent – environnement dans l'approche RL.

2.2.2 Stratégie de contrôle

Elle définit la façon dont l'agent apprenant se comporte à un instant donné. Grossièrement, il s'agit d'une application de l'espace d'états dans l'espace des actions. C'est ce qu'en psychologie on pourrait qualifier de règles stimuli/réponses. La stratégie de contrôle est le cœur du processus de RL puisqu'elle est suffisante pour déterminer le comportement de l'agent.

Dans le cas général, la stratégie de contrôle est de nature stochastique et par conséquent, nous la définirons par la distribution de probabilités π qui à chaque état $s \in S$, et une action $a \in \mathrm{A}(s)$, associe une probabilité $\pi(s,a)$ de prendre l'action a dans l'état s:

$$\pi : s,a \rightarrow \pi(s,a), \quad \forall s \in S, a \in A(s) \tag{2.1}$$

Si la stratégie de contrôle est déterministe, nous utiliserons la notation suivante :

$$\pi : s \rightarrow a = \pi(s), \quad \forall s \in S, \pi(s) \in A(s) \tag{2.2}$$

2.2.3 Fonction de renforcement

Elle définit le but (la tâche) à atteindre (exécuter). Elle fait correspondre à chaque état (ou couple (état, action)) de l'environnement un scalaire, la récompense, indiquant la valeur immédiate de l'utilité de l'état. C'est une fonction scalaire fixe définie par le concepteur dans l'unique but de caractériser les états et elle échappe au contrôle de l'agent qui généralement est incapable d'en prédire la valeur. Lorsque l'agent amène son environnement dans un état dit "*bon*", il reçoit un renforcement externe positif (un crédit). Inversement, si l'agent amène son environnement dans un état dit "*mauvais*", il reçoit un renforcement externe négatif (une pénalité). La fonction de renforcement exprime donc le score immédiat associé à chaque transition d'un état vers un autre suite aux actions entreprises par l'agent.

Cependant, l'objectif ultime de l'agent dans un problème RL est de maximiser la somme des récompenses reçues ou espérées à long terme. Par conséquent, il ne doit pas se laisser distraire par des crédits immédiats généreux aux prix d'une détérioration de son état, mais plutôt favoriser les actions le rapprochant le plus de son objectif ultime. Par exemple, si l'agent cherche à apprendre à jouer au jeu d'échecs, il serait tentant de croire que la prise d'une pièce de l'adversaire doit être récompensée par un crédit positif; ce qui aura pour conséquence de détourner l'attention de l'agent de son vrai but qui est la victoire, le rendant

ainsi vulnérable aux pièges tendus par un adversaire rusé. La fonction de renforcement doit être fixée à priori pour aider l'agent à remplir sa tâche, mais en aucun cas, elle ne doit servir à communiquer des connaissances à l'agent afin de le guider vers son but. De telles connaissances pourraient être passées à l'agent via la fonction d'évaluation si elles concernent un modèle prédictif de l'environnement, ou via la stratégie de contrôle si elles concernent une stratégie de contrôle déjà existante (Benreji, 1992). Cependant, la fonction de renforcement peut servir de support à l'agent pour changer de politique.

2.2.4 Fonction d'évaluation

Le concept de fonction d'évaluation est un concept fondamental de la programmation dynamique, largement repris dans les méthodes RL. L'objectif de la fonction d'évaluation est de définir une relation d'ordre sur l'ensemble des états en attribuant à chaque état une *valeur*. Alors que la fonction de renforcement peut être vue comme un indicateur immédiat de la qualité d'une action, la fonction d'évaluation est un indicateur de la qualité du choix au long terme. Grossièrement, il s'agit ici d'une estimation de ce que l'on peut espérer du gain accessible à partir d'un état particulier si l'on repart de cet état dans le futur. De manière plus imagée, un état peut ne produire qu'une faible récompense immédiate, alors qu'il est un passage obligé vers des états qui recevront une forte récompense, et de ce fait il a une forte valeur. La récompense associée à un état est un indicateur de gain immédiat, alors que la valeur d'un état se veut un estimateur des gains à venir. Dans le cadre d'un système décisionnel, il est évident que l'on est plus intéressé par un profit à long terme que par un gain à court terme.

Dans tout ce qui suit, la valeur d'un état sera définie, tel que suggéré par Barto (1983), comme étant la valeur moyenne de la somme escomptée de tous les renforcements futurs espérés. Puisque ces derniers dépendent de la stratégie de contrôle suivie, il s'en suit qu'une fonction d'évaluation est toujours associée à une stratégie de contrôle et n'indique la valeur d'un état que sous cette stratégie. Plus précisément, la fonction d'évaluation V^{π} de la stratégie de contrôle π est définie par (Barto, 1983) :

$$V^{\pi}(s) = \mathrm{E}^{\pi}\left\{\sum_{t=0}^{\infty} \gamma^{t} r(s_t, \pi(s_t)) \middle| s_0 = s\right\} \tag{2.3}$$

où E^{π} désigne la valeur espérée lorsque l'agent suit strictement la stratégie π. γ est un facteur d'escompte, compris entre 0 et 1, qui permet de fixer l'horizon d'évaluation. On peut interpréter le facteur d'escompte de plusieurs manières. On peut voir γ^{t} comme étant la probabilité qu'une action prise plus tôt puisse altérer la valeur du signal de renforcement après t itérations. On peut aussi voir $\gamma^{t} r(s_t, \pi(s_t))$ comme étant la contribution ou la trace de l'action $\pi(s_t)$ dans la valeur du signal de renforcement après t itérations.

2.3 Méthodes de résolution d'un problème RL

Il existe trois classes fondamentales de méthodes permettant de résoudre un problème RL : la *programmation dynamique* PD, les méthodes de *Monte Carlo* MC, et l'apprentissage par *différences temporelles* TD. Chaque classe possède des avantages et des inconvénients. Par exemple, la PD, méthode d'optimisation particulièrement adaptée au contrôle optimal, possède l'avantage d'avoir des fondements mathématiques bien établis (Bertsekas, 1995). Elle permet d'estimer la fonction d'évaluation, concept qu'elle a elle même introduite (Bellman, 1957), de façon incrémentale en mettant à jour les estimations des valeurs des états à partir d'autres estimations. L'apprentissage se fait en ligne. Cependant, elle présente l'inconvénient de nécessiter de disposer d'un modèle complet de l'environnement. Les méthodes MC, quand à elles, ne nécessitent pas de modèle et sont conceptuellement plus simples. Elles présentent également un autre avantage qui est l'apprentissage de la fonction d'évaluation à partir d'expériences. Elles n'estiment la fonction d'évaluation que dans les états visités (Sutton, 1998). Mais les méthodes MC sont inadaptées à un calcul pas à pas incrémental. En effet, il faut attendre la fin de chaque expérience pour effectuer la mise à jour des caractéristiques de l'agent. Enfin, l'approche TD ne nécessite pas de modèle, est incrémentale mais est beaucoup plus complexe à analyser. Il s'agit de méthodes combinant les idées de l'approche MC et de l'approche PD. Comme dans les méthodes MC les méthodes TD peuvent apprendre directement à partir des expériences brutes sans notion de modèle de

l'environnement dynamique. Comme pour les méthodes à base de PD, elles mettent à jour leurs estimations à partir d'autres estimations apprises, sans avoir à attendre la fin de l'expérience en cours (Sutton, 1988).

2.3.1 Programmation dynamique

Le terme PD correspond à un ensemble d'algorithmes pouvant être utilisés pour calculer les politiques/stratégies optimales étant donné un modèle MDP parfait de l'environnement. L'idée clef de la PD, et du RL en général, est l'utilisation de fonctions d'évaluation pour organiser et structurer la recherche de bonnes stratégies de contrôle. L'estimation de telles fonctions est de ce fait l'élément central des méthodes RL. Les autres algorithmes classiques de la PD sont d'un intérêt limité pour l'approche RL du fait de leur coût en matière de calcul et surtout de l'hypothèse de l'existence d'un modèle de l'environnement, néanmoins ils sont d'une grande utilité théorique.

Dans la PD, on cherche à optimiser la fonction d'évaluation pour ensuite en extraire une stratégie de contrôle optimale π^*. Soit V^* la fonction d'évaluation optimale associée à π^* et définie par :

$$V^*(s) = \max_{\pi} V^{\pi}(s) \qquad \forall s \in S \tag{2.4}$$

Il a été établi que tel que défini par les équations (2.3) et (2.4), V^* vérifie l'équation d'optimalité de Bellman (Bellman, 1957) :

$$V^*(s) = \max_{a \in A(s)} \left[r(s,a) + \gamma \sum_{s' \in S} P(s,a,s') V^*(s') \right] \tag{2.5}$$

Où $P(s,a,s')$ est la probabilité de transition de l'état s vers l'état s' sous l'effet de l'action a.

L'équation (2.5) permet également de calculer une stratégie de contrôle optimale si V^* est connue :

$$\pi^*(s) = \arg\max_{a \in A(s)} \left[r(s,a) + \gamma \sum_{s' \in S} P(s,a,s') V^*(s') \right] \tag{2.6}$$

Malheureusement, les équations (2.5) et (2.6) nécessitent la connaissance d'un modèle de l'environnement, en l'occurrence les fonctions P et r. Ceci n'est presque jamais le cas dans les applications réelles. Cependant, on peut surmonter cet handicap en utilisant une autre fonction d'évaluation utilisant une représentation explicite de l'action prise dans chaque état. C'est la fonction Q définie par :

$$Q^\pi(s,a) = r(s,a) + \gamma \sum_{s' \in S} P(s,a,s') V^\pi(s') \tag{2.7}$$

Autrement dit, $Q^\pi(s,a)$ est la valeur de l'état s dans lequel l'action a est prise et en supposant que l'agent utilise strictement la stratégie π par la suite. Soit $Q^* = Q^{\pi^*}$. En appliquant l'équation d'optimalité de Bellman, on trouve :

$$Q^*(s,a) = r(s,a) + \gamma \sum_{s' \in S} P(s,a,s') \left\{ \max_{a' \in A(s')} Q^*(s',a') \right\} \tag{2.8}$$

D'où :

$$V^*(s) = \max_{a \in A(s)} Q^*(s,a) \qquad \forall s \in S \tag{2.9}$$

$$\pi^*(s) = \arg\max_{a \in A(s)} Q^*(s,a) \qquad \forall s \in S \tag{2.10}$$

Par conséquent, si Q^* est connue, alors π^* peut être calculée sans recours à un modèle de l'environnement. Cet avantage de la fonction Q par rapport à la fonction V est au prix d'une plus grande complexité de calcul et d'une large utilisation de l'espace mémoire requis.

2.3.2 Algorithme TD(λ) pour l'estimation de la fonction d'évaluation

Résoudre un problème RL revient donc principalement à estimer une fonction d'évaluation. Pour estimer les fonctions d'évaluation (V ou Q, on a omis intentionnellement le π dans le souci d'alléger le texte), on peut éviter le recours à un modèle de l'environnement en estimant l'espérance mathématique d'une variable par sa valeur instantanée. Étant donné que cet estimateur est non biaisé, on peut l'utiliser sans risque d'erreur dans des algorithmes statistiques à base de moyennes comme l'algorithme du gradient descendant. Ainsi, la fonction d'évaluation devient:

$$\begin{aligned} V(s) &= \sum_{t=0}^{\infty} \gamma^t r(s_t, \pi(s_t)), \qquad s_0 = s \\ &= r + \gamma \sum_{t=0}^{\infty} \gamma^t r(s_{t+1}, \pi(s_{t+1})), \qquad r = r(s, \pi(s)) \\ &= r + \gamma V(s'). \end{aligned} \tag{2.11}$$

où s' est l'état observé suite à l'exécution de l'action a dans l'état s et r est la valeur du signal de renforcement suite à la transition de l'environnement de l'état s vers l'état s'.

Cette propriété de récursivité de la fonction d'évaluation est à l'origine du très populaire algorithme TD(0) dont le principe est de corriger la valeur actuelle de l'état, $V(s)$, dans la direction de son estimé $r + \gamma V(s')$ vu que cette dernière comprend la valeur réelle du signal de renforcement et est donc vraisemblablement plus précise. Plus précisément,

l'algorithme TD(0) utilise le signal d'erreur $\hat{r} = r + \gamma V(s') - V(s)$, appelé aussi signal de renforcement interne, pour mettre à jour la fonction d'évaluation de la manière suivante :

$$V(s) \leftarrow V(s) + \alpha[r + \gamma V(s') - V(s)] \tag{2.12}$$

Si le taux d'apprentissage α est correctement ajusté, par une décroissance lente, et si la stratégie de contrôle reste inchangée, il est garanti que l'algorithme TD(0) converge vers la vraie fonction d'évaluation voir (Sutton, 1988). Par contre, cette convergence risque d'être très lente puisque TD(0) permet de mettre la fonction V à jour seulement dans le dernier état visité s. Cependant, d'après la définition même de la fonction d'évaluation (Éq. 2-3 et Éq. 2-11), la valeur du signal de renforcement affecte la valeur de la fonction d'évaluation dans tous les états récemment visités. Cette propriété est pleinement exploitée par l'algorithme TD(λ) de Sutton, dont TD(0) n'est qu'un cas particulier. Principalement, l'algorithme TD(λ) suggère d'associer à chaque état une trace d'éligibilité qui permet de mesurer la responsabilité de l'état dans les renforcements ultérieurs reçus par l'agent depuis sa dernière visite à cet état. Le signal de renforcement interne est ensuite distribué à tous les états récemment visités suivant la règle de mise à jour suivante (Sutton, 1988):

$$V(u) \leftarrow V(u) + \alpha[r + \gamma V(s') - V(s)]e(u), \qquad \forall u \in S \tag{2.13}$$

où :

$$e(u) \leftarrow \begin{cases} \gamma\lambda e(s) + 1 & \text{if } u = s \text{ (état actuel)} \\ \gamma\lambda e(u) & \text{si } u \text{ est déjà visité} \\ 0 & \text{sinon} \end{cases} \tag{2.14}$$

Bien que plus exigeant en terme de puissance de calcul, l'algorithme TD(λ) converge plus rapidement pour $\lambda > 0$, du moins en ce qui concerne l'estimation de la fonction d'évaluation (Dayan, 1992). Par contre, les algorithmes RL utilisant la méthode TD(λ) où les

traces d'éligibilité sont cumulatives (Éq. 2-14) manifestent de faibles performances lorsque λ tend vers 1 (Singh, 1994). Cela est dû au fait que l'éligibilité d'un état fréquemment visité peut dépasser sensiblement 100%. Dans de telles situations, Singh et Sutton ont introduit de nouvelles traces d'éligibilité appelées traces substitutives définies par (Singh, 1994) :

$$e(u) \leftarrow \begin{cases} 1 & \text{if } u = s \text{ (état actuel)} \\ \gamma\lambda e(u) & \text{si } u \text{ est déjà visité} \\ 0 & \text{sinon} \end{cases} \tag{2.15}$$

2.4 Principaux algorithmes RL

Dans ce qui suit, nous allons présenter et analyser les deux principaux algorithmes de l'apprentissage par renforcement.

2.4.1 Algorithme du Q-Learning

L'estimation de la fonction Q^* peut se faire d'une manière directe en utilisant la même idée que l'algorithme TD(0) :

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a' \in A(s')} Q(s',a') - Q(s,a)\right] \tag{2.16}$$

Sous l'hypothèse minimale que tous les états soient visités un nombre de fois suffisamment grand, et que le facteur d'apprentissage α soit positif et lentement décroissant, il a été démontré que la règle de mise à jour (Éq. 2-16) est asymptotiquement convergente (Watkins, 1992). L'équation (2.16) est à l'origine du très populaire algorithme du Q-Learning (QL) (Watkins, 1989 et Watkins, 1992) dont l'algorithme générique est présenté dans la figure 2-2. Les principales raisons de la popularité du QL sont la garantie de

convergence ainsi que la facilité d'extraction d'une stratégie de contrôle optimale une fois que la fonction d'évaluation optimale Q^* est correctement estimée. En plus, la convergence du QL est indépendante du choix d'actions pendant la phase d'apprentissage.

En réalité, la méthode QL est une implémentation adaptative (en-ligne) de l'algorithme *Value Iteration* (VI), très connu en programmation dynamique (Bellman, 1957 et Bertsekas, 1987), mais qui a l'avantage de ne pas nécessiter de modèle du système. Dans les méthodes VI, la résolution du problème d'optimisation consiste d'abord à estimer la fonction d'évaluation et en extraire ultérieurement une stratégie de contrôle.

Extrait de (Watkins, 1992)

Initialize Q arbitrarily, e.g., $Q(s,a)=0$, for all $s \in S$ and $a \in A(s)$.
Repeat (for each episode)
  Initialize s
  Repeat (for each step of episode)
    Choose a from s using a policy derived from Q
    Take action a, observe r, s'

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a' \in A(s')} Q(s',a') - Q(s,a)\right]$$

    $s \leftarrow s'$;
  until s is terminal
Output a deterministic policy π^*, such that

$$\pi^*(s) = \arg\max_{a \in A(s)} Q^*(s,a)$$

Figure 2-2 Algorithme Q-Learning.

2.4.2 Algorithme acteur-critique

L'estimation de la fonction d'évaluation V ne permet pas à elle seule de dériver directement une stratégie de contrôle optimale en l'absence d'un modèle complet de l'environnement. Cependant, elle peut permettre à l'agent d'améliorer sa stratégie de contrôle par exploration. Le principe est que l'agent choisit occasionnellement d'exécuter (explorer) une action différente de celle calculée à partir de la stratégie de contrôle. L'utilité de cette dernière dans un état s étant exactement égale à sa valeur $V(s)$, la variation de celle-ci, provoquée par le changement de trajectoire de l'agent à la suite de l'exploration, peut être

utilisée pour comparer les deux actions. Si la valeur de l'état augmente, l'action explorée est donc meilleure et sera substituée dans la stratégie de contrôle. Dans le cas contraire, l'agent renforce ses convictions dans sa stratégie de contrôle. En l'absence d'exploration, la stratégie de contrôle reste constante mais l'agent continue de mettre à jour la fonction d'évaluation. L'agent peut ainsi s'améliorer progressivement jusqu'à l'obtention d'une solution quasi optimale. Cette approche est utilisée dans la famille d'algorithmes connus sous le nom d'algorithmes *Acteur-Critique* (AC) dont le plus connu est sans doute l'algorithme AHC (Adaptive Heuristic Critic) de Barto *et al.* qui l'ont utilisé pour le problème du pendule inverse (Barto, 1983). Dans les méthodes AC, on trouve deux éléments adaptatifs en constante interaction. Le premier élément, appelé *Acteur*, implémente la stratégie de contrôle, mais doit aussi explorer de nouveaux choix d'actions afin de maximiser sa fonction d'évaluation qu'implémente le deuxième élément appelé *Critique*. Ce dernier doit se mettre constamment à jour pour tenir compte des éventuelles modifications de la stratégie de contrôle. L'efficacité de cet algorithme dépend considérablement du mécanisme d'exploration , ainsi que du mécanisme de mise à jour de la stratégie de contrôle.

Extrait de (Sutton, 1998, pp. 151-152)

Initialize the value function V arbitrarily, e.g., $V(s) = 0$, for all $s \in S$.

Initialize the policy π uniformly, i.e., $\pi(s,a) = \frac{1}{|\mathrm{A}(s)|}$, for all $s \in S, a \in \mathrm{A}(s)$

Repeat (for each episode)
  Initialize s
  Repeat (for each step of episode)
    Compute an action a from s with a probability $\pi = e^{p(s,a)} / \sum_{b \in A(s)} e^{p(s,b)}$
    Take action a, observe r, s'
    $\hat{r} \leftarrow r + \gamma V(s') - V(s)$
    $V(s) \leftarrow V(s) + \alpha \hat{r}$
    $p(s,a) \leftarrow p(s,a) + \beta \hat{r}(1 - \pi(s,a))$
    $s \leftarrow s'$
  until s is terminal

Output a deterministic policy π^* (if necessary), such that

$$\pi^*(s) = \arg\max_{a \in A(s)} p(s,a)$$

Figure 2-3 Algorithme AHC.

En réalité, la méthode AC est une implémentation adaptative (en-ligne) de l'algorithme *Policy Iteration* (PI), très connu en programmation dynamique (Bellman, 1957 et Bertsekas, 1987), mais qui ne nécessite pas de modèle du système. Dans les méthodes PI, la résolution du problème d'optimisation consiste à alterner l'estimation de la fonction d'évaluation et l'amélioration de la stratégie de contrôle.

2.4.3 Choix d'une architecture RL

S'il existe un algorithme RL qui semble faire l'unanimité, c'est bien l'algorithme QL. D'une part, sa convergence a été établie dès le départ (Watkins,1992) puisqu'elle découle directement de celle de l'algorithme TD(0). D'autre part, le choix de l'algorithme QL élimine la lourde tâche de concevoir un mécanisme d'amélioration de la stratégie de contrôle. Ces deux propriétés sont dues au fait qu'on estime directement la fonction d'évaluation optimale Q^* pour ensuite en extraire une stratégie de contrôle qui est forcément optimale. Cependant, ces deux avantages de l'algorithme QL perdent sensiblement du terrain quand la vitesse de convergence est un facteur important, ou lors de l'implémentation pratique de cet algorithme. En effet, la convergence de l'algorithme QL suppose une évaluation exhaustive de toutes les actions possibles dans tous les états. Cela dit, on peut surmonter cette faiblesse par un choix judicieux du mécanisme d'exploration de l'espace des stratégies de contrôle, ce qui permet à l'agent de découvrir les meilleures actions dans les premières phases d'apprentissage. En effet, d'après l'équation (2.10) il suffit d'estimer une fonction d'évaluation Q qui ordonne l'espace d'états/actions de la même manière que Q^* pour arriver à une stratégie de contrôle optimale. Hélas, l'utilisation d'un mécanisme d'exploration dans l'algorithme QL le prive de tirer profit du puissant algorithme TD(λ) qui permet d'accélérer considérablement la convergence. En effet, il est implicitement requis dans l'équation (2.16) que l'action exécutée soit celle calculée à partir de l'équation (2.10) pour justifier l'utilisation des traces d'éligibilité (Éq. 2-14 et Éq. 2-15). Bien que plusieurs travaux aient été effectués afin de concilier exploration et traces d'éligibilité, les solutions proposées ne sont que des compromis intermédiaires. Par exemple, l'algorithme SARSA(λ) (Rummery, 1994) renonce à l'utilisation de l'opérateur

"max" dans l'équation (2.16), perdant ainsi le principal atout de l'algorithme QL, en l'occurrence la garantie de sa convergence. À noter cependant que, tout récemment, Singh *et al.* (Singh, 2000) ont démontré qu'une forme de convergence peut être garantie si l'algorithme SARSA(0) est combiné avec certaines méthodes d'exploration. Tandis que Watkins propose tout simplement de remettre à zéro toutes les traces d'éligibilité après chaque exploration (choix d'une action différente de celle calculée à partir de l'équation (2.10)). Une autre solution aux résultats empiriques supérieurs mais à l'implémentation plus complexe a été proposée dans (Peng, 1994, 1996). En particulier, la méthode de Peng ne requiert par la remise à zéro des traces d'éligibilité. Aussi, Wiering et Schmidhuber (Wiering, 1998) proposent de ne mettre à jour la valeur d'une paire (état, action) que lorsque cette dernière est à nouveau utilisée.

Cependant, même si la convergence de l'algorithme QL est garantie, il reste qu'elle peut être très lente et qu'elle l'est davantage quand le nombre d'actions possibles dans chaque état est relativement important ou pire encore si l'espace d'actions est continu (comme c'est souvent le cas dans les applications de contrôle). Cela est dû au fait que dans chaque état, on a besoin d'apprendre et de stocker la valeur de l'utilité de chaque action et de calculer ensuite le maximum de ces valeurs stockées. Or, si le nombre d'actions possibles est très grand, l'espace mémoire nécessaire pour le stockage de la fonction Q devient très coûteux sinon irréalisable physiquement. En plus, le mécanisme de sélection d'actions (Éq. 2-10) devient lui aussi coûteux du point de vue de la complexité des calculs. En effet, dans de tels cas, on n'a souvent pas d'autre choix que de recourir à un mécanisme de généralisation sur l'espace d'actions en utilisant un approxiamteur universel comme les réseaux de neurones, les systèmes flous, les réseaux RBF (Radial Basis Functions), etc., ayant comme entrées l'état s de l'environnement et l'action a de l'agent et comme sortie la fonction d'évaluation d'actions $Q(s, a)$. Pour de tels approximateurs, la procédure d'apprentissage n'est généralement pas une tâche difficile à réaliser. Par contre, le calcul de leur points optimums peut être considérablement difficile vu que ces optimums sont généralement stockés dans la partie non linéaire de l'approximateur (par exemple, les centres des fonctions d'appartenance des variables d'entrée pour un système neuro-flou, centres des fonctions radiales dans les réseaux RBF, etc). Bien sûr, on peut toujours résoudre ce problème en utilisant une méthode de recherche dans l'espace d'actions dans le but de trouver l'action

dont l'utilité est la plus grande (Santamaria, 1998). Par exemple, Baird a proposé d'utiliser une méthode d'optimisation se basant sur un gradient local ascendant (Baird, 1993). Par contre, une telle solution ne peut que ralentir davantage la convergence et alourdir le processus d'apprentissage. Par ailleurs, il a été établi que lorsque la méthode QL utilise un approximateur universel, elle peut devenir divergente (Bertsekas, 1996). L'exemple le plus significatif dans ce sens est celui de Baird (1995) où un PDM composé de sept états et représenté par un approximateur linéaire dont les vecteurs caractéristiques sont linéairement indépendants et pour lequel une solution exacte existe. Baird a démontré que pour certaines valeurs initiales, la fonction d'évaluation apprise est divergente. Cependant, des recherches récentes ont démontré que ce problème peut être résolu en modifiant la méthode de mise à jour de la fonction d'évaluation Q. Par exemple, Precup *et al.* (Precup, 2001) proposent, dans le cas d'une tâche épisodique, de pondérer les mises à jour de la fonction Q pour chaque épisode par la probabilité relative que l'épisode en question utilise la stratégie cible (glouton) ou une autre stratégie comportementale (dérivée ou non de la stratégie glouton). Cette proposition est basée sur le principe d'évaluer une stratégie de contrôle tout en en suivant une autre (Sutton, 1998, pages 124-126). Finalement, mentionnons que, tout récemment, Millan *et al.* (Millan, 2002) ont proposé une extension de l'algorithme QL qui permet de l'appliquer au cas où l'espace d'actions est continu. Cette extension est basée sur l'utilisation de méthodes d'apprentissage non supervisé pour la partition de l'espace d'états et sur l'utilisation de la fonction Q pour calculer le barycentre des actions discrètes associées à chaque région de l'espace d'états. L'action ainsi calculée est continue. Finalement, on peut aussi reprocher aux algorithmes de type QL le fait que la stratégie de contrôle soit très sensible aux variations de la fonction d'évaluation lors de sa mise à jour pendant la phase d'apprentissage, d'où un manque de robustesse quand l'apprentissage se fait en ligne. L'algorithme AC est plus robuste puisque la stratégie de contrôle évolue de façon incrémentale sans être directement influencé par les variations de la fonction d'évaluation.

Les problèmes d'explosion dimensionnelle sont très bien connus en programmation dynamique et plus particulièrement dans les méthodes VI dont fait partie l'algorithme QL. Cependant, ils peuvent être considérablement réduits en utilisant une représentation explicite de la stratégie de contrôle comme c'est le cas dans les méthode PI dont fait partie

l'algorithme AC. Ces méthodes, dites directes, de recherche d'une stratégie de contrôle opèrent en modifiant les paramètres de l'acteur non pas par exploration mais plutôt en cherchant directement à maximiser la fonction d'évaluation. Cette opération d'optimisation se fait généralement en calculant, ou en estimant, le gradient de la fonction d'évaluation par rapport aux paramètres modifiables de l'acteur. Ces paramètres sont par la suite modifiés dans le sens ascendant de ce gradient (Williams, 1992 ; Williams, 1993, Baird, 1999). Dans son algorithme REINFORCE, Williams (1992) fut le premier à proposer une méthode d'estimation du gradient de la fonction d'évaluation par rapport aux paramètres de l'acteur. Plus récemment, Konda et Tsitskilis (Konda, 2001) proposent d'utiliser un critique dont la fonction d'évaluation est du même type que celui utilisé dans le QL. Ils calculent le gradient de cette fonction d'évaluation par rapport au paramètres de l'acteur en combinant une méthode d'optimisation des PDM proposée récemment par Marbach et Tsitskilis (Marbach, 2001) avec la méthode TD(λ). Cette dernière permet d'éviter le recours aux simulations du PDM dans le but d'estimer le retour espéré ou le surplus de coût sur un horizon donné. Dans le souci de réduire le problème d'explosion dimensionnelle qu'invoque l'utilisation d'une fonction d'évaluation de type (état, action), Konda et Tsitskilis proposent d'estimer uniquement la projection de cette fonction d'évaluation sur un espace de dimension inférieure définie par un ensemble de fonctions de base déterminé par la paramétrisation de l'acteur. Le gros avantage de cette approche est qu'elle garantit la convergence vers une solution optimale et de ce fait se compare avantageusement avec la méthode QL surtout qu'elle promet d'être, du moins théoriquement, plus rapide. Notons que cette méthode est également utilisée dans l'architecture GARIC bien que de façon approximative. Néanmoins, on peut lui reprocher de souffrir du même problème d'explosion dimensionnelle que la méthode AHC. Sutton *et al.* (Sutton, 2001) ont aussi proposé une méthode semblable quoiqu'elle soit plus générale puisqu'elle s'applique aussi bien à la formulation escomptée du problème RL qu'à la formulation moyennée traitée par Konda et Tsitskilis. Fait intéressent, il a été démontré qu'en utilisant des approximateurs universels, aussi bien pour la solution proposée par Konda *et al.* que la solution proposée par Sutton *et al.* que les processus d'estimation de la fonction d'évaluation et la mise à jour de la stratégie de contrôle sont convergents. Finalement, on peut aussi mentionner la méthode PEGASUS proposée par Andrew Ng *et al.* (Ng, 2001) et dont l'objectif est de

chercher directement dans l'espace des stratégies de contrôle. Le but est de trouver ou de construire une stratégie de contrôle optimale, *i.e.* correspondant à la fonction d'évaluation maximale. La méthode de recherche utilisée par PEGASUS est basée sur l'évaluation simultanée de plusieurs stratégies de contrôle à l'aide de scénarios générés en utilisant un modèle POMDP (Partially Observable Markov Decision Process) déterministe construit à partir d'un modèle (PO)MDP général (stochastique) de l'environnement. Par conséquent, cette méthode requiert un modèle de l'environnement et de ce fait ne peut être utilisée dans la classe de problèmes traités dans le cadre de ce cette thèse.

Puisque dans cette thèse, on s'intéresse surtout aux applications de contrôle en ligne du RL où l'espace d'actions est typiquement continu, et pour toutes les raisons déjà mentionnées, nous arrêtons notre choix sur l'investigation des architectures AC.

2.4.4 Équilibre exploration/exploitation

La difficulté et le succès des méthodes RL reposent sur le compromis entre l'exploration et l'exploitation. L'agent exploite ce qu'il connaît afin d'obtenir la plus grande récompense, mais pour découvrir les actions conduisant au meilleur rendement, il a besoin d'explorer plusieurs possibilités. Ce dilemme fait que si on utilise exclusivement l'un des deux aspects de l'alternative, on ne pourra pas atteindre le but fixé.

L'apprentissage d'une stratégie de contrôle optimale implique donc un compromis entre deux objectifs apparemment contradictoires : l'exploitation de la stratégie de contrôle déjà apprise afin de maximiser la fonction d'évaluation et l'exploration robuste de l'espace d'actions dans le but de découvrir des actions encore plus payantes. Les méthodes de type glouton (greedy en anglais) représentent une extrémité possible du compromis en se basant uniquement sur l'exploitation et en souffrant, par conséquence, de la non-globalité de l'optimum trouvé. Cependant, elles convergent rapidement. Les méthodes de type recherche aléatoire représentent l'autre extrémité en procédant à une exploration permanente de l'espace d'actions et en souffrant, par conséquent, de leur lenteur excessive. Cependant, elles convergent souvent vers un optimum global. Par conséquent, tout compromis entre la vitesse de convergence et la qualité de l'optimum trouvé passe

forcément par une bonne stratégie d'équilibre exploration/ exploitation. Intuitivement, on peut imaginer un scénario où l'exploration est favorisée pendant le début de la phase d'apprentissage tout en laissant progressivement la place à l'exploitation. Si l'environnement est stationnaire, on peut même se contenter d'exploiter la stratégie de contrôle apprise durant la phase d'apprentissage. Cependant, en pratique, il est très pertinent de maintenir un certain niveau d'exploration soit parce que l'environnement réel est non stationnaire, soit parce que la stratégie de contrôle apprise n'est optimale que dans les états visités pendant la phase d'apprentissage. Par conséquent, La réussite de l'équilibre exploration/exploitation est un élément déterminant dans le succès et la qualité des performances des algorithmes RL.

Il existe plusieurs heuristiques pour assurer l'équilibre exploration/exploitation. Par exemple, on peut utiliser une stratégie de type glouton mais initialiser la fonction d'évaluation de façon optimiste de sorte qu'aucune action ne soit éliminée avant qu'elle ne soit testée et qu'elle ne s'avère effectivement moins payante. On trouve une telle approche dans plusieurs algorithmes RL (Sutton, 1990, Thurn, 1992, Moore, 1993). Cependant, cette heuristique n'est efficace que dans le début de la phase d'apprentissage et des actions optimales malchanceuses risquent d'être écartées dès les premières phases de l'apprentissage. Une autre heuristique consiste à suivre la stratégie glouton en choisissant les actions maximisant la fonction d'évaluation, mais à l'occasion, avec une probabilité ε, choisir aléatoirement une autre action (Sutton, 1998). En utilisant un ε variable, on peut contrôler le niveau d'exploration au fur et à mesure que l'apprentissage progresse. Une telle stratégie de contrôle est appelée quasi-glouton. On peut aussi combiner ces deux heuristiques et utiliser un ε adaptatif par rapport à la valeur de l'état (Berenji, 1992). Soulignons, finalement, que lorsque la stratégie de contrôle est stochastique comme dans le cas de l'algorithme AHC, l'équilibre exploration/exploitation est assuré de façon presque systématique puisque chaque action, optimale ou non, a toujours une chance d'être sélectionnée.

2.5 Implémentation d'un agent AC

Tel que formulé précédemment, le problème RL est réduit à un problème d'apprentissage supervisé dans lequel le critique joue le rôle de superviseur. La formulation du problème RL à l'aide d'un PDM fini nous a certes permis d'exploiter les méthodes bien établies de la programmation dynamique, en plus d'avoir un cadre mathématique bien formalisé pour étudier les problèmes RL. Malheureusement, cela implique également de fortes contraintes : états et actions de type discret. Or, dans les applications de contrôle réalistes, les états et les actions sont soit de type continu, soit de nombre tellement grand qu'il est pratiquement impossible de les énumérer individuellement (*e. g.* dans un *look-up table*).

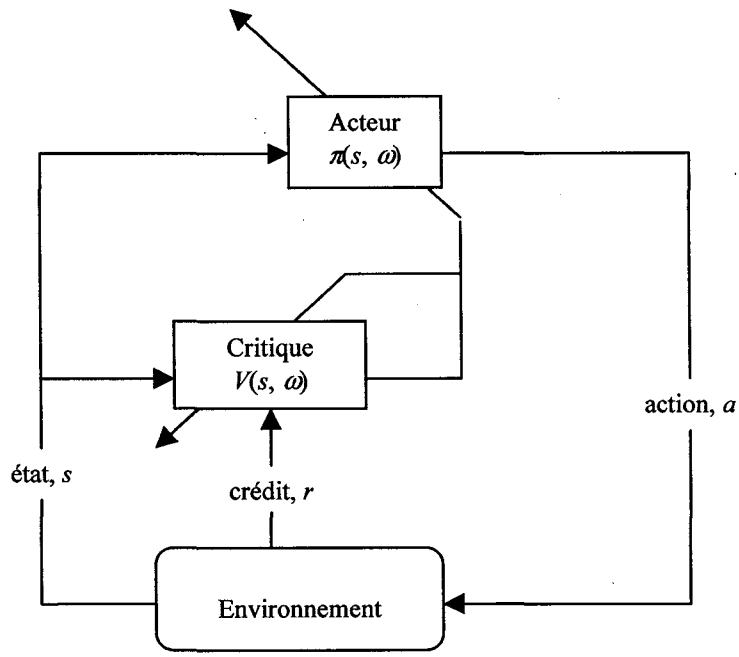


Figure 2-4 L'architecture acteur critique.

