



HAL
open science

Des comportements flexibles aux comportements habituels : Meta-apprentissage neuro-inspiré pour la robotique autonome

Erwan Renaudo

► **To cite this version:**

Erwan Renaudo. Des comportements flexibles aux comportements habituels : Meta-apprentissage neuro-inspiré pour la robotique autonome . Robotique [cs.RO]. Université Pierre et Marie Curie (Paris 6), 2016. Français. NNT: . tel-01698240

HAL Id: tel-01698240

<https://hal.science/tel-01698240>

Submitted on 1 Feb 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Une Thèse de doctorat par
Erwan RENAUDO

présentée à

Université Pierre et Marie Curie

École Doctorale Informatique, Télécommunications, Électronique (Paris)

Institut des Systèmes Intelligents et de robotique

pour obtenir le grade de

Docteur en Informatique

Des comportements flexibles aux comportements habituels

Meta-apprentissage neuro-inspiré pour la robotique autonome

6 juin 2016

Jury

David Filliat	– INRIA / ENSTA ParisTech Flowers	Rapporteur
Nicolas Rougier	– Mnemosyne, INRIA Bordeaux	Rapporteur
Jacques Malenfant	– LIP6, UPMC	Examineur
Costas Tzafestas	– IRAL, National Technical University of Athens	Examineur
Raja Chatila	– ISIR, UPMC / CNRS	Co-directeur de thèse
Mehdi Khamassi	– ISIR, UPMC / CNRS	Co-directeur de thèse
Benoît Girard	– ISIR, UPMC / CNRS	Invité
Veronique Serfaty	– Direction Générale de l'Armement	Invitée

Informations

Institut des Systèmes Intelligents et de Robotique
Université Pierre et Marie Curie
ISIR - CNRS UMR 7222
Boite courrier 173
4 Place Jussieu
75252 Paris cedex 05 - France

Mots clefs

Architecture de contrôle robotique • Robotique bio-inspirée • Apprentissage par renforcement • Comportement instrumental • Sélection de l'action

Résumé

Dans cette thèse, nous proposons d'intégrer la notion d'habitude comportementale au sein d'une architecture de contrôle robotique, ainsi que les mécanismes qui permettent de l'acquérir en parallèle du comportement planifié. Les architectures de contrôle robotiques adressent la question, dans le cadre robotique, d'incarner dans le monde réel le processus de planification des actions du robot. Elles coordonnent les capacités motrices de ce dernier avec ses capacités décisionnelles afin de rendre le comportement du robot réactif à son environnement mais également capable de prendre des décisions pour accomplir des buts à long terme (KORTENKAMP et SIMMONS 2008). Or, ces architectures sont pas ou peu dotées de capacités d'apprentissage leur permettant d'intégrer les expériences précédentes du robot, fonction qui est pourtant essentielle à la délibération (INGRAND et GHALLAB 2014). En neurosciences et en psychologie, l'apprentissage confirme cette idée : il a été montré comme une capacité essentielle pour adapter le comportement à des contextes changeants, mais également pour exploiter au mieux les contextes stables (DICKINSON 1985). Les différents types d'apprentissage sont modélisés par des algorithmes d'apprentissage par renforcement direct et indirect (SUTTON et BARTO 1998), combinés pour exploiter leurs propriétés au mieux en fonction du contexte (DAW et al. 2005).

Nous proposons premièrement une architecture de contrôle robotique inspirée de ces modèles, qui intègre en parallèle une capacité à planifier, à l'instar du comportement et la capacité d'apprendre et d'utiliser des habitudes comportementales. Nous montrons que, d'un point de vue robotique, la combinaison de ces algorithmes offre un avantage face à un changement de condition dans la tâche, en terme de robustesse de la performance. En revanche, l'hypothèse faite par les modèles du vivant que la capacité de planification permet de prendre des décisions plus informées que lorsque le comportement est habituel ne s'applique pas. Dans un second temps, nous proposons une série de méthodes pour coordonner ces capacités en fonction de l'information qu'à l'agent sur la stabilité du contexte. Ces méthodes n'améliorent pas la performance du robot en soi, mais nous permettent de mieux identifier les contraintes liées à la planification. Dans une troisième partie, nous étendons l'étude de notre architecture à d'autres tâches afin de généraliser l'intérêt de cette notion d'habitude pour la robotique, et confirmons qu'elle permet d'améliorer l'apprentissage du robot.

Remerciements

TODO.

Ce travail a été financé par une bourse de la Délégation Générale de l'Armement ainsi que par l'Agence Nationale de la Recherche (bourses ANR-12-CORD-0030, ANR-11-IDEX-0004-02) et Sorbonne-Universités SU-15-R-PERSU-14 PERSU.

Table des matières

Remerciements	5
Table des figures	10
Liste des tableaux	11
1 Introduction	13
1.1 Contexte général	13
1.2 Problématique	14
1.3 Plan de la thèse	14
2 Architectures de contrôle robotiques	17
2.1 Délibération en robotique	17
2.2 Architectures robotiques	19
2.3 Lien avec les architectures cognitives	23
2.4 Conclusion	23
3 Apprentissage par renforcement pour la prise de décision	25
3.1 Définition	26
3.2 Modélisation du problème	26
3.3 Programmation dynamique	30
3.4 Apprentissage par renforcement	33
3.5 Méthodes d'ensemble pour l'apprentissage par renforcement	39
3.6 Conclusion	41
4 Étude et modèles du comportement des mammifères, inspiration pour la robotique	43
4.1 Étude du comportement	44
4.2 Modèles du comportement instrumental	48
4.3 Conclusion	56
5 Résumé de l'état de l'art	59

6	Proposition d'une architecture robotique combinant comportement habituel et dirigé vers un but	61
6.1	Architecture proposée	62
6.2	Illustration expérimentale	70
6.3	Conclusion	77
7	Étude de critères pour le changement de comportement	79
7.1	Critères	80
7.2	Architecture	82
7.3	Résultats expérimentaux	84
7.4	Discussion	92
8	Extension à d'autres tâches	97
8.1	Expérience d'interaction	98
8.2	Expérience de navigation	107
8.3	Conclusion	121
9	Discussion et perspectives	123
9.1	Résumé des contributions	123
9.2	Discussions et perspectives	124
9.3	Conclusion	128
	Publications	129
	Bibliographie	131
	Glossaire	143
	Index	145

Table des figures

2.1	Shakey et Sense-Plan-Act	20
3.1	Modélisations de tâches en AR	28
3.2	Problème du chariot inversé	29
3.3	Planification, action, apprentissage & Dyna	37
4.1	Boîte de Thorndike	44
4.2	Dopamine et RPE	49
4.3	Tâches et possibilités de former des séquences	55
6.1	Architecture de contrôle proposée	63
6.2	Expert habituel	64
6.3	GD : Différentes méthodes de mise à jour des valeurs	66
6.4	Expérience de poussée de blocs	70
6.5	Évolution de l’environnement de simulation	71
6.6	Module de perception	72
6.7	Résultats : Récompense moyenne	75
6.8	Résultats : Performance	76
7.1	Récompense moyenne et entropie dans les experts seuls	81
7.2	Architecture modifiée : position du meta-contrôleur	83
7.3	Résultats : Performance méthodes basées signaux	85
7.4	Résultats : Performance méthodes basées fusion	86
7.5	Résultats : Histogrammes de performance	88
7.6	Résultats : Taux de sélection méthodes basées signaux	89
7.7	Résultats : Performance conversation ou oubli du plan	90
7.8	Résultats : Temps de planification conversation ou oubli du plan	91
7.9	[Résultats : Histogrammes conversation ou oubli du plan	92
7.10	Résultats : Distribution des jeux de paramètres	93
8.1	Architecture pour HRI	99
8.2	Plan HATP	100
8.3	Tâche d’interaction	101
8.4	Résultats : Performance	103
8.5	Résultats : Nombre d’actions décidées et évolution des poids	104