Dans de telles situations, on doit opter pour l'utilisation d'une représentation paramétrique compacte de la stratégie de contrôle et de la fonction d'évaluation de l'agent. Dans ce cas, plusieurs choix de représentation s'offrent à nous parmi lesquelles on peut citer :

- Les régressions linéaires;
- Les réseaux RBF (Radial Basis Functions);

- Les réseaux de neurones artificiels (RNA);
- Les systèmes d'inférence flous (SIF);

Dans tous ce qui suit, les agents AC (Fig. 2-4) que nous étudierons seront représentés par des SIF de type Takagi-Sugeno (TS) (Takagi, 1985) d'ordre zéro. Ce choix est dicté par les critères suivants :

- L'agent doit être en mesure d'intégrer aisément toutes les connaissances et heuristiques disponibles a priori.
- Les solutions apprises par l'agent doivent être lisibles par des experts humains, que ce soit pour des fins d'apprentissage, ou encore pour des fins d'analyse et de validation. Les SIF offrent indéniablement une solution naturelle pour satisfaire ces deux critères.
- Les paramètres ajustables dans les SIF de type TS d'ordre zéro sont faciles à interpréter et par conséquent faciles à initialiser et à ajuster.
- Les SIF de type TS d'ordre zéro sont des approximateurs universels capables de simuler n'importe quel type de fonctions mathématiques pourvu que le nombre de règles floues soit suffisamment grand (Kosko, 1992). En plus, leurs excellentes capacités de généralisation leur permettent généralement d'atteindre un très haut niveau de performance même avec un nombre restreint de règles floues.
- Les SIF de type TS d'ordre zéro sont entraînables par plusieurs types d'algorithmes d'apprentissage supervisé tels que :
  - Les algorithmes du gradient descendant et de rétro-propagation (Horikawa, 1992);
  - L'algorithme des moindres carrées récursifs (RLS) (Takagi, 1985; Jang, 1991);
  - Les algorithmes génétiques (Karr, 1991; Thrift, 1991),

2.5.1 Systèmes d'inférence flous

La logique floue, introduite dans le milieu des années 60 par Zadeh (Zadeh, 1965) a pour principale motivation d'étendre explicitement le concept de la logique multi-valuée introduite dans les années 20 par Jan Lukasiewicz à la notion d'appartenance dans la théorie classique des ensembles. Contrairement à cette dernière, dans la logique floue, l'appartenance est graduelle si bien qu'un élément peut appartenir à la fois à un ensemble et à son complément; avec des degrés d'appartenance différents. Zadeh a surtout eu le mérite de se servir de la logique floue pour introduire un formalisme linguistique dans lequel les sous-ensembles flous représentent des variables linguistiques semblables à celles utilisées par les humains pour décrire des variables ou des situations du monde réel (Zadeh, 1975). Les travaux ultérieurs ont permis de développer ces concepts et d'amener la logique floue à maturité ainsi qu'a élargir ses champs d'application. En particulier, les travaux de Mamdani (Mamdani, 1977a) sur le raisonnement approximatif par les SIF et son application au contrôle (Mamdani, 1977b) ont considérablement contribué à l'émergence et au succès des systèmes de contrôle flou. Takagi et Sugeno ont ensuite proposé un SIF très semblable à celui de Mamdani mais mieux adapté à l'approximation de fonctions et à la modélisation des systèmes non linéaires (Takagi, 1985).

2.5.1.1 Ensembles flous

Dans cette partie, nous nous contenterons de présenter quelques concepts de base de la théorie des sous-ensembles flous. Pour alléger le texte, on parlera dorénavant d'ensembles flous au lieu de sous-ensemble flous.

La théorie des ensembles flous est une extension, plus souple et plus flexible, de la théorie des ensembles classiques. Soit X un ensemble de référence, il est possible de classer des éléments de X par la définition d'ensembles classiques décrits par des fonctions d'appartenance binaires :

$$\mu_E(x) = \begin{cases} 1 & \text{si } x \in E \subset \mathrm{X} \\ 0 & \text{sinon} \end{cases}$$

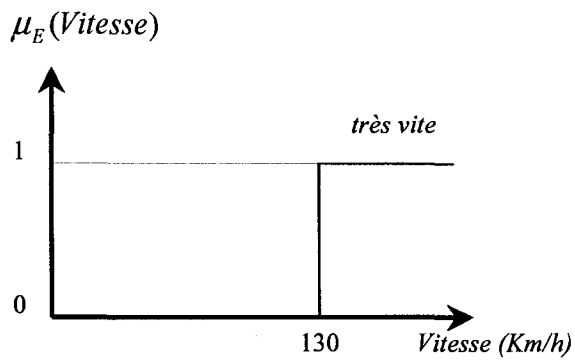


Figure 2-5 Notion d'excès de vitesse définie par un ensemble classique.

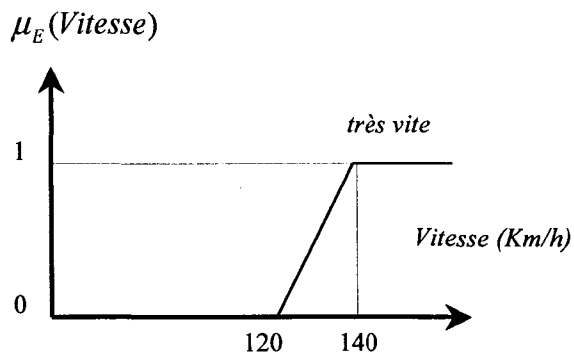


Figure 2-6 Notion d'excès de vitesse définie par un ensemble flou.

À titre d'exemple, prenons le cas de la perception d'une vitesse et plus spécifiquement la notion d'excès de vitesse. La description classique de l'ensemble E associé à cette notion sera effectuée par la définition d'une vitesse maximale, par exemple 130 Km/h, au-delà de laquelle un automobiliste risque d'écoper d'une contravention pour excès de vitesse (Fig. 2-5). La même notion sera plus fidèlement représentée si la méthode de représentation prend en compte l'imprécision inhérente à ce genre de catégories ainsi que la subjectivité d'une telle notion. En effet, le danger représenté par un automobiliste roulant à 128 Km/h n'est-il pas sensiblement le même que s'il roulait à 130 Km/h (imprécision)? N'est-il pas plus dangereux de rouler à 60 Km/h dans une zone urbaine que de rouler à 130 Km/h sur une autoroute (subjectivité)? Les ensembles flous satisfont parfaitement ces deux contraintes en graduant l'appartenance et en la liant au contexte. Ils sont caractérisés par des fonctions d'appartenance continues :

$$\mu_E(x) = \begin{cases} \in [0, 1] & \text{si } x \in E \subset \mathrm{X} \\ 0 & \text{sinon} \end{cases}$$

Comme le montre cette équation, l'appartenance à un ensemble flou est graduée de l'exclusion totale à l'appartenance totale (Fig. 2-6). Dans le formalisme linguistique défini par Zadeh (Zadeh, 1975), les propriétés définies par les ensembles flous se nomment des *termes linguistiques* ou *étiquettes floues* (*e.g. lent, vite, très vite*). Chaque étiquette floue

représente un ensemble de valeurs numériques sans pour autant en avoir l'exclusivité. Les fonctions d'appartenance des termes flous se chevauchent (Fig. 2-7).

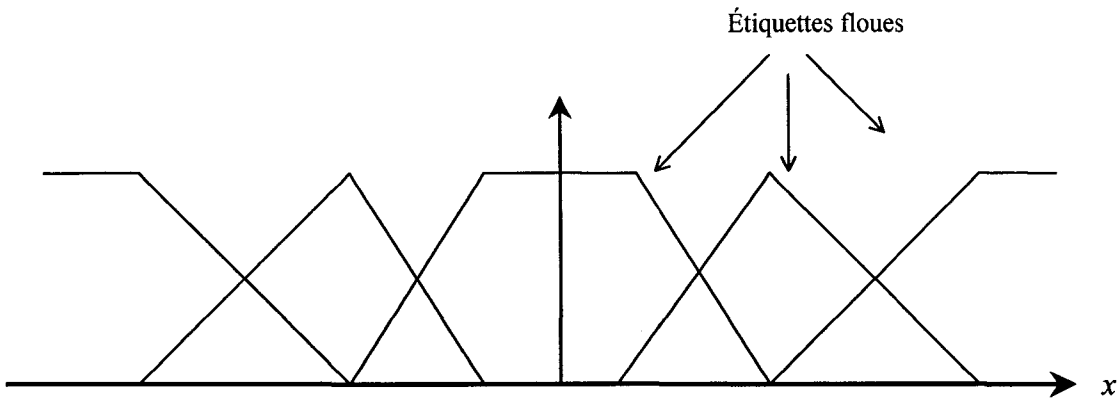


Figure 2-7 Partition floue du domaine de définition d'une variable linguistique.

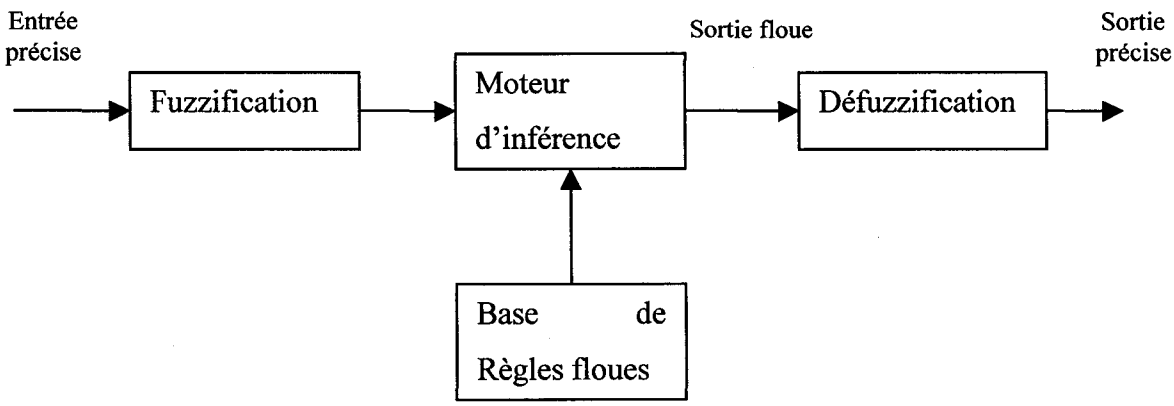


Figure 2-8 Structure de base d'un système d'inférence flou.

2.5.1.2 Structure de base d'un SIF

L'architecture typique d'un SIF (Fig. 2-8) comprend quatre composants principaux :

1. Un étage de fuzzification dont le but est de transformer les entrées précises (ou floues) en étiquettes floues appropriées. Par exemple *la vitesse est de 120 Km/h* peut devenir *la vitesse est trop élevée* avec un degré d'appartenance de 20% et *la vitesse est élevée* avec un degré d'appartenance de 80%.

2. Une base de règles floues caractérisée par une collection de règles floues du type « **si** *prémisse* **alors** *conclusion* » où les prémisses (et parfois les conclusions) constituent des propositions floues exprimées par « x est X » ou par une conjonction ou une disjonction de prédicats de cette forme. Un SIF est alors une collection de relations entre propositions floues ayant comme particularité la déduction de conclusions même quand les prédicats en prémisse ne sont qu'en partie satisfaits.

3. Un moteur d'inférence constituant le noyau de la structure et dont le rôle est d'assurer le raisonnement approximatif du SIF. D'une part, il doit définir tous les opérateurs (composition, conjonction, implication) impliqués dans le calcul des propositions floues afin de déduire des conclusions locales à partir de chaque règle floue. La règle d'inférence utilisée dans ce cas est une généralisation du modus ponens qui prend en considération l'aspect graduel de la notion d'appartenance inhérent aux ensembles flous :

Règle floue R^r : **Si** x est X^r **alors** y est Y^r

μ_{X^r} μ_{Y^r}

fait observé : x est X'^r

$\mu_{X'^r}$

Conclusion déduite y est Y'^r

$\mu_{Y'^r}$

La conclusion de chaque règle R^r est obtenue en déterminant la fonction d'appartenance $\mu_{Y'^r}$ de y. Ce calcul est réalisé à partir de la règle R^r et de $\mu_{X'^r}$. D'autre part, le moteur d'inférence doit aussi définir un mécanisme d'agrégation des conclusions intermédiaires de chaque règle. Cette agrégation peut être effectuée de plusieurs manières dont la plus utilisée

est la méthode du centroïde qui est particulièrement intéressante si les conclusions inférées sont de nature précise.

Plusieurs techniques d'inférence existent dans la littérature (Wang, 1994) mais on ne s'y attardera pas davantage étant donné qu'on utilise le SIF essentiellement comme un approximateur universel. Ceci implique que les entrées sont toujours précises ($\mu_{X'r}$ est un singleton, ce qui implique qu'on n'effectue plus d'opérations de composition), que le moteur d'inférence produit toujours une sortie précise ($\mu_{Y'r}$ est un singleton) dont le poids est défini par le degré de vérité (force de déclenchement) de la prémisse et que les opérateurs de conjonction, d'implication et d'agrégation doivent être différentiables.

4. Un étage de défuzzification dont le rôle est de produire une décision non floue (précise) à partir de la sortie floue du moteur d'inférence en sélectionnant un élément précis appartenant à l'ensemble des valeurs numériques qu'elle couvre. La sélection d'un élément représentatif de la sortie floue peut se faire par plusieurs méthodes dont le plus répandue est la méthode du centre de gravité. Cela dit, la sortie du moteur d'inférence étant précise dans un SIF de type TS, cet étage n'est plus nécessaire.

2.5.1.3 SIF de type Takagi-Sugeno d'ordre zéro

Les SIF de type TS sont des systèmes à base de règles floues de type « **Si** *prémisse* **alors** *conclusion* » possédant la particularité que les conclusions sont des fonctions numériques des entrées. Dans le cas des systèmes TS d'ordre zéro, cette fonction est une constante. Un SIF de type TS d'ordre zéro à n entrées et m sorties est alors décrit par une base de R règles floues de la forme :

$$R^r : \text{Si } x_1 \text{ est } X_1^r \ \ldots \text{ et } x_n \text{ est } X_n^r \text{ alors } y_1 = y_1^r \ldots \text{ et } y_m = y_m^r,\ r = 1 \cdots R \qquad (2.17)$$

où X_i^r est l'une des L_i étiquettes floues X_i^l, $l = 1 \cdots L_i$, décrivant l'entrée x_i, et y_j^r est la valeur locale de la sortie y_j lorsque seule la règle R^r est active.

Le calcul de la sortie discrète du SIF se fait alors en deux étapes :

1. **Fuzzification :** l'étape de fuzzification consiste à calculer le degré d'appartenance de l'entrée x_i à chacune des L_i étiquettes floues X_i^l la décrivant.

2. **Inférence floue :** elle se compose des deux étapes suivantes :

   a) Calcul de la valeur du degré de vérité de la prémisse avec l'opérateur de conjonction (ET) implémenté par le produit. Soit :

$$\mu^r(x_1,...,x_n) = \prod_{i=1}^{n} \mu_{X_i^r}(x_i), \qquad r = 1 \cdots R \tag{2.18}$$

   b) Calcul des conclusions finales par la méthode du centre de gravité :

$$y_j = \frac{\sum_{r=1}^{R} \mu^r(x_1,\ldots,x_n) y_j^r}{\sum_{r=1}^{R} \mu^r(x_1,\ldots,x_n)}, \qquad j = 1 \cdots m \tag{2.19}$$

2.5.1.4 Partition de l'espace d'états

La partition de l'espace d'états est définie par le type de partition des différentes variables d'état (entrées). Deux principaux types de partition s'offrent alors au concepteur :

- **Partition globale** : dans cette structure, chaque variable d'entrée x_i est décrite par un nombre limité L_i d'étiquette floues X_i^l intervenant alternativement, mais jamais simultanément dans chacune des règles floues. La base de règles est alors construite à partir de toutes les combinaisons possibles des étiquettes floues décrivant chacune des variables d'entrée. Le nombre total de règles est dans ce cas défini par :

$$R = \prod_{i=1}^{n} L_i \tag{2.20}$$

Comme le montre cette équation, le problème de cette structure réside dans l'accroissement exponentiel de la taille de la base de règles en fonction du nombre de variables d'entrée ainsi que le nombre d'étiquettes floues pour chacune de ces variables. En revanche, l'interprétation et, par le fait même, l'initialisation de la base de règles sont aisées. En outre, chaque variable d'entrée peut être partitionée séparément. Cette structure est de loin la plus utilisée (Mamdani, 1977, Jang, 1992).

- **Partition locale** : dans cette structure (Wang, 1994), chaque règle possède son propre jeu d'étiquettes floues. Il en résulte que chaque variable d'entrée x_i possède autant d'étiquettes floues X_i^l que le système a de règles :

$$R = L_i, \ \forall i \tag{2.21}$$

Comme le montre cette équation, l'avantage de ce type de partition est que la base de règles ne croît pas de façon exponentielle avec le nombre d'étiquettes floues associées aux variables d'entrée. Cependant, le prix à payer se situe au niveau de la lisibilité du système ainsi qu'au niveau de la couverture complète de l'espace d'états, d'où des problèmes de généralisation et même d'instabilité inhérents à cette structure.

Dans toute cette thèse, nous utiliserons exclusivement la partition globale. Ce choix implique que le nombre total de règles est toujours donné par l'équation (2.20) et que chaque variable participe obligatoirement dans chacune des règles.

Le choix des autres caractéristiques structurelles du système d'inférence floue, notamment le type de fonction d'appartenance ainsi que le nombre de termes linguistiques pour chaque variable d'entrée, sera laissé à la discrétion du concepteur. Nous nous intéresserons alors exclusivement à l'apprentissage des caractéristiques paramétriques du SIF, à savoir :

- les conclusions locales y_j^r,
- les paramètres des fonctions d'appartenance (centres, largeur,s etc.).

2.5.2 Apprentissage supervisé des paramètres d'un SIF

En représentant un SIF par un réseau de neurones artificiels (RNA), il devient possible de puiser dans l'arsenal des méthodes d'apprentissage supervisé présentes dans le domaine neuronal. On parle alors de systèmes hybrides communément appelés système neuro-flous. La méthode de rétro-propagation du gradient représente la méthode neuronale la plus utilisée dans le cadre d'un apprentissage supervisé. Dans ce cas, le but de l'apprentissage est de minimiser l'écart entre la sortie générée par le SIF et la sortie préconisée par le superviseur. Le succès des systèmes neuro-flous est le fruit de l'intégration des caractéristiques sémantiques des SIF et les capacités d'apprentissage des RNA.

2.5.2.1 Algorithme du gradient descendant

Il s'agit d'un algorithme d'apprentissage supervisé dont le but est de réduire l'écart entre une sortie désirée (celle du superviseur) et la sortie effective du système d'apprentissage, en minimisant une fonction de coût E^{ω} où ω est le vecteur des paramètres ajustables. La fonction de coût est généralement une fonction quadratique de l'erreur définie par :

$$E^{\omega} = \sum_{t=1}^{N} E_t^{\omega} = \sum_{t=1}^{N} \frac{1}{2}(y^*(x_t) - y(x_t, \omega))^2 \tag{2.22}$$

où x_t représente le vecteur d'entrée à l'instant t, $y(x_t, \omega)$ la sortie calculée par l'agent, $y^*(x_t)$ est la sortie désirée fournie par le superviseur, et E_t^{ω} est l'erreur instantanée d'estimation.

L'apprentissage consiste à trouver le vecteur de paramètres optimal ω^* correspondant à un minimum global de la fonction E^{ω}. Dans ce but, l'agent modifie son vecteur de paramètres dans la direction opposée du gradient de la fonction E^{ω} (Widrow, 1985):

$$\omega_{n+1} = \omega_n - \eta \frac{\partial E^{\omega}}{\partial \omega_n} = \omega_n + \eta \sum_{t=1}^{N} (y^*(x_t) - y(x_t, \omega_n)) \frac{\partial y(x_t, \omega_n)}{\partial \omega_n} \tag{2.23}$$

où ω_n est le vecteur d'apprentissage à l'itération *n*, et $\eta > 0$ est le taux d'apprentissage.

Si le taux d'apprentissage est choisi suffisamment petit, la règle d'apprentissage définie par l'équation (2.23) est garantie de converger vers un minimum (pas forcément global) de la fonction E^{ω} (Widrow, 1985). Cependant, l'équation (2.23) présente l'inconvénient d'exiger que tous les exemples fournis par le superviseur soient parcourus avant de mettre à jour le vecteur de paramètres, et elle ne peut donc être utilisée pour un apprentissage en ligne. Pour combler cette lacune, on peut se contenter de minimiser la valeur instantanée E_t^{ω} de l'erreur en mettant à jour le vecteur de paramètres après chaque exemple d'apprentissage :

$$\omega_{t+1} = \omega_t - \eta_t \frac{\partial E_t^{\omega}}{\partial \omega_t} = \omega_t + \eta_t (y^*(x_t) - y(x_t, \omega_t)) \frac{\partial y(x_t, \omega_t)}{\partial \omega_t} \tag{2.24}$$

Le prix à payer pour cette simplification est une convergence beaucoup plus incertaine et un taux d'apprentissage difficile à fixer. Il est généralement d'usage d'opter pour un taux d'apprentissage variable (lentement décroissant) pour amortir les oscillations aux alentours du minimum trouvé.

La principale difficulté avec l'algorithme du gradient descendant est la nécessité de calculer toutes ces dérivées partielles qu'implique l'équation (2.24). Dans le cas où le système d'apprentissage peut être représenté par un RNA, on peut y arriver de façon élégante en utilisant l'algorithme de rétro-propagation du gradient introduit au milieu des années 80 (Rumelhart, 85). Or, justement un SIF de type TS peut être représenté par un RNA. Cette représentation connexionniste des SIF, que nous détailleront dans les deux sections qui suivent, a été initialement proposée par Horikawa *et al.* (Horikawa, 1992) et reprise par la suite par d'autres pionniers des systèmes *neuro-flous* (Lin, 1994, Wang, 1994).

2.5.2.2 RNA et algorithme de rétro-propagation

Un RNA typique est formé de plusieurs couches de neurones en cascade, les neurones d'une même couche ne pouvant pas interagir entre eux. Chaque neurone implémente une fonction de base qui peut soit lui être unique, soit être commune à toute la couche dont il fait partie, voire même à tout le RNA. Typiquement, un neurone est un système à plusieurs entrées et une seule sortie (voir Fig. 2-9).

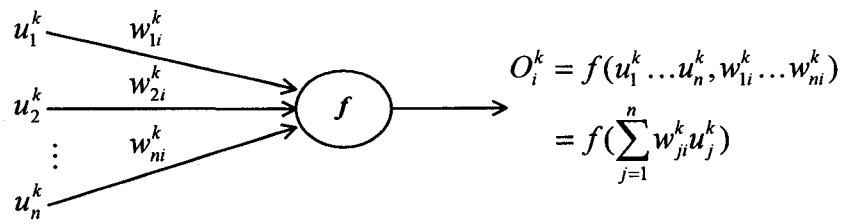


Figure 2-9 Structure de base d'un neurone dans un RNA.

Les entrées sont généralement liées au neurone par des poids synaptiques w_{ji}^k et le neurone calcule sa sortie en appliquant une fonction générique (par exemple la fonction sigmoïde) à la somme pondérée de toutes les entrées.

Les paramètres d'un RNA sont en général ajustés par l'algorithme de rétro-propagation dont le principe est d'exploiter la structure connexionniste du RNA afin de simplifier et d'uniformiser le calcul du gradient de l'erreur.

L'algorithme de rétro-propagation permet de calculer le gradient local de la sortie d'un RNA dans n'importe quel neurone d'une couche donnée à partir de la valeur du gradient dans les neurones de la couche suivante aux quels la sortie de ce neurone est connectée. De la même manière que l'entrée est propagée dans le sens direct (*Forward*) pour calculer la sortie du RNA, le gradient de la sortie est rétro-propagé dans le sens inverse (*Backward*), c'est à dire de la couche de sortie vers la couche d'entrée. Considérons le neurone i de la couche k représenté par la figure 2-9. Pour calculer le gradient de l'erreur à la sortie de ce neurone, l'algorithme de rétro-propagation utilise la règle de chaîne suivante :

$$\frac{\partial y}{\partial w_{ji}^k} = \frac{\partial y}{\partial O_i^k}\frac{\partial O_i^k}{\partial w_{ji}^k}, \quad \text{et} \quad \frac{\partial y}{\partial O_i^k} = \sum_l w_{il}^{k+1}\frac{\partial y}{\partial O_l^{k+1}} \tag{2.25}$$

Comme le montre cette équation, il suffit de rétro-propager le gradient de la couche de sortie vers les couches antérieures pour estimer le gradient local dans tous les neurones du RNA.

Cependant, on peut aussi considérer les poids synaptiques w_{ji}^k comme étant des paramètres internes du neurone. Dans ce cas, le RNA peut performer n'importe quelle tâche de calcul décomposable, en représentant par exemple chaque étape de calcul par une couche spécialisée dont les neurones possèdent la structure adéquate et implémentent chacun une fonction bien spécifique plutôt qu'une fonction générique commune à tous les neurones du RNA. L'avantage d'une telle approche réside dans l'interprétation beaucoup plus facile des paramètres du RNA et une structure de ce dernier déduite directement de la décomposition du problème à résoudre. L'inconvénient de cette approche réside surtout dans le fait que le concepteur doit réécrire l'algorithme de rétro-propagation en tenant compte des spécificités des neurones de chaque couche. En effet, la fonction d'activation est différente d'une couche à l'autre contrairement aux RNA conventionnels.

2.5.2.3 Représentation d'un SIF de type TS par un RNA

Dans le cas des SIF, le vecteur de paramètres est formé des conclusions locales y_j^r et/ou des paramètres des fonctions d'appartenance. Pour des raisons de clarté, nous supposerons que les fonctions d'appartenance sont des fonctions gaussiennes définies par :

$$\mu_{X_i^j}(x_i) = \exp(-\frac{1}{2}(\frac{x_i - c_i^j}{\sigma_i^j})^2) \tag{2.26}$$

où c_i^j et σ_i^j spécifient la localisation de la moyenne et la largeur de l'étiquette floue X_i^j décrivant l'entrée x_i.

Réécrivons dans ce cas les équations (2.18) et (2.19) pour mettre en évidence les paramètres d'apprentissage du SIF:

$$\mu^r(x,c,\sigma)=\prod_{i=1}^{n}\mu_{X_i^r}(x_i,c_i^r,\sigma_i^r),\qquad r=1\ldots R \tag{2.27}$$

$$y_j(x,\omega)=\sum_{r=1}^{R}\hat{\mu}_r y_j^r,\quad \hat{\mu}_r=\frac{\mu^r(x,c,\sigma)}{\sum_{r=1}^{R}\mu^r(x,c,\sigma)} \tag{2.28}$$

$x=[x_1\ldots x_n]$, $c=[c_1^1\ldots c_1^{L1}\ldots c_n^1\ldots c_n^{Ln}]$, $\sigma=[\sigma_1^1\ldots\sigma_1^{L1}\ldots\sigma_n^1\ldots\sigma_n^{Ln}]$, $\omega=[c,\sigma,y_1^1\ldots y_m^R]$.

où c et σ sont deux vecteurs de paramètres regroupant respectivement les centres et les largeurs des fonctions d'appartenance des différentes variables d'entrée.

Un SIF de type TS peut être représenté par un RNA à deux ou trois couches cachées où chaque couche représente une des étapes du calcul de la sortie du SIF (Fig. 2-10).

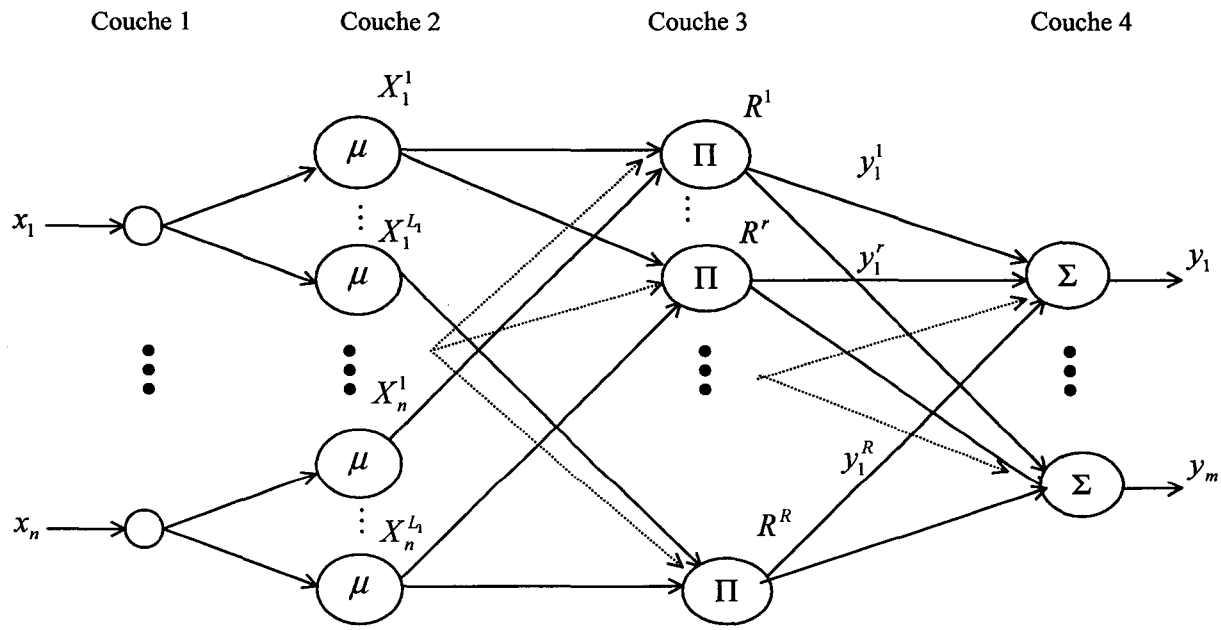


Figure 2-10 Représentation d'un SIF de type TS d'ordre zéro par un RNA.

Dans ce qui suit, nous allons décrire en détail le rôle de chaque couche pour ensuite montrer comment rétro-propager le gradient de l'erreur à partir des neurones de la couche de sortie (couche 4) vers les neurones de la couche cachée implémentant les fonctions d'appartenance (couche 2).

- **Couche 1 :** cette couche forme la couche d'entrée du RNA. Elle est formée de n neurones servant uniquement à transmettre les valeurs du vecteur d'entrée à la couche suivante. Par conséquent, le vecteur d'entrée et la sortie de chaque neurone de cette couche sont donnés par :

$$O_i^1 = u_i^1 = x_i, \quad i = 1 \ldots n, \tag{2.29}$$

- **Couche 2** : cette couche cachée du RNA implémente l'opération de fuzzification. Ses neurones calculent les fonctions d'appartenance de chacune des entrées aux différentes étiquettes floues la décrivant. Le nombre de neurones dans cette couche est donc exactement égal au nombre total des étiquettes floues utilisées pour la partition des différentes variables d'état. Le vecteur d'entrée, et la sortie de chaque neurone de cette couche sont donnés par :

$$O_l^2 = \exp\left(-\frac{1}{2}\left(\frac{u_i^2 - c_i^j}{\sigma_i^j}\right)^2\right), \; u_i^2 = O_i^1 \tag{2.30}$$

où : $\quad j = 1 \ldots L_i, \quad i = 1 \ldots n,$ et $\quad l = \begin{cases} j & \text{si } i = 1 \\ j + \sum_{k=1}^{i-1} L_k & \text{sinon} \end{cases}$

- **Couche 3 :** cette deuxième couche cachée du RNA implémente la première étape de traitement de l'inférence floue. Plus précisément, elle a pour rôle de calculer le degré de vérité des prémisses de chacune des règles floues. Le nombre de neurones dans cette couche est donc exactement égal au nombre total des règles floues du système

d'inférence. Le vecteur d'entrée, et la sortie de chaque neurone de cette couche sont donnés par :

$$O_r^3 = \prod_{\substack{l=1 \\ O_l^2 \in R^r}}^{L} u_l^3 \text{, et} \qquad u_l^3 = O_l^2 \qquad \text{avec } r = 1 \ldots R \tag{2.31}$$

où $O_l^2 \in R^r$ indique que l'étiquette floue associée à la sortie O_l^2 participe dans la règle R^r.

- **Couche 4** : cette couche, formant la couche de sortie du RNA, implémente la deuxième étape de traitement de l'inférence floue. Elle contient m neurones dont le rôle est de calculer les sorties numériques du SIF. Le vecteur d'entrée, la sortie, ainsi que les poids synaptiques de chaque neurone de cette couche sont donnés par :

$$O_j^4 = \frac{\sum_{r=1}^{R} u_r^4 y_j^r}{\sum_{r=1}^{R} u_r^4}, \; u_r^4 = O_r^3, \text{ et } w_{rj}^4 = y_j^r \qquad \text{avec } j = 1 \ldots m \tag{2.32}$$

2.5.2.4 Rétro-propagation de l'erreur

- **Couche 4** : En utilisant les équations (2.28) et (2.32), on peut aisément calculer la valeur du gradient à la sortie de chacun des neurones de cette couche :

$$\frac{\partial E_t^{\omega}}{\partial O_j^4} = -(y_j^* - y_j), \text{ et } \frac{\partial O_j^4}{\partial y_j^r} = \frac{O_r^3}{\sum_{k=1}^{R} O_k^3} = \hat{\mu}_r \tag{2.33}$$

D'où :

$$\frac{\partial E_t^{\omega}}{\partial y_j^r} = \frac{\partial E_t^{\omega}}{\partial O_j^4}\frac{\partial O_j^4}{\partial y_j^r} = -(y_j^* - y_j)\hat{\mu}_r \tag{2.34}$$

Par conséquent, les valeurs des conclusions locales y_j^r peuvent être mises à jour de la façon suivante :

$$y_j^r(t+1) = y_j^r(t) + \eta(y_j^* - y_j)\hat{\mu}_r \tag{2.35}$$

- **Couche 3** : Il n'existe pas de paramètres à ajuster dans cette couche. On se contentera alors de calculer le gradient associé à chaque neurone de cette couche. En utilisant les équations (2.28) et (2.31), on trouve :

$$\frac{\partial E_t^{\omega}}{\partial O_r^3} = \sum_{j=1}^{m}\frac{\partial E_t^{\omega}}{\partial O_j^4}\frac{\partial O_j^4}{\partial O_r^3} \tag{2.36}$$

où

$$\frac{\partial O_j^4}{\partial O_r^3} = \frac{\partial}{\partial O_r^3}\left(\frac{\sum_{r=1}^{R} y_j^r O_r^3}{\sum_{r=1}^{R} O_r^3}\right) = \frac{y_j^r \sum_{r=1}^{R} O_r^3 - \sum_{r=1}^{R} y_j^r O_r^3}{\left(\sum_{r=1}^{R} O_r^3\right)^2} = \frac{y_j^r - y_j}{\sum_{r=1}^{R} O_r^3} \tag{2.37}$$

et $\frac{\partial E_t^{\omega}}{\partial O_j^4}$ est donné par l'équation (2.32).

- **Couche 2** : En utilisant les équations (2.28) et (2.30), on peut aisément calculer la valeur du gradient de l'erreur à la sortie de chacun des neurones de cette couche :

$$\frac{\partial E_t^{\omega}}{\partial O_l^2} = \sum_{\substack{r=1 \\ O_l^2 \in R^r}}^{R} \frac{\partial E_t^{\omega}}{\partial O_r^3} \frac{\partial O_r^3}{\partial O_l^2} \tag{2.38}$$

où

$$\frac{\partial O_r^3}{\partial O_l^2} = \prod_{\substack{O_k^2 \in R^r \\ k \neq l}} O_k^2 \tag{2.39}$$

et $\frac{\partial E_t^{\omega}}{\partial O_l^2}$ est donné par l'équation (2.36).

- Par conséquent, on peut aisément calculer le gradient de l'erreur instantanée par rapport aux paramètres des fonctions d'appartenance :

$$\frac{\partial E_t^{\omega}}{\partial c_i^j} = \frac{\partial E_t^{\omega}}{\partial O_l^2} \frac{\partial O_l^2}{\partial c_i^j}, \text{ et } \frac{\partial E_t^{\omega}}{\partial \sigma_i^j} = \frac{\partial E_t^{\omega}}{\partial O_l^2} \frac{\partial O_l^2}{\partial \sigma_i^j}, \qquad l = \begin{cases} j & \text{si } i = 1 \\ j + \sum_{k=1}^{i-1} L_k & \text{sinon} \end{cases} \tag{2.40}$$

où

$$\frac{\partial O_l^2}{\partial c_i^j} = \frac{(x_i - c_i^j)}{(\sigma_i^j)^2} O_l^2, \text{ et } \frac{\partial O_l^2}{\partial \sigma_i^j} = \frac{(x_i - c_i^j)^2}{(\sigma_i^j)^3} O_l^2 \tag{2.41}$$

- Par conséquent, les valeurs des centres et des écart types des fonctions d'appartenance peuvent être mises à jour de la façon suivante :

$$c_i^j(t+1) = c_i^j(t) - \eta \frac{\partial E_t^{\omega}}{\partial c_i^j} \tag{2.42}$$

$$\sigma_i^j(t+1) = \sigma_i^j(t) - \eta \frac{\partial E_t^\omega}{\partial \sigma_i^j} \tag{2.43}$$

Il en résulte d'après l'équation (2.41) que le taux d'apprentissage utilisé pour la mise à jour des paramètres c et σ doit être soigneusement choisi (petit) car le gradient de l'erreur par rapport à ces paramètres peut être d'une forte amplitude. En outre, pour la stabilité de l'apprentissage, il est prudent d'opter pour un taux d'apprentissage plus agressif (grand) pour la mise à jour des conclusions y_j^r constituant la partie linéaire du SIF et d'opter pour un taux d'apprentissage plus conservateur (petit) pour la mise à jour des paramètres c et σ constituant la partie non linéaire du SIF.

2.6 Conclusion

Dans ce chapitre, nous avons effectué une étude bibliographique détaillée pour introduire la problématique de l'apprentissage par renforcement et présenter ses principaux algorithmes. L'analyse de ces algorithmes nous a motivé à consacrer cette thèse à l'étude des algorithmes AC en tant qu'alternative aux méthodes de contrôle adaptatif.

Nous avons également présenté les principales caractéristiques des systèmes neuro-flous dans le but de s'en servir pour l'implantation des différents éléments adaptatifs des agents RL.

Dans le prochain chapitre, nous allons analyser en détail les principaux algorithmes AC et proposer un ensemble de concepts et d'outils pour en améliorer les performances. Plus précisément, nous allons analyser en profondeur le mécanisme d'interaction entre le critique et l'acteur dans les méthodes AC existantes et en démontrer l'inefficacité. La solution que nous proposerons permettra au critique de rester cohérent avec l'acteur et à ce dernier d'apprendre plus rapidement. Nous nous retarderons aussi sur les risques associés à l'utilisation des traces d'éligibilité cumulatives dans le cas particulier où des systèmes neuro-flous sont utilisés pour l'implantation de l'acteur et du critique.

3 NOUVELLE ARCHITECTURE ACTEUR-CRITIQUE: L'AGENT FNAC

3.1 Introduction

Dans le chapitre précédent, nous avons présenté une formulation du problème de l'apprentissage par renforcement et analysé ses principaux algorithmes. Nous en sommes venus à la conclusion de consacrer l'essentiel de cette thèse à l'investigation de l'architecture AC. Nous proposons de l'étudier en tant qu'alternative possible de la commande adaptative directe (sans identification d'un modèle du système à contrôler). Par conséquent, on s'intéresse principalement aux cas où la stratégie de contrôle est déterministe et continue. Nous commencerons, dans ce chapitre, par l'étude du cas particulier où il n'existe qu'une seule variable de contrôle. Le cas du contrôle multivariable, quasiment non traité jusqu'à présent par les autres méthodes AC déterministes, sera traité au chapitre suivant.

Le présent chapitre est divisé en trois parties. Dans la première partie, nous commencerons par mettre en évidence les principales faiblesses des architectures AC existantes. Nous proposerons ensuite une nouvelle architecture, appelée FNAC (Fuzzy Neural Actor Critic), qui non seulement permet de combler les faiblesses des architectures AC existantes, mais aussi permet d'étendre leur champ d'applications. Dans la deuxième partie de chapitre, nous allons présenter les détails de l'implémentation de l'architecture FNAC à l'aide des systèmes neuro-flous. Enfin, la dernière partie de ce chapitre sera consacrée à la validation de la FNAC par la présentation des résultats de simulation.

3.2 Les architectures AC continues

Au chapitre précédant, nous avons présenté l'algorithme AHC. Cet algorithme est à la base de la quasi-totalité des algorithmes AC qui existent dans la littérature. Cependant, tel que présenté, il ne peut être utilisé que lorsque la stratégie de contrôle est discrète. Quoiqu'à priori ceci n'est pas un défaut en soi, puisque qu'on peut considérer la stratégie de contrôle discrète comme étant la quantification d'une stratégie continue, il en devient autrement lorsque on se penche sur l'implémentation de l'algorithme. En effet, et bien qu'il utilise une représentation explicite de la stratégie de contrôle, l'algorithme AHC n'échappe pas aux problèmes de l'explosion dimensionnelle tant reprochés aux algorithmes de type QL. En effet, la mise en œuvre de la fonction des préférences $p(s,a)$ (Fig. 2-3) requiert autant d'espace mémoire que la fonction Q. D'ailleurs, cette dernière peut jouer élégamment le rôle de la première tout en éliminant la nécessité d'estimer en plus la fonction V. Il s'agit probablement d'une autre raison de la marginalisation des méthodes AC comparativement aux méthodes QL.

Cela dit, lorsque la stratégie de contrôle est déterministe et continue, l'algorithme AC peut être reformulé de sorte qu'aucune fonction de type $F(s,a)$, comme c'est le cas dans le QL avec la fonction d'évaluation Q et dans AHC avec la fonction de préférence p, ne soit utilisée; éliminant ainsi le phénomène de l'explosion dimensionnelle. Rappelons que la raison principale d'être de la fonction Q est qu'elle nous permet de comparer toutes les actions possibles dans un état donné s. Or, sans être aussi exhaustive, la fonction V peut jouer ce rôle. En effet, elle permet de comparer l'action recommandée $\pi(s)$, calculée à partir de la stratégie de contrôle de l'acteur, avec n'importe quelle autre action pourvu que cette dernière soit celle effectivement exécuté par l'agent. En effet, si une action a est effectivement exécutée au lieu de l'action $\pi(s)$ et si l'agent suit la stratégie π strictement par la suite, on sait, d'après l'équation (2.7), que l'utilité (la valeur) de l'action a dans l'état s est définie par :

$$Q(s,a) = r(s,a) + \gamma V(s'(s,a)) = r + \gamma V(s')$$

Alors que l'utilité de l'action $\pi(s)$ est tout simplement égale à $V(s)$:

$$Q(s,\pi(s)) = r(s,\pi(s)) + \gamma V(s'(s,\pi(s))) = V(s)$$

On peut alors comparer ces deux actions en formant le signal de renforcement interne :

$$\hat{r} = Q(s,a) - Q(s,\pi(s)) = r + \gamma V(s') - V(s), \tag{3.1}$$

Dans ce cas, on peut dire que l'action $\pi(s)$ est plus utile que l'action a si le renforcement interne est négatif et vice versa. Par conséquent, l'agent peut améliorer sa stratégie de contrôle uniquement par exploration en utilisant la règle de mise à jour suivante :

$$\pi(s) \leftarrow \pi(s) + \alpha\hat{r}(a - \pi(s)) \tag{3.2}$$

Autrement dit, l'action recommandée $\pi(s)$ doit être corrigée dans la direction de l'action explorée a si celle-ci est plus utile ($\hat{r} > 0$), ou dans la direction opposée sinon ($\hat{r} \leq 0$). Dans ce dernier cas, $\pi(s)$ sera corrigé dans la direction de $2\pi(s) - a$ tel que démontré par le raisonnement suivant :

$$\pi(s) \leftarrow \pi(s) + \alpha(-\hat{r})(\pi(s) - a).$$
$$\pi(s) \leftarrow \pi(s) + \alpha(-\hat{r})(\pi(s) - a + \pi(s) - \pi(s)).$$
$$\pi(s) \leftarrow \pi(s) + \alpha(-\hat{r})((2\pi(s) - a) - \pi(s)).$$

Finalement, mentionnons que si l'action $\pi(s)$ est déjà optimale, l'agent ne devra pas procéder à l'exploration et par conséquent, l'équation (3.2) ne risque pas de perturber le choix d'action dans l'état en question.

Par conséquent, on dispose d'un mécanisme d'amélioration de la stratégie de contrôle qui permet de découvrir progressivement les actions optimales et de les insérer dans la stratégie de contrôle. Ce principe est à la base des architectures GARIC (Berenji, 1992) et RFNN (Lin, 1994) où on trouve deux architectures (Fig. 3-1) d'apprentissage par renforcement très semblables bien que développées séparément d'après leurs auteurs. La seule différence

notable entre ces deux architectures se situe au niveau de l'implémentation des deux éléments adaptatifs, en l'occurrence l'acteur et le critique.

On se référera désormais à ces deux architectures par le nom GARIC vu que cette architecture fût la première à être publiée. La stratégie de contrôle étant déterministe dans les deux cas, les auteurs suggèrent d'utiliser un explorateur stochastique pour résoudre le dilemme de l'équilibre exploration/exploitation. Il s'agit d'un explorateur gaussien centré sur la valeur de l'action recommandée et dont la variance est modulée par la valeur de l'état.

Cela dit, on peut voir l'architecture GARIC comme une adaptation de l'architecture AHC dans le cas où la stratégie de contrôle serait déterministe et où l'espace d'actions est continu. Cependant, elle traîne avec elle deux lacunes sérieuses de l'architecture AHC, auxquelles elle vient ajouter au moins une autre lacune propre à elle. Ces lacunes seront démontrées dans la section qui suit.

Extrait de (Berenji, 1992)

Initialize the value function V, e.g., $V(s) = 0$ for all states,
Initialize the policy π randomly,
Repeat (for each episode)
    Initialize s
    Repeat (for each step of episode)
        Compute $\pi(s)$, $V(s)$
        $a \leftarrow$ Gaussian random value with mean $\pi(s)$ and standard deviation $\sigma(V(s))$
        Take action a, observe reward r, and next state s'
        $\hat{r} = r + \gamma V(s') - V(s)$
        $V(s) \leftarrow V(s) + \beta\hat{r}$
        $\pi(s) \leftarrow \pi(s) + \alpha\hat{r}\left[\frac{a - \pi(s)}{\sigma(V(s))}\right]$
        $s' \leftarrow s$
    Until s is terminal

Figure 3-1 L'algorithme GARIC.

3.3 L'agent FNAC

Dans ce qui suit, les principaux éléments distinctifs de l'agent FNAC seront introduits progressivement pour combler des lacunes, mises en évidence au préalable, communes ou non aux deux principales méthodes AC citées ci-haut.

3.3.1 Le critique doit rester consistent avec l'acteur

Dans tous les algorithmes AC que nous avons présenté précédemment, la mise à jour du critique est effectuée de façon systématique quelle que soit l'action réellement exécutée par l'agent. Il en résulte que certains états risquent de voir leurs '*valeurs*' injustement diminuer suite à une exploration non fructueuse d'une action différente de celle recommandée par l'acteur (voir Éq. 2-12 et 2-13). Or, une telle exploration renforcera la confiance de l'agent dans sa stratégie de contrôle. Par conséquent, il n'y a aucune raison pour que le critique altère négativement la fonction d'évaluation puisque la stratégie de contrôle reste la même. Or, toute modification de la fonction d'évaluation risque d'entraîner à son tour la modification de la stratégie de contrôle qui rappelons-le tend toujours à maximiser la fonction d'évaluation, d'où une *perturbation* globale du processus d'apprentissage provoquant son ralentissement. L'exemple qui suit se veut une illustration très claire de ce phénomène.

Exemple 3.1

Considérons le système simple représenté sur la figure 3-2 où le but de l'agent est d'atteindre l'état terminal $s = 0$ le plus rapidement possible. L'agent reçoit un crédit de +1 lorsqu'il entre dans cet état, et un crédit nul partout ailleurs. Dans chaque état, l'agent peut choisir de se déplacer vers la gauche, *action* = -1, se rapprochant ainsi davantage de son but ou de se déplacer vers la droite, *action* = +1, s'éloignant ainsi davantage de son but.

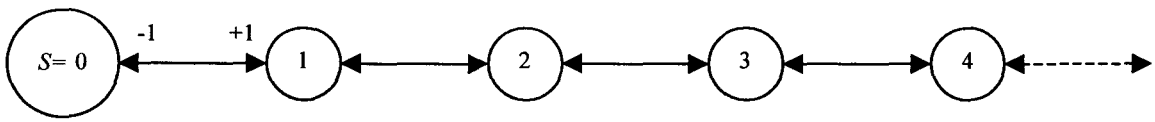


Figure 3-2 Illustration des problèmes d'apprentissage avec un critique qui suit aveuglement l'action exécutée par l'agent.

La stratégie de contrôle suivie par l'agent est la suivante :

$$a_\pi = \begin{cases} -1 & \text{si} \quad \pi(s) \geq 0.5 \\ +1 & \text{sinon} \end{cases}$$

et la dynamique de l'environnement est définie par :

$$s' = s + a_\pi$$

Dans cet exemple, où la stratégie de contrôle est discrète, nous utiliserons l'architecture AHC (Fig. 2-3) avec les valeurs numériques suivantes : $\gamma = 0.9$, $\alpha = 0.6$, $\beta = 0.6$, $\lambda = 0$, et on suppose que l'état initial est toujours l'état $s = 2$. Nous avons choisi le couple (acteur, critique) initial du tableau 3-1 tel que la stratégie de contrôle initiale est optimale et la fonction d'évaluation ordonne correctement les états. Le but de la simulation est d'observer l'impact d'une exploration non fructueuse sur le processus d'apprentissage.

State	1	2	3	4
V	0.7500	0.6500	0.5600	0.4790
π	0.5200	0.5200	0.5200	0.5200

Tableau 3-1 Couple (acteur, critique) de l'agent au début de la simulation.

Supposons que l'agent explore accidentellement l'action non optimale $a = +1$, lors de ses deux premières visites à l'état $s = 2$ tout en suivant strictement sa stratégie de contrôle le reste du temps. Nous allons observer et comparer le nombre d'itérations nécessaires à l'agent pour retrouver l'état terminal $s = 0$. À noter que la solution optimale est 6 itérations. Les résultats de cette simulation sont présentés dans le tableau 3-2 ci-dessous.

On voit que l'agent met 8 itérations avant de retrouver son but. Analysons le processus d'apprentissage pas à pas. À l'instant $t=1$, l'agent, se trouvant dans l'état $s=2$, a exécuté l'action non optimale $a=+1$ au lieu de l'action optimale calculée par l'acteur $a_\pi=-1$; ce qui l'a conduit à l'état suivant $s'=3$. Puisque l'agent se retrouve dans un état encore plus loin de son but, il en résulte un renforcement interne négatif et par conséquent, à l'instant $t=2$, la valeur de cet état a diminué en passant de 65% à 56.24%. Pourtant, l'exploration n'a fait que renforcer la conviction de l'agent dans l'action optimale calculée par l'acteur $a=a_\pi=-1$, son degré de confiance passant de 52% à 60.76%. Il n'y a donc aucune raison de s'inquiéter et de diminuer la valeur de l'état $s=2$; puisque la stratégie de contrôle est restée inchangée. En effet, dans ce cas ci, seule la valeur de l'état $s=1$, résultat de l'exécution de l'action recommandée dans l'état $s=2$, doit affecter la valeur de ce dernier. Autrement, la fonction d'évaluation ne suit plus la stratégie de contrôle.

Or, dans tous les algorithmes AC présentés ci-dessus, la valeur de l'état diminue systématiquement chaque fois qu'un renforcement interne négatif est produit, exploration ou pas. Le critique n'évalue plus la stratégie de contrôle de l'acteur mais plutôt celle suivie par l'agent. Un effet d'avalanche se produira alors et affectera négativement et le critique et l'acteur. En effet, un autre renforcement interne négatif verra le jour lors de la transition de l'environnement de l'état $s=3$ à l'état $s=2$ à cause de la sous-estimation de la valeur de ce dernier. Par conséquent, à l'instant $t=3$, l'agent a diminué la valeur de l'état $s=3$, mais, aussi et surtout, a diminué sa conviction dans l'action optimale $a=-1$, son degré de confiance passant de 52% à 48.77%, et elle sera dorénavant substituée par l'action non optimale $a=+1$. Désormais, la stratégie de contrôle n'est plus optimale. La deuxième exploration dans l'état $s=2$, à l'instant $t=3$, aura pour conséquence d'emmener l'agent successivement dans l'état $s=3$ puis dans l'état $s=4$. De façon similaire, à l'instant $t=6$, la valeur de l'état $s=4$, ainsi que le choix d'action dans cet état seront affectés négativement.

State	1	2	3	4
V	0.7500	0.6500	0.5600	0.4790
π	0.5200	0.5200	0.5200	0.5200
$t = 1,\ s = 2,\ a_\pi = -1,\ a = 1,\ s' = 3$				
V	0.7500	0.5624	0.5600	0.4790
π	0.5200	0.6076	0.5200	0.5200
$t = 2,\ s = 3,\ a_\pi = -1,\ a = -1,\ s' = 2$				
V	0.7500	0.5624	0.5277	0.4790
π	0.5200	0.6076	0.4877	0.5200
$t = 3,\ s = 2,\ a_\pi = -1,\ a = 1,\ s' = 3$				
V	0.7500	0.5099	0.5277	0.4790
π	0.5200	0.6601	0.4877	0.5200
$t = 4,\ s = 3,\ a_\pi = 1,\ a = 1,\ s' = 4$				
V	0.7500	0.5099	0.4697	0.4790
π	0.5200	0.6601	0.5457	0.5200
$t = 5,\ s = 4,\ a_\pi = -1,\ a = -1,\ s' = 3$				
V	0.7500	0.5099	0.4697	0.4453
π	0.5200	0.6601	0.5457	0.4863
$t = 6,\ s = 3,\ a_\pi = -1,\ a = -1,\ s' = 2$				
V	0.7500	0.5099	0.4632	0.4453
π	0.5200	0.6601	0.5392	0.4863
$t = 7,\ s = 2,\ a_\pi = -1,\ a = -1,\ s' = 1$				
V	0.7500	0.6090	0.4632	0.4453
π	0.5200	0.7591	0.5392	0.4863
$T = 8,\ s = 1,\ a_\pi = -1,\ a = -1,\ s' = 0$				
V	0.9000	0.6090	0.4632	0.4453
π	0.6700	0.7591	0.5392	0.4863

Tableau 3-2 L'apprentissage avec un critique classique.

State	1	2	3	4
V	0.7500	0.6500	0.5600	0.4790
π	0.5200	0.5200	0.5200	0.5200
$t = 1,\ s = 2,\ a_\pi = -1,\ a = 1,\ s' = 3$				
V	0.7500	0.6500	0.5600	0.4790
π	0.5200	0.6076	0.5200	0.5200
$t = 2,\ s = 3,\ a_\pi = -1,\ a = -1,\ s' = 2$				
V	0.7500	0.6500	0.5750	0.4790
π	0.5200	0.6076	0.5350	0.5200
$T = 3,\ s = 2,\ a_\pi = -1,\ a = 1,\ s' = 3$				
V	0.7500	0.6500	0.5750	0.4790
π	0.5200	0.6871	0.5350	0.5200
$t = 4,\ s = 3,\ a_\pi = -1,\ a = -1,\ s' = 2$				
V	0.7500	0.6500	0.5810	0.4790
π	0.5200	0.6871	0.5410	0.5200
$t = 5,\ s = 2,\ a_\pi = -1,\ a = -1,\ s' = 1$				
V	0.7500	0.6650	0.5810	0.4790
π	0.5200	0.7021	0.5410	0.5200
$t = 6,\ s = 1,\ a_\pi = -1,\ a = -1,\ s' = 0$				
V	0.9000	0.6650	0.5810	0.4790
π	0.6700	0.7021	0.5410	0.5200

Tableau 3-3 L'apprentissage avec le critique de l'agent FNAC.

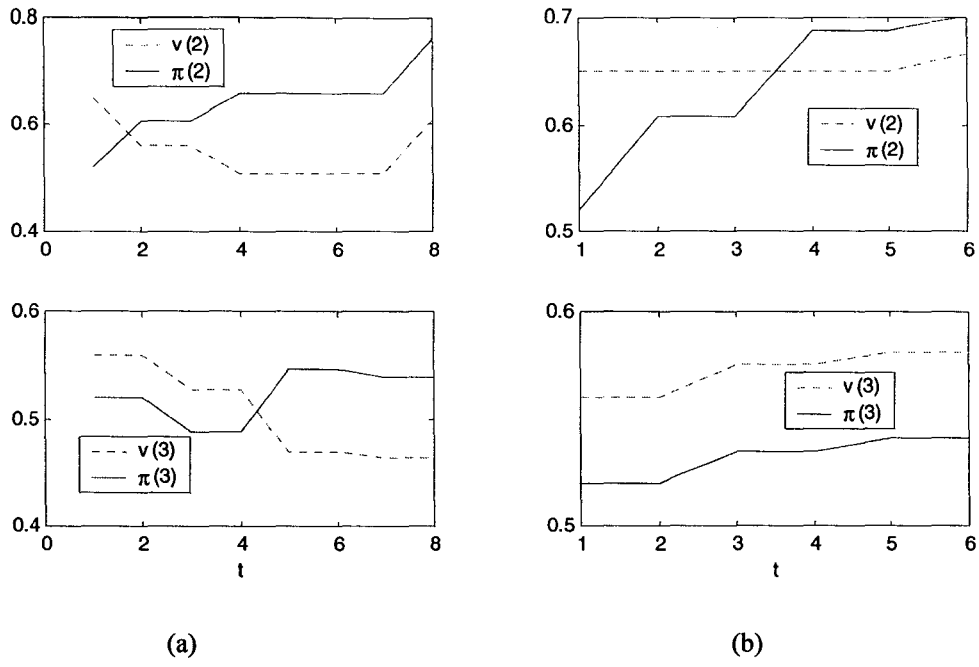


Figure 3-3 Variations de la stratégie et de la fonction d'évaluation dans les états s = 2 et s = 3. (a) Critique classique et (b) Critique de l'agent FNAC.

Sur la figure 3-3, nous avons représenté les variations de la valeur des états provoquées par l'exploration. On peut remarquer qu'en général, l'exploration a renforcé le choix des actions optimales (courbe π (2) croissante) mais que paradoxalement, les valeurs des états ont diminué comme si la stratégie de contrôle de l'acteur n'est plus optimale. Cette inconsistance entre l'acteur et le critique provoque donc une perturbation générale du processus d'apprentissage. Malgré que ces perturbations ne soient que temporaires, la stratégie de contrôle de l'acteur et celle suivie par l'agent redevenant ultérieurement identiques, le processus d'apprentissage risque d'être considérablement ralenti.

Signalons enfin que l'utilisation de traces d'éligibilité, un outil puissant utilisé pour accélérer la convergence des algorithmes TD(λ), accentue davantage ce problème puisque la sous-évaluation accidentelle d'un état risque de se propager vers les états qui l'ont précédé. Par exemple, pour un $\lambda = 0.9$, nous avons noté que l'agent met 12 itérations avant de retrouver son but; les paramètres d'apprentissage étant identiques.

3.3.2 Le critique de l'agent FNAC est vigilant

Puisque l'exploration de nouvelles actions est indispensable dans les méthodes RL en général et les méthode AC en particulier, nous proposons de mettre à jour la fonction d'évaluation dépendamment du résultat de l'exploration. Dans le cas où l'action explorée s'avère meilleure que l'action calculée par l'acteur (renforcement interne positif), on corrige cette dernière dans la direction de l'action explorée et la valeur de l'état où l'exploration a eu lieu augmente ainsi qu'éventuellement les valeurs des états récemment visités. Par contre, si l'action explorée s'avère moins bonne que l'action de l'acteur (renforcement interne négatif), on corrige cette dernière dans la direction opposée de l'action explorée, mais par contre on garde intacte la valeur de l'état en question tout en remettant à zéro toutes les traces d'éligibilité pour préserver les valeurs des états précédents. Par conséquent, la règle de mise à jour de l'acteur reste à priori inchangée, mais celle du critique devient :

$$V(u) \leftarrow V(u) + \beta(r + \gamma V(s') - V(s))e(u) \tag{3.3}$$

où

$$e(u) \leftarrow \begin{cases} \gamma\lambda e(u) + \delta_u^s & \text{si } r + \gamma V(s') - V(s) \geq 0 \\ 0 & \text{sinon} \end{cases}, \text{ où } \delta_u^s = \begin{cases} 1 & \text{si } u = s \\ 0 & \text{sinon} \end{cases} \tag{3.4}$$

Autrement dit, si l'exploration révèle une action meilleure que celle calculée par l'acteur, le critique est mis à jour en anticipant que l'action explorée fera désormais partie de la stratégie de contrôle. Dans le cas contraire, l'acteur renforce sa confiance dans sa stratégie, voire même essaye de s'ajuster dans la direction opposée à celle de l'action explorée. La stratégie de contrôle reste donc sensiblement la même, mais l'état actuel de l'agent n'est pas celui qui succédera dorénavant à l'état où l'exploration a eu lieu. La valeur de ce dernier ne doit donc pas être modifiée ni celles des états le précédant.

En résumé, le critique reste concentré uniquement sur l'évaluation de la stratégie de contrôle de l'acteur, et la paire (acteur, critique) ne peut donc que s'améliorer après une opération d'exploration, quel qu'en soit le résultat.

Le tableau 3-3, ci-dessus, illustre les résultats de cette technique appliquée à l'exemple 3-1. On voit que l'agent met exactement 6 itérations avant de retrouver son but ; ce qui constitue une solution optimale. La raison est qu'aucun état n'a vu sa valeur diminuer injustement au cours de cette simulation (Fig. 3-3). Tout au contraire, les valeurs des états se sont ajustées à la hausse pour s'approcher davantage des valeurs optimales (Fig. 3-3). On peut donc constater que la solution que nous avons proposée améliore le processus de l'apprentissage de l'agent. Ce constat sera appuyé et consolidé empiriquement en se basant sur les résultats des simulations sur des problèmes plus complexes et plus standards traités dans le reste de ce chapitre ainsi que dans le prochain chapitre. Néanmoins, une analyse théorique plus approfondie ainsi qu'une démonstration formelle plus soutenue seront nécessaires pour affirmer la supériorité de la solution proposée

3.3.3 L'acteur doit renforcer équitablement les actions

Utilisé en tant que comparateur d'actions, le signal de renforcement interne donné par l'équation (3.2) offre certes un moyen efficace pour déterminer dans quelle direction l'acteur doit s'ajuster. Cependant, l'amplitude de cet ajustement est proportionnelle à la valeur de l'état dans lequel l'exploration a eu lieu. Ainsi, une meilleure action découverte dans un état de faible valeur (en terme de valeur absolue) a moins de chance de faire sa place dans la stratégie de contrôle que si elle était découverte dans un autre état de valeur plus importante. L'exemple qui suit est une illustration très claire de ce phénomène.

Exemple 3.2

Considérons à nouveau le système de la figure 3-2.

State	1	2	3	...	8	9	10
V	1.0000	0.9000	0.8100	...	0.4783	0.4305	0.3874
π	0.7500	0.2500	0.7500	...	0.7500	0.2500	0.7500

Tableau 3-4 Couple (acteur, critique) de l'agent au début de la simulation.

Dans cet exemple, nous utiliserons également l'architecture AHC avec les valeurs numériques suivantes : $\gamma = 0.9$, $\alpha = 0.35$, $\beta = 0.1$, $\lambda = 0$, et on suppose que l'état initial est toujours l'état $s = 9$. Partant du couple (acteur, critique) du tableau 3-4, l'agent a pour mission de trouver l'état terminal $s = 0$ le plus rapidement possible. Les valeurs du tableau 3-4 sont optimales exceptées les valeurs de l'acteur dans les états $s = 9$ et $s = 2$, où nous avons intentionnellement substitué deux valeurs erronées mais égales. Le but de cette simulation est d'observer et comparer le nombre d'itérations nécessaires à l'agent pour retrouver les actions optimales dans ces deux états. À noter l'écart très important entre les valeurs numériques de la fonction d'évaluation dans ces deux états.

Les résultats de cette simulation sont illustrés sur les figures 3-4 et 3-5. On y a représenté l'évolution de la mise à jour de la stratégie de contrôle dans les deux états qui nous intéressent. Les paliers correspondent aux itérations où l'agent se trouve en dehors de l'état considéré. On voit qu'un total de 13 visites est nécessaire pour que l'agent retrouve l'action optimale dans l'état $s = 9$ (Fig. 3-4), comparativement à 5 visites dans le cas de l'état $s = 2$ (Fig. 3-5). La raison de cet écart est toute simple et est clairement illustrée par la courbe de la figure 3-8, où on a représenté le signal de renforcement interne dans les deux états, qui d'après l'équation (3.2) module le facteur d'apprentissage α. On voit que la pénalité accompagnant l'exécution de l'action non optimale $a = 1$ est d'amplitude nettement moins importante dans l'état $s = 9$ que dans l'état $s = 2$ de plus grande valeur. Par conséquent, l'agent se corrige avec plus de vigueur dans l'état $s = 2$, et y recouvrera donc plus rapidement l'action optimale. Les actions ne sont pas critiquées équitablement, mais, au contraire, sont discriminées selon l'amplitude de la valeur de l'état où elles ont été exécutées. Soulignons que la partie positive de cette courbe (Fig. 3-8) correspond au renforcement de la stratégie de contrôle dans les autres états voisins où elle est déjà optimale.

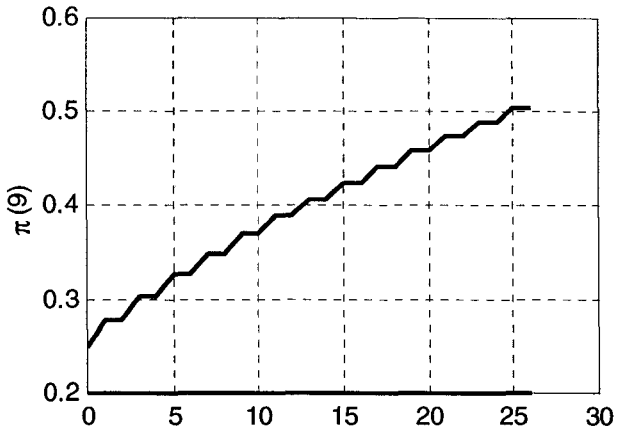


Figure 3-4 Courbe d'apprentissage de l'acteur non équitable dans l'état s = 9.

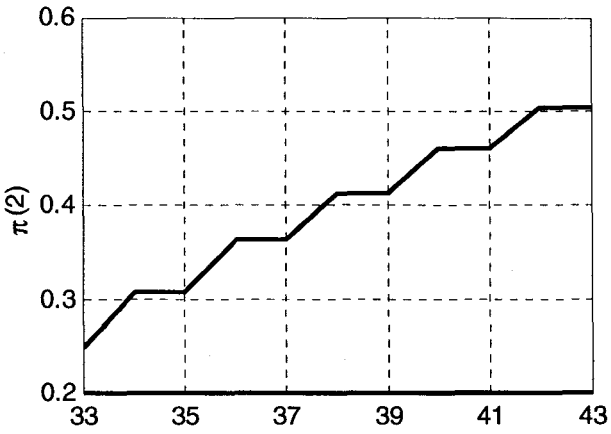


Figure 3-5 Courbe d'apprentissage de l'acteur non équitable dans l'état s = 2.

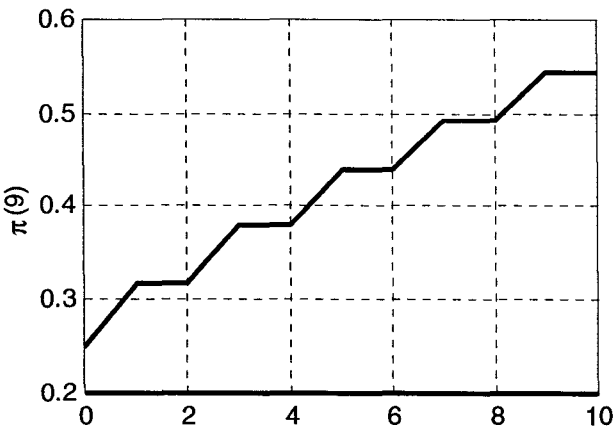


Figure 3-6 Courbe d'apprentissage de l'acteur équitable dans l'état s = 9.

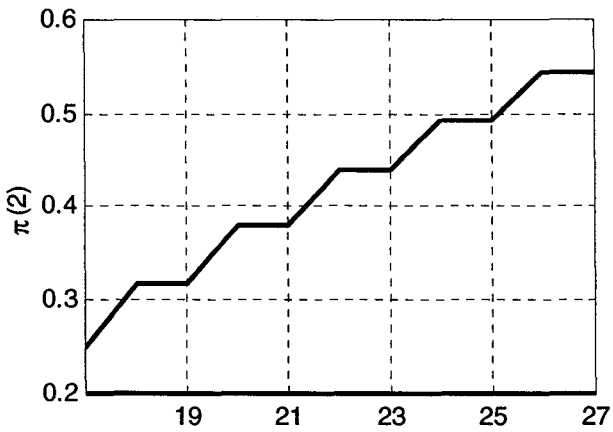


Figure 3-7 Courbe d'apprentissage de l'acteur équitable dans l'état s = 2.

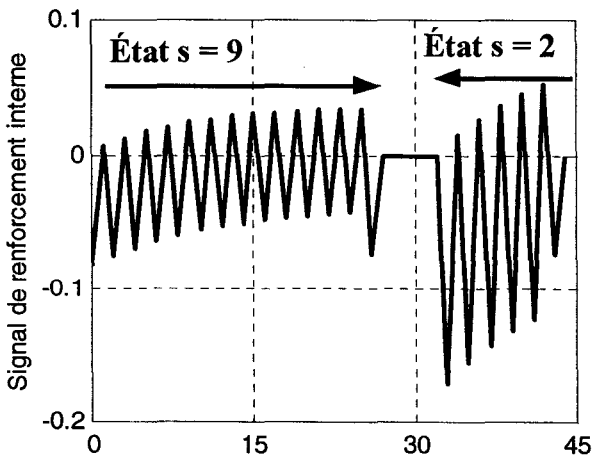


Figure 3-8 Signal de renforcement interne pour l'acteur non équitable.

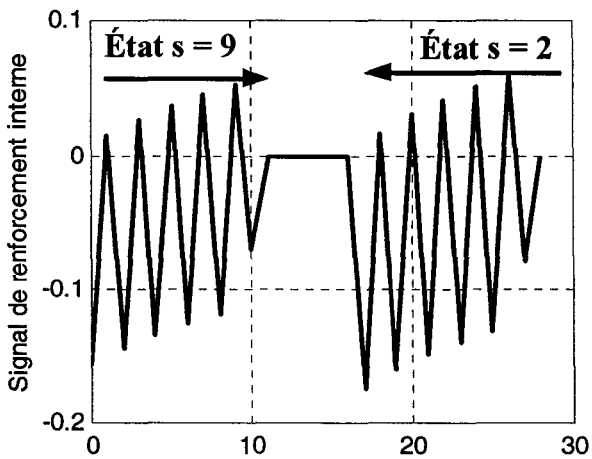


Figure 3-9 Signal de renforcement interne pour l'acteur équitable.

3.3.4 L'acteur de l'agent FNAC est équitable

Pour critiquer les actions de façon équitable, nous proposons de normaliser le signal de renforcement interne de sorte qu'une action rapprochant l'agent de son but soit toujours bien récompensée indépendamment de l'état où elle a été exécutée, et inversement.

Étant donnée la forme exponentielle de la fonction d'évaluation, nous proposons de normaliser le signal de renforcement interne (Éq. 3-2) en le divisant par la valeur absolue de la valeur de l'état dans lequel l'action a été exécutée. La règle de mise à jour de l'acteur devient :

$$\pi(s) \leftarrow \pi(s) + \alpha \left[\frac{r + \gamma W(s') - V(s)}{\varsigma + |V(s)|} \right] (a - \pi(s)) \tag{3.5}$$

où $\varsigma \geq 0$ est un seuil minimum utilisé pour éviter une division par zéro. Comme nous le verrons plus loin, notre stratégie d'exploration à seuil élimine quasiment tout risque de division par zéro excepté quand on utilise une stratégie de contrôle de type ε-greedy au quel cas, on posera $\varsigma = |\tau|$

Les figures 3-6 et 3-7 illustrent les résultats de cette technique appliquée à l'exemple 3.2. On voit que l'agent met exactement le même nombre de visites (5 visites) pour retrouver l'action optimale dans chacun des deux états perturbés. Sur la figure 3-9, nous avons représenté la courbe du nouveau signal de renforcement interne utilisé par l'acteur et défini par :

$$\hat{r}_{\pi} = \frac{r + \gamma W(s') - V(s)}{\varsigma + |V(s)|} \tag{3.6}$$

On voit que l'amplitude de ce dernier est quasiment la même dans les deux états considérés (partie négative de la courbe).

3.3.5 Traces d'éligibilité

Les traces d'éligibilité sont un outil très efficace utilisé pour accélérer la convergence des algorithmes d'apprentissage par renforcement en permettant d'actualiser la fonction d'évaluation et la stratégie de contrôle dans l'état actuel de l'environnement mais aussi dans tous les états récemment visités. Dans le cas où la stratégie de contrôle serait discrète, il est toujours possible d'utiliser, sans trop de modifications, les traces d'éligibilité utilisées dans l'architecture AHC (Barto, 1983) et définies par :

$$p(u,b) \leftarrow p(u,b) + \alpha \hat{r}(s) e(u,b) \tag{3.6}$$

où

$$e(u,b) \leftarrow \gamma \lambda e(u,a) + \delta_{u,b}^{s,a}, \quad \text{où } \delta_{u,b}^{s,a} \begin{cases} 1 & \text{si } u = s \text{ et } b = a \\ 0 & \text{sinon} \end{cases} \tag{3.7}$$

Cependant, dans le cas où l'espace d'actions serait continu ou de très grande dimension, il n'est plus possible d'utiliser une telle forme d'éligibilité, d'une part à cause de l'espace mémoire physique qu'occuperaient de telles traces, mais d'autre part, et surtout, à cause de la nature même du mécanisme d'amélioration de la stratégie de contrôle. En effet, les actions ne sont plus renforcées sur une base individuelle, mais seulement sur une base comparative entre deux actions. Dans l'architecture GARIC, l'usage des traces d'éligibilité a été tout simplement écarté aussi bien pour l'acteur que pour le critique (Lin, 1994) sous prétexte que seul l'algorithme TD(0) est à ce jour garanti d'être optimal (Sutton, 1988). Cependant, cette garantie étant conditionnelle à l'usage d'approximateurs universels linéaires, Lin et Berenji n'ont pas respecté cette condition dans les deux architectures GARIC qu'ils ont proposées où ils ont utilisé des approximateurs non linéaires. D'autre part, Tsitsiklis et Van Roy ont démontré plus tard que l'algorithme TD(λ) converge dans le cas général, *i.e.* $\lambda \in [0,1]$ (Tsitsiklis, 1997). Mieux encore, ils ont démontré que l'erreur d'estimation de la fonction d'évaluation est bornée par :

$$\max(\left|V(s,\omega^*) - V^*(s)\right|)\frac{1-\gamma\lambda}{1-\gamma},$$

où ω^* est le vecteur de paramètres optimal associé au meilleur approximateur universel linéaire indépendamment de l'algorithme d'apprentissage utilisé.

On voit que cette borne est une fonction décroissante de λ. Par conséquent, malgré qu'il ne s'agit là que d'une borne, il est justifié de croire que plus la valeur de λ est grande, meilleure serait la précision de l'approximateur universel. Au fait, des études empiriques ont démontré que la vitesse de convergence de l'algorithme TD(λ) est meilleure pour des valeurs de λ inférieures à 1 (Sutton, 1988 ; Singh, 1996).

Dans le cas de l'agent FNAC, on propose d'utiliser systématiquement les traces d'éligibilité pour l'estimation de la fonction d'évaluation de la même manière standard suggérée dans l'architecture AHC à deux différences près. D'une part, les traces d'éligibilité sont remises à zéro après chaque exploration non fructueuse (Éq. 3-4). D'autre part, on y introduit une adaptation des traces d'éligibilité aux cas où l'acteur et le critique sont implémentés à l'aide des réseaux neuro-flous (voir section suivante). En ce qui concerne l'acteur, nous proposons une nouvelle forme de traces d'éligibilité adaptées aux stratégies de contrôle continues :

$$\pi(u) \leftarrow \pi(u) + \alpha\frac{\hat{r}(s)}{\varsigma + \left|V(s)\right|}e(u) \tag{3.8}$$

où

$$e(u) \leftarrow \gamma\lambda e(u) + (a - \pi(s))\delta_u^s \tag{3.9}$$

Si en plus, on tient à la consistance entre l'acteur et le critique, comme il est proposé dans la section 3.2.2, on peut utiliser la définition suivante des traces d'éligibilité :

$$e(u) \leftarrow \begin{cases} a - \pi(s) & \text{si } u = s, \\ \gamma\lambda e(u) & \text{si } \hat{r} \geq 0 \text{ et } u \neq s \\ 0 & \text{sinon} \end{cases} \tag{3.10}$$

On voit que contrairement à l'équation (3.4), la trace d'éligibilité de l'état dans lequel l'exploration a eu lieu n'est pas remise à zéro, mais tout simplement réinitialisée comme si cet état était visité pour la première fois. À noter également que les équations (3.9) et (3.10) correspondent à des traces d'éligibilité de type substitutives. Il est en effet impossible d'utiliser des traces de type cumulatives dans ce cas-ci puisque chaque trace incorpore aussi le signal d'erreur. C'est pour cette raison que cette trace peut être négative. Dans le cas du critique, on parle d'une trace d'un état. Dans le cas de l'acteur, la trace que nous avons introduite est celle de l'état dans lequel l'exploration a eu lieu mais elle est modulée par le signal d'erreur formé par écart entre l'action exécutée et l'action calculée par l'acteur.