8.6	Résultats : Taux de sélection	105
8.7	Architecture avec inhibition sélective de la planification	107
8.8	Environnement de navigation	109
8.9	Navigation : point de vue du robot	110
8.10	Navigation : carte obtenue et états construits	111
8.11	Résultats : Performance des experts	114
8.12	Résultats : apprentissage de l'expert habituel	115
8.13	Résultats : Temps de planification par expert	116
8.14	Résultats : Performance de la combinaison	117
8.15	Résultats : Probabilité et taux de sélection des experts	118
8.16	Résultats : apprentissage de l'expert habituel combiné	119
8.17	Modèle fourni à l'expert dirigé vers un but	122
9.1	Comparaison KERAMATI et al. (2011) et RENAUDO et al. (2015c)	127

Liste des tableaux

6.1	Valeurs testées et choisies des paramètres des experts dans la première version de l'architecture.	73
7.1	Valeurs testées et choisies des paramètres des experts dans la seconde version de l'architecture.	84
8.1	Valeurs choisies des paramètres des experts dans l'expérience de navigation.	113



Chapitre 1

Introduction

1.1 Contexte général	13
1.2 Problématique	14
1.3 Plan de la thèse	14

1.1 Contexte général

Du bras robotique industriel enfermé dans sa cage à *BigDog*, qui évolue en forêt, le robot a vu son environnement changer, se déstructurer et devenir de plus en plus incertain. Les applications futures des robots tendent vers ce passage d'un contexte complètement prévisible à un contexte changeant et incertain. L'aspirateur robot s'invite au domicile pour réduire le temps consacré à la tâche nécessaire du nettoyage. La voiture autonome, dont de nombreux prototypes sont développés, propose de repenser notre rapport au déplacement et d'augmenter la sécurité des passagers. Le robot *mule*, qui déplace de lourdes charges d'un point à un autre ou en suivant une personne, a des applications potentielles dans le domaine militaire (WOODEN et al. 2010). Dans l'agriculture, les robots peuvent servir à automatiser les traitements des maladies des plantes (OBERTI et al. 2014) en extérieur. Enfin, plusieurs entreprises développent ou commercialisent des robots comme *compagnon de la famille*. En dehors de l'aspect « animal domestique artificiel », le robot compagnon a un intérêt pour le traitement de l'autisme (DAUTENHAHN et WERRY 2004 ; CHEVALIER et al. 2016) en étant une interface avec laquelle l'enfant a plus de facilités à interagir. Toutes ces applications ont en commun de demander au robot qu'il possède une capacité minimale à prendre des décisions pour s'adapter au nouveau contexte.

Cependant, dans les exemples évoqués précédemment, le robot reste en général spécialisé pour une tâche : il est donc programmé en fonction des cas d'utilisation et dotés d'une capacité de délibération minimale et suffisante pour gérer la majorité des autres cas auquel il sera confronté.

Or, avancer vers l'autonomie comportementale d'un robot nécessite de le doter de capacités de délibération adaptatives ; ces capacités doivent lui permettre d'agir correctement

dans les situations qu'il a déjà rencontré, mais également de trouver des solutions dans des situations nouvelles. Il est donc nécessaire que le robot intègre ses expériences quotidiennes pour pouvoir les exploiter dans le futur.

1.2 Problématique

Nous souhaitons donc aborder la question de la délibération adaptative en robotique au travers de la ré-exploitation des connaissances et comportements précédemment efficaces. Cette ré-exploitation permet d'une part de réduire le besoin d'en appeler au processus coûteux de planification lorsqu'un comportement déjà expérimenté permet de se comporter correctement ; d'autre part de concentrer ce processus de planification dans les situations où il est nécessaire, c'est-à-dire lorsque le contexte change vers une situation nouvelle. Pour cela, nous nous inspirons de l'étude des mammifères, qui montre une dichotomie dans le type de comportement employé par les animaux et dans le type de mécanisme d'apprentissage, en fonction de leur entraînement à accomplir une tâche donnée (e.g. leur familiarité à cet environnement, le temps qu'ils ont passé à y apprendre, l'incertitude et la nouveauté liés à cet environnement, et le caractère stationnaire ou au contraire dynamique de cet environnement) (DICKINSON 1985). Les modèles computationnels qui cherchent à expliquer les comportements d'apprentissage des mammifères s'appuient principalement sur la combinaison de plusieurs mécanismes d'apprentissage pour expliquer les différents types de comportements au sein d'un même agent (DAW et al. 2005). Nous détaillons au chapitre 4 ces modèles.

À partir de ces bases, nous proposons d'étudier une architecture robotique dont la couche de décision est dotée de la capacité d'apprendre des habitudes comportementales (dont le robot devra apprendre tout seul en fonction de quels niveaux de familiarité et de stabilité de l'environnement ces habitudes deviennent fiables et pertinentes) en plus de sa capacité à planifier. Nous nous intéressons en particulier à deux questions principales :

- Ce mécanisme d'apprentissage des habitudes a-t-il un intérêt pour la robotique ? Permet-il, par exemple, d'éviter effectivement temps et coût de calcul de la planification lorsque les habitudes comportementales sont efficaces ?
- Quelles sont l'organisation et les informations nécessaires au transfert du contrôle d'un comportement de type planifié à un comportement de type habituel dans un cadre robotique ?

1.3 Plan de la thèse

Ce manuscrit se compose de 8 chapitres.

- Dans le chapitre 2, nous présenterons le processus de délibération pour la robotique. Nous examinerons son évolution, d'abord comme restreint à la seule capacité à planifier, puis son incarnation dans des architectures de contrôle robotique, visant à permettre au robot d'agir efficacement dans le monde réel.

- Dans le chapitre 3, nous ferons un détour par le domaine de l'apprentissage automatique pour introduire l'Apprentissage par Renforcement (AR) qui sera à la base des modèles présentés dans la suite du manuscrit. Nous présenterons le problème d'apprendre par renforcement ainsi que les algorithmes classiques qui permettent d'y trouver des solutions. Nous évoquerons également les méthodes d'ensemble pour l'AR dans lesquelles s'inscrivent les modèles du chapitre suivant.
- Dans le chapitre 4, nous nous intéresserons au comportement des mammifères et à la dichotomie relevée entre comportement planifié (dit comportement dirigé vers un but) et comportement habituel, ainsi que leurs mécanismes d'apprentissage respectifs. Nous examinerons les modèles qui cherchent à expliquer cette dichotomie et représentent une source d'inspiration pour répondre à la problématique.

Les chapitres suivants présentent notre contribution :

- Dans le chapitre 6, nous proposerons une première version d'une architecture de contrôle robotique dotée d'une capacité d'apprentissage d'habitudes, que nous évaluons dans une tâche simulée, afin de valider le principe de coordonner au sein d'un robot une capacité de planification avec une capacité d'apprentissage d'habitudes.
- Dans le chapitre 7, nous étudierons un ensemble de méthodes pour coordonner décision planifiée et décision habituelle, et proposons une extension du processus de planification visant à le rendre plus efficace dans une tâche réaliste.
- Dans le chapitre 8, nous étendons notre architecture à d'autres tâches (interaction homme-robot et navigation), afin de vérifier l'intérêt de notre principe dans des contextes différents. Les résultats de navigation qui y sont présentés sont issus d'une expérimentation sur robot réel. Nous testons également dans cette expérience une autre méthode de coordination issue des constatations des résultats précédents.

Pour finir, nous discuterons dans le chapitre 9 des différents choix qui ont été faits au cours de ce travail et des différentes pistes à explorer pour outrepasser les limitations actuelles de l'architecture proposée.

Chapitre 2

Architectures de contrôle robotiques

2.1	Délibération en robotique	17
2.1.1	Planification	17
2.1.2	Raisonnement basé cas	18
2.2	Architectures robotiques	19
2.2.1	Approche délibérative	19
2.2.2	Approche réactive	19
2.2.3	Approche hybride	21
2.2.4	Planification dans les architectures robotiques	22
2.3	Lien avec les architectures cognitives	23
2.4	Conclusion	23

Dans ce chapitre, nous examinons la manière dont un robot autonome peut être doté d'une capacité de délibération qui lui permet de prendre des décisions face aux situations qu'il expérimente. Nous examinons d'abord les bases de la prise de décision en robotique, sous la forme des outils de planification automatique, puis les solutions qui permettent leur incarnation dans un robot agissant dans le monde.