Pour bien préciser l'idée derrière cette définition, nous proposons d'analyser, pas à pas, le processus d'apprentissage dans l'exemple suivant.

Exemple 3.3

Considérons la situation ci-dessous (Fig. 3-10) où l'agent alterne exploration et exploitation. Nous allons considérer le cas extrême où $\lambda = 1$. À noter que les états marqués par un cercle plein sont meilleurs que ceux marqués par un cercle vide.

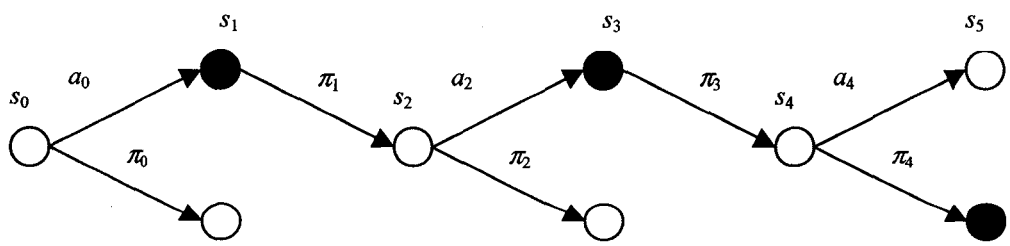


Figure 3-10 Illustration de l'effet des traces d'éligibilité définies pour l'acteur.

À l'instant $t = 0$, l'agent exécute l'action explorée a_0 au lieu de l'action π_0 calculée par l'acteur. Il en résulte une amélioration de son état. D'après les équations (3.8) (où nous avons omis intentionnellement la normalisation du signal de renforcement interne pour ne pas alourdir les calculs) et (3.10), l'acteur mettra à jour sa stratégie de contrôle comme suit :

$$\pi_0^1 = \pi(s_0, t=1) \leftarrow \pi_0^0 + \alpha(r_0 + \gamma V_1 - V_0)(a_0 - \pi_0).$$

où : $\pi_0^0 = \pi(s_0, t=0) = \pi_0$.

À l'instant $t = 1$, l'agent exécute l'action π_1 calculée par l'acteur. D'après les équations (3.8) et (3.10), l'acteur mettra à jour sa stratégie de contrôle uniquement dans l'état $s = 0$ où il y'a eu exploration. La mise à jour se fait comme suit :

$$\pi_0^2 \leftarrow \pi_0^1 + \alpha\gamma(r_1 + \gamma V_2 - V_1)(a_0 - \pi_0).$$

$$\pi_0^2 \leftarrow \pi_0^0 + \alpha(r_0 + \gamma V_1 - V_0)(a_0 - \pi_0) + \alpha\gamma(r_1 + \gamma V_2 - V_1)(a_0 - \pi_0).$$

$$\pi_0^2 \leftarrow \pi_0^0 + \alpha(r_0 + \gamma r_1 + \gamma^2 V_2 - V_0)(a_0 - \pi_0).$$

On dispose donc d'une meilleure estimation du signal de renforcement interne vu que, comparativement à $r_0 + \gamma V_1$, l'expression $r_0 + \gamma r_1 + \gamma^2 V_2$ constitue une meilleure estimation de V_0 puisque qu'elle contient deux valeurs exactes du signal de renforcement externe dont la fonction d'évaluation n'est que la somme cumulative escomptée.

À l'instant $t = 2$, l'action explorée s'avère encore une fois meilleure. Par conséquent, l'acteur met à jour l'action calculée dans l'état $s = 0$ mais également, le critique augmente la valeur de l'état $s = 0$ où l'exploration a eu lieu. Il en résulte que l'action a_0 est encore meilleure que prévu et l'acteur doit s'ajuster dans sa direction lorsque il se trouve dans l'état $s = 0$.

À l'instant $t = 4$, l'action explorée s'avère cette fois moins bonne que celle calculée par l'acteur. Dans ce cas, toutes les traces d'éligibilité sont remises à zéro. Seule l'action π_0 calculée par l'acteur est modifiée dans la direction opposée de l'action exécutée a_0.

Pour conclure, mentionnons qu'utilisées adéquatement, les traces d'éligibilité avec un λ proche de 1 rendent les méthodes TD(λ) moins sensibles aux effets non markoviens (Peng, 1994) et que l'utilisation de traces d'éligibilité pour l'acteur est aussi très utile pour appliquer les méthodes RL aux problèmes non markoviens (Kimura, 1998).

3.3.6 L'explorateur stochastique

Pour adresser le dilemme de l'équilibre exploration/exploitation, l'agent FNAC utilise la combinaison d'un explorateur à seuil avec un autre explorateur aléatoire uniforme. L'explorateur à seuil explore l'espace d'actions autour de l'action recommandée de façon uniforme dès que l'agent se trouve dans un état dont la valeur est en dessous d'un seuil prédéterminé. Ce type d'explorateur offre au moins deux avantages hormis une mise en œuvre plus simple. L'usage d'un seuil nous permet de contrôler la pertinence d'une exploration en mesurant la *distance* entre l'état actuel de l'agent et les états terminaux (Échec/Succès). Cette distance est ici définie en terme du nombre de *pas* (itérations) que l'agent mettra avant d'aboutir dans un état terminal. Si l'état terminal est un état d'échec, le seuil d'exploration doit être fixé de sorte que dès qu'il s'approche dangereusement de cet état, l'agent remet en question sa stratégie de contrôle et commence à explorer de nouvelles alternatives. Similairement, si l'état terminal est un état de succès, le seuil d'exploration doit être fixé de sorte que dès qu'il s'éloigne considérablement de cet état, l'agent remet en question sa stratégie de contrôle et commence à explorer de nouvelles alternatives.

Revenons à l'exemple 3.1. La valeur optimale d'un état s étant donnée par :

$$V^*(s) = \sum_{n=s}^{n=1} \gamma^{s-n} r(n,-1) = \gamma^{s-1} \tag{3.11}$$

sa distance optimale par rapport l'état terminal $s = 0$ est donnée par :

$$d = 1 + \frac{\ln(V^*(s))}{\ln(\gamma)} \tag{3.12}$$

Par conséquent, on doit déclencher l'exploration à chaque fois que la fonction d'évaluation est inférieure au seuil défini par :

$$V_{\min} \leq \tau = \begin{cases} \gamma^{d_{\max}-1} & \text{si l'état terminal est un succès} \\ -\gamma^{d_{\min}-1} & \text{si l'état terminal est un échec} \end{cases} \tag{3.13}$$

Cependant, la valeur d'un état peut être inférieure au seuil d'exploration alors que l'agent ne prenait que des actions optimales. Une telle situation arrive par exemple lorsque l'agent est très éloigné de son objectif. Dans ce cas, nous proposons de ne procéder à l'exploration que lorsqu'un signal de renforcement interne négatif est observé ; ce qui indique que l'agent ne suit plus un chemin optimal. D'autre part, la valeur d'un état peut être supérieure au seuil d'exploration sans qu'elle ne soit pour autant optimale. Dans ce cas, il se peut qu'il existe d'autres actions plus utiles qui permettent à l'agent de s'éloigner davantage du danger ou de s'approcher encore plus de son but. Pour de telles situations, nous proposons d'utiliser une stratégie de contrôle du type $\varepsilon - greedy$, c'est à dire une stratégie qui est optimale la plupart du temps, mais qui à l'occasion sélectionne ses actions aléatoirement avec une probabilité ε :

$$\pi^*_\varepsilon(s) = \begin{cases} \pi^*(s) & \text{avec une probabilité 1-}\varepsilon, \\ \textit{action aléatoire} & \text{avec une probabilité } \varepsilon. \end{cases} \tag{3.14}$$

L'usage d'une telle stratégie de contrôle permet de trouver des solutions plus robustes puisque l'exploration aléatoire peut être vue comme une perturbation aléatoire et non prévisible. On peut même lui passer la main dans les premiers épisodes de l'apprentissage en attendant que la fonction d'évaluation devienne plus riche en information. Par exemple,

on peut utiliser un ε variable et décroissant. Enfin, une telle stratégie permet également de minimiser les risques de convergence vers des solutions sous optimales. En effet, son effet perturbateur permet de mettre à l'épreuve la stabilité de l'optimum trouvé.

On peut aussi améliorer l'équilibre exploration/exploitation en jouant sur l'initialisation de la fonction d'évaluation. Par exemple, en commençant par un critique optimiste, où tous les états sont supposés être très bons au départ, on laisse le temps au critique d'apprendre la fonction d'évaluation puisque l'agent n'explore pas et l'acteur demeure inchangé. Au fur et à mesure que l'apprentissage progresse, l'agent reçoit des critiques négatives formées par le critique et active par conséquent le processus d'exploration. Le temps que l'acteur découvre les actions optimales, le critique aura déjà dévalorisé les valeurs de tous les états tellement la stratégie de contrôle est chaotique au départ. Il en résulte une période d'exploration intensive. Cependant, l'acteur s'améliorant constamment, les valeurs des états s'améliorent progressivement, et l'agent explore de moins en moins. L'exploitation le remporte progressivement sur l'exploration.

Les figures 3-11 et 3-12 résument les principales caractéristiques que nous proposons dans l'architecture FNAC.

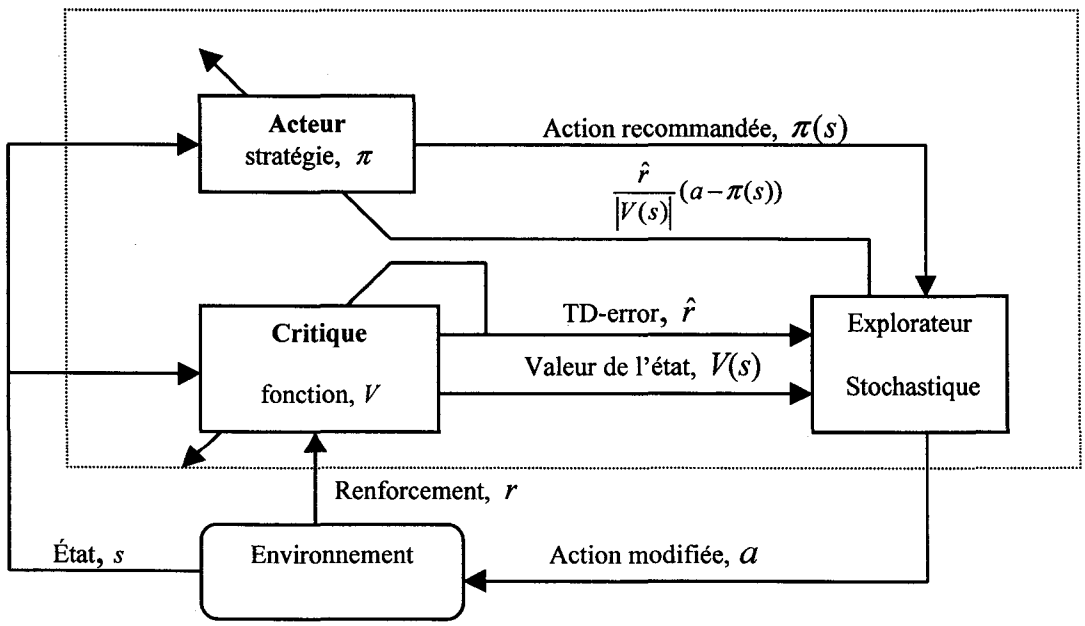


Figure 3-11 Architecture de l'agent FNAC.

Initialize the value function V and the policy π randomly for all states
Repeat (for each episode)
Initialize s
Repeat (for each step of episode)
Compute $\pi(s)$, $V(s)$
$a \leftarrow \pi(s)$
if $V(s) < \tau$, $a \leftarrow \min(\pi_{max}, \max(\pi_{min}, \pi(s) + (\pi_{max} - \pi_{min})rand)$
if $rand < \varepsilon$, $a \leftarrow \pi_{min} + (\pi_{max} - \pi_{min})rand$
for each state u,
$E_V(u) \leftarrow \gamma\lambda_V E_V(u)$
$E_\pi(u) \leftarrow \gamma\lambda_\pi E_\pi(u)$
for current state s,
$E_V(s) \leftarrow E_V(s) + 1$
Take action a, observe reward r, and next state s'
$\hat{r} = r + \gamma V(s') - V(s)$
if $a \neq \pi(s)$ and $\hat{r} < 0$,
Reset eligibility traces: $E_V(u) \leftarrow 0$, $E_\pi(u) \leftarrow 0$ for each state u
for current state s,
$E_\pi(s) \leftarrow E_\pi(s) + a - \pi(s)$
for each state u
$V(u) \leftarrow V(u) + \beta\hat{r}E_V(u)$
$\pi(u) \leftarrow \pi(u) + \alpha\frac{\hat{r}}{\varsigma + |V(s)|}E_\pi(u)$
$s \leftarrow s'$
Until current state s is terminal.

Figure 3-12 Algorithme générique de l'agent FNAC.

3.4 Implémentation neuro-floue de l'agent FNAC

Les deux éléments adaptatifs de l'agent FNAC, à savoir l'acteur et le critique, peuvent être implémentés à l'aide des SIF de type TS. Deux choix s'offrent alors au concepteur. Le premier choix consiste à opter pour une implémentation séparée de chacun des deux éléments par un SIF différent. L'inconvénient d'une telle approche est un usage abusif des ressources matérielles pour le stockage des paramètres des deux SIF et une redondance inutile du calcul de la partition de l'espace d'états. Par contre, une telle approche présente l'avantage d'éliminer les interférences pouvant résulter lorsque les gradients de l'acteur et

du critique ne tirent pas dans la même direction ou ne tirent pas avec la même vigueur. En effet, d'après l'équation (2.38), le gradient dans la couche 2 (couche de fuzzification) du SIF est donné par la somme pondérée des gradients associés aux différentes sorties du SIF. Un autre avantage de cette approche réside dans le fait qu'on n'est pas obligé d'avoir la même complexité de structure pour les deux éléments.

L'autre choix s'offrant au concepteur consiste à utiliser un seul SIF, à plusieurs sorties, aussi bien pour implémenter l'acteur que pour implémenter le critique. L'avantage d'une telle approche est une utilisation minimale des ressources matérielles (mémoires, processeurs). Par contre, le réglage des paramètres communs aux deux éléments adaptatifs de l'agent devient nettement plus difficile à cause notamment des interférences causées par l'utilisation simultanée de deux gradients de natures très différentes. De telles interférences peuvent arriver lorsque la minimisation des erreurs d'approximation de l'acteur et du critique implique deux partitions différentes de l'espace d'états. Elles auront donc lieu plus particulièrement au niveau de l'ajustement des paramètres des fonctions d'appartenance du SIF.

En réalité, on peut minimiser l'effet des interférences dues à un apprentissage simultané de la fonction d'évaluation et de la stratégie de contrôle moyennant quelques heuristiques simples. D'une part, les amplitudes des erreurs de prédiction à la sortie des différents neurones de la couche de sortie du SIF doivent être normalisées avant d'être rétro-propagées aux couches cachées. On s'assure ainsi d'avoir un gradient à la sortie de la couche de fuzzification (couche 2) qui ne fait aucune discrimination entre les différentes sorties du SIF. D'autre part, et en examinant le processus d'apprentissage, on peut jouer sur les taux d'apprentissage (normalisation) associés au critique et à l'acteur de sorte qu'on favorise l'un des deux dépendamment de la progression de l'apprentissage. Par exemple, durant les premiers essais, l'accent devrait être surtout mis sur l'estimation de la fonction d'évaluation sur laquelle se basera l'acteur pour renforcer et améliorer sa stratégie de contrôle. Néanmoins, au fur et à mesure que l'apprentissage progresse, l'agent doit se concentrer davantage sur sa stratégie de contrôle.

Finalement, rappelons que dans le cadre d'un problème RL, il n'est pas nécessaire d'identifier avec précision la fonction d'évaluation mais simplement de lui trouver un estimé qui ordonne l'espace d'états de la même manière. D'après Shannon (1950) et Samuel (1959), la fonction d'évaluation peut toujours être adéquatement estimée par un approximateur linéaire. Par conséquent, l'apprentissage en ligne des fonctions d'appartenance n'est pas toujours nécessaire. Dans ce cas, on peut toujours opter pour cette implémentation unique en se contentant de modifier uniquement les décisions locales (conclusions) du SIF, tout en s'assurant que les fonctions d'appartenance couvrent adéquatement l'ensemble de l'espace d'états. L'avantage d'une telle approche est que l'on dispose d'un approximateur universel linéaire et par conséquent entraînable, avec garantie de convergence, par l'algorithme TD(λ). En plus, cette implémentation est nettement plus facile à mettre en œuvre puisque ne nécessitant pas le recours à l'algorithme de rétro-propagation. Cependant, la solution obtenue par un tel SIF risque d'être sous-optimale en plus d'avoir de faibles capacités de généralisation. En fait, une telle solution est exactement équivalente à l'utilisation de deux SIF différents pour représenter l'acteur et le critique avec cependant la contrainte que l'espace d'états doit être partitionné exactement de la même façon par les deux SIF.

En résumé, le choix de l'architecture d'implémentation de l'agent FNAC doit se faire en fonction du problème à résoudre:

- Pour les problèmes de petite taille (dimension de l'espace d'états petite), on peut utiliser deux SIF différents pour implémenter chacun de l'acteur et du critique;

- Pour les problèmes de taille plus importante, on conseille d'utiliser un SIF unique à plusieurs sorties. Dépendamment de la complexité de la tâche de l'agent, on optera pour un apprentissage ou pour un réglage manuel des fonctions d'appartenance. Pour les tâches aisées, on peut se permettre le luxe de chercher une stratégie de contrôle la plus optimale que possible en optant pour l'apprentissage des paramètres des fonctions d'appartenance. Tandis que pour les tâches plus difficiles, on optera plutôt pour un réglage manuel des fonctions d'appartenance (e. g. distribution uniforme). Dans ce cas, on sacrifie l'optimalité de la stratégie de contrôle apprise au

profit de la vitesse de convergence, voire de la convergence tout simplement, de l'apprentissage.

Dans tous ce qui suit, et sauf mention du contraire, nous opterons pour le deuxième choix, *i.e.* l'acteur et le critique sont représentés par le même SIF. Nous ferons également l'hypothèse qu'il existe une seule variable de contrôle.

3.4.1 Détails de l'implémentation

3.4.1.1 Entrées et sorties

Pour compléter l'implémentation neuro-floue de l'agent FNAC, on doit définir les entrées, les sorties, et les sorties désirées du SIF :

Le vecteur d'entrée est formé par les valeurs des différentes variables d'état à un instant donné. Sa taille est donc égale à la dimension de l'espace d'états.

Le SIF possède deux sorties : l'action π calculée par l'acteur et la valeur $V(s)$ de l'état actuel de l'environnement calculée par le critique.

Pour sa mise à jour, le SIF reçoit d'une part l'estimé $r+\gamma V(s')$, plus précise que $V(s)$ étant donné qu'il incorpore la vraie valeur r du signal de renforcement, de la valeur de l'état actuel de l'environnement. La différence entre ces deux valeurs, $\hat{r}=r+\gamma V(s')-V(s)$, sera utilisée pour la mise à jour du SIF. D'autre part, le SIF reçoit la valeur de l'action a effectivement exécutée par l'agent (action recommandée ou explorée). L'erreur, $a-\pi$, associée à cette sortie sera modulée par la valeur $\hat{r}$ du signal de renforcement interne, afin que le SIF se corrige dans la direction de l'action a si elle est meilleure ($\hat{r}\geq 0$) ou dans la direction opposée, $2\pi-a$, sinon ($\hat{r}\leq 0$).

La figure 3-13 représente le schéma fonctionnel du FNAC.

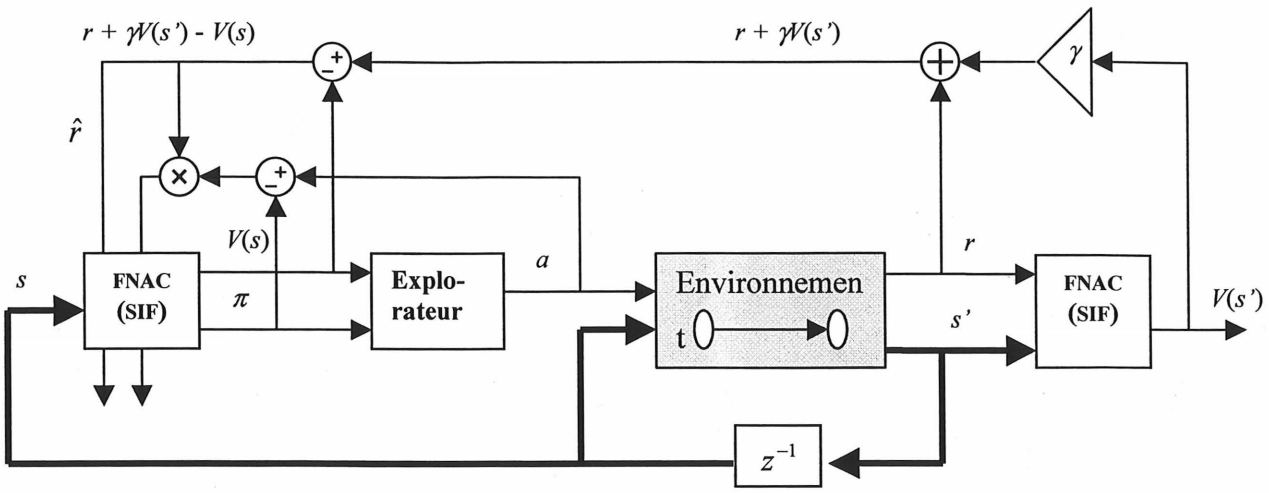


Figure 3-13 Implémentation neuro-floue de l'agent FNAC.

3.4.1.2 Traces d'éligibilité

Les traces d'éligibilité (ET) que nous avons introduites auparavant sont associées aux états discrets de l'espace d'états. Dans le cas des SIF, nous allons plutôt les associer aux règles floues. Cependant, on doit tenir compte du recouvrement de ces règles. En effet, à chaque mise à jour du SIF, plusieurs règles floues peuvent être actives simultanément et doivent de ce fait être considérées dans la présente mise à jour (systématiquement assuré par l'algorithme du gradient descendant (Éq. 2-33)), mais aussi dans les prochaines mises à jour en utilisant leur combinaison ET. D'après les équations (2.14), (3.9) et (2.33), nous pouvons définir les ET cumulatives (AET) de la manière suivante (Bradtke, 1996):

$$\sum\nolimits_V (R^r) \leftarrow \gamma\lambda_v \sum\nolimits_V (R^r) + \hat{\mu}_r \tag{3.15}$$

$$\sum\nolimits_\pi (R^r) \leftarrow \gamma\lambda_\pi \sum\nolimits_\pi (R^r) + \hat{\mu}_r (a - \pi) \tag{3.16}$$

où $\sum_V (R^r)$ et $\sum_\pi (R^r)$ sont respectivement les ET du critique et de l'acteur associés à la règle floue R^r dont le degré d'activation est $\hat{\mu}_r$. Elles utilisent respectivement les facteurs d'oubli λ_V et λ_π. Dans le cas où il n'existe aucun recouvrement entre les règles floues,

c'est-à-dire lorsqu'une seule règle est active à la fois, les équations (3.15) et (3.16) sont exactement équivalentes aux définitions des AET discrètes données par les équations (2.14) et (3.9) respectivement.

Par contre, nous allons introduire pour la première fois des ET substitutives (RET) associées au SIF. Rappelons que le principe des traces d'éligibilité est que pour un facteur d'oubli $\lambda = 0$, la mise à jour n'affecte que le dernier état visité, alors que pour un facteur d'oubli $\lambda = 1$, la mise à jour affecte tous les états récemment visités et devrait en principe conduire l'agent à apprendre plus rapidement. Lorsqu'un état est fréquemment visité pendant un essai d'apprentissage, l'usage des AET peut provoquer une sur-correction de sa valeur. Dans le cas des SIF, le problème peut être encore plus sévère puisque chaque règle floue peut rester active assez longtemps pour voir exploser son ET. Elle peut rester active parce que l'agent tourne en rond ou tout simplement parce que la partition de l'espace d'états est tellement minimale que l'agent met du temps avant de quitter l'hyper cube associé à une règle donnée, i.e. où la règle est active.

L'idée des RET, introduite par Singh et Sutton (Singh, 1996), est justement d'éviter de trop se rappeler (particulièrement quand λ tend vers 1) des états (règles) fréquemment visités en réinitialisant leur ET (à 1 d'après l'équation (2.15)). Cette approche est facile à mettre en œuvre si l'on utilise des tables de conversion ou des approximateurs linéaires avec des caractéristiques binaires telle que la méthode des BOITES (Barto, 1983). Le problème avec les SIF est qu'on ne sait pas à quelle valeur il faut réinitialiser les ET. Sutton (1998, pages 202-205) propose une méthode appelée la méthode du codage brut et dont le principe consiste à traiter toutes les règles active comme étant *également* active, i.e. elles ont toutes accès à la totalité du signal de renforcement interne. Dans une telle perspective, les RET seront définies par :

$$\sum_V(R^r) \leftarrow \begin{cases} 1 & \text{si } \hat{\mu}_r > 0 \\ \gamma\lambda_v \sum_V(R^r) & \text{sinon} \end{cases}$$

Nous reprochons à cette approche de se limiter à l'usage du signe du gradient. En ce qui nous concerne, nous proposons de comparer la ET de chaque règle avec son degré

d'activation. Si ce dernier est plus grand, on l'utilise pour réinitialiser la ET. Sinon, la ET décroît exponentiellement. D'où :

$$\sum_V(R^r) \leftarrow \max\left(\gamma\lambda_V \sum_V(R^r), \hat{\mu}_r\right) \tag{3.17}$$

Dans le cas où il n'existe aucun recouvrement entre les règles floues, c'est à dire lorsqu'une seule règle est active à la fois, l'équation (3.17) est exactement équivalente à la définition des RET discrètes données par l'équation (2.15) des RET.

3.5 Application : Problème du pendule inverse

Pour illustrer et comparer les performances de l'agent FNAC, nous allons l'appliquer pour résoudre le problème du pendule inverse qui est un problème typique utilisé pour tester des méthodes d'apprentissage. Nous l'utiliserons afin d'illustrer les performances de l'agent FNAC en les comparant avec celles des méthodes RL voisines.

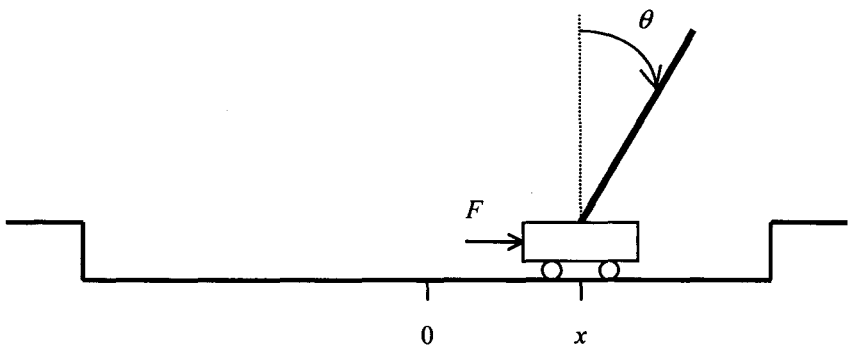


Figure 3-14 Schéma du pendule inverse.

3.5.1 Description du système

Dans ce problème (Figure 3-14), un pendule rigide est fixé sur un chariot mobile. Le chariot peut se déplacer uniquement de façon rectiligne le long d'une paire de rails à

longueur finie. Quant à lui, le mouvement du pendule est limité au plan vertical défini par le pendule et les rails. La seule variable de contrôle est la force F appliquée sur le chariot.

Le modèle complet du système est décrit par 4 variables d'état, ainsi qu'une variable de contrôle:

- θ : l'angle formé par le pendule et l'axe vertical;
- $\dot{\theta}$: la vitesse angulaire du pendule;
- x : la position du chariot sur les rails;
- $\dot{x}$: la vitesse de déplacement du chariot;
- F : la force appliquée sur le chariot.

La dynamique de ce système est modélisée par les équations différentielles suivantes (Barto, 1983):

$$\ddot{\theta} = \frac{g\sin(\theta) + \cos(\theta)\left[\frac{-F - ml\dot{\theta}^2\sin(\theta) + \mu_c sgn(\dot{x})}{m_c + m}\right] - \frac{\mu_p \dot{\theta}}{ml}}{l\left[\frac{4}{3} - \frac{m\cos^2(\theta)}{m_c + m}\right]} \tag{3.18}$$

$$\ddot{x} = \frac{F + ml\left[\dot{\theta}^2\sin(\theta) - \ddot{\theta}\cos(\theta)\right] - \mu_c sgn(\dot{x})}{m_c + m} \tag{3.19}$$

où g représente l'accélération due à la force de gravité ($g = -9.8\,\mathrm{m/s^2}$), m_c la masse du chariot ($m_c = 1.0\,\mathrm{kg}$), m la masse du pendule $m = 0.1\,\mathrm{kg}$, l la demi longueur du pendule ($l = 0.5\,\mathrm{m}$), μ_c le coefficient de friction du chariot sur les rails ($\mu_c = 0.0005$) et enfin μ_p le coefficient de friction du pendule sur le chariot ($\mu_p = 0.000002$). Ces équations

différentielles, reprises intégralement de (Barto, 83) et simulées par le biais de la méthode d'Euler avec un pas de calcul de 20 ms, ne sont, bien entendu, pas connues de l'agent.

Le but de l'agent consiste à maintenir le pendule en équilibre le plus longtemps possible tout en s'assurant que le chariot ne dépasse pas la limite des rails. Plus précisément, on se fixe comme objectif de maintenir l'angle θ dans l'intervalle [-12°, 12°] et la position du chariot dans l'intervalle [-2-4 m, 2-4 m] pendant au moins 100000 pas de calcul. La force F doit rester inférieure à 10 N.

La fonction de renforcement consiste uniquement à punir l'agent en cas d'échec :

$$r = \begin{cases} 0 & \text{si } |\theta| \leq 12^\circ \text{ et } |x| \leq 2.4 \text{ m} \\ -1 & \text{sinon} \end{cases} \tag{3.20}$$

3.5.2 Étude comparative des performances de l'agent FNAC

Dans cette section, nous allons analyser les performances de l'agent sous différents angles en respectant les mêmes protocoles de tests utilisés par les pionniers du domaine (Sutton, 1998 et Randlov, 1998). Ces protocoles se basent sur les définitions suivantes :

- Essai : il correspond au temps écoulé avant que l'agent ne se trouve dans un état terminal. Dans ce cas-ci, les états terminaux correspondent à ceux où l'angle du pendule dépasse les 12 degrés ou la position du chariot dépasse les 2.4 mètres.

- Épisode : nombre d'essais nécessaires à l'agent pour réussir sa mission. Dans ce cas-ci, la mission de l'agent consiste à garder le système en équilibre pendant 100000 itérations.

- Indice de performance : pour mesurer les performances d'un agent RL, il est commun d'observer le nombre moyen d'essais nécessaires à l'agent pour apprendre une stratégie de contrôle satisfaisante. Dans ce cas-ci, on effectue la moyenne sur 30 épisodes. Il existe une certaine pratique (Lin, 1994) qui impose une limite maximale du nombre

d'essais par épisode. Au delà de cette limite, Lin considère que l'agent a échoué dans sa mission, et l'épisode n'est pas pris en considération dans le calcul de la moyenne. Une telle pratique pourrait cacher des problèmes de convergence car seuls les épisodes où l'agent a été 'chanceux', dans ses explorations, seront retenus. Dans notre étude, cette pratique n'est pas utilisée : l'agent doit réussir chaque épisode. En effectuant une moyenne sur 30 épisodes, notre indice de performance est nettement moins biaisé.

- La courbe de performance : elle donne la variation de l'indice de performance en fonction du facteur d'oubli λ dans les algorithmes TD(λ).

- La courbe d'apprentissage : elle permet de visualiser la durée des essais pendant un épisode d'apprentissage. On en tracera aussi une version moyennée sur 10 épisodes. L'avantage de cette dernière est qu'elle donne une mesure plus précise de la vitesse d'apprentissage puisqu'elle tient compte de la durée des essais.

Sauf mention du contraire, les conditions initiales du système seront nulles, l'acteur initialisé aléatoirement, et le critique initialisé de façon optimiste : la fonction d'évaluation initiale est maximale partout.

3.5.2.1 Courbes de performance

Dans un premier temps, nous avons comparé la courbe de performance de l'agent FNAC avec celle d'un agent AC classique qui lui est identique mais dont le critique n'est pas vigilent et l'acteur n'est pas équitable (Fig. 3-11). Les paramètres d'apprentissage sont identiques dans les deux cas : $\gamma = 0.95$, $\alpha = 5$, $\beta = 0.35$, $\tau = -0.1$, $\lambda_\pi = 0$, et $\varepsilon = 0$. Les traces d'éligibilité sont de type cumulatives. Les résultats sont présentés sur les figures 3-15 et 3-16 où on a présenté les courbes de performance (moyennées sur 30 épisodes) des deux agents. Plus la courbe d'apprentissage est basse, plus vite est l'apprentissage de l'agent.

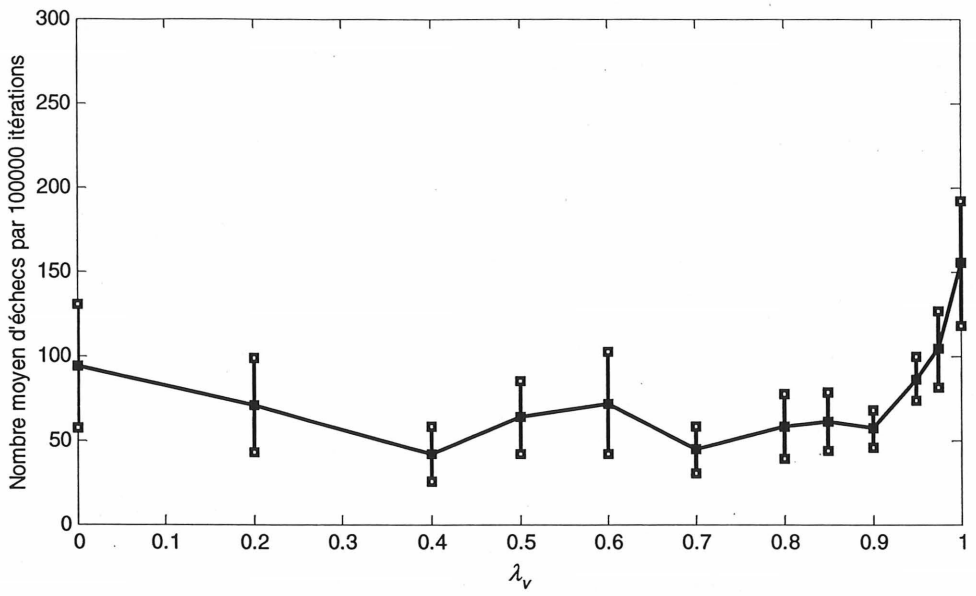


Figure 3-15 Courbes de performance de l'agent FNAC. Les traits verticaux indiquent un écart type.

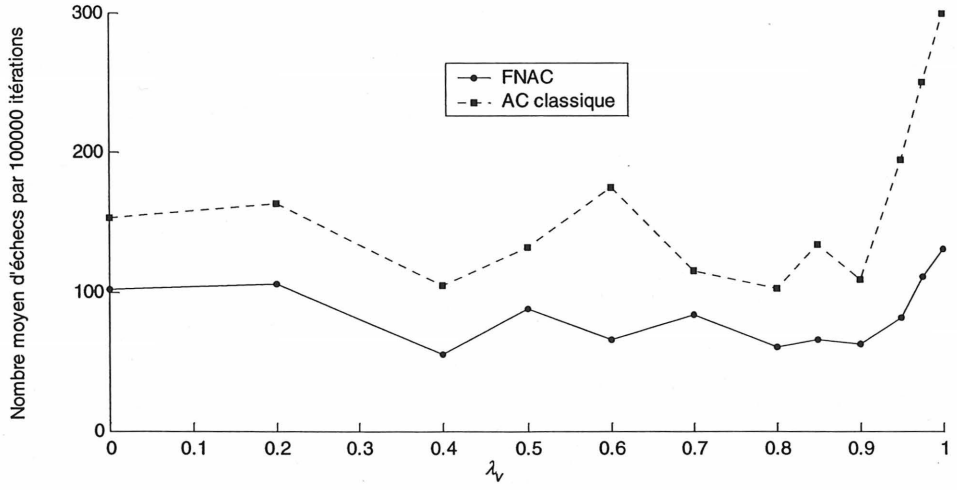


Figure 3-16 Courbes de performance de l'agent FNAC (trait plein) et d'un agent AC classique (trait pointillé).

Ces résultats démontrent clairement la supériorité des performances de l'agent FNAC validant ainsi la contribution des améliorations proposées à savoir la consistance entre l'acteur et le critique et renforcement équitable des actions explorées. Finalement, mentionnons que les performances de l'agent FNAC se sont aussi avérées supérieures à

celles rapportées dans (Sutton, 1998) où une architecture SARSA (une méthode QL) fut utilisée et à celles rapportées dans (Berenji, 1992) où une architecture GARIC fut utilisée.