2.1 Délibération en robotique

2.1.1 Planification

Les premiers travaux qui cherchent à donner des capacités de délibération aux robots s'inspirent directement des travaux de l'intelligence artificielle : ils se concentrent sur la capacité de planification du robot, et exploitent des langages basés sur la logique du premier ordre (comme STRIPS, (FIKES et NILSSON 1971)). Appliquée à des problèmes robotiques, cette planification vise à contrôler le robot, et consiste principalement à trouver l'enchaînement d'actions symboliques qui lui permettent d'atteindre un ou plusieurs buts. L'exemple le plus célèbre est le langage PDDL (MCDERMOTT et al. 1998), dérivé de STRIPS, et ses extensions, qui sépare d'un côté la description du domaine (*domain*

description) et de l'autre la description du problème (*problem description*). La première rassemble un ensemble de faits (*predicates*), qui décrit les relations existant dans le problème (ROOM(x) exprime le fait que x soit une pièce) ainsi que les opérateurs (ou actions), qui, à condition que leurs préconditions soient vérifiées, modifient ces faits (voir algorithme 2.1 pour un exemple d'opérateur). Le second décrit le problème à résoudre en terme d'état initial, d'états buts et d'objets présents (ces derniers pouvant être l'agent ou des parties de lui). Résoudre le problème revient à trouver la suite d'opérateurs qui mène de l'état initial aux états buts.

Algorithme 2.1 : Exemple d'opérateur PDDL

```
( :action MoveTo
:parameters (Agent A, Location L)
:preconditions
(ROOM(L) and not A.IsIn(L))
:effect
A.IsIn = L
)
```

Ce processus peut se faire en partant de l'état initial (*forward chaining*, correspond aux méthodes classiques de recherche dans un graphe), en partant de l'état but (*backwards chaining*). Enfin, pour résoudre les problèmes d'ordre de résolution des buts, qui apparaissent quand résoudre un but empêche d'en résoudre un autre, la planification partiellement ordonnée propose de ne planifier que les relations strictement nécessaires et de laisser l'ordre entre les plans indépendants se définir lors de leur exécution.

Les méthodes de planification s'étendent désormais au delà de ces seuls outils : l'intégration de la théorie des probabilités a permis de sortir du cadre déterministe ; les processus de décision markoviens (MDP) permettent une approche Bayésienne du problème de décision. Nous reviendrons sur ce formalisme dans le chapitre suivant.

2.1.2 Raisonnement basé cas

Une extension intéressante pour notre problématique est le raisonnement basé cas (CBR, BERGMANN et al. (1998)). Pour un problème défini, la planification classique cherche une suite d'opérateurs qui répond au problème d'aller de l'état initial à l'état final. Le CBR associe le problème avec sa solution trouvée par la planification. Face à un nouveau problème, cette approche enchaîne les étapes *Recherche-Réutilisation-Révision-Rétention* : *Recherche* consiste à chercher dans la *base des cas* pour essayer de trouver un problème similaire ; *Réutilisation* adapte la solution précédente au problème courant ; *Révision* consiste à évaluer le plan obtenu dans le problème pour vérifier son adéquation et l'absence d'effets indésirables ; *Rétention* ajoute ce nouveau couple problème-solution à la base des cas.

Le CBR représente une forme d'apprentissage par coeur des expériences précédentes. Une limite est qu'au fur et à mesure que la base de cas croît, l'algorithme s'attache

aux détails des solutions sans essayer d'extraire de ces solutions des règles récurrentes DE MANTARAS et al. (2005). Des développements plus récents proposent une approche statistique HÜLLERMEIER (2007), tandis que CBR a été couplé à de l'apprentissage par renforcement pour des problèmes de transfert de connaissances entre problèmes similaires CELIBERTO et al. (2011).

2.2 Architectures robotiques

2.2.1 Approche délibérative

Quand NILSSON (1969) applique STRIPS sur le robot *Shakey*, il découpe le problème du contrôle séquentiellement (figure 2.1), selon une architecture qualifiée de *Sense-Model-Plan-Act*. Le robot mesure son environnement à l'aide de ses capteurs pour former un modèle du monde, puis utilise ce modèle pour planifier une solution au problème auquel il est confronté. Une fois la solution trouvée, elle peut être exécutée partiellement et les actionneurs du robot utilisés pour agir sur l'environnement. Si NILSSON (1969) décrit cette organisation des capacités du robot comme « naturelle », elle souffre de limites importantes : créer un modèle pertinent est une tâche difficile, et calculer une solution prend du temps, durant lequel le robot est inerte et aveugle à son environnement. Si ce dernier est dynamique, l'exécution du plan est probablement obsolète et les actions du robot ne sont plus adaptées à la situation réelle.

2.2.2 Approche réactive

Brooks prend le contre-pied en proposant une approche à niveaux, et en montrant qu'il n'est pas nécessaire de planifier pour obtenir des comportements complexes. Il est ainsi possible d'obtenir un robot réactif en ayant une hiérarchie de comportements évalués en parallèle, à des échelles de temps différentes (BROOKS 1986). Ces *niveaux de compétence* gèrent chacun un type de comportement, et le principe de *subsumption* permet à des comportements de plus haut niveau d'inhiber les comportements plus bas en remplaçant leur sortie. Dans la même idée, Arkin propose pour la navigation les *motor schemas*, des comportements primitifs encodés sous la forme de champs de potentiel, dont l'interaction génère des comportements plus complexes (ARKIN 1989).

Cette approche réactive est cependant limitée par plusieurs facteurs. Dans la proposition de BROOKS (1986), les comportements ont un niveau de priorité défini, qui indique quels sont les comportements qu'ils subsument et ceux par qui ils sont subsumés. Ce mécanisme rigide est pointé par HARTLEY et PIPITONE (1991) comme source d'un manque de modularité de l'architecture et d'une dépendance des comportements de haut niveau envers ceux de bas niveau. La notion de priorité des comportements, qui génère la hiérarchie, est également peu pertinente lorsque les comportements sont équivalents (e.g. « atterrir » et « décoller » dans le cas d'un drone volant n'ont pas de relation hiérarchique).

Basées sur les travaux de Brooks, les architectures *basées comportements* étendent l'idée de combiner de multiples comportements simples pour créer des comportements au ni-

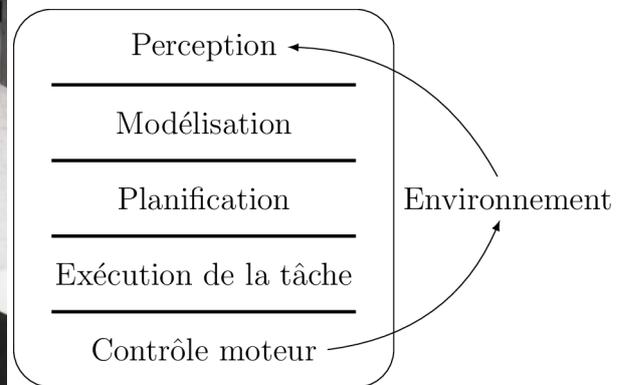
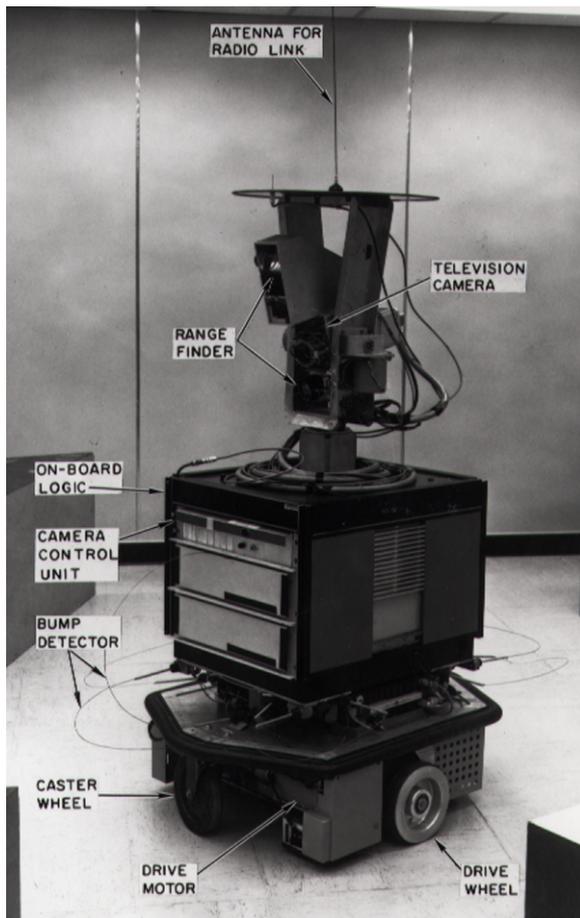


Figure 2.1 – *Gauche* : Photo Shakey sous licence CC BY-SA 3.0. *Droite* : Architecture « traditionnelle » Sense-Plan-Act, d'après BROOKS (1986)

veau d'un agent (MATARIĆ et MICHAUD 2008). Ces comportements simples peuvent être connectés à la fois aux entrées sensorielles du robot et à ses actionneurs comme aux entrées et sorties des autres comportements. En créant des réseaux de comportements, il est possible d'exploiter des représentations internes distribuées (e.g. une carte pour la navigation peut être représentée par un ensemble de comportements qui encodent la présence de stimulus; percevoir un stimulus déclenche le comportement moteur correspondant) et d'obtenir des comportements plus riches qu'avec les architectures purement réactives. Cette approche représente un niveau intermédiaire entre ces dernières et l'*approche hybride* que nous décrivons ensuite.

2.2.3 Approche hybride

L'évolution suivante consiste à réconcilier l'approche délibérative, directement inspirée de l'IA, et l'approche réactive. En effet, la nature réactive des comportements de cette dernière rend difficile la prise en compte de buts à long terme. La proposition d'une architecture *hybride* vise à bénéficier des avantages des deux approches. Arkin propose *AuRA*, qui couple ses *motor schemas* avec des planificateurs (de mission, spatial) et un séquenceur de plans (ARKIN et al. 1988; ARKIN 1990; ARKIN et BALCH 1997). FIRBY (1989) esquisse avec ses travaux sur les *reactive action packages* les prémisses des architectures hybrides à trois niveaux. BONASSO (1991) propose en parallèle de coupler un planificateur dont les opérateurs sont des comportements de subsomption.

Dans ce principe, une multitude d'architectures différentes ont été proposées. GAT (1998) justifie une organisation à trois niveaux par le besoin de faire l'interface entre le niveau symbolique, qui maintient des représentations à long terme, et le niveau sous-symbolique, qui s'appuie directement sur les données des capteurs. Ces différents niveaux portent plusieurs noms selon les travaux mais nous retiendrons les dénominations suivantes :

- Couche fonctionnelle (ou *couche des compétences, contrôleur*) : ce niveau correspond aux boucles sensorimotrices réactives qui forment les compétences du robot. Elles fonctionnent à une échelle de temps rapide, et gèrent les fonctions bas-niveau spécialisées (e.g. déplacement avec évitement d'obstacles, saisie d'un objet). Ces fonctions n'ont pas d'état interne, ou des états éphémères : ainsi, il n'y a pas d'accumulation des erreurs de modélisation au niveau du contrôle, et de déconnexion entre les informations utilisées pour le contrôle et le monde réel.
- Couche décisionnelle (ou *délibérative*) : ce niveau contient les capacités de planification à long terme du robot. Cette couche est chargée de déterminer les tâches à effectuer par le robot. Bien qu'elle soit en général chargée du calcul de plans à un niveau symbolique, on peut également y retrouver la planification de trajectoires lorsque le robot est dédié à des applications de robotique mobile.
- Couche exécutive (ou *séquenceur*) : Cette couche est l'interface entre les contrôleurs numériques de la couche fonctionnelle et les plans symboliques issus des planificateurs de la couche délibérative. Son rôle est d'exécuter le plan calculé, c'est-à-dire d'appeler, dans l'ordre, les modules de la couche fonctionnelle, et de superviser les actions effectuées, l'environnement, en accord avec le plan.

Une version à deux couches des architectures hybrides a également été proposée ; elle vise à corriger certaines limites de l'organisation à trois couches, identifiés par VOLPE et al. (2000, 2001) :

- la couche délibérative n'ayant pas accès aux raisons de l'échec de la couche fonctionnelle, cette information ne peut pas être utilisée pour corriger le plan établi. Plus généralement, l'information perdue entre les différentes couches conduit à des réactions du robot inadaptées à sa situation actuelle, notamment quand les connaissances de la couche délibérative ont été données a priori.
- note également la difficulté de définir une véritable hiérarchie entre les couches, avec des architectures où l'une domine (FIRBY 1989 ; BORRELLY et al. 1998 ; ESTLIN et al. 1999), et des travaux où planification et exécution se confondent (KNIGHT et al. 2000).

De nombreuses architectures ont été proposées sur ce principe : l'architecture 3^T est une évolution des travaux de BONASSO (1991) (BONASSO et al. 1997). ATLANTIS (GAT 1998) donne plus d'importance à la couche exécutive, qui est en charge d'appeler la couche délibérative quand nécessaire. L'architecture du LAAS (ALAMI et al. 1998) transfère le rôle de supervision de la couche exécutive vers la couche délibérative. CLARAty (VOLPE et al. 2000) est un exemple d'architecture à deux couches, dans laquelle la couche fonctionnelle est organisée selon le niveau d'abstraction des compétences du robot.

2.2.4 Planification dans les architectures robotiques

Deux approches principales ont été développées dans le cadre de la planification dans les architectures robotiques : l'approche HTN repose sur une hiérarchie entre des opérateurs « composés » qui permet de trouver un plan à la granularité des buts définis par le problème, puis de décomposer ces opérateurs comme un nouveau problème à résoudre (NAU et al. 1999). Ce nouveau problème est résolu en trouvant un plan qui sera lui-même décomposé s'il met en jeu des opérateurs composés. Ce processus est répété jusqu'à ce que les seuls opérateurs utilisés soient des opérateurs élémentaires, compréhensibles par la couche de supervision du robot. L'approche HTN vise à accélérer le temps de planification en considérant la hiérarchie inhérente à une tâche, plutôt que de la découvrir au moment de la planification.

L'approche planificateur / ordonnanceur séquence les tâches en accordant un intérêt particulier à l'usage des ressources et du temps CHIEN et al. (2000) ; MUSCETTOLA (1994). Pour permettre la réactivité de la couche délibérative, elle procède incrémentalement pour disposer à tout moment d'un plan exécutable si nécessaire.

Enfin, la relation au planificateur dans l'architecture peut être envisagée de deux façons différentes : comme une ressource que le superviseur peut solliciter pour obtenir un plan (comme dans GAT (1998)) ou comme un processus qui planifie et supervise en permanence pour s'adapter à d'éventuels problèmes (BONASSO et al. 1997). La première relation semble s'adapter principalement à la logique des HTN tandis que la seconde correspond plutôt à l'approche planificateur / ordonnanceur.

2.3 Lien avec les architectures cognitives

Bien qu'au moins aussi ancienne que l'étude des architectures robotiques, celle des architectures cognitives s'est développée en parallèle, en abordant des questions similaires sur les capacités qui forment la cognition d'un être vivant, et par extension d'un système artificiel. LANGLEY et al. (2008) cite, par exemple, la capacité de reconnaître et de catégoriser, de prendre des décisions, de prévoir à long terme ou d'agir dans le monde, sans oublier celle d'apprendre. KHAMASSI et al. (2016) identifie les points communs entre les problématiques de la robotique et celles considérées dans les sciences du vivant. Les architectures robotiques, initialement voulues pour être fonctionnelles, tendent naturellement vers les architectures cognitives comme le montrent les travaux qui adressent la question de l'architecture cognitive des robots (VERNON et al. 2007a; KURUP et LEBIERE 2012; VERNON et al. 2007b). Nous limitons notre étude aux architectures robotiques, celle des architectures cognitives constituant un domaine à part entière.