Nous avons également comparé la courbe de performance d'un agent FNAC n'utilisant pas de traces d'éligibilité de l'acteur avec celle d'un autre agent FNAC qui en utilise. Les paramètres d'apprentissage sont sensiblement les même que ceux donnés plut haut. Les résultats sont présentés sur la figure 3-17. Dans la majorité des cas, l'agent apprend plus vite en utilisant des traces d'éligibilité pour l'apprentissage de la stratégie de contrôle. Ces résultats démontrent clairement le gain apporté par l'utilisation des traces d'éligibilité que nous avons proposées pour l'apprentissage de l'acteur. On remarque, par exemple, que pour un facteur d'oubli $\lambda_v = \lambda_\pi = 0.7$, il suffit en moyenne moins de 25 essais pour que l'agent apprenne à accomplir sa tâche correctement.

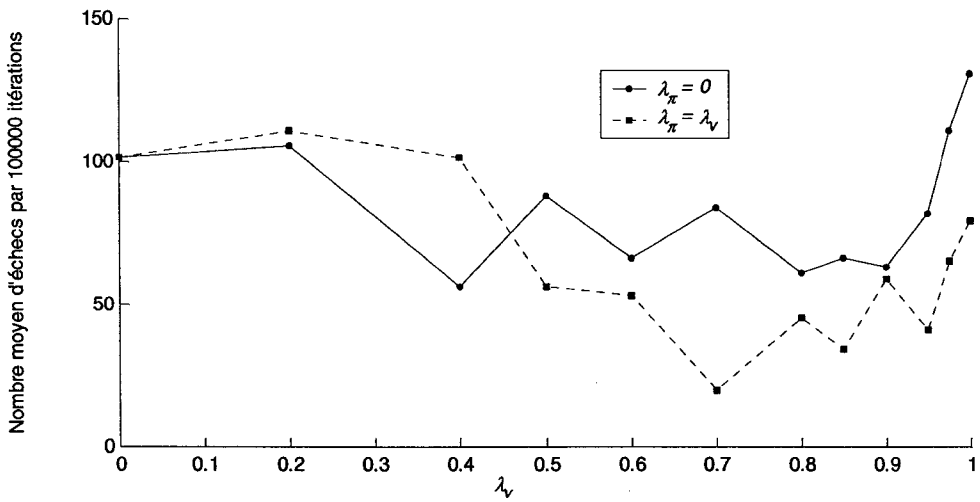


Figure 3-17 Effet des traces d'éligibilité de l'acteur sur la courbe de performance de l'agent FNAC.

3.5.2.2 Courbes d'apprentissage

Le but des courbes d'apprentissage est d'observer, en la visualisant, la progression de l'apprentissage pendant un épisode. On peut les utiliser pour mieux estimer la vitesse de convergence des algorithmes d'apprentissage car elles tiennent compte de la longueur des essais. On représente sur ces courbes l'évolution de la durée des essais pendant un épisode

d'apprentissage. Ces courbes ont été générées en utilisant les paramètres d'apprentissage suivants : $\gamma = 0.95$, $\alpha = 5$, $\beta = 0.35$, $\tau = -0.1$, $\lambda_\pi = \lambda_V = 0.7$, et $\varepsilon = 0.01$.

Sur la courbe droite de la figure 3-18, nous avons représenté deux courbes d'apprentissage typiques d'un agent FNAC appliqué au problème du pendule inverse. La courbe de droite représente la courbe d'apprentissage pour un seul épisode, tandis que la courbe de gauche représente la courbe d'apprentissage moyenne sur 10 épisodes.

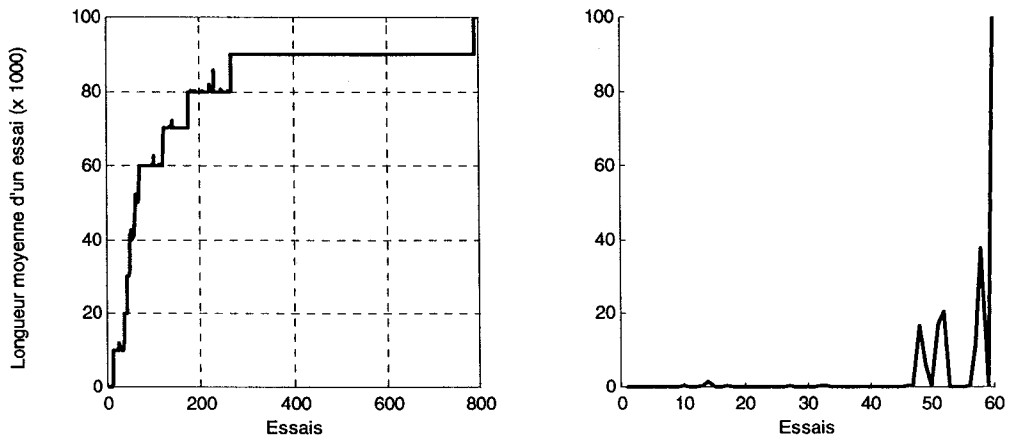


Figure 3-18 Courbes d'apprentissage de l'agent FNAC. (a) représente la moyenne sur 10 épisodes des courbes d'apprentissage dont une courbe typique est représentée sur la courbe (b).

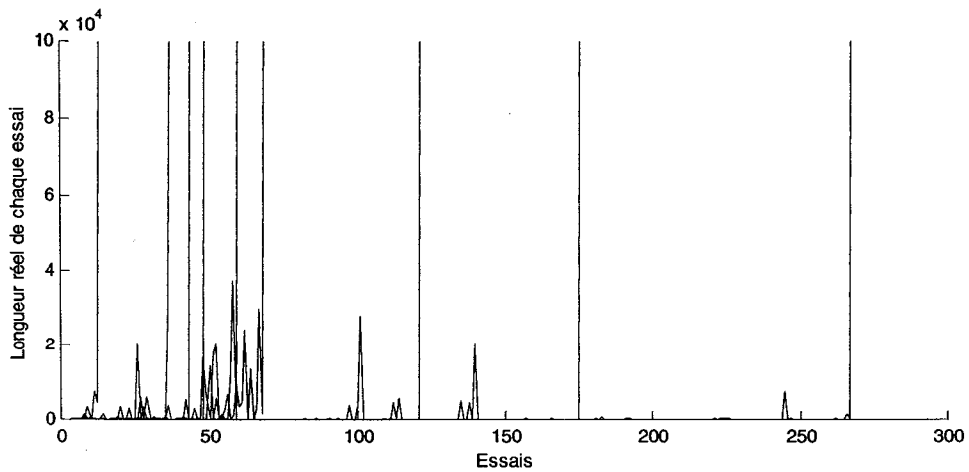


Figure 3-19 Courbes d'apprentissage de l'agent FNAC.

Sur la figure 3-18 sont superposées les courbes d'apprentissage de l'agent pendant les 10 épisodes sur lesquels nous avons effectué la moyenne de la figure 3-18. On voit que la longueur des épisodes peut varier considérablement. Le but de la présentation de ces courbes est de montrer à quel point la pratique soulevée auparavant, et qui consiste à ne considérer que les épisodes dont la longueur est inférieure à une certaine valeur (50 essais/épisodes dans le cas de (Lin, 1994)) peut amplifier les performances d'un agent. Intuitivement, on aurait pu s'attendre à des courbes d'apprentissage croissantes (Fig. 3-18, courbe de droite) en supposant que l'agent s'améliore constamment essai après essai. Cependant, ceci n'est pas le cas pour plusieurs raisons. D'une part, les changements constants de la fonction d'évaluation pour suivre la stratégie de contrôle et de cette dernière, qui à son tour tente constamment de maximiser la première, font en sorte que la trajectoire dans l'espace d'états varie tout aussi fréquemment. Par conséquent, d'un essai à l'autre, l'agent s'améliore mais peut être confronté à des situations qu'il n'a jamais visitées ou où il n'a pas encore appris comment agir adéquatement. Finalement, l'équilibre exploitation/exploration implique que la stratégie de contrôle suivie par l'agent est stochastique, même si celle de l'acteur est déterministe. On peut notamment attribuer l'échec d'un essais après une durée relativement importante à l'utilisation d'une stratégie de type quasi glouton (*ε-greedy*).

Sur la courbe gauche de la figure 3-18, nous avons représenté la durée moyenne (sur 10 épisodes) des essais. Pour effectuer la moyenne, l'agent est réinitialisé au début de chaque épisode. On peut lire, par exemple, sur cette courbe qu'après 400 essais, l'agent apprend en moyenne à maintenir l'équilibre du pendule pendant près de 30 minutes (90000 itérations). On peut y lire également qu'en moyenne l'agent s'améliore constamment après chaque essai.

3.5.2.3 Effet des méthodes d'exploration stochastique

Lors de notre étude des méthodes d'exploration, nous avons vu qu'il existe deux types de méthodes d'exploration : celles basées sur la fonction d'évaluation et celles purement aléatoires. Si l'utilité des premières n'est plus à démontrer particulièrement dans les méthodes AC, il est légitime de se poser des questions sur la pertinence des méthodes

d'exploration purement aléatoires. Dans ce qui suit, nous allons analyser l'utilité de deux de ces méthodes : l'exploration par l'usage d'une stratégie de contrôle quasi glouton (*ε-greedy*) et l'exploration par variations des conditions initiales.

3.5.2.3.1 Agent glouton ou quasi-glouton ?

Sur la figure 3-20, nous avons représenté les courbes d'apprentissage de deux agents : un agent glouton ($\varepsilon = 0$; courbe de gauche) et un agent quasi-glouton ($\varepsilon = 0.01$; courbe de droite). Nous remarquons que l'agent glouton apprend à accomplir sa tâche nettement plus vite. La raison est probablement due au fait que dans le cas de l'agent quasi-glouton, chaque exploration non basée sur la valeur d'un état provoque un changement de trajectoire. L'agent quasi glouton doit apprendre à agir correctement sur une plus grande portion de l'espace d'états. Cependant, on peut s'attendre à ce que la stratégie de contrôle apprise dans ce cas soit plus générale.

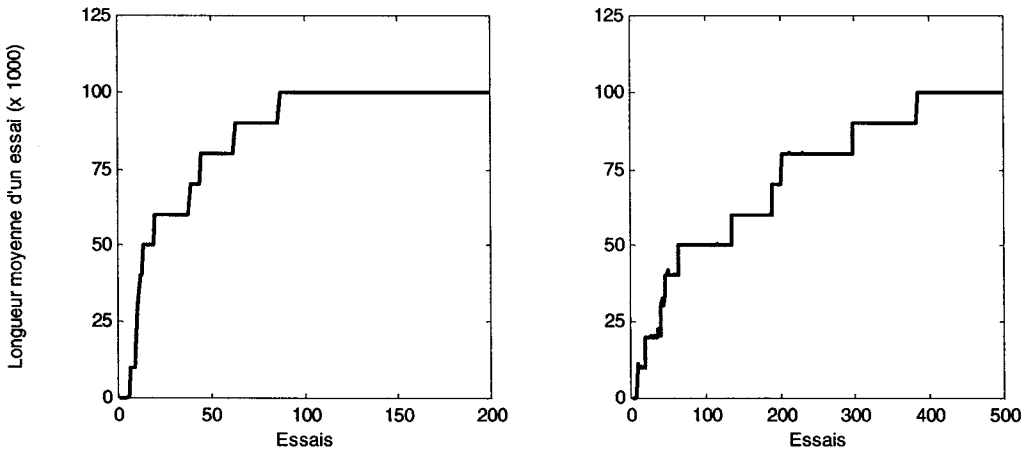
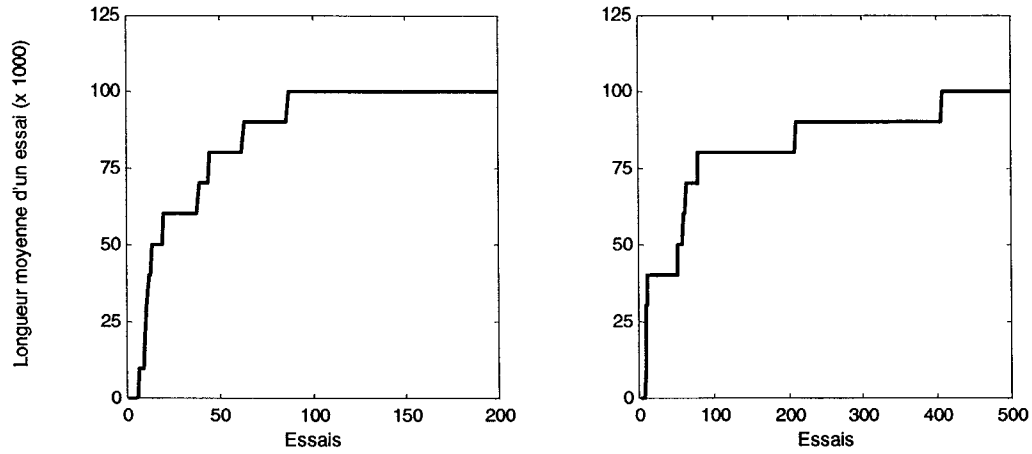


Figure 3-20 Courbes d'apprentissage. (a) stratégie glouton et (b) stratégie quasi glouton ($\varepsilon = 0.01$).

3.5.2.3.2 Effet des conditions initiales :

Sur la figure 3-21, nous avons représenté les courbes d'apprentissage de deux agents gloutons : un agent qui apprend toujours à partir de conditions initiales nulles (angle du pendule et position du chariot nuls, courbe de gauche) et un agent qui apprend à



(a) (b)

Figure 3-21 Effet des conditions initiales sur les courbes d'apprentissage. (a) conditions initiales nulles et (b) conditions initiales aléatoires.

partir de conditions initiales aléatoires (courbe de droite). On remarque que les performances du premier agent sont largement supérieures. On peut justifier ce résultat de deux manières. D'une part, la tâche du premier agent est plus facile vu qu'il lui suffit d'apprendre la bonne stratégie du contrôle sur une partie réduite de l'espace d'états : les trajectoires de l'agent commencent toujours au même point et ne se séparent que lorsque le processus d'exploration est déclenché. D'autre part, cet état étant le meilleur état possible, l'agent aurait eu l'occasion de le visiter au moins une fois. D'ailleurs, des résultats similaires ont été rapportées dans (Lin, 1994).

Cependant, nous avons remarqué que la rapidité d'apprentissage du premier agent n'est pas nécessairement attribuable aux conditions initiales nulles, mais plutôt au fait qu'elles soient fixes. Pour le démontrer, nous avons comparé les courbes d'apprentissage de deux agents apprenant tous les deux à partir de conditions initiales fixes : nulles pour le premier et aléatoires (mais fixes pendant toutes la durée d'un épisode) pour le deuxième. Les résultats sont illustrés sur la figure 3-22. On remarque que les performances sont sensiblement identiques dans les deux cas.

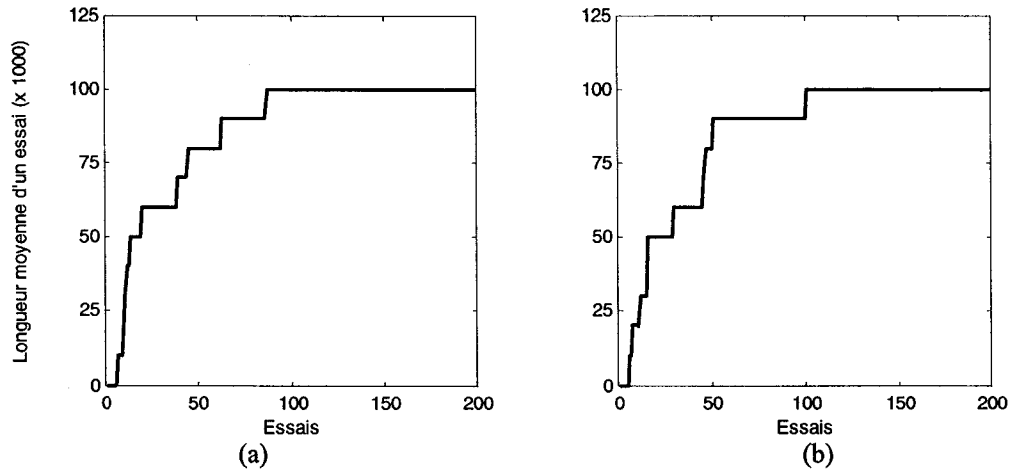


Figure 3-22 Courbes d'apprentissage de l'agent FNAC pour des conditions initiales fixes. (a) conditions initiales nulles et (b) conditions initiales aléatoires.

Cela dit, on remarque que les courbes d'apprentissage sont très semblables dans le cas d'un agent apprenant à partir de conditions initiales aléatoires (courbe de droite, figure 3- 20) et un agent quasi glouton (courbe de droite, figure 3- 19). On peut donc affirmer que l'utilisation de conditions initiales aléatoires peut être considérée dans le cadre d'une bonne stratégie d'équilibre exploitation/exploration. Au mieux de notre connaissance, nous sommes les premiers à établir une telle équivalence. Notons cependant que la méthode *Monte Carlo with Exploring Starts* (voir Sutton, 1998, pages 119-120) requiert aussi, et montre l'importance, l'utilisation de conditions initiales aléatoires au début de chaque essai. Cependant, elle le fait dans le but de calculer le retour moyen (le cumulatif des renforcement externe) de chaque pair (état, action) sur une longue séquence de pas de calcul. Dans notre cas, l'usage des conditions initiales aléatoires vise essentiellement à emmener l'agent à visiter des régions de l'espace d'état que son mécanisme d'exploration interne n'a pas réussi à l'y conduire. À vrai dire, on peut se contenter d'utiliser des conditions initiales variables par exemple en utilisant les centres des fonctions d'appartenance.

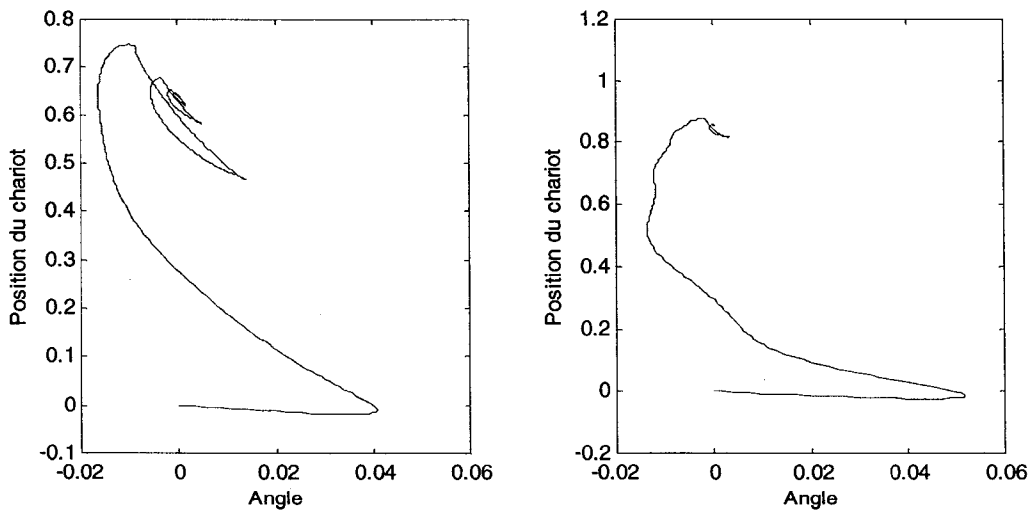
On peut aussi conclure que l'exploration basée uniquement sur les fonctions d'évaluation est non seulement suffisante mais qu'elle permet un apprentissage plus rapide. Par conséquent, on peut se demander quel intérêt peut-on avoir à utiliser les autres méthodes

d'exploration purement aléatoires. Dans ce qui suit, on tentera de répondre à cette question. La clé de notre investigation sera basée sur l'hypothèse que l'usage modéré des méthodes d'exploration purement aléatoires permet à l'agent d'apprendre des stratégies de contrôle plus générales et plus robustes.

3.5.2.3.3 Analyse d'un épisode d'apprentissage

Dans cette section, nous allons observer et comparer, après un seul épisode d'apprentissage, la globalité des solutions apprises par les deux agents : un agent glouton ($\varepsilon = 0$) et un agent quasi glouton ($\varepsilon = 0.01$). Les paramètres d'apprentissage sont : $\gamma = 0.95$, $\alpha = 5$, $\beta = 0.35$, $\tau = -0.1$, $\lambda_V = 0.8$ et $\lambda_\pi = 0$. Dans les deux cas, les conditions initiales sont nulles pendant la phase d'apprentissage.

Comme nous l'avons déjà constaté, l'agent glouton apprend plus rapidement. Dans la présente simulation, l'agent glouton a réussi sa mission après seulement 17 essais comparativement à 38 essais pour l'agent quasi glouton. Les deux agents réussissent également leurs missions même en absence d'apprentissage.



(a) (b)

Figure 3-23 Trajectoires des deux agents en mode exploitation. (a) un agent glouton et (b) un agent quasi glouton ($\varepsilon = 0.01$).

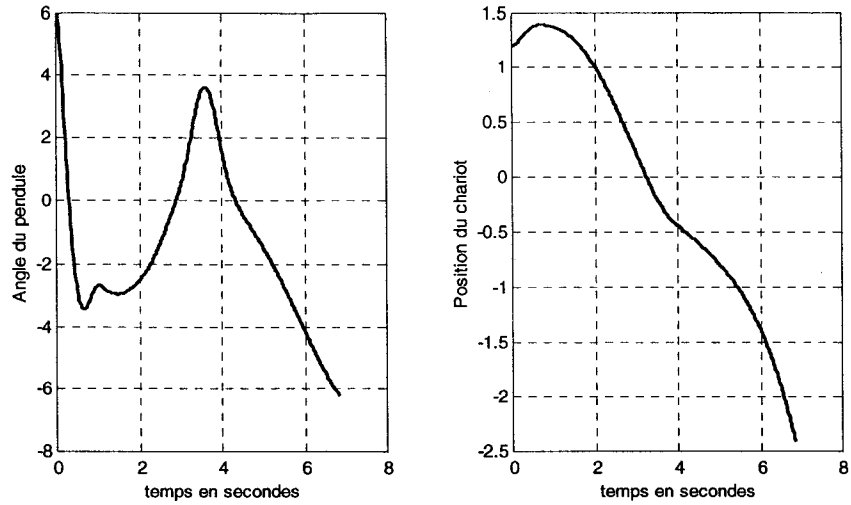


Figure 3-24 Variations des variables d'état d'un agent glouton en mode exploitation. Les conditions initiales sont différentes de celles utilisées pendant l'apprentissage. L'agent a échoué.

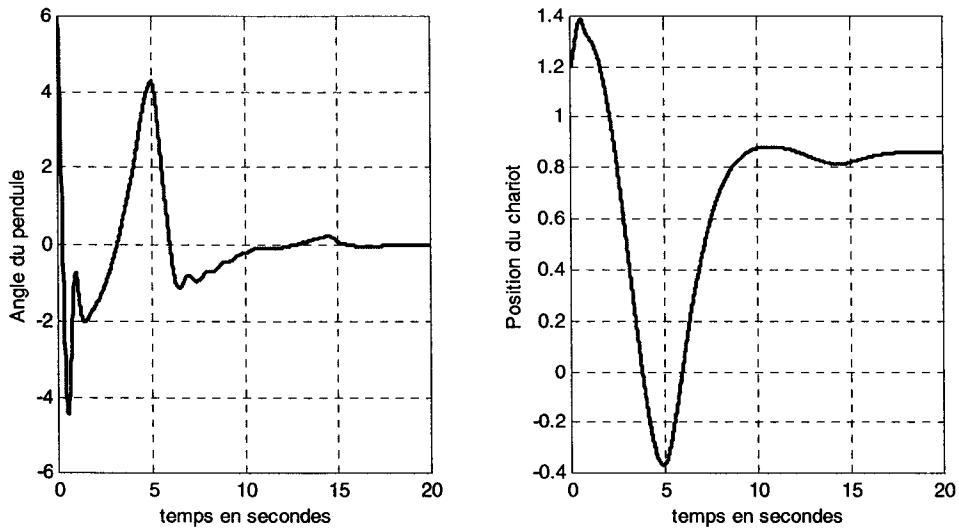


Figure 3-25 Variations des variables d'état d'un agent quasi-glouton en mode exploitation. Les conditions initiales sont différentes de celles utilisées pendant l'apprentissage.

Sur la figure 3-23, nous avons représenté leurs trajectoires pendant une phase test (les conditions initiales étant toujours nulles) où ils fonctionnent en mode exploitation seulement : les stratégies de contrôle restent fixes et sont suivies strictement par les agents. On peut remarquer que les deux agents accomplissent bien leurs missions. Les résultats de ce test sont présentés sur les figures 3-24 et 3-25. On remarque que l'agent glouton (Fig. 3-

24) a échoué, le chariot heurté la limite des rails. Par conséquent, la stratégie de contrôle qu'il a appris pendant la phase d'apprentissage est limitée à une partie de l'espace d'états. Par contre, l'agent quasi-glouton (Fig. 3-34) réussit à accomplir sa tâche quelques soient les conditions initiales. Sa stratégie de contrôle est donc plus générale.

Cependant, nous avons voulu vérifier notre hypothèse qu'un agent quasi glouton ce dernier possède de meilleures capacités de généralisation. Nous avons alors décidé de mettre les deux agents à l'épreuve en changeant les conditions initiales. Pendant cet essai, les deux agents fonctionnent en mode exploitation seulement. Les conditions initiales que nous avons utilisées sont les suivantes : $\theta = 6°$, $\dot{\theta} = 0$, $x = 1.2\,\text{m}$ et $\dot{x} = 0$. Cette hypothèse, que nous avons déjà avancée plus haut, est confirmée par la figure 3-26 (courbe (a)) où nous avons tracé l'ensemble des états visités pendant tout l'épisode d'apprentissage.

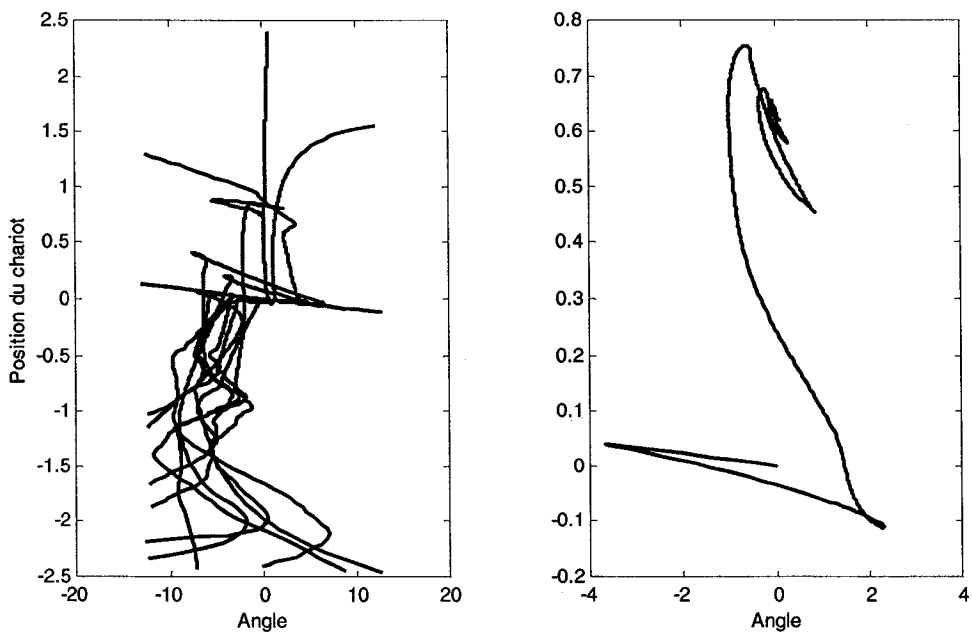


Figure 3-26 Trajectoires d'un agent glouton. (a) totalité de l'épisode. (b) dernier essai de l'épisode.

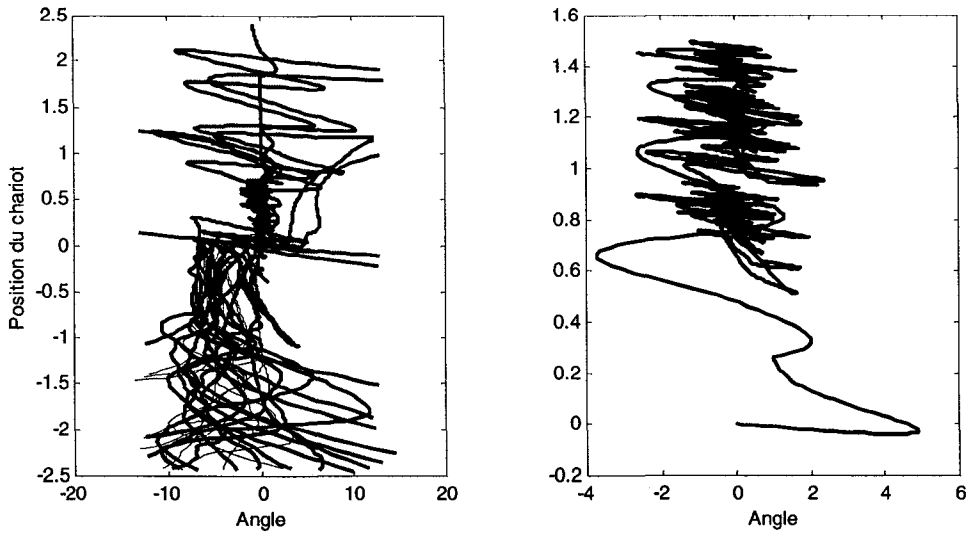


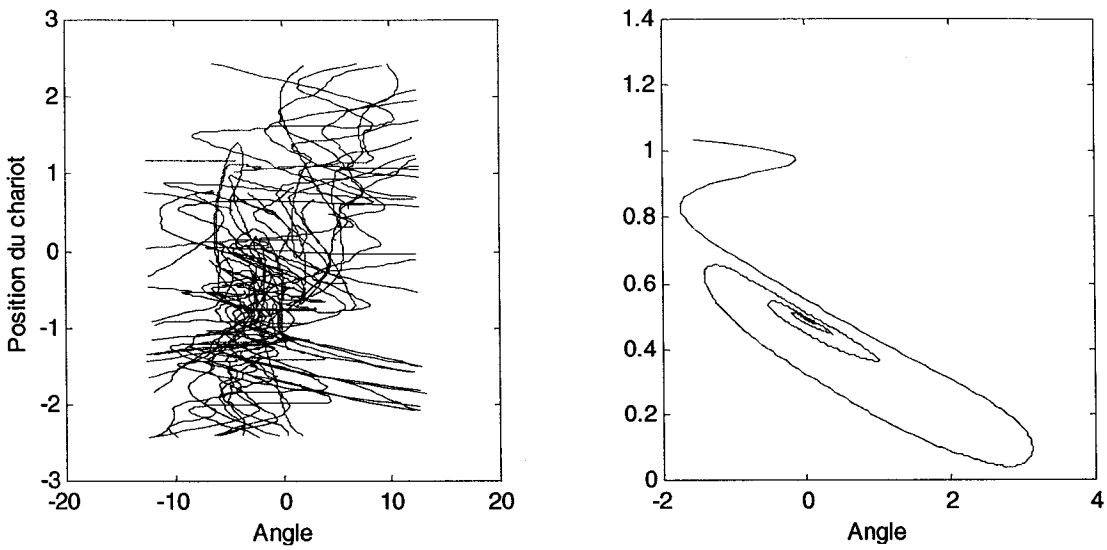
Figure 3-27 Trajectoires de l'agent quasi glouton ($\varepsilon = 0.01$). (a) totalité de l'épisode. (b) dernier essai de l'épisode.

On voit que la couverture de l'espace d'états pendant la phase d'apprentissage est limitée comparativement à celle de l'agent non glouton (Fig. 3-27, courbe (b)). Or, justement ce dernier réussit très bien à maintenir le pendule en équilibre comme l'illustre la figure 3-25 où nous avons présenté les variations des variables d'état pendant les 20 premières secondes (1000 itérations).

Nous avons conduit le même ensemble d'expérience avec un agent glouton utilisant des conditions initiales aléatoires au début de chaque essai. Les résultats (figures 3-28 et 3-29) sont très semblables à ceux d'un agent quasi glouton. Ce qui nous emmène encore une fois à conclure qu'il existe une certaine équivalence entre ces deux méthodes d'exploration. Par conséquent, le concepteur pourra librement choisir laquelle des deux méthode d'exploration lui convient le mieux.

On peut donc conclure que dépendamment de la complexité de la tâche, le concepteur peut choisir entre la vitesse ou la qualité (globalité) de l'apprentissage. Dans le premier cas, il doit se limiter à une exploration stochastique de l'espace d'états en se basant uniquement sur la fonction d'évaluation. Dans le deuxième cas, il peut combiner plusieurs méthodes

d'exploration purement aléatoires pour s'assurer une plus grande couverture de l'espace d'états.



(a) (b)

Figure 3-28 Trajectoires de l'agent glouton avec des conditions initiales aléatoires. (a) totalité de l'épisode. (b) dernier essai de l'épisode.

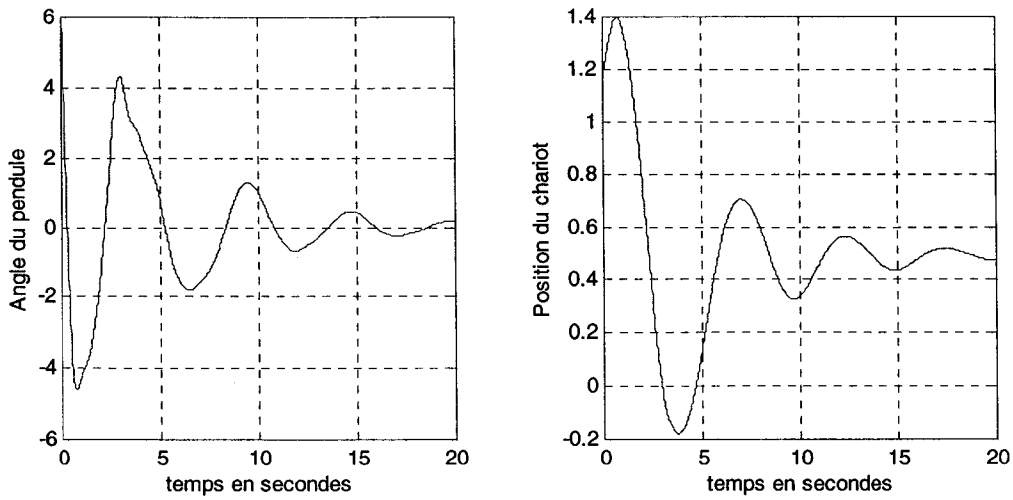


Figure 3-29 Variations des variables d'état d'un agent glouton en mode exploitation. Les conditions initiales sont aléatoires pendant la phase d'apprentissage.

3.5.2.4 Initialisation du critique

Le choix de la méthode d'initialisation du critique joue un rôle primordial en ce qui concerne la vitesse de convergence. Mais contrairement aux méthodes d'apprentissage supervisé, l'importance de la méthode d'initialisation du critique ne vient pas du fait qu'on cherche le meilleur estimé pour amorcer l'apprentissage, mais plutôt du rôle qu'elle joue dans le mécanisme d'équilibre exploration/exploitation. En effet, dépendamment de l'attitude de l'agent au début de l'apprentissage (optimiste, pessimiste, etc), il aura tendance à déclencher rapidement ou non le processus d'exploration.

Dans ce qui suit, nous allons comparer les performances d'un même agent utilisant quatre méthodes d'initialisation différentes :

1. Optimiste : les valeurs initiales de tous les états sont maximales. Dans ce cas, l'agent ne déclenche son processus d'exploration qu'une fois que des états terminaux ont été visités, ce qui aurait pour conséquence de diminuer les valeurs des états visités récemment par l'agent. Par conséquent, la première phase de l'apprentissage est une phase d'exploitation exclusivement. L'agent attend sagement d'estimer la fonction d'évaluation de la stratégie de contrôle actuelle avant de la remettre en question.

2. Limites : les valeurs initiales de tous les états sont égales au seuil d'exploration. Dans ce cas, l'agent déclenche l'exploration à la première visite de chaque état mais systématiquement augmente la valeur de ce dernier pour indiquer qu'il est tout de même meilleur que son successeur et donc de sa valeur actuelle. À moins que la valeur de cet état soit ultérieurement diminuée légitimement ce qui indiquerait qu'il est réellement mauvais, la prochaine fois qu'il est visité, aucune exploration n'aura lieu. La première phase de l'apprentissage commence par une exploration excessive qui permet de diviser l'espace d'états en deux classes : une classe où l'agent explore fréquemment et une autre classe où il exploite strictement sa stratégie de contrôle.

3. Pessimiste : les valeurs initiales de tous les états sont minimales. L'agent explore excessivement en début d'apprentissage. La stratégie de contrôle change continuellement et, par conséquent, la fonction d'évaluation. L'agent panique.

4. Aléatoire : les valeurs initiales de tous les états sont aléatoires.

Les résultats de simulation sont présentés sur la figure 3-30 où on a représenté les courbes de performance dans les 4 cas.

Ils est assez aisé de conclure qu'une attitude optimiste est garante des meilleurs performances. À noter également que, sans surprise, l'attitude pessimiste offre les pires performances.