2.4 Conclusion

Nous avons présenté les bases de la planification en robotique, ainsi que les architectures qui leur permettent d'agir dans le monde. Les différentes approches des architectures de contrôle robotiques abordent chacune différemment la question de rendre la machine complexe qu'est le robot capable de fonctionner dans un environnement réel, bruité, imprévisible et changeant. Concevoir une architecture robotique qui répondrait à tous les problèmes auxquels est confronté le robot étant une tâche très difficile, le domaine a jusqu'à présent choisi d'adapter l'architecture à l'application voulue pour le robot (KORTENKAMP et SIMMONS 2008). L'évolution naturelle des architectures robotiques semble être les architectures cognitives, qui ont cherché, en parallèle des premières, à comprendre les mécanismes de la cognition des êtres vivants.

Chapitre 3

Apprentissage par renforcement pour la prise de décision

3.1	Définition	26
3.2	Modélisation du problème	26
3.2.1	Critère d'optimalité	27
3.2.2	Notion de valeur et politique optimale	29
3.3	Programmation dynamique	30
3.3.1	Policy Iteration	31
3.3.2	Value Iteration	31
3.4	Apprentissage par renforcement	33
3.4.1	Apprentissage direct	33
3.4.2	Apprentissage indirect	35
3.4.3	Compromis exploration/exploitation et sélection de l'action	38
3.5	Méthodes d'ensemble pour l'apprentissage par renforcement	39
3.5.1	Méthodes d'ensemble	39
3.5.2	Techniques d'agrégation	40
3.6	Conclusion	41

Dans ce chapitre, nous nous intéressons à l'Apprentissage par Renforcement (abrégé AR dans la suite) comme outil pour la prise de décision d'agents artificiels. Cette méthode d'apprentissage est utilisée aussi bien comme outil d'optimisation pour le contrôle optimal d'agent que comme outil de modélisation du comportement biologique (SUTTON et al. 1992). Comme nous le verrons dans le chapitre 4, l'AR est à la base de la plupart des modèles du comportement instrumental chez les mammifères ainsi que de leur application en robotique.

Nous commencerons par définir l'apprentissage par renforcement, donner le formalisme sur lequel il s'appuie et évoquer les différentes méthodes qui s'y rattachent. Nous nous concentrerons sur la dichotomie entre apprentissage direct et indirect (ou model-free et model-based comme nous le définirons plus tard) qui est au coeur des modèles de comportement et de notre travail, puis évoquerons les méthodes d'ensemble pour l'AR. Les

modèles que nous étudierons plus loin ainsi que la séparation faite par les architectures robotiques font que nous nous concentrerons sur les méthodes discrètes, mais nous discuterons de l'utilisation des méthodes continues dans le chapitre 9.

3.1 Définition

L'apprentissage par renforcement regroupe un ensemble de méthodes destinées à répondre à un problème de décision séquentielle. Étant donné un agent autonome plongé dans un environnement, les méthodes d'AR permettent à ce dernier d'apprendre un comportement (une suite de choix) optimal selon un critère, ici la maximisation d'un signal de récompense scalaire.

L'idée sous-jacente est que l'agent apprend en interaction avec l'environnement (SUTTON et BARTO 1998), et obtient une information partielle sur cette interaction : une évaluation de son comportement. L'apprentissage par renforcement se situe à mi-chemin entre l'apprentissage supervisé et l'apprentissage non-supervisé : si contrairement au premier, l'agent ne connaît pas la réponse à apporter dans une certaine situation, il obtient une indication sur l'intérêt de la réponse qu'il vient de fournir. On notera l'importance de la notion d'*essai-erreur* : l'agent doit agir (ou simuler son action) pour apprendre. Cela pose la question du compromis d'exploration-exploitation : à un état donné des connaissances de l'agent, doit-il choisir la solution qu'il a apprise comme étant de meilleure valeur, ou doit-il essayer une autre action, dans l'espoir de découvrir un comportement finalement plus intéressant ?

3.2 Modélisation du problème

Afin d'appliquer l'apprentissage par renforcement à une variété de problèmes, il est nécessaire de définir une modélisation de ces derniers qui puisse être manipulée par l'algorithme. L'approche usuelle consiste à modéliser le problème à l'aide des Processus de Décision Markoviens (MDP) (PUTERMAN 1994). Dans ce cadre, on définit le problème à l'aide des éléments suivants :

- un ensemble d'états \mathcal{S} , qui définit les situations dans lesquelles l'agent peut se trouver.
- un ensemble d'actions \mathcal{A} , qui représente les opérations qui peuvent faire évoluer la situation actuelle.
- une fonction de transition T qui exprime comment les opérations transforment les situations.
- une fonction de récompense R qui indique l'intérêt d'une action ou d'une situation particulière dans laquelle se trouve l'agent.

Le n -uplet $\langle \mathcal{S}, \mathcal{A}, T, R \rangle$ constitue un Processus Décisionnel de Markov. Dans ce problème, on appellera *politique* le modèle du comportement de l'agent et on le notera π . π est définie comme $\pi : \mathcal{S} \rightarrow \mathcal{A}$ si la politique est déterministe, ou $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ dans le cas stochastique. L'effet des actions sur l'état courant est représenté par la fonction $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ qui décrit la probabilité de passer d'un état à l'autre en faisant une

certaines actions. Le but que l'agent doit atteindre est représenté par la fonction $R : \mathcal{S} \rightarrow \mathbb{R}$ ou $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ où on associe des états ou des transitions désirables à une récompense positive, et des situations ou actions à éviter par l'attribution d'une récompense négative.

On suppose en général que la Propriété de Markov est respectée : dans un état $s \in \mathcal{S}$, l'agent possède toute l'information nécessaire pour prendre une décision, i.e choisir l'action $a \in \mathcal{A}$ qui lui permet de répondre au problème (maximiser la mesure de récompense). En général, l'environnement est considéré comme stationnaire, c'est-à-dire que \mathcal{T}, \mathcal{R} ne changent pas au cours du temps. Nous verrons que c'est une hypothèse qui ne tient pas dans la plupart des situations réalistes dans lesquelles peut se trouver un animal ou un robot, et en particulier dans les problèmes que nous considérons dans les chapitres suivants.

Dans les cas les plus simples, le problème est modélisé explicitement (par exemple en neuroscience, voir figure 3.1a ou en intelligence artificielle 3.1b), avec des états et actions directement représentatifs du problème (e.g. des états s_0 et s_1 abstraits reliés par des actions *Presser Levier* et *Entrer dans la Mangeoire* pour une tâche de modélisation du comportement). Il est également possible de définir la notion d'état comme un vecteur aléatoire de N descripteurs $S_k = (D_0, \dots, D_n)$. Ces derniers sont des variables aléatoires discrètes ou continues, et un état rencontré par l'agent correspond à un tirage pour chacun des descripteurs dans leur domaine respectif. Selon que l'on décrit les actions par la fonction de transition T ou directement par leur effet (e.g. " a_i rajoute 1 unité à la valeur de D_j "), on obtient une modélisation explicite ou implicite du problème. Enfin, le choix du point de vue influencera également la modélisation résultante : si on prend celui d'un observateur, on pourra décrire des transitions indépendantes de l'action d'un agent (par exemple l'apparition de nouvelles ressources qui modifient l'état courant dans lequel l'agent se trouve), alors que du point de vue de l'agent, ces événements représentent du bruit dans ses propres actions.

3.2.1 Critère d'optimalité

Une fois le problème formalisé, nous devons définir le critère d'optimalité qui permettra d'évaluer la politique suivie par l'agent et guider l'apprentissage. Puisque l'agent cherche à maximiser les récompenses reçues au cours de ses interactions, il est important qu'il prenne en compte le futur dans l'évaluation de sa politique plutôt que seulement la récompense instantanée r_t reçue. Il pourra ainsi discriminer entre deux options immédiatement équivalentes mais dont l'une mène à une récompense plus importante, ou éviter une option légèrement récompensée qui mène en fait à une punition importante.