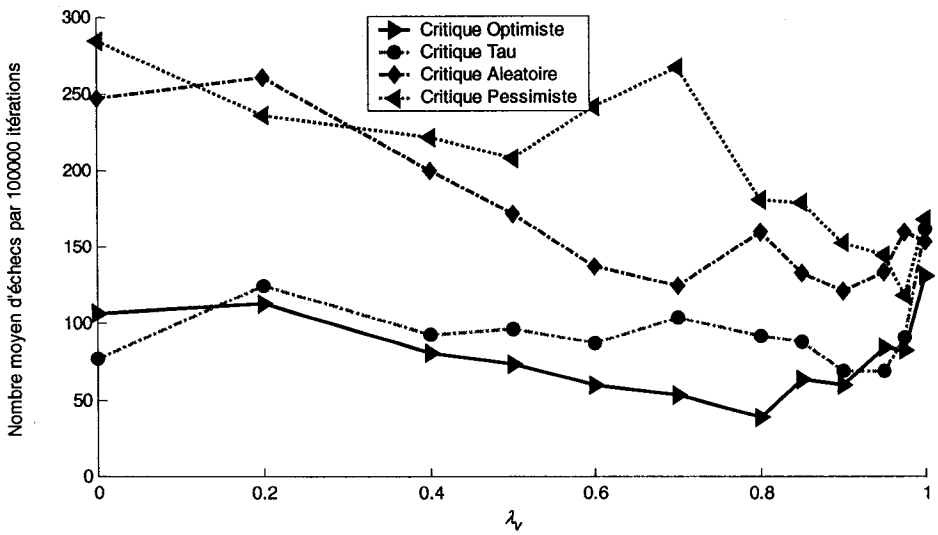


Figure 3-30 Courbes de performance d'un agent FNAC pour différentes méthodes d'initialisation.

3.5.2.5 Effet du type de traces d'éligibilité

L'analyse de la courbe de performance (figure 3-15) indique qu'en utilisant des AET, l'agent FNAC apprend moins rapidement lorsque $\lambda = 1$ (*i.e.* mise à jour effectuée sur l'ensemble d'un essai). Comme nous l'avons déjà mentionné auparavant, cela est dû à une surévaluation des traces de chaque règle ce qui provoque une mise à jour excessive des paramètres ajustables de la règle. Pour y remédier, nous avons le choix entre la diminution des coefficients d'apprentissage (afin de diminuer les amplitudes des mises à jour) ou l'utilisation des RET (comme nous l'avons introduit auparavant).

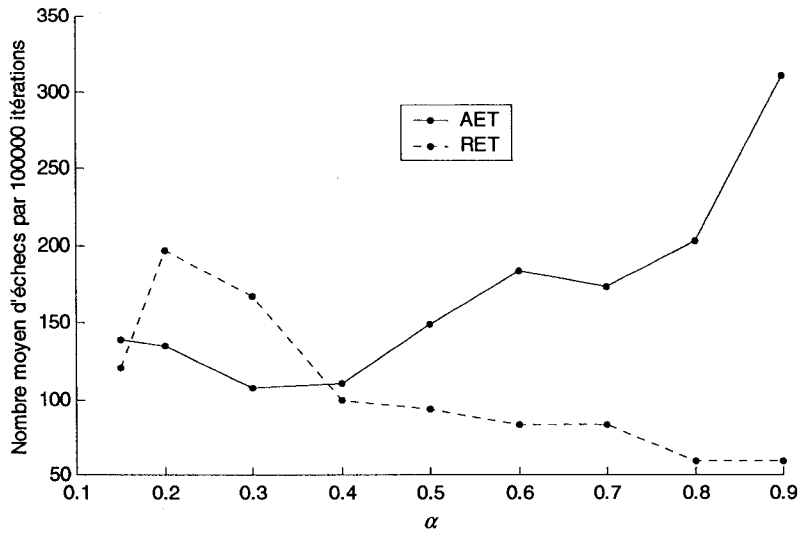


Figure 3-31 Variations de l'indice de performance de l'agent FNAC ($\lambda = 1$) en fonction du taux d'apprentissage du critique et du type de traces d'éligibilité.

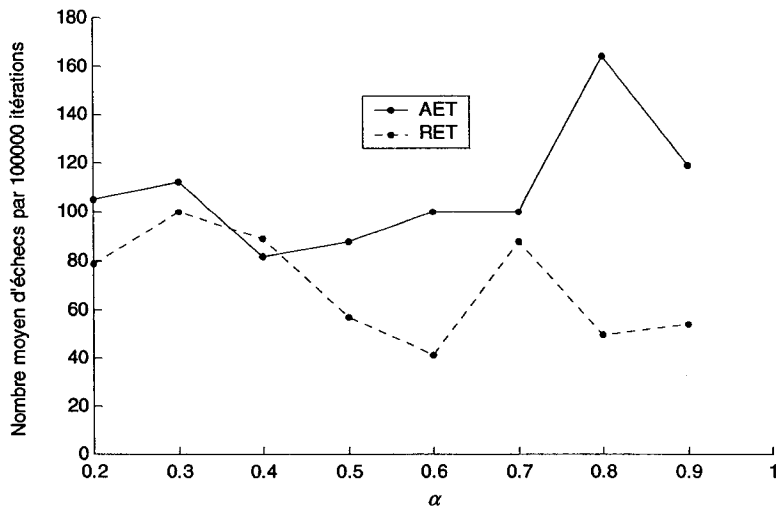


Figure 3-32 Variations de l'indice de performance de l'agent FNAC ($\lambda = 0.95$) en fonction du taux d'apprentissage du critique et du type de traces d'éligibilité.

Sur la figure 3-31, nous avons représenté les variations de l'indice de performance de deux agents à la mémoire longue ($\lambda = 1$) en fonction du taux d'apprentissage du critique. On remarque une nette amélioration des performances de l'agent en utilisant des RET. Fait intéressent, l'usage des RET permet d'augmenter la valeur du taux d'apprentissage, ce qui

ne peut qu'accélérer l'apprentissage. On remarque également les même caractéristiques pour ($\lambda = 0.95$) quoique l'écart de performance n'est pas aussi important (Fig. 3-32).

D'autre part, en comparant avec les données de la figure 3-15, on voit que les performances obtenues en utilisant les RET et un ($\lambda = 1$) sont meilleures que toutes celles obtenues avec les AET; quelque soit la valeur de λ.

On peut donc affirmer que les traces d'éligibilité substitutives que nous avons introduites (Éq. 3-17) se sont avérées très utiles. Cependant, nos simulations indiquent que pour les valeurs de λ inférieures à 0.9, l'usage des AET donne presque toujours de meilleurs résultats que celui des RET.

Finalement, mentionnons que les RET que nous avons proposés sont aussi applicables dans le cas des méthodes de type QL ainsi qu'avec tous les approximateurs universels linéaires.

3.6 Conclusions et discussions

Dans le présent chapitre, nous avons commencé par une revue des principales méthodes AC déterministes et continues. Nous avons présenté leurs principales caractéristiques et surtout leurs limitations. Par la suite, nous avons présenté l'architecture FNAC afin de répondre à ces limitations. L'architecture FNAC permet aussi de résoudre le problème d'implémentation en utilisant les systèmes neuro-flous. Nos simulations indiquent que l'architecture FNAC permet un apprentissage plus rapide comparativement à toutes les autres méthodes AC existantes.

Les principales contributions présentées dans ce chapitre sont les suivantes :

- Le critique de l'agent FNAC est prudent dans son estimation de la fonction d'évaluation pendant la phase d'exploration. Nous avons proposé notamment de distinguer entre les explorations fructueuses et celles qui ne le sont pas. Par conséquent, la fonction d'évaluation tend à rester la plus consistante que possible avec la stratégie de contrôle.

- L'acteur de l'agent FNAC est équitable dans son appréciation des résultats d'exploration. Nous avons proposé de normaliser le signal de renforcement interne utilisé pour la mise à jour de l'acteur. Par conséquent, les actions explorées sont critiquées en fonction du changement relatif de la valeur de l'état où l'exploration a eu lieu. Il en résulte que si une bonne action est découverte, elle est incorporée dans la stratégie de contrôle indépendamment de la valeur de l'état où elle est découverte.

- L'agent FNAC combine plusieurs méthodes d'exploration.

- Nous avons introduit des traces d'éligibilité pour l'acteur et montré qu'elles permettent généralement un apprentissage plus rapide.

- Nous avons introduit de nouvelles traces d'éligibilité de type substitutives spécialement adaptées aux systèmes neuro-flous mais applicables à toutes les méthodes RL. Elles sont aussi utilisables avec tout les approximateurs linéaires.

- Nous avons établi l'équivalence entre deux méthodes d'exploration. La première méthode d'exploration est l'usage d'une stratégie quasi-glouton. La deuxième est l'exploration par variation des conditions initiales que nous avons nous-même introduit.

- Nous avons démontré empiriquement que même si l'exploration basée uniquement sur la fonction d'évaluation permettait un apprentissage plus rapide, les méthodes d'exploration purement aléatoires conduisent généralement vers des solutions plus globales.

Bien que les solutions proposées ont été validées par simulations, il serait utile, voire même nécessaire, de les soutenir avec une analyse formelle rigoureuse et un développement théorique approfondi afin de définir leur domaine de validité ainsi que les conditions attachées. Cette direction de recherche nous paraît très prometteuse vues les études récemment publiées sur les méthodes acteur-critique utilisant la fonction Q (Konda, 2001, Sutton, 1999).

Une autre limitation de notre travail concerne la méthodologie de comparaison des différents algorithmes RL. En effet, il aurait été certainement plus rigoureux d'effectuer les

comparaisons en optimisant les paramètres d'apprentissage de chaque algorithme plutôt que d'utiliser les même paramètres pour tous les algorithmes considérés. Cependant, on ne l'a pas fait ici car on est parti de l'hypothèse que les performances des autres algorithmes rapportées dans la littérature ont été vraisemblablement optimisées et dans cette perspective nos résultats restent quand même supérieurs. D'autre part, il est à noter que vu le nombre de considérable de paramètres à optimiser : (α, β, α, λ, γ, ε, τ, les paramètres du SIF tel que le nombre d'étiquettes floues décrivant chaque variable d'état ainsi que la forme des fonctions d'appartenance, etc.) fait en sorte que l'optimisation de tous ces paramètres constitue un problème non trivial d'optimisation combinatoire qui dépasse les objectifs de cette thèse. Dans la perspective d'une continuité de ce travail, nous proposons de développer un module d'optimisation à base d'algorithmes génétiques ou d'autres méthode de recherche heuristiques.

4 L'AGENT FNAC POUR LE CONTRÔLE MULTIVARIABLE

4.1 Introduction

Le problème du contrôle multivariable peut être traité assez aisément par les méthodes QL ou encore par les méthodes AC discrètes. Dans les deux cas, des fonctions de type $F(s,a)$ sont invoquées avec les problèmes d'explosion dimensionnelle qui en résultent. Dans le cas des méthodes AC continues, le problème de répartition des crédits devient plus sévère. En effet, le critique ne formant qu'un seul et unique signal de renforcement interne, il est à priori impossible de déterminer la contribution de chacune des variables de contrôle dans la valeur de ce signal. Par exemple, après une exploration fructueuse où une ou plusieurs variables de contrôle ont été explorées, comment déterminer lesquelles sont réellement responsables de l'amélioration de la valeur de l'état. Dans ce chapitre, nous allons présenter deux approches pour résoudre ce problème.

4.1.1 Exploration multidimensionnelle

La première approche consiste à considérer chaque combinaison d'actions comme une macro action et, par conséquent, lui attribuer la totalité du signal de renforcement interne. Chaque exploration porte simultanément sur toutes les variables de contrôle. Le processus d'amélioration de la stratégie de contrôle consiste alors à chercher dans chaque état la meilleure combinaison des variables de contrôle. À la suite d'une opération d'exploration, les différents éléments de chaque combinaison reçoivent le même signal de renforcement. Si celui-ci est positif, alors tous les éléments sont substitués par ceux de la combinaison explorée. Sinon, ils sont modifiés dans la direction opposée. Cette approche, simple et facile à mettre en œuvre, présente l'inconvénient de voir la dimension de l'espace de recherche des stratégies de contrôle exploser exponentiellement. Elle reste toutefois plus intéressante que les méthodes QL puisque le problème d'explosion dimensionnelle n'est

qu'implicite et par conséquent ne nécessite ni plus de mémoire pour le stockage ni plus de temps de calcul pour la sélection d'actions. En effet, seule la fonction d'évaluation des états est à estimer, la stratégie de contrôle étant toujours explicitement représentée. Cependant, cette méthode d'exploration présente l'inconvénient que même si certains éléments de la combinaison initialement calculée par l'acteur sont éventuellement meilleurs que ceux de la combinaison explorée, ils sont quand même modifiés, ce qui risque de ralentir considérablement le processus d'apprentissage.

4.1.2 Exploration unidimensionnelle

La seconde approche consiste à critiquer individuellement les différentes variables de contrôle. À chaque exploration, l'agent choisit au hasard une et une seule variable à explorer. Les autres variables prennent strictement les valeurs calculées par l'acteur. La motivation de cette approche est donc de découvrir individuellement les valeurs optimales de chacune des variables de contrôle dans un état donné. La combinaison optimale de variables de contrôle associée à un état donné est construite progressivement; une variable à la fois. Elle ne peut donc que s'améliorer au fur et à mesure que l'agent explore. Cette approche d'exploration, plus disciplinée, a pour but d'apprendre plus rapidement des stratégies de contrôle acceptables. Il n'est cependant pas justifié d'anticiper qu'elle peut conduire plus rapidement vers une stratégie de contrôle optimale. En effet, la découverte de celle-ci exige généralement que tout les états de l'espace de recherche soient visités un nombre de fois suffisamment grand afin d'en évaluer l'utilité. Par conséquent, les deux approches devraient conduire ultimement au même optimum global. La figure 4-1 décrit l'algorithme d'apprentissage de l'agent FNAC multivariable.

Le but de l'étude que nous effectuerons ici sera limité à démontrer que les deux approches fonctionnent effectivement et à vérifier l'hypothèse que la deuxième approche permet généralement un apprentissage plus rapide. Nous utiliserons le problème de la conduite d'un vélo pour valider les deux approches.

```
Initialize the value function optimistically
Initialize the policy randomly,
Repeat (for each trial)
        Initialize the current state
        Repeat (for each step of the current trial)
                for the current state
                        Compute the recommended actions,
                        Compute the current state's value,
                        explored actions ← recommended actions
                        if (the state value is under a certain threshold),
                                if (multidimensional exploration),
                                        for each actor's output,
                                                Explore randomly around the recommended actions,
                                else
                                        Choose one actor's output randomly,
                                        Explore randomly around the selected output,
                for each state,
                                Compute the eligibility traces for the critic,
                        for each actor's output, compute the eligibility traces,
                Take explored actions, observe reward, and next state,
                        Compute the next state's value,
                Compute the TD-error,
                if (TD-error < 0),
                        Reset all eligibility traces,
                else
                        Update the critic,
                Update the actor,
                current state ← next state,
        Until current state s is terminal
```

Figure 4-1 Algorithme générique de l'agent FNAC multivariable.

4.2 Problème de conduite d'un vélo

Pour démontrer les performances de l'agent FNAC multivariable au-delà du simple problème de la pendule inverse, nous l'avons confronté à une tâche nettement plus difficile : conduire un vélo. L'agent doit apprendre à garder l'équilibre tout en suivant une trajectoire bien spécifique.

4.2.1 Description du système

Dans ce problème, l'agent, assimilé à un cycliste humain (poids et posture), doit apprendre pour la première fois à conduire un vélo sans superviseur.

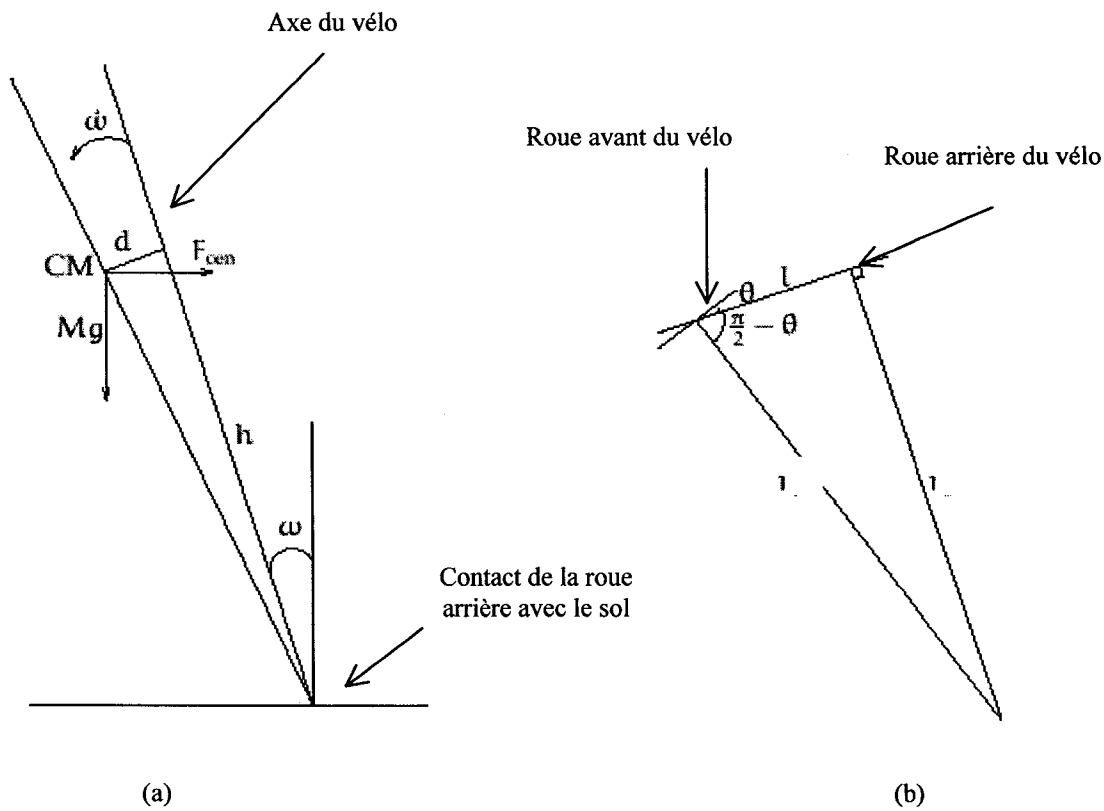


Figure 4-2 (a) Vue d'arrière du vélo. (b) Vue de dessus du vélo.

L'agent dispose de capteurs lui permettant d'observer son état constitué des 5 variables d'état suivantes (Fig. 4-2) :

- L'angle ω formé par l'axe du vélo et la verticale;
- La vitesse angulaire $\dot{\omega}$;
- L'accélération angulaire $\ddot{\omega}$;
- L'angle $\theta \in \left[-60°, 60°\right]$ de déviation du guidon par rapport à l'axe horizontal du vélo formé par les points de contact des deux roues avec le sol;
- La vitesse angulaire $\dot{\theta}$.

Le cycliste (l'agent) dispose de deux variables de contrôle : le couple $T \in [-2\text{N}, 2\text{N}]$ appliqué sur le guidon ainsi que le déplacement $d \in [-2\,\text{cm}, 2\,\text{cm}]$ du centre de masse du système (vélo + cycliste) par rapport à l'axe du vélo. Le modèle cinématique du vélo, ainsi que les paramètres correspondants, sont repris intégralement de (Randlov, 1998) et sont présentés à l'annexe 1.

Le but ultime de l'agent est de garder le système en équilibre le plus longtemps possible tout en s'assurant de ne pas trop s'éloigner d'une trajectoire désirée.

4.2.2 Problème de maintien de l'équilibre

Dans un premier temps, nous avons réduit la tâche de l'agent en cherchant seulement à maintenir le système en équilibre. Ainsi, on se place dans les mêmes conditions d'apprentissage que (Randlov, 1998) pour comparer les performances d'apprentissage. Plus précisément, on se fixe comme objectif de maintenir, pendant au moins 100000 itérations (1000 secondes), l'angle ω dans l'intervalle [-12° , 12°].

La fonction de renforcement consiste uniquement à punir l'agent en cas d'échec :

$$r = \begin{cases} 0 & \text{si } |\omega| \leq 12^\circ \\ -1 & \text{sinon} \end{cases} \tag{4.1}$$

Le SIF implémentant l'agent FNAC multivariable possède 5 entrées (les variables d'état) et 3 sorties (la valeur de l'état courant ainsi que les deux variables de contrôle). Nous avons utilisé une partition globale et uniforme où chaque entrée est décrite par trois étiquettes floues à fonctions d'appartenance triangulaires. Les paramètres (centres et largeurs) de fuzzification sont choisis pour assurer une couverture uniforme de l'espace d'états, ce qui correspond à un total de 243 règles floues (Annexe 2). Les paramètres d'apprentissage utilisés sont les suivants : $\gamma = 0.99$, $\alpha = 5$, $\beta = 0.25$, $\tau = -0.05$, $\lambda_\pi = 0$, et $\varepsilon = 0$. Le facteur d'escompte γ est choisi très proche de l'unité car la tâche de l'agent est assez critique et il est donc important de pouvoir prédire l'échec le plutôt possible. Pour la même raison, nous avons choisi un seuil d'exploration relativement bas afin de mettre l'agent en mode exploration dès l'apparition d'un signe d'échec. Enfin, nous avons utilisé une stratégie glouton pour permettre à l'agent de vivre des expériences plus longues.

4.2.2.1 Courbes de performance

Les résultats, présentés sur les figures 4-3 et 4-4, démontrent clairement l'efficacité de l'agent FNAC multivariable aussi bien en utilisant une approche par exploration multidimensionnelle qu'unidimensionnelle. En effet, avec un indice d'apprentissage moyen inférieur à 500 essais par épisode, les performances de l'agent FNAC sont nettement supérieures à celles rapportées dans (Randlov, 1998) où l'indice d'apprentissage moyen était supérieur à 2000 essais par épisodes dans le meilleur des cas. Dans (Randlov, 1998) une architecture de type SARSA fut utilisée et implémentée par un système de BOITES identique à celui de (Barto, 1983) à 5 entrées (les mêmes que nous avons utilisées) et 9 sorties. La discrétisation de l'espace d'états fut discrétisé en 3456 boites, comparativement à 243 boites floues dans notre cas. Le système de boites est en réalité l'équivalent d'un SIF où il n'existe aucun recouvrement entre les différentes étiquettes décrivant chaque variable d'entrée. À chaque instant, une seule règle est active, d'où une faible capacité de généralisation.

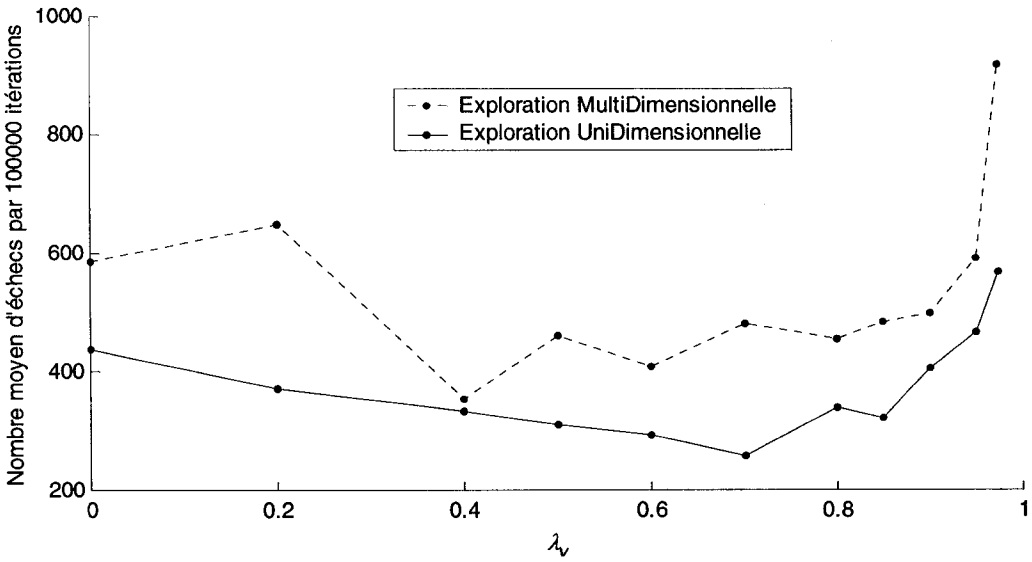


Figure 4-3 Courbes de performance de l'agent FNAC multivariable.

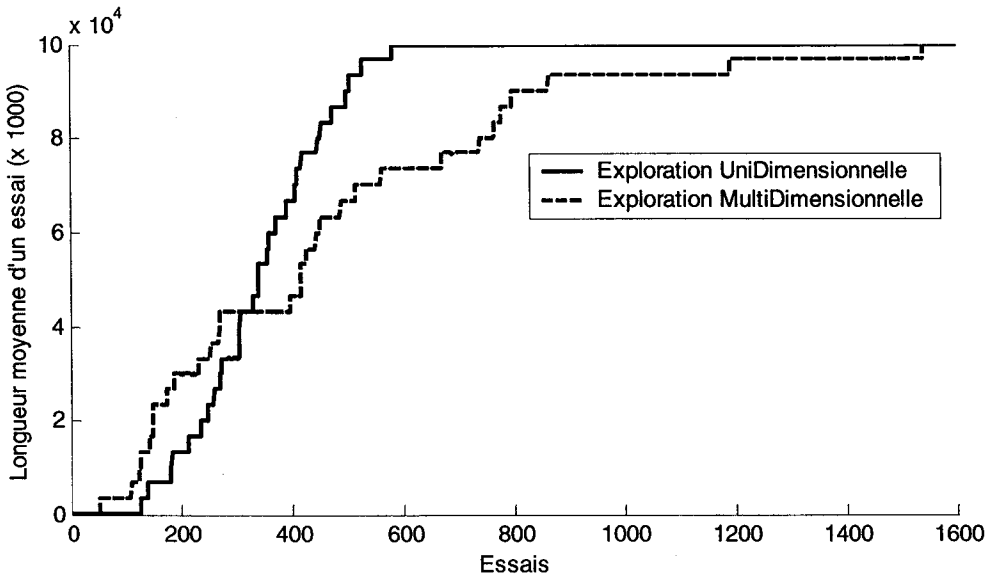


Figure 4-4 Courbes d'apprentissage de l'agent FNAC multivariable.

Par conséquent, la supériorité des performances de l'agent FNAC est attribuable à la fois à l'utilisation d'une architecture AC mais aussi aux capacités de généralisation des SIF. Un autre avantage de l'agent FNAC est la continuité des variables de contrôle évitant ainsi de

converger vers des lois de commande de type tout ou rien comme dans le cas de Randlov. Soulignons finalement que, tel qu'anticipé, l'approche par exploration unidimensionnelle offre les meilleures performances.

4.2.2.2 Analyse de la stratégie de contrôle apprise

Les paramètres d'apprentissage étant identiques à ceux utilisés ci haut avec en plus $\lambda_V = 0.8$, nous avons observé le comportement du cycliste dans une phase test où aucun apprentissage supplémentaire n'est effectué. Les résultats correspondant aux 120 premières secondes sont présentés sur les figures 4-5 et 4-6. On constate, sur la figure 4-6, que la trajectoire de l'agent n'est ni rectiligne ni même oscillatoire : elle est ellipsoïdale. L'analyse des variables de contrôle (Fig. 4-5) indique que le centre de masse à une moyenne négative et est donc responsable de cette allure de la trajectoire. Cette anomalie de la solution apprise est du même genre constaté dans le cas du problème du pendule : la position du chariot en régime permanent n'est pas nulle (Fig. 3-29). L'agent

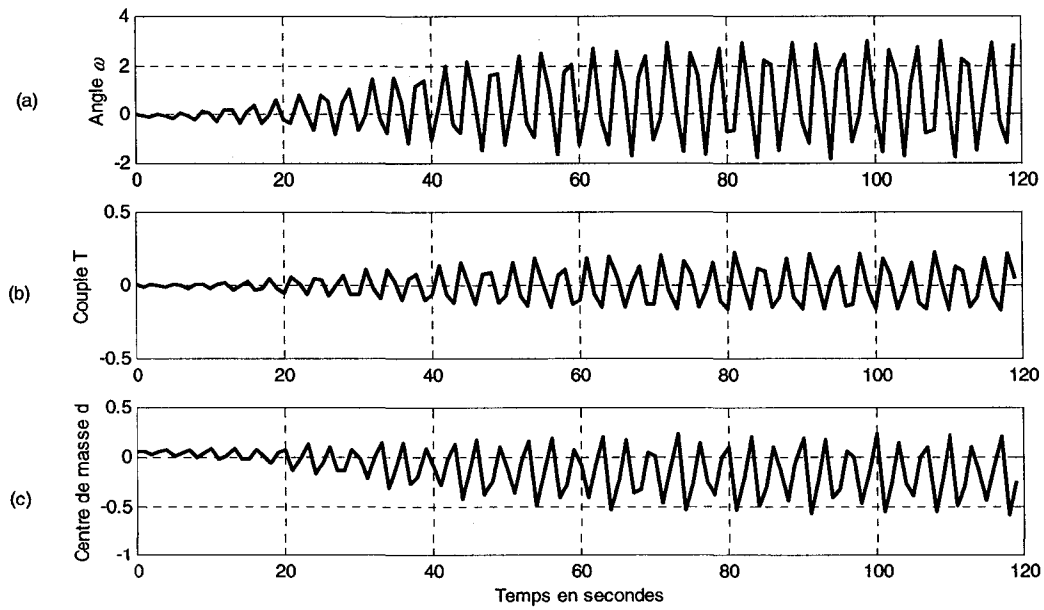


Figure 4-5 Évolution des variables du système une fois la phase d'apprentissage terminée. (a) angle d'inclinaison du vélo, (b) couple appliqué sur le guidon et (c) déplacement du centre de masse.

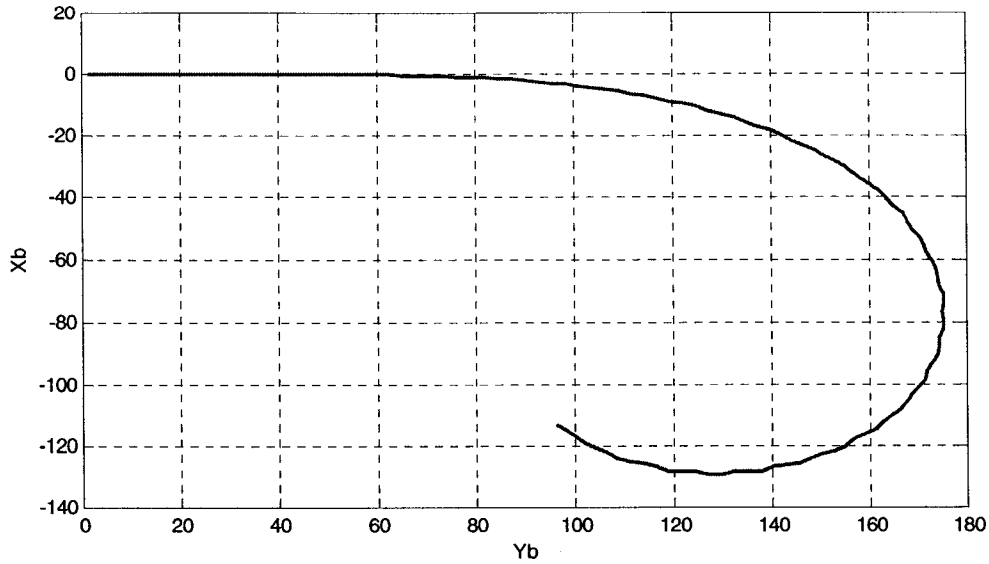


Figure 4-6 Trajectoire du vélo une fois l'apprentissage terminé.

n'a pas de souci à raffiner sa stratégie de contrôle pourvu que celle-ci lui évite les pénalités. Par contre, si on veut que l'agent apprenne une stratégie de contrôle plus performante, il faut le lui indiquer en enrichissant dans la mesure du possible la définition de la fonction de renforcement. Par exemple, on peut diviser chaque épisode d'apprentissage en deux parties. Dans la première partie, la fonction de renforcement est définie pour indiquer que la tâche de l'agent consiste seulement à apprendre une stratégie de contrôle suffisamment bonne pour atteindre son but. Alors que dans le seconde partie de l'épisode, la fonction de renforcement tient en compte également de la qualité de la solution apprise. Par exemple, dans le cas du pendule, on peut redéfinir la fonction de renforcement pour réduire l'erreur sur la position du chariot.

Une série de tests a été ensuite menée pour évaluer la robustesse de la stratégie de contrôle de l'agent une fois l'apprentissage terminé. Dans un premier test, nous avons simulé une mauvaise posture du cycliste en perturbant aléatoirement la position du centre de gravité du système (vélo + cycliste). Nous avons superposé à la variable de contrôle, définie par le déplacement du centre du système, un signal aléatoire uniforme d'amplitude 1 cm. Dans le second test, nous avons fait varier la masse du cycliste en la multipliant et en la divisant par

deux. Dans ces deux tests, l'agent réussi bien à conduire le vélo sans aucun apprentissage supplémentaire. On peut donc affirmer que la stratégie de contrôle apprise est assez robuste.

4.2.3 Problème de poursuite d'une trajectoire

Dans ce problème, nous nous proposons de soumettre l'agent apprenant à résoudre la totalité du problème de la conduite du vélo : se déplacer d'un point à l'autre sans aucune chute. Ce problème peut être abordé de deux façons. La première façon consiste à associer un renforcement positif à la zone cible ainsi que de re-concevoir judicieusement la fonction de renforcement afin que l'agent apprenant puisse avoir une idée de la trajectoire à suivre. Autrement, l'agent risque de ne jamais tomber sur la zone cible et ainsi lui associer un crédit positif. Cette approche, combinée à l'apprentissage par formation (*shaping*), est utilisée dans (Randlov, 1998). Par contre, on doit l'idée d'enrichissement de la fonction de renforcement afin de favoriser l'émergence de certains comportements à (Mataric, 1994). L'autre façon de définir la zone cible consiste à définir explicitement une trajectoire reliant le point de départ au point d'arrivée. Pour transmettre cette information à l'agent, on augmente l'état de l'environnement en lui ajoutant une nouvelle variable définie par la distance euclidienne entre l'état observé de l'environnement et la trajectoire prédéfinie. Une telle approche suppose l'absence de tout obstacle ou du moins que l'environnement où évolue l'agent soit stationnaire. D'après notre étude de la littérature existante, nous sommes les premiers à utiliser cette approche. Par contre, la première approche est plus générale et peut s'appliquer au-delà du problème de poursuite de trajectoire.

Ainsi, l'état du système sera constitué des 6 variables d'état suivantes :

- L'angle ω formé par l'axe du vélo et la verticale;
- La vitesse angulaire $\dot{\omega}$;
- L'accélération angulaire $\ddot{\omega}$;

- L'angle $\theta \in]-60°, 60°]$ de déviation du guidon par rapport à l'axe horizontal du vélo formé par les points de contact des deux roues avec le sol;

- La vitesse angulaire $\dot{\theta}$;

- La distance euclidienne $S(X_b, Y_b)$ entre le point (X_b, Y_b) de contact de la roue arrière du vélo avec le sol et la trajectoire à suivre. Dans cette simulation, le but de l'agent est de suivre soit une trajectoire rectiligne soit une trajectoire circulaire. Nous avons défini la trajectoire rectiligne comme étant celle de l'axe du vélo (axe verticale selon la formulation du problème se trouvant en annexe 1). Par conséquent, on a $S(X_b, Y_b) = X_b$. La trajectoire circulaire que nous avons utilisée est le cercle de rayon R tangent à l'axe du vélo. Par conséquent, on a $S(X_b, Y_b) = \sqrt{(X_b - R)^2 + Y_b^2} - R$.

Le but de l'agent est de garder le système en équilibre le plus longtemps possible tout en s'assurant de ne pas trop s'éloigner de la trajectoire désirée. Plus précisément, on se fixe comme objectif de maintenir, pendant au moins 100000 itérations, l'angle ω dans l'intervalle [-12° , 12°] et la position de la roue arrière ne doit pas s'éloigner de plus de 5 mètres de la trajectoire désirée.

La fonction de renforcement consiste uniquement à punir l'agent en cas d'échec :

$$r = \begin{cases} 0 & \text{si } |\omega| \leq 12° \text{ et } |S(X_b, Y_b)| \leq 5\,\text{m} \\ -1 & \text{sinon} \end{cases} \tag{4.2}$$

4.2.3.1 Expérience no1 : trajectoire rectiligne

Dans cette expérience, le but de l'agent est se déplacer le long de l'axe des y avec la contrainte de ne pas s'en éloigner de plus de 5 mètres. Quoique cette contrainte peut à priori sembler très souple, il est à noter qu'on peut anticiper que la trajectoire réelle de l'agent soit beaucoup plus précise. En effet, la distance de 5 mètres étant synonyme

d'échec, l'agent choisira ses actions pour garder sa trajectoire dans une zone de confort plus restreinte.

Nos résultats de simulation, avec les mêmes paramètres d'apprentissage utilisés précédemment, indiquent que l'agent met en moyenne 2868 essais par épisode pour accomplir sa tâche. Cela peut sembler surprenant, étant donné qu'on aurait pu imaginer que la complexité de cette tâche n'est pas si élevée par rapport au problème de maintien d'équilibre. En réalité, cette différence est surtout due à l'augmentation de la dimension de l'espace d'états et elle n'est donc que d'ordre numérique.

Nous avons essayé d'aider l'agent dans son apprentissage en lui suggérant de résoudre le problème en deux temps. Dans un premier temps, il ne s'occupe que du problème du maintien d'équilibre. Une fois cette tâche apprise, la fonction de renforcement est mise à jour pour lui indiquer que la tâche comprend désormais la poursuite d'une trajectoire. C'est le principe de l'apprentissage par formation qui consiste à accomplir des tâches complexes en relaxant les contraintes et en simplifiant les objectifs au début de l'apprentissage (Selfridge, 1985). Paradoxalement, nos simulations indiquent qu'en utilisant cette approche, l'agent met en moyenne 3700 essais par épisode pour accomplir sa tâche. Sans remettre en question l'apprentissage par formation, on constate qu'il n'a pas été efficace dans ce cas.

4.2.3.1.1 Courbes d'apprentissage

Sur la figure 4-7, nous avons représenté, en guise de courbe d'apprentissage, l'ensemble des trajectoires du vélo pendant les 500 premiers essais d'un épisode où 2512 essais ont été nécessaires à l'agent pour apprendre à accomplir sa tâche. La lecture de ces trajectoires se fait comme suit. Toutes les trajectoires se terminant à l'intérieur de l'enveloppe, formée par les droites $y_b = \pm 5$m, correspondent au cas où l'échec est dû à la chute du cycliste. C'est le cas par exemple des trajectoires 1 et 3 sur la figure 4-7. Les trajectoires, par exemple 2 et 4, se terminant à la limite de l'enveloppe correspondent au cas où l'agent s'est éloigné trop de sa trajectoire.