Il existe plusieurs critères d'optimalité mais leur écriture la plus générale est celle du critère de récompense actualisée, présentée dans l'équation (3.1). Le paramètre $\gamma \in [0, 1]$ est le facteur d'atténuation et détermine la longueur de l'horizon considéré.

$$R_t = \sum_{i=0}^{\infty} \gamma^i \cdot r_{t+i+1} \quad (3.1)$$

Lorsque $\gamma = 0$, les politiques considérées comme optimales seront celles qui rapportent

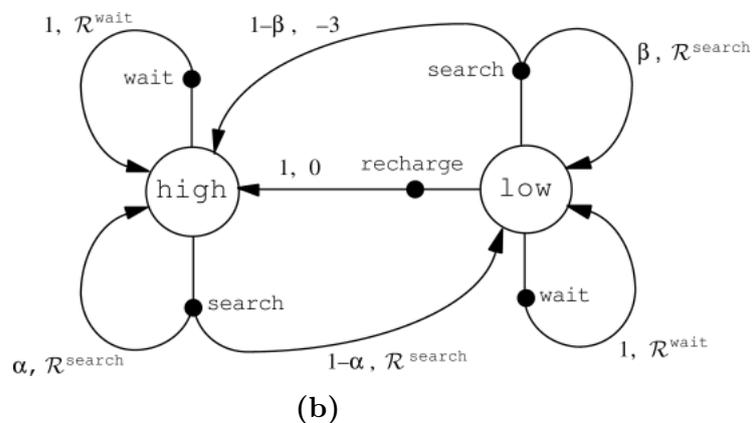
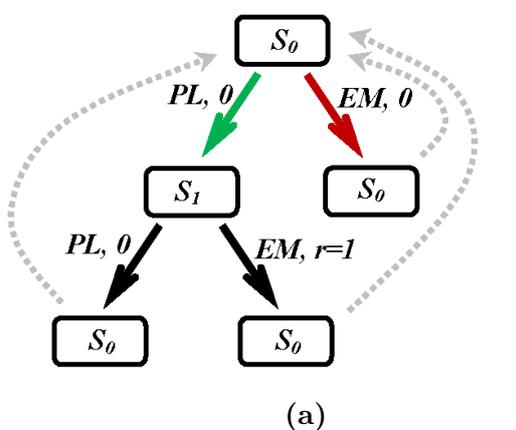


Figure 3.1 – 3.1a : Illustration de la modélisation d’une tâche de conditionnement instrumental dans une « Boîte de Skinner » sous forme de MDP. L’animal, dans un état initial S_0 , peut effectuer deux actions, Presser Levier (PL) et Entrer dans la Mangeoire (EM) qui ne rapportent pas de récompense ($r = 0$). Lorsqu’il a atteint l’état S_1 , mettre sa tête dans la mangeoire (EM) lui permet d’obtenir de la nourriture (ce qui est modélisé par une récompense positive $r = 1$) et il est ramené dans l’état S_0 . Image extraite de KERAMATI et al. (2011) sous licence Creative Commons Attribution. ; 3.1b : tâche de rechargement de batterie adaptée de SUTTON et BARTO (1998). Ici, les états correspondent explicitement à un état physique de la batterie de l’agent, et la tâche est stochastique : le couple de valeurs associé à chaque action représente la probabilité de chaque transition (“search” dans l’état high a une probabilité α de laisser l’agent dans le même état et $1 - \alpha$ de l’amener dans l’état low).

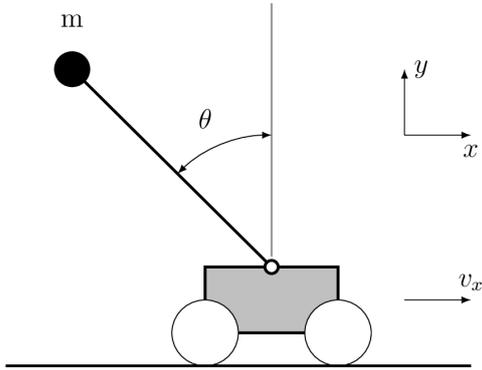


Figure 3.2 – *Problème du pendule inversé ou cartpole problem, décrit notamment dans (SUTTON et BARTO 1998, p59). Un chariot se déplace sur un axe x . Le but de l'agent est de maintenir à la verticale la tige fixée par un pivot sur le chariot.*

immédiatement de la récompense à l'agent, ce qui suppose que la fonction de récompense du problème donne de l'information à chaque étape de décision (ce qui se justifie dans des tâches où le comportement peut être évalué immédiatement, par exemple dans la tâche classique de pendule inversé (voir figure 3.2), mais revient à superviser l'apprentissage). Au fur et à mesure que γ tend vers 1, les politiques qui mènent à des récompenses de plus en plus lointaines regagnent de l'intérêt, ce qui valorisera la politique d'un agent qui agit à long terme.

Certains problèmes peuvent être qualifiés d'*épisodiques* : dans ce cas, l'agent doit effectuer une séquence d'actions avant d'arriver dans un état dit *terminal*. Le problème est alors réinitialisé et l'agent replacé dans un état initial (comme dans le cas illustré sur la figure 4.1a). Dans ces cas, l'équation (3.1) peut être simplifiée en :

$$R_t = \sum_{i=0}^T r_{t+i+1} \quad (3.2)$$

avec T l'étape où l'état terminal est atteint. Ce choix de modélisation est lié notamment au point de vue : une tâche qui semble épisodique à un agent peut en fait être modélisée de façon cyclique du point de vue d'un observateur. Sauf exception, nous considérerons dans la suite des problèmes continus, dans la mesure où nous nous intéressons à un robot plongé dans le monde, qui apprend sur le long terme.

3.2.2 Notion de valeur et politique optimale

À partir de la définition de l'optimalité d'une politique π et de la modélisation du problème, nous pouvons déterminer les *valeurs d'état* $V^\pi(s)$ (resp. *valeurs de transition* $Q^\pi(s, a)$) du problème. Ces dernières correspondent au retour moyen attendu à partir de l'état courant s (resp. en faisant l'action a dans s) si l'agent suit ensuite la politique π .

$$V^\pi(s) = E_\pi \{R_t | s_t = s\} = E_\pi \left\{ \sum_{i=0}^{\infty} \gamma^i \cdot r_{t+i+1} | s_t = s \right\} \quad (3.3)$$

$$Q^\pi(s, a) = E_\pi \{R_t | s_t = s, a_t = a\} = E_\pi \left\{ \sum_{i=0}^{\infty} \gamma^i \cdot r_{t+i+1} | s_t = s, a_t = a \right\} \quad (3.4)$$

En exprimant $V^\pi(s)$ en fonction de $V^\pi(s')$ (s' étant l'état suivant s) et en explicitant l'espérance, on peut écrire (3.3) sous la forme récursive suivante :

$$V^\pi(s) = \sum_{s' \in \mathcal{S}} T(s, \pi(s), s') (R(s, a, s') + \gamma V^\pi(s')) \quad (3.5)$$

Par cette expression, on comprend que la valeur d'un état correspond à la somme pondérée de la récompense immédiate qu'il peut y obtenir en faisant l'action proposée par π plus la valeur des états voisins atteignables depuis celui-ci. En pratique, plus un état sera proche de l'état récompensé, plus sa valeur sera élevée.

Avec l'évaluation de la politique donnée par V , il est possible de chercher la politique optimale. Comme précisé par (SUTTON et BARTO 1998, p75), une politique π est meilleure qu'une politique π' si son *retour attendu* est plus grand. La valeur des états V^π étant définie comme le *retour attendu moyen*, π est donc meilleure que π' si $\forall s \in \mathcal{S}, V^\pi(s) \geq V^{\pi'}(s)$. Si plusieurs politiques peuvent être qualifiées de meilleures, elles ne correspondent cependant qu'à une seule fonction de valeur optimale notée V^* et définie comme :

$$\forall s \in \mathcal{S}, V^*(s) = \max_{\pi} V^\pi(s) = \max_a \sum_{s' \in \mathcal{S}} T(s, a, s') (R(s, a, s') + \gamma V^*(s')) \quad (3.6)$$

et

$$\forall (s, a) \in \mathcal{S} \times \mathcal{A}, Q^*(s, a) = \max_{\pi} Q^\pi(s, a) = \sum_{s' \in \mathcal{S}} T(s, a, s') \left(R(s, a, s') + \max_b \gamma Q^*(s', b) \right) \quad (3.7)$$

dans le cas de la fonction de valeur de transition.

L'enjeu pour un agent est de trouver une politique optimale, donc la fonction de valeur optimale, en l'estimant au cours de ses interactions avec l'environnement du problème. Nous décrivons dans les sections suivantes les différentes méthodes pour estimer ces fonctions de valeur, en fonction des hypothèses faites sur l'information dont dispose l'agent.

3.3 Programmation dynamique

Si l'agent a accès au modèle de la tâche – c'est-à-dire qu'il connaît les fonctions de transition et de récompense de cette tâche –, accomplir cette tâche revient à résoudre un problème de planification pour trouver la politique qui maximisera le critère d'optimalité. Le but des méthodes de programmation dynamique est justement d'exploiter le modèle pour fournir une estimation des valeurs que l'agent pourra utiliser pour décider d'une politique.

Les deux algorithmes principaux pour calculer la politique optimale par programmation dynamique sont *policy iteration* (PI, HOWARD 1960) et *value iteration* (VI, BELLMAN 1957) (nous emploierons dans la suite les dénominations en langue anglaise).

3.3.1 Policy Iteration

Policy Iteration (PI) se décompose en deux étapes : l'évaluation de la politique (*policy evaluation*) et amélioration de la politique (*policy improvement*). La première étape permet de calculer la fonction de valeur V^π pour évaluer la politique courante. La seconde détermine la nouvelle politique en choisissant, dans chaque état, l'action qui maximise sa valeur. PI consiste à partir d'une politique initiale π_0 à mettre à jour successivement les valeurs puis la politique pour obtenir une politique optimale π^* qui correspond à la fonction de valeur optimale V^* . L'algorithme complet est donné dans l'algorithme 3.1.

Algorithme 3.1 : Policy Iteration (adapté de SUTTON et BARTO (1998))

Initialiser V^π et π arbitrairement

1 Évaluation de la politique

répéter

(a) $\Delta \leftarrow 0$

(b) **pour tous** $s \in S$ **faire**

i. $v \leftarrow V^\pi(s)$

ii. $V_\pi(s) \leftarrow \sum_{s'} T(s, \pi(s), s') [R^\pi(s) + \gamma V^\pi(s')]$

iii. $\Delta \leftarrow \max(\Delta, |v - V_\pi(s)|)$

fin

jusqu'à $\Delta < \varepsilon$ (*avec ε petit et positif*)

2 Amélioration de la Politique

stable \leftarrow *vrai*

pour tous $s \in S$ **faire**

(a) $b \leftarrow \pi(s)$

(b) $\pi(s) \leftarrow \operatorname{argmax}_a \sum_{s'} T(s, a, s') [R^\pi(s) + \gamma V^\pi(s')]$

(c) **si** $b \neq \pi(s)$ **alors** *stable* \leftarrow *faux*

fin

si non stable **alors** aller en 1

retourner V_π et π

3.3.2 Value Iteration

Plutôt que d'attendre la convergence de l'évaluation de la politique, il est possible de faire l'étape d'amélioration directement après l'avoir partiellement évaluée. En effet, après quelques itérations de l'évaluation, les estimations des valeurs sont suffisantes pour fixer la nouvelle politique si la décision est prise de manière gloutonne. *Value Iteration* (VI) pousse cette idée à l'extrême en n'effectuant qu'une seule étape d'évaluation puis en améliorant l'estimation de la fonction de valeur directement avec le retour maximal qui peut être attendu dans l'état s' .

Les propriétés de convergence de VI ne sont pas changées par rapport à PI. Selon les

Algorithme 3.2 : Value Iteration (adapté de SUTTON et BARTO (1998))

Initialiser V^π *arbitrairement*

Mise à jour des valeurs

répéter

 (a) $\Delta \leftarrow 0$

 (b) **pour tous** $s \in S$ **faire**

 1. $v \leftarrow V^\pi(s)$

 2. $V^\pi(s) \leftarrow \max_a \sum_{s'} T(s, a, s') [R_\pi(s) + \gamma V(s')]$

 3. $\Delta \leftarrow \max(\Delta, |v - V^\pi(s)|)$

fin

jusqu'à $\Delta < \varepsilon$ (*pour* ε *petit et positif*)

Calcul d'une politique déterministe

pour tous $s \in S$ **faire**

$\pi(s) \leftarrow \operatorname{argmax}_a \sum_{s'} T(s, a, s') [R_\pi(s) + \gamma V(s')]$

fin

problèmes, il est possible d'améliorer la vitesse de convergence en augmentant le nombre d'évaluations avant amélioration.

Ces deux méthodes permettent donc de trouver une solution optimale dans le cas d'un problème stationnaire pour lequel on dispose d'un modèle. Cependant, elles souffrent de plusieurs défauts :

- En dehors de problèmes bien déterminés, il est rare que l'agent ait accès au modèle T, R du problème directement, ces méthodes de calcul ne peuvent donc pas être utilisées. L'agent doit effectivement interagir avec son environnement pour obtenir de l'information et ne peut pas simuler en avance le résultat de sa politique.
- La dimension des espaces d'état et d'action peut rendre l'utilisation de ces méthodes très coûteuse : si on imagine un problème de navigation dans un labyrinthe de taille $1m \times 1m$, discrétisé en 5×5 carrés dans lesquelles l'agent peut aller dans 4 directions, on obtient déjà 100 couples état-action à évaluer. Si on augmente à 8 le nombre de directions possibles et qu'on divise la taille du côté des états par 2, on multiplie le tout par 8, pour un problème plutôt simple. Si on rajoute une variable binaire pour représenter un niveau de batterie, on se retrouve avec 1600 couples à évaluer. PI et VI devant itérer jusqu'à convergence, on se rend compte de l'effet de l'explosion combinatoire sur le coût de calcul des algorithmes.

Dans les sections suivantes, nous présenterons les méthodes d'apprentissage par renforcement en elles-mêmes, qui permettent de contrebalancer le premier défaut. Le deuxième peut être résolu à l'aide d'approximateurs de fonctions et de méthodes continues qui ne seront pas traitées dans ce manuscrit.

3.4 Apprentissage par renforcement

Comme nous l'avons évoqué, l'agent dispose rarement d'une version complète ou fiable du modèle du problème. Le modèle donné par le concepteur de l'agent peut ne représenter qu'une sous-partie de l'espace (e.g. une sous-partie d'un labyrinthe), manquer de précision (e.g. considérer comme déterministe la saisie d'un objet qui en fait échoue dans 15% des cas) ou on peut souhaiter ne pas donner de modèle et laisser l'agent l'estimer, pour accroître son autonomie de comportement dans des environnements nouveaux. Dans ces cas, il n'est pas possible de calculer directement $V(s)$ (ou $Q(s, a)$) par programmation dynamique. Deux solutions s'offrent cependant à l'agent : il peut chercher à estimer le modèle, pour pouvoir appliquer les méthodes présentées précédemment, ou chercher à estimer directement les valeurs. On parlera respectivement d'apprentissage indirect (voir 3.4.2) et direct (voir 3.4.1) du comportement, puisque soit l'agent apprend un modèle du problème pour en déduire ensuite le comportement, soit il apprend les valeurs et donc directement la meilleure action à faire dans chaque état.

3.4.1 Apprentissage direct

Lorsque l'agent ne dispose pas de modèle mais interagit effectivement avec l'environnement, il obtient un retour d'information sur le problème qu'il peut utiliser pour estimer la fonction de valeur. Cette information ne concernant que l'état précédent s , l'action effectuée par l'agent et le nouvel état courant s' , l'agent ne pourra mettre à jour que localement sa fonction de valeur à chaque étape, et devra expérimenter effectivement tous les états pour obtenir une estimation complète de la fonction de valeur.

Cette mise à jour à chaque interaction est appelée *apprentissage par différence temporelle* (nous utiliserons le sigle TD de l'anglais *Temporal Differences learning*) (SUTTON et BARTO 1987). Une fois de plus, il s'agit d'estimer V^π ou Q^π pour les états ou couples (état, action) connus.

$$V_{k+1}(s) = V_k(s) + \alpha \cdot \delta_k \quad (3.8)$$

avec $\alpha \in [0, 1]$ le taux d'apprentissage et δ_k l'erreur de prédiction sur la valeur de l'état s à l'étape k .