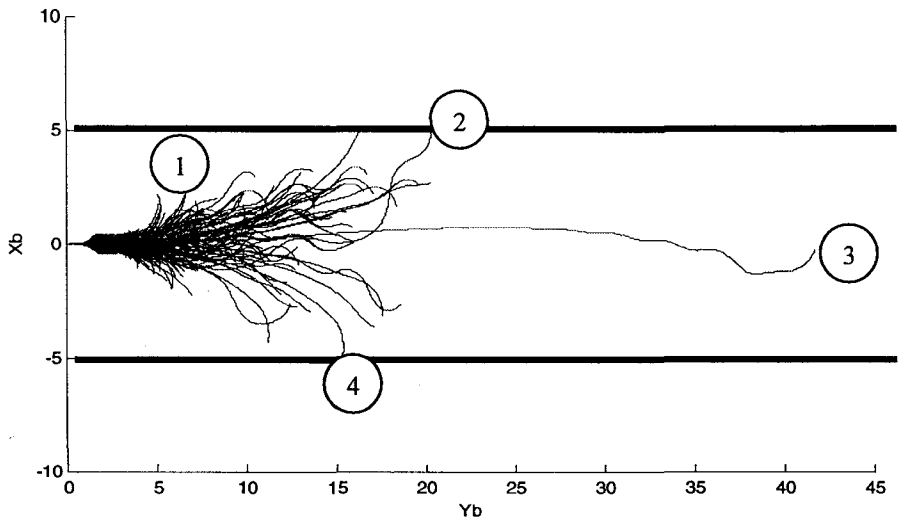


Figure 4-7 Trajectoires de l'agent pendant les 500 premiers essais de l'épisode.

On peut donc voir clairement que les premiers essais de l'agent sont surtout consacrés à son objectif premier : maintenir l'équilibre. Au fur et à mesure que l'apprentissage progresse, les trajectoires deviennent plus longues et l'agent se met à chercher aussi une solution pour contrôler sa trajectoire. Ce qu'il réussit plutôt bien dans le dernier essai, représenté sur la courbe 4 de la figure 4-7 où il arrive à parcourir près de 43 mètres avant de tomber. En atteignant son deuxième objectif, l'agent a dû oublier son premier objectif. La suite de l'apprentissage consiste donc surtout à concilier les deux objectifs.

4.2.3.1.2 Analyse de la stratégie de contrôle apprise

Sur les figures 4-8 et 4-9, nous avons représenté les variations des variables d'état ω et x_b ainsi que les deux variables de contrôle d et T dans le but d'analyser le comportement de l'agent une fois l'apprentissage terminé.

On constate que bien que le vélo reste à l'intérieur de l'enveloppe, sa trajectoire est oscillatoire. L'analyse des variables de contrôle indique que la principale variable de contrôle utilisée par l'agent est le déplacement de son centre de masse. En effet, on peut remarquer une corrélation évidente entre l'allure de cette variable et celle de la trajectoire

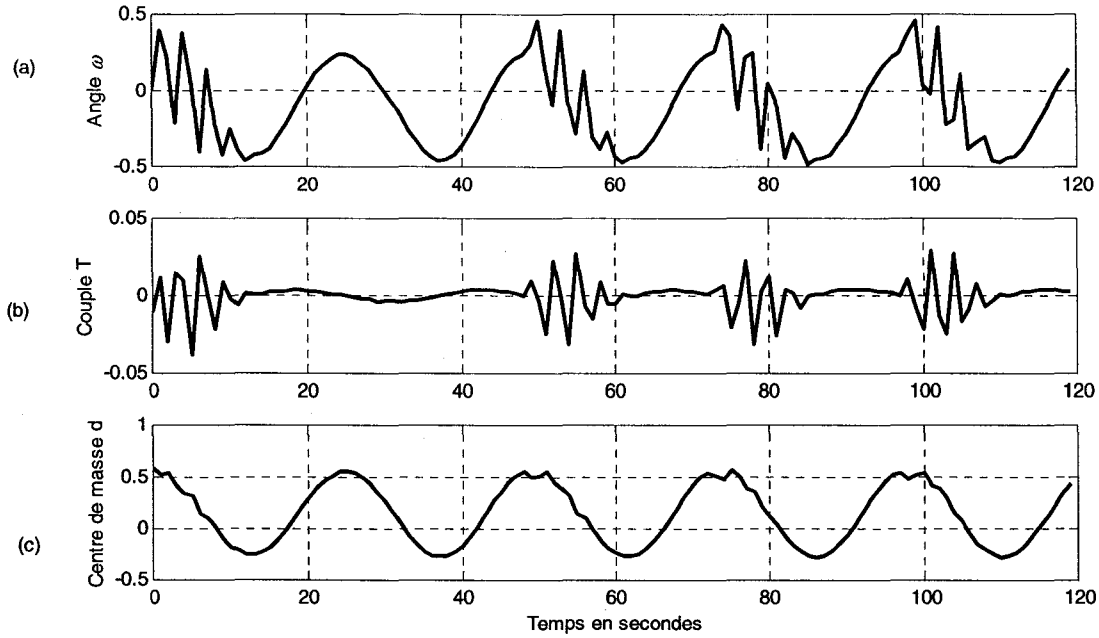


Figure 4-8 Évolution des variables du système une fois la phase d'apprentissage terminée. (a) angle d'inclinaison du vélo, (b) couple appliqué sur le guidon et (c) déplacement du centre de masse.

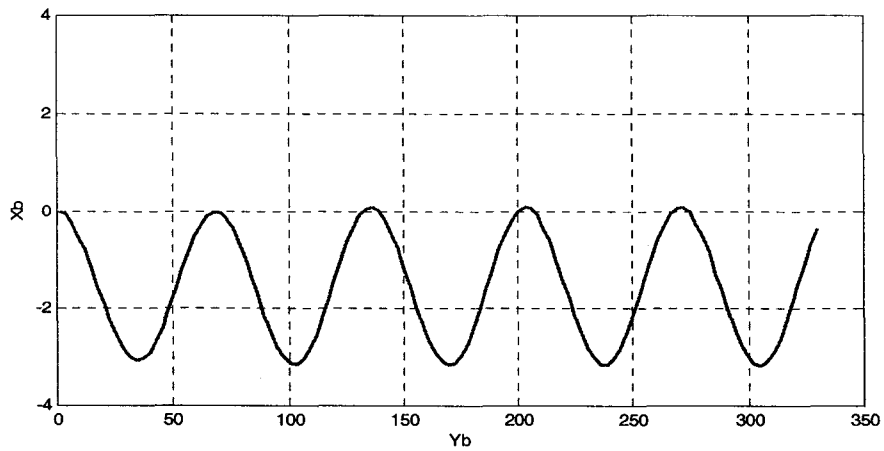


Figure 4-9 Trajectoire du vélo une fois l'apprentissage terminé.

du vélo. D'ailleurs, l'amplitude peu élevée du couple T indique également que l'agent ne s'en sert presque pas. C'est exactement ce qu'on constate chez un cycliste humain. Pour plus de détails sur la stratégie de contrôle apprise, nous avons présenté, en annexe 2, une partie de la base de règles de l'agent.

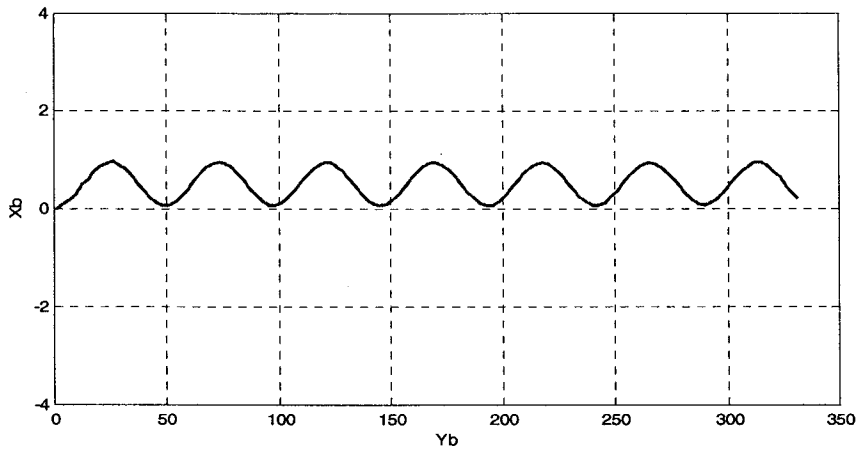


Figure 4-10 Trajectoire du vélo une fois l'apprentissage terminé.

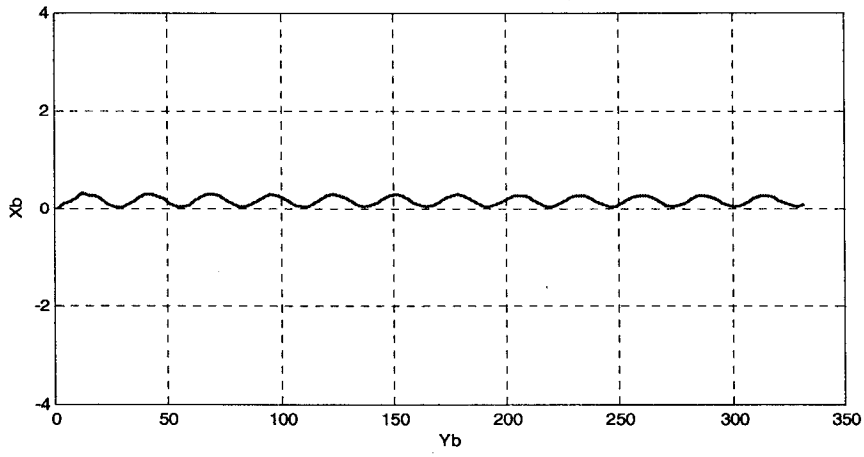


Figure 4-11 Trajectoire du vélo une fois l'apprentissage terminé.

Comme nous l'avons mentionné dans la section précédente, l'amélioration de la stratégie de contrôle apprise repose surtout sur l'enrichissement de la fonction de renforcement. Dans le cas présent, on peut diminuer la largeur de l'enveloppe ce qui aurait nécessairement diminué l'amplitude des oscillations du cycliste. Malheureusement, cette solution rendrait la tâche de l'agent plus difficile et par conséquent, allongerait considérablement le nombre d'essais par épisode avant que l'agent n'apprenne à accomplir sa tâche. Par contre, en enrichissant progressivement la fonction de renforcement tel que proposé dans la section précédente, on a réussi à améliorer l'allure de la trajectoire du cyclise comme le montre les figures 4-10 et 4-11.

Sur la figure 4-10, l'enveloppe a été réduite à ±3m, mais l'agent est initialisé avec les paramètres appris dans la simulation précédente : il sait comment garder le système en équilibre tout en restant à l'intérieur de l'enveloppe ±5m.

De façon similaire, sur la figure 4-11, l'enveloppe a été réduite à ±1m, mais l'agent est initialisé avec les paramètres appris dans la simulation précédente : il sait comment garder le système en équilibre tout en restant à l'intérieur de l'enveloppe ±3m.

4.2.3.2 Expérience no2 : trajectoire circulaire

La seconde tâche que l'agent devait accomplir consiste à poursuivre une trajectoire circulaire de rayon 30 mètres sans s'en éloigner de plus de 5 mètres. L'agent réussi bien sa mission même s'il lui faut en moyenne 15538 essais par épisode pour y arriver. Cette tâche est donc plus difficile que la précédente.

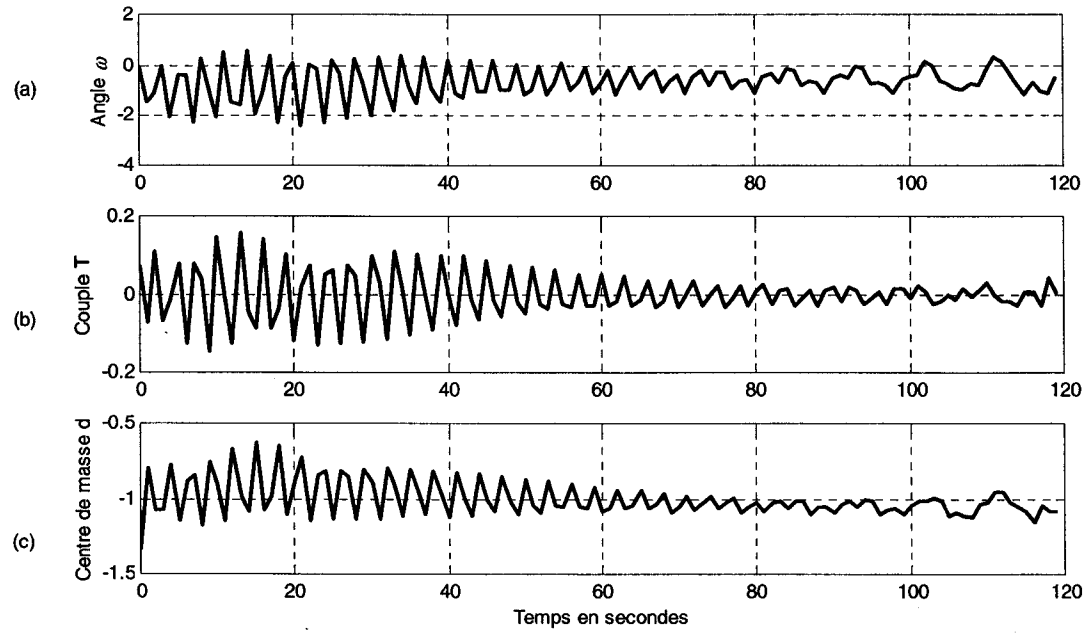


Figure 4-12 Évolution des variables du système une fois la phase d'apprentissage terminée.

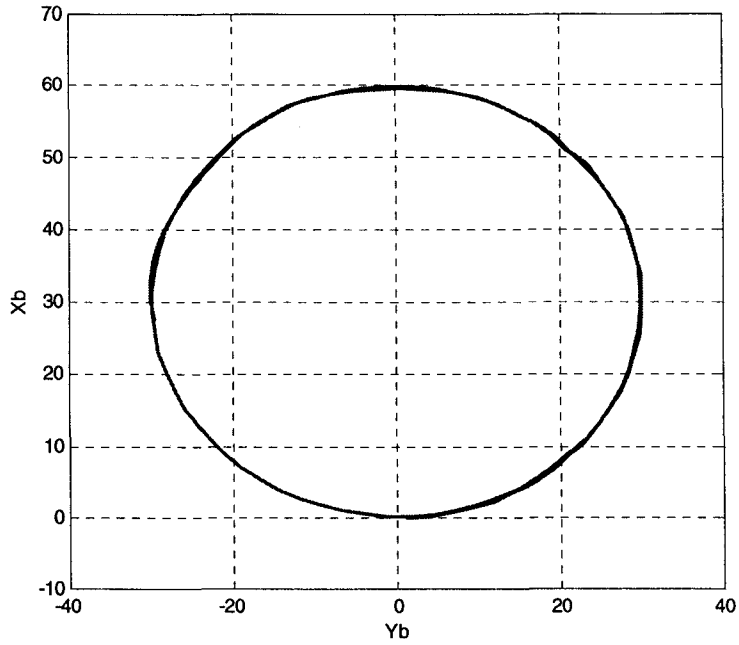


Figure 4-13 Trajectoire du vélo une fois l'apprentissage terminé. Le but de l'agent est de suivre une trajectoire circulaire de rayon 30 mètres.

Encore une fois, l'analyse des résultats sur les figures 4-12 et 4-13 indique que l'agent se fie surtout au déplacement de son centre de masse pour suivre sa trajectoire. En effet, l'agent garde son centre de masse constamment incliné vers la droite pour parcourir le cercle dans le sens horaire. L'agent semble réussir parfaitement (dans le sens que la trajectoire est très régulière) sa mission. Ceci n'est pas toujours le cas par contre. Sur la figure 4-14, correspondant à un autre épisode d'apprentissage et un cercle de rayon 50 mètres, on voit que même si l'agent accomplit correctement sa mission, sa trajectoire est quand même assez irrégulière. Pour minimiser le risque de tomber sur de telles solutions, la fonction de renforcement peut être rendue adaptative comme proposé précédemment.

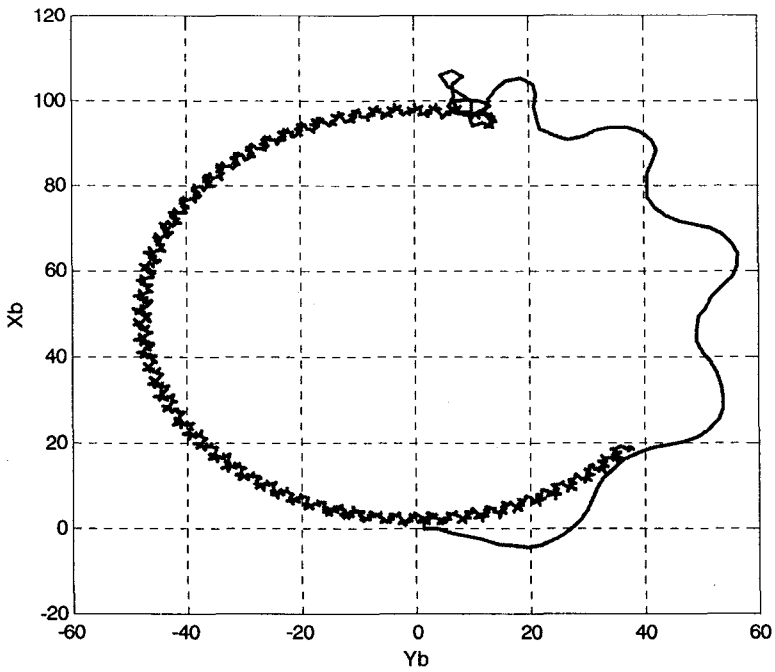


Figure 4-14 Trajectoire du vélo une fois l'apprentissage terminé. Le but de l'agent est de suivre une trajectoire circulaire de rayon 50 mètres.

4.3 Conclusion

Dans ce chapitre, nous avons généralisé l'architecture FNAC introduite au chapitre 3 au cas des systèmes de contrôle multivariable, où la stratégie de contrôle est déterministe et continue. Nous avons démontré que dans le cadre du contrôle multivariable, l'exploration unidimensionnelle, i.e. explorer une variable de contrôle à la fois, permet un apprentissage plus rapide. Nous avons appliqué la solution proposée, avec succès, pour résoudre un problème complexe qui est celui de la conduite d'un vélo. Les performances obtenues sont supérieures à celles des méthodes de type QL appliquées aux mêmes problèmes (Randlov, 1998). Finalement, nous avons proposé d'adresser le problème de poursuite de trajectoire comme une partie intégrale du problème d'apprentissage. Dans ce cadre, nous avons proposé d'augmenter la dimension du vecteur d'état, observé par l'agent, en lui ajoutant une mesure de la distance entre la trajectoire réelle du vélo et la trajectoire désirée. L'inconvénient d'une telle approche est l'augmentation de la dimension de l'espace d'états;

ce qui aura pour effet d'augmenter exponentiellement l'espace de recherche. Cependant, notre approche à l'avantage de décharger, et donc de simplifier, la fonction de renforcement. En comparaison, Randlov (Randlov, 1998) résout le problème de déplacement entre deux points donnés en pénalisant l'agent en fonction de l'écart entre sa trajectoire effective et la droite définie par les deux points. Une telle solution impliquerait la combinaison de deux signaux de renforcement de natures différentes. Le premier signal de renforcement, correspondant à la chute du vélo, est un signal retardé que l'agent reçoit uniquement à la fin d'un essai. Le second signal de renforcement, correspondant à la déviation de l'agent du chemin optimum vers son objectif, est un signal immédiat que l'agent reçoit après chaque itération. Rappelons que cette dernière est généralement définie par la somme cumulative des renforcement espérés à long terme. Par conséquent, l'amplitude du second signal de renforcement doit être soigneusement définie de pondérer correctement les contributions respectives des deux signaux dans la valeur de chaque état. Par contre, l'approche de Randlov peut être mieux adapté dans le cas où les objectifs du contrôle sont multiples et qu'ils impliquent d'autres variables en plus des variables d'état. Ce serait le cas, par exemple, si on doit minimiser explicitement l'énergie utilisée par l'agent pour accomplir sa tâche.

CONCLUSIONS ET PERSPECTIVES

Contributions

Cette thèse présente une nouvelle architecture de contrôle intelligent des systèmes, nommée FNAC. Elle tente d'améliorer et d'étendre le champ d'application d'une classe des méthodes d'apprentissage par renforcement appelée méthode acteur-critique. Les principales caractéristiques distinctives de ces méthodes de contrôle sont leurs capacités d'apprendre des stratégies de contrôle efficaces par le seul biais d'interaction d'un agent avec son environnement. Aucun modèle de ce dernier n'est nécessaire. En plus, l'agent a toute la latitude d'explorer de nouvelles actions et d'enrichir son processus de prise de décision en conséquence. Dans le cas particulier des méthodes AC, l'agent apprend exclusivement par exploration. Cette capacité d'apprendre de leurs propres expériences et, à l'occasion, de remettre en question leurs propres stratégies de contrôle justifie amplement de qualifier ces agents d'intelligents.

Les principales contributions de cette thèse portent essentiellement sur l'amélioration et l'accélération du processus d'apprentissage notamment au niveau de l'interaction entre le critique et l'acteur. On s'est intéressé aussi à chacun de ces deux éléments de façon individuelle. Parmi ces principales contributions, ont peut citer :

- Le processus d'interaction entre le critique et l'acteur est amélioré en distinguant entre les explorations fructueuses ou non. En plus, l'acteur apprend plus vite en adaptant le signal de renforcement interne en fonction de la valeur de l'état où l'exploration a eu lieu.

- Des traces d'éligibilité ont été introduites pour l'acteur afin de lui permettre d'apprendre plus vite et de mieux se comporter dans les processus non markoviens.

- L'acteur et le critique sont implémentés en utilisant des systèmes neuro-flous. Les traces d'éligibilité ont été redéfinies en conséquence. Nous avons également proposé une nouvelle forme de traces d'éligibilité substitutives qui permettent d'utiliser simultanément un taux d'apprentissage et un facteur d'oubli élevés.

- L' agent FNAC combine plusieurs méthodes d'exploration. Nous en avons démontré la pertinence en ce qui concerne la qualité des solutions apprises.

- Nous avons étendu l'usage des méthodes AC continues au cas du contrôle mutlivariables.

- Les performances de l'agent FNAC sont nettement supérieures à celles des autres méthodes RL.

- Finalement, nous avons effectué un ensemble d'études empiriques visant à aider le concepteur à mieux choisir les paramètres d'apprentissage.

Perspectives

Malgré la capacité des méthodes RL d'apprendre uniquement par interaction de type essai/erreur avec un environnement dynamique, il n'en demeure pas moins que cet apprentissage est lent. Le nombre élevé d'essais nécessaires pour accomplir leurs tâches limite l'applicabilité de ces méthodes à des processus physiques réels. Au moins deux axes de recherche existent pour accélérer l'apprentissage des méthodes RL. Le premier axe consiste à relaxer l'hypothèse que le système d'apprentissage doit être complètement autonome. On peut, par exemple, développer des méthodes RL basées sur des modèles des processus à contrôler : l'agent a accès à ces modèles et peut éventuellement les améliorer. On peut aussi entraîner l'agent pendant la phase d'apprentissage en lui donnant à l'occasion des conseils pour mieux l'orienter dans l'amélioration de son processus de prise de décision (Clouse, 1992 et Maclin, 1994).

Le second axe est celui du contrôle hiérarchique où les méthodes RL sont utilisées pour l'arbitrage entre plusieurs contrôleurs conventionnels spécialisés (Singh, 1992, Dietterich,

2000 et Hengst, 2000). Le principal gain, que promet cette approche, se situe au niveau de la réduction de la dimension de l'espace d'états par la formation de ce que les auteurs appellent les macro-états, qui correspondent grosso modo aux domaines de définition de chaque contrôleur local. Drummond (2002) propose, quand à lui, d'accélérer l'apprentissage par la décomposition de l'objectif général de l'agent en sous-objectifs intermédiaires identifiés automatiquement. Ensuite, les solutions intermédiaires apprises sont composées pour former la solution globale. Foster et Dayan (Foster, 2002) ont également proposé d'utiliser un algorithme d'apprentissage non supervisé, qui permet de classifier l'espace d'états en se basant sur le profil de la fonction d'évaluation. Ces deux dernières propositions ont en commun de construire graduellement l'espace d'états en se limitant aux régions effectivement visitées par l'agent. En réduisant la taille de l'espace de recherche, l'agent apprend plus vite.

Il faut également noter que les méthodes RL peuvent être combinées avec les méthodes de contrôle conventionnelles afin d'en étendre le domaine d'applicabilité. Par exemple, Andersen *et al.* (Andersen, 1996) ont utilisé le RL pour superviser un contrôleur PI conventionnel appliqué à un problème de chauffage thermique. Ils mentionnent que la composante RL de leur système de contrôle a permis d'ajuster correctement l'amplitude du signal de contrôle dans certains états. Randlov (2000) a aussi proposé d'utiliser l'approche RL pour le contrôle supervisé où le but de l'agent serait de maintenir ou d'amener le système dans une zone prédéfinie où un contrôleur conventionnel peut être conçu pour exécuter la tâche de façon optimale. Kretchmar (2000) proposent d'utiliser le RL pour définir des contraintes robustes garantissant la stabilité des systèmes de contrôle adaptatifs utilisant les réseaux de neurones. Un autre axe de recherche exploré de façon de plus en plus intensive est celui de l'intégration des méthodes d'abstraction temporelle développées pour les problématiques de planification. Un exemple de cette intégration est celui de la technique des *options* proposée par Precup (1999) et qui consiste à disposer d'une population de contrôleurs locaux opérant en boucle fermée et conçus principalement à l'aide des théories conventionnelles du contrôle. La main est passée alternativement à un contrôleur local pendant une certaine période de temps à durée fixe ou variable. Dans ce cas, le problème RL consiste à trouver la meilleure stratégie de sélection des contrôleurs locaux et d'évaluer ses performances. Puisque la durée d'utilisation de chaque contrôleur

local est variable, le problème ne peut plus être modélisé par un PDM mais doit plutôt être modélisé par un semi PDM (pour la théorie des processus semi PDM, voir Bradtke, 1995).

En résumé, on assiste dernièrement à une tendance suggérant d'utiliser l'apprentissage par renforcement à un niveau plus élevé et d'utiliser les autres méthodes de contrôle et d'apprentissage conventionnelles pour le calcul où l'apprentissage des décisions (actions) à bas niveau.

Finalement, nous pensons qu'en combinant les techniques RL et les techniques des algorithmes génétiques, on peut améliorer le processus d'amélioration de la stratégie de contrôle tout en supprimant, du moins explicitement, la nécessité d'un module d'exploration. Cette approche est utilisée avec succès par Lin (2000) pour la conception et le contrôle d'un système de roulement magnétique. Cependant, l'approche utilisée par Lin ne semble pas être réellement incrémentale à moins que l'on dispose d'un modèle de l'environnement à contrôler.

ANNEXE 1 : MODÈLE DU VÉLO

(Randlov, 2001)

A.1 The Bicycle Task

The bicycle must be held upright within $\pm 12^\circ$ measured from vertical position. If the angle from the vertical to the bicycle falls outside this interval, the bicycle has fallen, and the agent receives punishment -1. The bicycle is modelled by the following non-linear differential equations. One simplification was made to ease the derivation of the equations: The front fork was assumed to be vertical, which is unusual but not impossible. This, however, made the task a bit more difficult for the agent.

There are two important angles in this problem: The angle θ of the direction of the bicycle from straightforward, and the angle ω the bicycle is tiled from vertical. The conservation of angular momentum of the tyres results in some important cross terms.

The equations do not model a bicycle exactly, as some second order cross effects were ignored during the derivation. However we believe that the largest problem of transferring to a real bicycle would be to build hardware that could withstand falling over a thousand times—not just without crashing but also without changing and thereby make the system unstationary.

The following equations describe the mechanics of the system. (See Figure 44.) The angle φ is the total angle of tilt of the centre of mass, and is defined as:

$$\varphi \stackrel{\text{def}}{=} \omega + \arctan\left(\frac{d}{h}\right)$$

The angular acceleration $\frac{d^2\omega}{dt^2} = \ddot{\omega}$ can be calculated as:

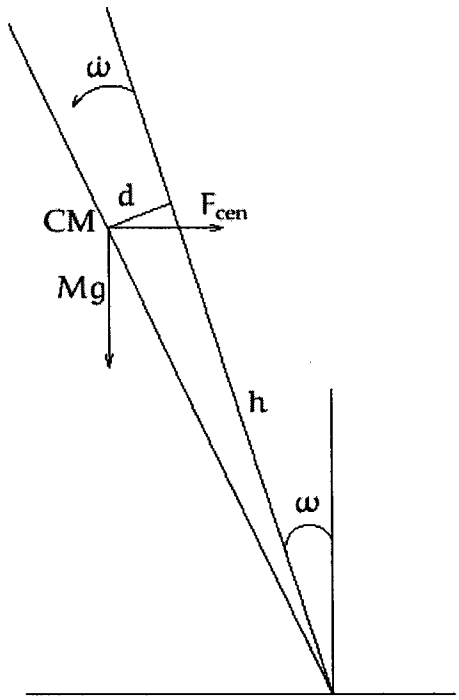


Figure 44: The bicycle as seen from behind. The thick line represents the bicycle. CM is the centre of mass of the bicycle and cyclist.

$$\ddot{\omega} = \frac{1}{I_{tot}}\left(Mhg\sin\varphi - \cos\varphi\left(I_{dc}\,\dot{\sigma}\dot{\theta} + \mathrm{sgn}(\theta)\cdot v^2\left(\frac{M_d r}{r_f} + \frac{M_d r}{r_b} + \frac{Mh}{r_{CM}}\right)\right)\right) \quad \text{(A.1)}$$

This equation is the mechanical equation for angular momentum. The physical contents of the right hand side are terms for the gravitation, effects of the conservation of angular momentum of the tyres and the fictional centrifugal force. The term $I_{dc}\,\dot{\sigma}\dot{\theta}$ is important for understanding why it is relative easier to ride a bicycle than to keep the balance on a bicycle standing still. The cross effects that originate from the conservation of angular momentum of the tyres stabilise the bicycle, and this effect is proportional to the angular velocity of the tyres $\dot{\sigma}$ and thereby to the velocity of the bicycle.

The angular acceleration $\ddot{\theta}$ of the front tyre and the handlebars is:

$$\ddot{\theta} = \frac{T - I_{dv}\,\dot{\sigma}\,\dot{\omega}}{I_{dl}} \tag{A.2}$$

These equations are not an exact analytical description, as some second (and higher) order terms have been ignored. The values of ω, $\dot{\omega}$, $\ddot{\omega}$, θ, $\dot{\theta}$ are sent to the agent at each time step. The agent returns the value of d and the torque T.

The front and back tyres follow different paths in a curve with different radii (see Figure 45). The front tyre follows the longest path. The radius for the front tyre is:

$$r_f = \frac{l}{|\cos(\frac{\pi}{2} - \theta)|} = \frac{l}{|\sin\theta|} \tag{A.3}$$

And for the back tyre:

$$r_b = l\left|\tan\left(\frac{\pi}{2} - \theta\right)\right| = \frac{l}{|\tan\theta|} \tag{A.4}$$

For the CM the radius can be calculated as:

$$r_{CM} = \left((l-c)^2 + \frac{l^2}{(\tan\theta)^2}\right)^{\frac{1}{2}} \tag{A.5}$$

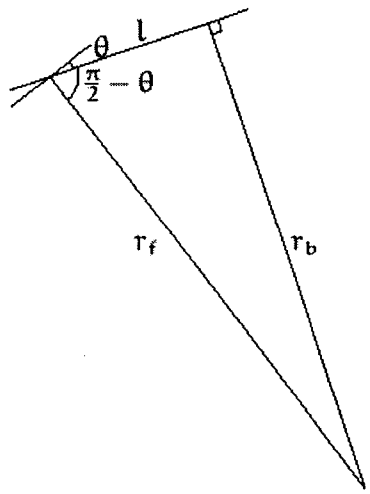


Figure 45: Seen from above. The thick line represents the front tyre.

The equations of the position of the tyres for the front tyre:

$$\begin{pmatrix} x_f \\ y_f \end{pmatrix}_{(t+1)} = \begin{pmatrix} x_f \\ y_f \end{pmatrix}_{(t)} + v\,dt \begin{pmatrix} -\sin(\psi + \theta + \mathrm{sgn}(\psi+\theta)\arcsin(\frac{v\,dt}{2r_f})) \\ \cos(\psi + \theta + \mathrm{sgn}(\psi+\theta)\arcsin(\frac{v\,dt}{2r_f})) \end{pmatrix} \tag{A.6}$$

And for the back tyre:

$$\begin{pmatrix} x_b \\ y_b \end{pmatrix}_{(t+1)} = \begin{pmatrix} x_b \\ y_b \end{pmatrix}_{(t)} + v\,dt \begin{pmatrix} -\sin(\psi + \mathrm{sgn}(\psi)\arcsin(\frac{v\,dt}{2r_b})) \\ \cos(\psi + \mathrm{sgn}(\psi)\arcsin(\frac{v\,dt}{2r_b})) \end{pmatrix} \tag{A.7}$$

Here ψ is the angle the bicycle has to the coordinate system. This number is only relevant if the agent is trying to learn to ride to a specific place.

We estimated the values of the moments of inertia to:

$$I_{tot} = \frac{13}{3}M_c h^2 + M_p\,(h + d_{CM})^2$$

The various moments of inertia for a tyre were estimated to (see Figure 46):

$$I_{dc} = M_d r^2$$
$$I_{dv} = \frac{3}{2} M_d r^2$$
$$I_{dl} = \frac{1}{2} M_d r^2$$

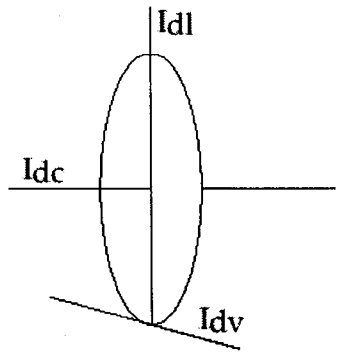


Figure 46: Axis for moments of inertia for a tyre.

Table 3 shows the values of the parameters used for the bicycle system.

Notation		Value
c	Horizontal distance between the point, where the front wheel touches the ground and the CM.	66 cm
CM	The Centre of Mass of the bicycle and cyclist as a total.	
d	The agent's choice of the displacement of the CM perpendicular to the plan of the bicycle.	
d_{CM}	The vertical distance between the CM for the bicycle and for the cyclist.	30 cm
h	Height of the CM over the ground.	94 cm
l	Distance between the front tyre and the back tyre at the point where they touch the ground.	111 cm
M_c	Mass of the bicycle.	15 kg
M_d	Mass of a tyre.	1.7 kg
M_p	Mass of the cyclist.	60 kg
ψ	Angle to the second axis.	
ψ_g	Angle to the goal.	
r	Radius of a tyre.	34 cm
$\dot{\sigma}$	The angular velocity of a tyre.	$\dot{\sigma} = \frac{v}{r}$
T	The torque the agent applies on the handlebars.	
v	The velocity of the bicycle.	10 km/h

Table 3: Notation and values for the bicycle system.

A.1.1 The simulation

Table 47 shows the simulation of the bicycle system. Since the update routine will have to be run a Very Large number of times it is a good idea to adjust for accumulation of rounding errors in the computer. Especially it seems the bicycle tend to 'shorten' if the positions of the tyres are left unrelated. In the original program this was compensated for right after the update of the tyre position (point 11) by:

$$l' = \sqrt{((x_f - x_b)^2 + (y_f - y_b)^2)}$$
$$x_b = (x_b - x_f)\frac{l - l'}{l'}$$
$$y_b = (y_b - y_f)\frac{l - l'}{l'}$$

---

1. Initialise $\omega = 0$, $\dot{\omega} = 0$, $\ddot{\omega} = 0$, $\theta = 0$, $\dot{\theta} = 0$, $\ddot{\theta} = 0$, $(x_f, y_f) = (0,0)$, $(x_b, y_b) = (l,0)$.
2. Translate the agent's choice of action into values for the distance d and torque T.
3. Calculate r_f, r_b and r_{CM} using equation (A.3), (A.4) and (A.5).
4. Calculate $\varphi = \omega + \arctan\left(\frac{d}{h}\right)$.
5. Calculate $\ddot{\omega}$ using equation (A.1).
6. Calculate $\ddot{\theta}$ using equation (A.2).
7. Update $\dot{\omega}$ and $\dot{\theta}$ by some suiting method. Using Euler's method: $\Delta\dot{\omega} = \ddot{\omega}\,dt$ and $\Delta\dot{\theta} = \ddot{\theta}\,dt$, where dt is a time step for the simulation.
8. Update ω and θ similarly.
9. If $|\omega| > \frac{\pi}{15}$ the bicycle has fallen. Tell the agent and return to 1.
10. If the handlebars have been turned more than 80°: $\theta = \text{sgn}(\theta)\frac{\pi}{180}80$.
11. Update (x_f, y_f) and (x_b, y_b) using equation (A.6) and (A.7).
12. Update $\psi = \arctan\left(\frac{x_b - x_f}{y_f - y_b}\right)$.
13. Jump back to 3. and repeat until a full time step in the 'real world' has passed, then let the agent do its part and return to 2.

Figure 47: The Bicycle simulation.

ANNEXE 2 : DÉTAILS DU SIF

UTILISÉ POUR LE PROBLÈME DU VÉLO

Le système d'inférence flou que nous avons utilisé pour le problème du vélo possède 6 entrées et 3 sorties. Les entrées sont les variables d'état :

- L'angle ω formé par l'axe du vélo et la verticale; $\omega_{\max} = -\omega_{\min} = 12°$.
- La vitesse angulaire $\dot{\omega}$; $\dot{\omega}_{\max} = -\dot{\omega}_{\min} = 30°/s$.
- L'accélération angulaire $\ddot{\omega}$; $\ddot{\omega}_{\min} = -\ddot{\omega}_{\max} = 115°/s^2$.
- L'angle θ de déviation du guidon par rapport à l'axe horizontal du vélo formé par les points de contact des deux roues avec le sol; $\theta_{\max} = \theta_{\min} = 60°$.
- La vitesse angulaire $\dot{\theta}$; $\dot{\theta}_{\max} = -\dot{\theta}_{\min} = 115°/s$.
- La distance S entre le point de contact de la roue arrière du vélo avec le sol et la droite parallèle à la position initial du vélo. $S_{\max} = -S_{\min} = 5m$.

Chaque variable d'entrée est définie par 3 étiquettes floues réparties uniformément sur son intervalle de variation tel qu'illustré par la figure A2-1 ci-dessous.

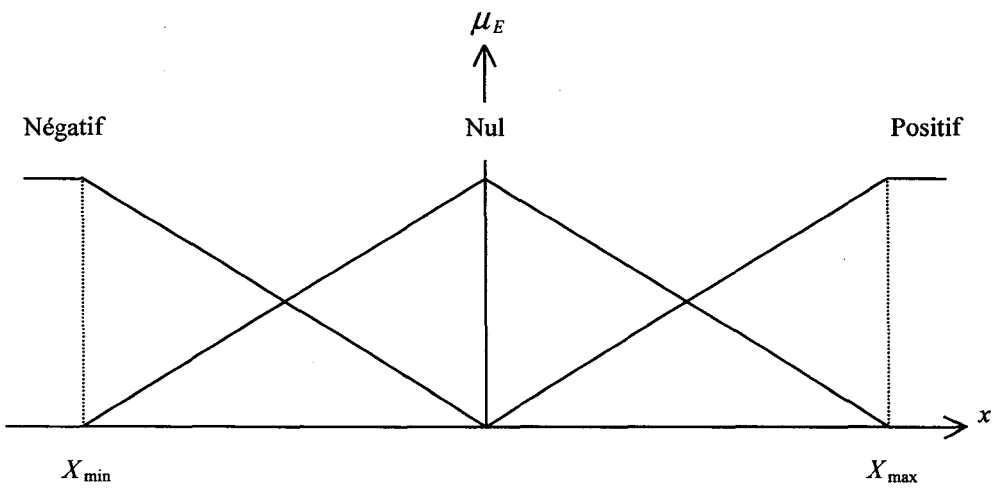


Figure A2-1 Étiquettes floues utilisée pour la partition des variable d'état.

Les 3 sorties numériques du SIF sont :

- La variable de contrôle d correspondant au déplacement du centre de masse du système (vélo + cycliste) . $d_{max} = -d_{min} = 2cm$. Les poids synaptiques correspondant à cette sortie sont initialisés de façon aléatoire.

- La variable de contrôle T correspondant au couple appliqué par le cycliste sur le guidon du vélo. $T_{max} = -T_{min} = 2N$. Les poids synaptiques correspondant à cette sortie sont initialisés de façon aléatoire.

- La sortie du critique V correspondant à la valeur (de la fonction d'évaluation) de l'état du système. $V_{max} = 0$; $V_{min} = -1$. Les poids synaptiques correspondant à cette sortie sont initialisés à zéro (valeur maximale).

Dans ce qui suit, nous allons donner des exemples de règles floues utilisées par l'agent avant et après l'apprentissage.

Exemple de règles floues du SIF avant l'apprentissage

Cas no 1 : S est Négatif

SI ω est Négatif ET $\dot{\omega}$ est Négatif ET $\ddot{\omega}$ est Négatif ET θ est Négatif ET $\dot{\theta}$ est Négatif **ALORS** T = -1.9N ET d = 1.08cm ET V = 0.

SI ω est Négatif ET $\dot{\omega}$ est Négatif ET $\ddot{\omega}$ est Négatif ET θ est Positif ET $\dot{\theta}$ est Positif **ALORS** T = -1.9N ET d = 0.3cm ET V = 0.

SI ω est Négatif ET $\dot{\omega}$ est Positif ET $\ddot{\omega}$ est Nul ET θ est Négatif ET $\dot{\theta}$ est Négatif **ALORS** T = -1.9N ET d = 1.2cm ET V = 0.

SI ω est Négatif ET $\dot{\omega}$ est Positif ET $\ddot{\omega}$ est Nul ET θ est Positif ET $\dot{\theta}$ est Positif **ALORS** T = 2.0N ET d = -0.1cm ET V = 0.

SI ω est Nul ET $\dot{\omega}$ est Négatif ET $\ddot{\omega}$ est Négatif ET θ est Négatif ET $\dot{\theta}$ est Négatif **ALORS** T = 1.6N ET d = -0.4cm ET V = 0.

SI ω est Nul ET $\dot{\omega}$ est Négatif ET $\ddot{\omega}$ est Négatif ET θ est Positif ET $\dot{\theta}$ est Négatif **ALORS** T = 0.2N ET d = 1.4cm ET V = 0.

SI ω est Nul ET $\dot{\omega}$ est Positif ET $\ddot{\omega}$ est Positif ET θ est Négatif ET $\dot{\theta}$ est Négatif **ALORS** T = 0.8N ET d = -1.0cm ET V = 0.

SI ω est Nul ET $\dot{\omega}$ est Positif ET $\ddot{\omega}$ est Positif ET θ est Positif ET $\dot{\theta}$ est Positif **ALORS** T = -1.6N ET d = 1.5cm ET V = 0.

SI ω est Positif ET $\dot{\omega}$ est Négatif ET $\ddot{\omega}$ est Nul ET θ est Négatif ET $\dot{\theta}$ est Négatif **ALORS** T = -0.2N ET d = -0.6cm ET V = 0.

SI ω est Positif ET $\dot{\omega}$ est Négatif ET $\ddot{\omega}$ est Nul ET θ est Positif ET $\dot{\theta}$ est Positif **ALORS** T = -0.2N ET d = 1.7cm ET V = 0.

SI ω est Positif ET $\dot{\omega}$ est Positif ET $\ddot{\omega}$ est Positif ET θ est Négatif ET $\dot{\theta}$ est Négatif **ALORS** T = 1.4N ET d = 1.3cm ET V = 0.

SI ω est Positif ET $\dot{\omega}$ est Positif ET $\ddot{\omega}$ est Positif ET θ est Positif ET $\dot{\theta}$ est Positif **ALORS** T = 1.9N ET d = 0.3cm ET V = 0.

Cas no 2 : S est Nul

SI ω est Négatif ET $\dot{\omega}$ est Négatif ET $\ddot{\omega}$ est Négatif ET θ est Négatif ET $\dot{\theta}$ est Négatif **ALORS** T = 0.20N ET d = -1.4cm ET V = 0.

SI ω est Négatif ET $\dot{\omega}$ est Négatif ET $\ddot{\omega}$ est Négatif ET θ est Positif ET $\dot{\theta}$ est Positif **ALORS** T = -0.5N ET d = 0.8cm ET V = 0.

SI ω est Négatif ET $\dot{\omega}$ est Positif ET $\ddot{\omega}$ est Nul ET θ est Négatif ET $\dot{\theta}$ est Négatif **ALORS** T = -1.5N ET d = -0.5cm ET V = 0.

SI ω est Négatif ET $\dot{\omega}$ est Positif ET $\ddot{\omega}$ est Nul ET θ est Positif ET $\dot{\theta}$ est Positif **ALORS** T = -02N ET d = 0.0cm ET V = 0.0.

SI ω est Nul ET $\dot{\omega}$ est Négatif ET $\ddot{\omega}$ est Négatif ET θ est Négatif ET $\dot{\theta}$ est Négatif **ALORS** T = -1.0N ET d = -0.1cm ET V = 0.

SI ω est Nul ET $\dot{\omega}$ est Négatif ET $\ddot{\omega}$ est Négatif ET θ est Positif ET $\dot{\theta}$ est Négatif **ALORS** T = -0.8N ET d = 1.7cm ET V = 0.

SI ω est Nul ET $\dot{\omega}$ est Positif ET $\ddot{\omega}$ est Positif ET θ est Négatif ET $\dot{\theta}$ est Négatif **ALORS** T = 0.9N ET d = -1.3cm ET V = 0.

SI ω est Nul ET $\dot{\omega}$ est Positif ET $\ddot{\omega}$ est Positif ET θ est Positif ET $\dot{\theta}$ est Positif **ALORS** T = -1.6N ET d = -0.73cm ET V = 0.

SI ω est Positif ET $\dot{\omega}$ est Négatif ET $\ddot{\omega}$ est Nul ET θ est Négatif ET $\dot{\theta}$ est Négatif **ALORS** T = 1.7N ET d = 0.9cm ET V = 0.

SI ω est Positif ET $\dot{\omega}$ est Négatif ET $\ddot{\omega}$ est Nul ET θ est Positif ET $\dot{\theta}$ est Positif **ALORS** T = 0.4N ET d = 1.8cm ET V = 0.

SI ω est Positif ET $\dot{\omega}$ est Positif ET $\ddot{\omega}$ est Positif ET θ est Négatif ET $\dot{\theta}$ est Négatif **ALORS** T = -1.5N ET d = 0.4cm ET V = 0.

SI ω est Positif ET $\dot{\omega}$ est Positif ET $\ddot{\omega}$ est Positif ET θ est Positif ET $\dot{\theta}$ est Positif **ALORS** T = -0.9N ET d = 1.8cm ET V = 0.

Cas no 3 : S est Positif

SI ω est Négatif ET $\dot{\omega}$ est Négatif ET $\ddot{\omega}$ est Négatif ET θ est Négatif ET $\dot{\theta}$ est Négatif **ALORS** T = -1.2N ET d = -1.8cm ET V = 0.

SI ω est Négatif ET $\dot{\omega}$ est Négatif ET $\ddot{\omega}$ est Négatif ET θ est Positif ET $\dot{\theta}$ est Positif **ALORS** T = 0.1N ET d = 1.1cm ET V = 0.

SI ω est Négatif ET $\dot{\omega}$ est Positif ET $\ddot{\omega}$ est Nul ET θ est Négatif ET $\dot{\theta}$ est Négatif **ALORS** T = -0.2N ET d = 2.0cm ET V = 0.

SI ω est Négatif ET $\dot{\omega}$ est Positif ET $\ddot{\omega}$ est Nul ET θ est Positif ET $\dot{\theta}$ est Positif **ALORS** T = 2N ET d = 1.0cm ET V = 0.

SI ω est Nul ET $\dot{\omega}$ est Négatif ET $\ddot{\omega}$ est Négatif ET θ est Négatif ET $\dot{\theta}$ est Négatif **ALORS** T = -1.2N ET d = -1.8cm ET V = 0.

SI ω est Nul ET $\dot{\omega}$ est Négatif ET $\ddot{\omega}$ est Négatif ET θ est Positif ET $\dot{\theta}$ est Négatif **ALORS** T = -1.3N ET d = -0.8cm ET V = 0.

SI ω est Nul ET $\dot{\omega}$ est Positif ET $\ddot{\omega}$ est Positif ET θ est Négatif ET $\dot{\theta}$ est Négatif **ALORS** T = 1.1N ET d = -1.5cm ET V = 0.

SI ω est Nul ET $\dot{\omega}$ est Positif ET $\ddot{\omega}$ est Positif ET θ est Positif ET $\dot{\theta}$ est Positif **ALORS** T = -0.2N ET d = 1.8cm ET V = 0.

SI ω est Positif ET $\dot{\omega}$ est Négatif ET $\ddot{\omega}$ est Nul ET θ est Négatif ET $\dot{\theta}$ est Négatif **ALORS** T = 0.6N ET d = -0.4cm ET V = 0.

SI ω est Positif ET $\dot{\omega}$ est Négatif ET $\ddot{\omega}$ est Nul ET θ est Positif ET $\dot{\theta}$ est Positif **ALORS** T = 0.8N ET d = -1.6cm ET V = 0.

SI ω est Positif ET $\dot{\omega}$ est Positif ET $\ddot{\omega}$ est Positif ET θ est Négatif ET $\dot{\theta}$ est Négatif **ALORS** T = -0.8N ET d = -0.8cm ET V = 0.

SI ω est Positif ET $\dot{\omega}$ est Positif ET $\ddot{\omega}$ est Positif ET θ est Positif ET $\dot{\theta}$ est Positif **ALORS** T = -1.9N ET d = 0.3cm ET V = 0.

Exemple de règles floues du SIF après l'apprentissage

Cas no 1 : S est Négatif

SI ω est Négatif ET $\dot{\omega}$ est Négatif ET $\ddot{\omega}$ est Négatif ET θ est Négatif ET $\dot{\theta}$ est Négatif **ALORS** T = -2.0N ET d = 1.2cm ET V = 0.006.

SI ω est Négatif ET $\dot{\omega}$ est Négatif ET $\ddot{\omega}$ est Négatif ET θ est Positif ET $\dot{\theta}$ est Positif **ALORS** T = -0.9N ET d = 2cm ET V = -1.

SI ω est Négatif ET $\dot{\omega}$ est Positif ET $\ddot{\omega}$ est Nul ET θ est Négatif ET $\dot{\theta}$ est Négatif **ALORS** T = -1.8N ET d = 0.9cm ET V = 0.015.

SI ω est Négatif ET $\dot{\omega}$ est Positif ET $\ddot{\omega}$ est Nul ET θ est Positif ET $\dot{\theta}$ est Positif **ALORS** T = 1.9N ET d = -0.1cm ET V = 0.001.

SI ω est Nul ET $\dot{\omega}$ est Négatif ET $\ddot{\omega}$ est Négatif ET θ est Négatif ET $\dot{\theta}$ est Négatif **ALORS** T = 1.6N ET d = -0.4cm ET V = 0.

SI ω est Nul ET $\dot{\omega}$ est Négatif ET $\ddot{\omega}$ est Négatif ET θ est Positif ET $\dot{\theta}$ est Négatif **ALORS** T = -2.0N ET d = 1.9cm ET V = -0.815.

SI ω est Nul ET $\dot{\omega}$ est Positif ET $\ddot{\omega}$ est Positif ET θ est Négatif ET $\dot{\theta}$ est Négatif **ALORS** T = -2.0N ET d = -2.0cm ET V = -0.570.

SI ω est Nul ET $\dot{\omega}$ est Positif ET $\ddot{\omega}$ est Positif ET θ est Positif ET $\dot{\theta}$ est Positif **ALORS** T = -1.6N ET d = 1.4cm ET V = -0.001.

SI ω est Positif ET $\dot{\omega}$ est Négatif ET $\ddot{\omega}$ est Nul ET θ est Négatif ET $\dot{\theta}$ est Négatif **ALORS** T = -0.1N ET d = -0.8cm ET V = 0.004.

SI ω est Positif ET $\dot{\omega}$ est Négatif ET $\ddot{\omega}$ est Nul ET θ est Positif ET $\dot{\theta}$ est Positif **ALORS** T = 1.1N ET d = 2.0cm ET V = -0.030.

SI ω est Positif ET $\dot{\omega}$ est Positif ET $\ddot{\omega}$ est Positif ET θ est Négatif ET $\dot{\theta}$ est Négatif **ALORS** T = 0.5N ET d = -2.0cm ET V = -1.053.

SI ω est Positif ET $\dot{\omega}$ est Positif ET $\ddot{\omega}$ est Positif ET θ est Positif ET $\dot{\theta}$ est Positif **ALORS** T = 1.9N ET d = 0.2cm ET V = -0.007.

Cas no 2 : S est Nul

SI ω est Négatif ET $\dot{\omega}$ est Négatif ET $\ddot{\omega}$ est Négatif ET θ est Négatif ET ET $\dot{\theta}$ est Négatif **ALORS** T = -0.1N ET d = -0.8cm ET V = 0.025.

SI ω est Négatif ET $\dot{\omega}$ est Négatif ET $\ddot{\omega}$ est Négatif ET θ est Positif ET $\dot{\theta}$ est Positif **ALORS** T = 0.4N ET d = 1.9cm ET V = -0.842.

SI ω est Négatif ET $\dot{\omega}$ est Positif ET $\ddot{\omega}$ est Nul ET θ est Négatif ET $\dot{\theta}$ est Négatif **ALORS** T = 0.1N ET d = -2.0cm ET V = -0.088.

SI ω est Négatif ET $\dot{\omega}$ est Positif ET $\ddot{\omega}$ est Nul ET θ est Positif ET $\dot{\theta}$ est Positif **ALORS** T = -0.6N ET d = 0.7cm ET V = 0.008.

SI ω est Nul ET $\dot{\omega}$ est Négatif ET $\ddot{\omega}$ est Négatif ET θ est Négatif ET $\dot{\theta}$ est Négatif **ALORS** T = -1.1N ET d = 0.3cm ET V = 0.004.

SI ω est Nul ET $\dot{\omega}$ est Négatif ET $\ddot{\omega}$ est Négatif ET θ est Positif ET $\dot{\theta}$ est Négatif **ALORS** T = -2.0N ET d = 2.0cm ET V = -1.298.

SI ω est Nul ET $\dot{\omega}$ est Positif ET $\ddot{\omega}$ est Positif ET θ est Négatif ET $\dot{\theta}$ est Négatif **ALORS** T = 2.0N ET d = -2.0cm ET V = -1.171.

SI ω est Nul ET $\dot{\omega}$ est Positif ET $\ddot{\omega}$ est Positif ET θ est Positif ET $\dot{\theta}$ est Positif **ALORS** T = 1.7N ET d = -1.1cm ET V = 0.005.

SI ω est Positif ET $\dot{\omega}$ est Négatif ET $\ddot{\omega}$ est Nul ET θ est Négatif ET $\dot{\theta}$ est Négatif **ALORS** T = 2.0N ET d = 0.0cm ET V = -0.006.

SI ω est Positif ET $\dot{\omega}$ est Négatif ET $\ddot{\omega}$ est Nul ET θ est Positif ET $\dot{\theta}$ est Positif **ALORS** T = -1.1N ET d = 2.0cm ET V = -0.100.

SI ω est Positif ET $\dot{\omega}$ est Positif ET $\ddot{\omega}$ est Positif ET θ est Négatif ET $\dot{\theta}$ est Négatif **ALORS** T = 2.0N ET d = -1.6cm ET V = -.950.

SI ω est Positif ET $\dot{\omega}$ est Positif ET $\ddot{\omega}$ est Positif ET θ est Positif ET $\dot{\theta}$ est Positif **ALORS** T = -1.0N ET d = 1.3cm ET V = 0.009.

Cas no 3 : S est Positif

SI ω est Négatif ET $\dot{\omega}$ est Négatif ET $\ddot{\omega}$ est Négatif ET θ est Négatif ET ET $\dot{\theta}$ est Négatif **ALORS** T = -1.2N ET d = 1.7cm ET V = 0.002.

SI ω est Négatif ET $\dot{\omega}$ est Négatif ET $\ddot{\omega}$ est Négatif ET θ est Positif ET $\dot{\theta}$ est Positif **ALORS** T = -1.4N ET d = 1.4cm ET V = -1.118.

SI ω est Négatif ET $\dot{\omega}$ est Positif ET $\ddot{\omega}$ est Nul ET θ est Négatif ET $\dot{\theta}$ est Négatif **ALORS** T = 0.18N ET d = 0.9cm ET V = -0.022.

SI ω est Négatif ET $\dot{\omega}$ est Positif ET $\ddot{\omega}$ est Nul ET θ est Positif ET $\dot{\theta}$ est Positif **ALORS** T = 0.6N ET d = 1.2cm ET V = -0.011.

SI ω est Nul ET $\dot{\omega}$ est Négatif ET $\ddot{\omega}$ est Négatif ET θ est Négatif ET $\dot{\theta}$ est Négatif **ALORS** T = -1.3N ET d = -1.7cm ET V = 0.

SI ω est Nul ET $\dot{\omega}$ est Négatif ET $\ddot{\omega}$ est Négatif ET θ est Positif ET $\dot{\theta}$ est Négatif **ALORS** T = -1.9N ET d = 2.0cm ET V = -0.678.

SI ω est Nul ET $\dot{\omega}$ est Positif ET $\ddot{\omega}$ est Positif ET θ est Négatif ET $\dot{\theta}$ est Négatif **ALORS** T = 2.0N ET d = -2.0cm ET V = -0.930.

SI ω est Nul ET $\dot{\omega}$ est Positif ET $\ddot{\omega}$ est Positif ET θ est Positif ET $\dot{\theta}$ est Positif **ALORS** T = -0.2N ET d = 1.7cm ET V = 0.001.

SI ω est Positif ET $\dot{\omega}$ est Négatif ET $\ddot{\omega}$ est Nul ET θ est Négatif ET $\dot{\theta}$ est Négatif **ALORS** T = 0.7N ET d = -0.5cm ET V = -0.001.

SI ω est Positif ET $\dot{\omega}$ est Négatif ET $\ddot{\omega}$ est Nul ET θ est Positif ET $\dot{\theta}$ est Positif **ALORS** T = -0.7N ET d = -1.3cm ET V = -0.011.

SI ω est Positif ET $\dot{\omega}$ est Positif ET $\ddot{\omega}$ est Positif ET θ est Négatif ET $\dot{\theta}$ est Négatif **ALORS** T = 2.0N ET d = -1.7cm ET V = -1.014.

SI ω est Positif ET $\dot{\omega}$ est Positif ET $\ddot{\omega}$ est Positif ET θ est Positif ET $\dot{\theta}$ est Positif **ALORS** T = -1.9N ET d = 0.2cm ET V = 0.004.

BIBLIOGRAPHIE

ANDERSON, C., HITTLE, D., KATZ, A., and R. KRETCHMAR (1996) Synthesis of Reinforcement Learning, Neural Networks, and PI Control Applied to a Simulated Heating Coil. *Journal of Artificial Intelligence in Engineering,* vol. 11, no. 4, pp. 423-431.

ATLASIS, A. and VASILAKOS (2002) The use of reinforcement learning algorithms in traffic control of high speed networks. In H. Zimmermann, G. Tselentis, M. Someren, and G. Dounias, *Advances in Computational Intelligence and Learning : Methods and Applications,* pp. 353-369. International series in intelligent technologies.

BAIRD, L. C. (1995) Residual algorithms : Reinforcement learning with function approximation. In *Proceeding of the Twelfth International Conference on Machine Learning*, pp. 30-37. Morgan Kaufmann, San Francisco.

BAIRD, L. C. (1999) Reinforcement learning through gradient descent. Doctoral Dissertation, Carnegie Mellon University, Pittsburgh.

BAIRD, L. C., and KLOPF, A. H. (1993) Reinforcement learning with high-dimensional, continuous actions. Tech. Rep. WL-TR-93-1147, Wright-Patterson Air Force Base Ohio: Wright Laboratory.

BARTO. A. G., SUTTON, R. S., and ANDERSON, C. W. (1983) Neuron-like elements that can solve difficult learning control problems. *IEEE Transaction on Systems, Man, and Cybernetics,* 13 :835-846.

BELLMAN, R. (1957) *Dynamic Programming.* Princeton University Press, Princeton, NJ.

BERENJI, H. R., and KHEDKAR, P. (1992) Learning and Tuning Fuzzy Logic Controllers Through Reinforcements. *IEEE Transaction on Neural Networks,* 3 :724-740.

BERTSEKAS, D. P. (1995) *Dynamic Programming and Optimal Control.* Athena Scientific, Belmont, Massachusetts. Volumes 1 and 2.

BOYAN, J. A. (1998) Learning Evaluation Functions for Global Optimization. Ph.D. thesis, Carnegie Mellon University.

BOYAN, J. A. (1999) Least-Squares Temporal Difference Learning. In Bratko, I., and Dzeroski, S., eds., *Machine Learning: Proceedings of the Sixteenth International Conference* (ICML'1999).

BRADTKE S. J., and M. O. DUFF (1995) Reinforcement learning methods for continuous-time Markov decision problems. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 393–400. The MIT Press.

BRATDKE, S. J. and BARTO, A. G. (1996) Linear least-squares algorithms for temporal difference learning. *Machine Learning*, 22:33-57.

CLOUSE, J., and UTGOFF, P. (1992) A teaching method for reinforcement learning systems. In *Proceeding of the Ninth International Conference on Machine Learning*, pp. 92-101. Morgan Kaufmann, San Mateo, CA.

CRITES, R. H., and BARTO, A. G. (1996) Improving elevator performance using reinforcement learning. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo (eds.), *Advances in Neural Information Processing Systems: Proceeding of the 1995 Conference*, pp. 1017-1023. MIT Press, Cambridge. MA.

DAYAN, P. (1992) The convergence of TD(λ) for general λ. *Machine Learning*, 8 :341-362.

DIETTERICH, T. (1998). The MAXQ Method for Hierarchical Reinforcement Learning. *Proceedings of the International Conference on Machine Learning, 1998.*

DIETTERICH, T. (2000). Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition, *Journal of AI Research*, 13 : 227-303

DRUMMOND, C. (2002) Accelerating Reinforcement Learning by Composing Solutions of Automatically Identified Subtasks, *Journal of Artificial Intelligence Research,* 16 : 59-104.

FOSTER, and D. DAYAN, P. (2002) Structure in the Space of Value Functions. *Machine Learning* 49 : 325-346.

GARCIA, F. and FLORENT, S. (2000). Efficient Asymptotic Approximation in Temporal Difference Learning. *European Conference on Artificial Intelligence ECAI'2000.*

GADALETA, S. AND DANGELMAYR, G. (1999). Optimal Chaos Control through reinforcement learning. *Chaos,* 9 : 775-1999.

HENGST, B. (2000). Generating Hierarchical Structure in Reinforcement Learning from State Variables. *Springer-Verlag, Lecture Notes in Artificial Intelligence series.*

HORIKAWA, S,. FURUHASHI, T., and UCHIKAWA, Y. (1992) On fuzzy modeling using fuzzy neural networks with the back-propagation algorithm. *IEEE Tran. Neural Networks*, 3(5)801-806.

HORNIK, K., STINCHCOMMBE, M., and WHITE, H. (1989) Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359-366.

HUI, T. and BROWN, T. (2002) Reinforcement Learning for Call Admission Control and Routing under Quality of Service Constraints in Multimedia Networks. *Machine Learning* 49: 111-139.

JANG, J. S. R. (1992) Self-learning fuzzy controllers based on temporal back propagation. *IEEE Tran. Neural Networks*, 3(5)714-726.

KAELBLING, L. P., LITTMAN, M. L., and MOORE, A. W. (1996) Reinforcement Learning : A survey. *Journal of Artificial Intelligence Research*, 4 :237-285.

KARR, C. (1991) Applying genetic algorithmes to fuzzy logics. *AI Expert*, march 1991, p. 38-43.

KELLER, J. M., YAGER R. R., and TAHANI, H. (1992) Neural Network implementation of fuzzy logic. *Fuzzy Sets Systems.*, 45:1-12.

KIMURA, H. and KOBAYASHI, S (1998) Reinforcement Learning for Continuous Action using Stochastic Gradient Ascent, The 5th International Conference on Intelligent Autonomous Systems, (IAS-5) pp.288--295.

KIMURA, H. and KOBAYASHI, S. (1999) Efficient Non-Linear Control by Combining Q-learning with Local Linear Controllers, 16th International Conference on Machine Learning, pp.210--219.

KOHONEN, T. K. (1989) Self-Organization and Associative Memory, 3rd ed. New York: Springer-Verlag.

KONDA, V. and TSITSIKLIS, J. (2000) Actor-critic algorithms. In NIPS 2000 editors, editor, Advances in Neural Information Processing Systems 12: Proceedings of the 1999 Conference. MIT Press.

KOSKO, B. (1992) Fuzzy systems as universal approximators. *Proc. IEEE Int. Conf. Fuzzy Syst.*, 1153-1162.

KRETCHMAR, R.M. (2000) A Synthesis of Reinforcement Learning and Robust Control Theory, Ph.D. Dissertation, Department of Computer Science, Colorado State University, Fort Collins.

LIN, C. J., and KIM, H. (1991) CMAC-based adaptive critic self-learning control. *IEEE Tran. Neural Networks*, 2(5)530-533.

LIN, C. T., and LEE, G. C. S. (1994) Reinforcement Structure/Parameter Learning for Neural-Networks-Based Fuzzy Logic Control Systems. *IEEE Transaction on Fuzzy Systems,* 2:46-63.

LIN, C. T. and LEE, C. S. (1996). *Neuro Fuzzy Systems*. Prentice Hall.

LIN, C. T., and JOU, C. P., GA-Based Reinforcement Learning for Control of a magnetic Bearing System, *IEEE Transaction on Systems, Man, and Cybernetics part B*, vol. 30 no. 2, pp. 276–289, April 2000.

MACLIN, R., and SHALIK, J. W. (1994). Incorporating advice into agents that learn from reinforcements. In *Proceeding of the Twelfth National Conference on Artificial Intelligence*, pp. 394-699. AAAI Press, Mentlo Park, CA.

MAMDANI E. H. (1977a) Application of fuzzy logic to approximate reasoning. *IEEE Tran. Computers,* 26:1182-1191.

MAMDANI E. H. (1977b) Application of fuzzy sets theory to control systems. in M. M. Gupta (Eds.) *Fuzzy Automata and Decision Processes*, 77-88, Amsterdam: North-Holland.

MARBACH, P., and TSITSIKLIS, J. N. (2001) Simulation-Based Optimization of Markov Reward Processes, *IEEE Transactions on Automatic Control*, Vol. 46, No. 2, pp. 191-209.

MATARIC, M. J. (1994). Reward functions for accelerated learning, *in* W. W. Cohen and H. Hirsh (eds.), *Machine Learning: Proceedings of the Eleventh International Conferenc*e, Morgan Kaufmann, CA.

MILLÁN, J. R., POSENATO, D., and DEDIEU, E. (2002) Continuous-Action Q-Learning. *Machine Learning*, 49 (2.3): 247-265,

MOORE, A. W., and ATKESON, C. G. (1993). Prioritized sweeping: Reinforcement learning with less data and less real time. *Machine learning*, 13:103-130.

NARENDRA, K. S. and THATHACHAR, A. L. (1970) Learning automata: a survey. *IEEE Tran. Syst. Man and Cybern.*, 4:323-334.

NG, A. Y. and M. JORDAN (2000) PEGASUS: A policy search method for large MDPs and POMDPs. In *Uncertainty in Artificial Intelligence*, Proceedings of the Sixteenth Conference.

PENG, J., and WILLIAMS, R. J. (1996). Incremental multi-step Q-leaning. *Machine Learning*, 22 :283-290.

SUTTON, R. S., PRECUP, D., and S. SINGH (1999) Between MDPs and semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning. In *Artificial Intelligence*, vol. 112, pp.181-211.

PRECUP, D., SUTTON, R., S., and DASGUPTA, S. (2001) Off-policy temporal-difference learning with function approximation. *In The Eighteenth Conference International Machine Learning (ICML'01)*, 417-424. Morgan Kaufmann.

RANDLØV, J., and ALSTROM, P. (1998). Learning to drive a bicycle using reinforcement learning and shaping. *Machine learning. Proceeding of the fifteenth international conference (ICML'98)*, 463-471.

RANDLØV, J., BARTO, A. G., and ROSENSTEIN, M. T. (2000). Combining Reinforcement Learning with a Local Control Algorithm. *Proceeding of the seventeenth international conference on machine learning (ICML'2000)*

RANDLØV, J. (2001). Solving Complex Problems with Reinforcement Learning, PhD Thesis, The Niels Bohr Institute, University of Copenhagen.

RUMELHART, D.E, HINTON, G. E., and WILLIAMS, R. J. (1986). Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland (Eds.). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. I, *Foundations*. Bradford/MIT Press, Cambridge, MA.

RUMMERY, G. A., and NIRANJAN, M. (1994) On-line Q-learning using connectionist systems. Technical Report CUED/F-INFENG/TR 166. Engineering Department, Cambridge University.

SANTAMARIA, J.C., SUTTON, R.S., RAM, A. (1998). Experiments with reinforcement learning in problems with continuous state and action spaces, Adaptive Behavior 6(2): 163-218.

SCHMIDHUBER, J. (1996) A general method for multi-agent learning and incremental self-improvement in unrestricted environments. In Yao, X. (Ed.), *Evolutionary Computation : Theory and Applications*. Scientific Publ. Co., Singapore.

SELFRIDGE, O.J., Sutton, R.S. and BARTO, A. G. (1985). Training and tracking in robotics. In A. Joshi (ed.), *Proceedings of the Ninth International Joint conference on Artificial Intelligence*, pp. 970-672. Morgan Kaufmann, San Mateo, CA.

SINGH, S. P. (1992). Transfer of learning by composing solutions of elemental sequential tasks. *Machine Learning*, 8(3):323-340.

SINGH, S. P. and SUTTON, R. S. (1994) Reinforcement learning with replacing eligibility traces. *Machine Learning*, 22:123-158.

SINGH, S. P. and BERTSEKAS, D. (1997) Reinforcement learning for dynamic channel allocation in cellular telephone systems. In *advances in Neural Information Processing systems: Proceeding of the 1996 Conference*, Pp. 974-980. MIT Press, Cambridge, MA.

SINGH, S., JAAKKOLA, T. LITTMAN, M. L. and SZEPESVÁRI, C. (2000) Convergence Results for Single-Step On-Policy Reinforcement-Learning Algorithms. *Machine Learning,* 38 (3): 287-308.

SINGH, S., LITMAN, D., KEARNS, M., WALKER, M. (2002) Optimizing Dialogue Management with Reinforcement Learning: Experiments with the NJFun System. *Journal of Artificial Intelligence Research,* 16:105-133.

SUGENO, M. and MURAKAMI, K. (1985) An experiment study of Fuzzy parking control using a model car. In M. Sugeno (Eds.), Industrial applications of fuzzy control, 125-138. Amsterdam: North-Holland.

SUTTON, R. S. (1988) Learning to predict by the method of temporal differences. *Machine Learning*, 3:9-44.

SUTTON, R. S., BARTO, A. G., and WILLIAMS, R. J. (1992) Reinforcement learning is direct adaptive optimal control. *IEEE Control Syst. Mag.* 12:19-22.

SUTTON, R. S. and BARTO, A. G. (1998) *Reinforcement Learning : An Introduction.* The MIT Press, Cambridge, Massachusetts.

SUTTON, R. S., MCALLESTER, SINGH, D. S, and MANSOUR, Y. (1999) Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems 12.* MIT Press.

TADIĆ, V. (2001) On the Convergence of Temporal-Difference Learning with Linear Function Approximation. *Machine Learning*, 42 (3): 241-267.

TAKAGI, and SUGENO, M. (1985) Fuzzy identification of systems and its applications to modeling and control. *IEEE Tran. Syst. Man and Cybern.*, 15(1):116-132.

TESAURO, G. J. (1992) Practical issues in temporal difference learning. *Machine Learning*, 8:257-277.

TESAURO, G. J. (1994) TD_Gammon, a self teaching backgammon program, achieves master-level play. Neural Computation, 6(2):215-219.

THRIFT, P. (1991) Fuzzy logic synthesis with genteic algorithms. *Proc. Of the third Int. Conf. On Genetic Algorithms*, p. 509-513.

THRUN, S. B. (1992) The role of exploration in learning control. In White, D. A., and Sofge, D. A. (Eds.), *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches.* Van Nostrand Reinhold, New York, NY.

TSITSILKIS, J. N. (1994) Asynchronous stochastic approximation and Q-Learning. *Machine Learning*, 8 (3).

TSITSIKLIS, J. N., and VAN ROY, B. (1997) An analysis of temporal-difference learning with function approximation. *IEEE Transaction on Automatic Control,* 42:674-690.

WANG, G. and MAHADEVAN, S. (1999). Hierarchical Optimization of Policy-Coupled Semi-Markov Decision Processes. *International Conference on Machine Learning (ICML-99).*

WANG, L. X. (1994) *Adaptive Fuzzy Systems and Control.* Englewood Cliffs, NJ: Prentice-Hall.

WATKINS, C. J. C. H. (1989) Learning from delayed rewards. Ph.D. thesis, Cambridge University.

WATKINS, C. J. C. H. and DAYAN, P. (1992) Q-learning. *Machine Learning*, 8:279-292.

WERBOS, P. J., (1990) A menu design for reinforcement learning over time. In W. T. Miller, III, R. S. Sutton, and P. J. Werbos (Eds.) *Neural Networks for Control*, chap. 3 Cambridge, MA: MIT Press.

WIDROW, B., GUPTA, N. K., and MAITRA, S. (1973) Punish/reward: Learning with a critic in adaptive threshold systems. *IEEE Tran. Syst. Man and Cybern.*, 3:455-465.

WIERING, M. and SCHMIDHUBER, J. (1998) Fast Online Q(λ). *Machine Learning*, 33 (1): 105-115.

WILLIAMS, R. J. (1988) On the use of backpropagation in associative reinforcement learning. *Proc. IEEE Int. Conf. Neural Networks*, 1:263-270, San Diego.

WILLIAMS, R. J. (1992) Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229-256.

WILLIAMS, R. J., and BAIRD, L. C. (1993) Analysis of Some Incremental Variants of Policy Iteration : First Steps Toward Understanding Actor-Critic Learning Systems. Technical Report NU-CCS-93-11. College of Computer Science, Northeastern University, Boston.

XU, X., HE, H. and HU, D. (2002) Efficient Reinforcement Learning Using Recursive Least-Squares Methods, *Journal of Artificial Intelligence Research*, 16 : 259-292.

ZADEH, L. A. (1965) Fuzzy sets. *Information and control*, 8:338-353.

ZADEH L. A. (1973) Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Tran. Syst. Man and Cybern.*, 15:28-44.

ZADEH L. A. (1975a) The concept of linguistic variable and its application to approximate reasonning, Part I. *Inf. Sci.* 9:199-249.

ZADEH L. A. (1975b) The concept of linguistic variable and its application to approximate reasonning, Part III. *Inf. Sci.* 9:199-249.

ZHANG, W., and DIETTERICH, T. G. (1996) High-performance job-shop scheduling with a time-delay TD(λ) network. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo (eds.), *Advances in Neural Information Processing Systems: Proceeding of the 1995 Conference*, pp. 1024-1030. MIT Press, Cambridge. MA.