Dans le cas de problèmes stationnaires, on fait en général décroître α pour permettre la convergence des valeurs optimales. Cependant, les problèmes étant rarement stationnaires, il est également possible de garder le taux d'apprentissage constant pour prendre en compte d'éventuelles modifications dans le problème.

TD(0)

La méthode la plus simple est appelée *TD(0)* et consiste, une fois dans l'état s' , à mettre à jour directement la valeur de l'état précédent $V_{k+1}(s)$ en fonction de l'erreur entre le retour reçu dans l'état courant s' (récompense et valeur estimée de l'état courant) et la

prédiction de la valeur de l'état précédent $V_k(s)$. On peut ainsi évaluer une politique π au fur et à mesure de son déroulement, à l'aide de l'équation (3.9) :

$$V_{k+1}(s) = V_k(s) + \alpha \underbrace{\left(\overbrace{r(s) + \gamma V_k(s')}^{\text{retour reçu}} - \overbrace{V_k(s)}^{\text{prédiction}} \right)}_{\delta_k : \text{erreur de prédiction}} \quad (3.9)$$

Cette méthode ne prend en compte que la transition effectivement expérimentée, indépendamment des autres options dans l'état s' . Il n'est donc pas possible de modifier π sans plus d'information sur la manière dont les actions amènent d'un état à l'autre, c'est-à-dire un modèle du problème. Cependant, plutôt que de chercher à estimer le modèle, il est possible de transformer la fonction de valeur $V(s)$ sur les états en fonction de la valeur des couples (état, action) $Q(s, a)$.

SARSA

SARSA (*State-Action-Reward-State-Action*) (RUMMERY et NIRANJAN 1994; RUMMERY 1995) permet donc d'apprendre une fonction Q qui affine l'estimation de la valeur de l'état en une estimation de la valeur de la transition, et permet donc de comparer les actions entre elles. Dans ce cas, la mise à jour de la valeur $Q(s, a)$ nécessite l'information de récompense instantanée $r(s, a)$ reçue en passant de s à s' mais également la décision de l'agent dans s' pour obtenir un équivalent de $V(s')$ et évaluer l'état s' (d'où le nom de l'algorithme). Dans SARSA, on utilise $Q(s', a')$, ce qui donne l'équation suivante :

$$Q_{k+1}(s, a) = Q_k(s, a) + \alpha \left(r(s, a) + \underbrace{\gamma Q_k(s', a')}_{\equiv V^\pi(s')} - Q_k(s, a) \right) \quad (3.10)$$

Il est ainsi toujours possible d'évaluer une politique π à l'aide des valeurs $Q^\pi(s, a)$ mais également de la modifier en choisissant une nouvelle action dans s à l'aide des méthodes décrites plus loin (voir 3.4.3). SARSA apprend *sur politique* : l'évaluation de la valeur de chaque transition est dépendante de la transition suivante (donc de la politique suivie par l'agent). Selon le degré d'exploration de la méthode de décision choisie, l'évaluation de l'état suivant peut-être sous-optimale et la convergence non assurée si la décision prise n'est pas *gloutonne*. SINGH et al. (1998) indique cependant qu'à condition de faire tendre vers zéro le taux d'exploration et d'évaluer une infinité de fois chaque transition, la politique converge tout de même vers la politique optimale.

Q-learning

Pour rendre indépendantes les valeurs de transition et la politique suivie par l'agent, l'apprentissage doit se faire *hors politique*. Pour cela, plutôt que d'évaluer la valeur de l'état suivant s' en fonction de l'action qui y serait décidée, on peut l'évaluer en fonction

de l'action optimale, c'est-à-dire celle qui maximise la valeur dans s' . C'est la proposition faite par WATKINS (1989) (WATKINS et DAYAN 1992).

$$Q_{k+1}(s, a) = Q_k(s, a) + \alpha \left(r(s, a) + \underbrace{\gamma \max_b Q_k(s', b)}_{\equiv V^\pi(s')} - Q_k(s, a) \right) \quad (3.11)$$

L'algorithme n'est donc plus sensible aux actions exploratoires de l'agent puisqu'il évalue désormais les transitions en fonction des actions optimales connues à l'étape k . Ici encore, la politique converge vers la politique optimale dès lors que les transitions peuvent être évaluées une infinité de fois.

Approche Acteur-Critique

Il est également possible de séparer la fonction de valeur V de la politique π grâce à une approche dite Acteur-Critique (WITTEN 1977; BARTO et al. 1983; KONDA et TSITSIKLIS 2003). Cette approche combine un *critique*, chargé de maintenir la fonction de valeur et d'évaluer l'erreur de prédiction, et un *acteur*, qui maintient des valeurs de *préférence* $p(s, a)$ indiquant sa volonté de choisir l'action a dans l'état s . Le critique met à jour la fonction de valeur selon l'équation (3.8) et l'acteur met à jour la préférence de la transition expérimentée $p(s, a)$ avec l'erreur de prédiction définie par l'équation (3.9).

Bien que (SUTTON et BARTO 1998) présentent les méthodes Acteur-Critique comme apprenant *sur politique*, DEGRIS et al. (2012) a proposé récemment une version *hors politique* qui converge comme les autres méthodes du même type et obtient des performances similaires sur des problèmes classiques (pendule inversé, montée de la colline, labyrinthe continu). Nous discuterons dans les perspectives du chapitre 9 comment cette approche peut apporter un certain nombre de possibilités pour étendre notre travail.

3.4.2 Apprentissage indirect

L'autre option lorsque l'agent ne dispose pas d'un modèle du problème, c'est de lui donner les capacités pour l'estimer. Cet apprentissage se fait, comme pour l'apprentissage direct, au fur et à mesure que l'agent interagit avec l'environnement du problème.

Chaque transition mène d'un état s à un état s' par l'effet de l'action a , et chaque expérience $(s, a, s', r(s, a))$ peut être intégrée avec les précédentes pour construire un modèle que l'agent pourra utiliser pour estimer les conséquences de ses actions dans un état donné. Si le résultat d'une action est stochastique, et que le modèle qui en résulte prend en compte tous les états but possibles, on parle de modèle *de distribution*. S'il ne renvoie qu'une seule possibilité choisie selon la stochasticité des transitions, on parle de *modèle échantillonné*.

Une fois le modèle estimé, il devient possible de l'utiliser pour déterminer une politique, par exemple en appliquant les algorithmes de programmation dynamique évoqués plus haut (voir 3.3), mais également pour affiner la fonction de valeur, comme proposé dans Dyna-Q (SUTTON 1992).

L'intérêt d'apprendre un modèle est pointé par SUTTON (1992) comme justifié par le besoin de raisonnements « cognitifs » (raisonner sur les causes et conséquences des actions). Les approches aux différences temporelles se contentent d'apprendre un comportement et sa valeur dans le problème considéré. Or, comme le dit Sutton, l'information (sensorielle) présente dans les états et leurs relations est bien plus riche que le signal de récompense unidimensionnel utilisé par les approches TD.

La manière de construire un modèle est souvent dépendante du type de problèmes considérés : dans des cas discrets, une méthode simple est d'évaluer la statistique des expériences de l'agent. En maintenant pour chaque couple (s, a) un compte $C(s, a, s')$ des fois où s' a été atteint, on peut en déduire la probabilité de la transition (s, a, s') en normalisant sur tous les états atteints depuis s en faisant a :

$$C(s, a) = \sum_{u \in \mathcal{S}} C(s, a, u) \quad (3.12)$$

$$P(s'|s, a) = T(s, a, s') = \frac{C(s, a, s')}{C(s, a)} \quad (3.13)$$

De la même manière, l'agent peut construire un modèle qui estime en moyenne la fonction de récompense :

$$R(s, a) = \frac{\sum_t r_t(s, a)}{C(s, a)} \quad (3.14)$$

Cette approche est utilisée par exemple dans E^3 (KEARNS et SINGH 2002) et R-MAX (BRAFMAN et TENNENHOLTZ 2002), où un nouvel état est d'abord « évalué » sur ces statistiques avant son intégration complète au modèle.

Pour un problème de contrôle, avec états et actions continues, SCHAAL et ATKESON (1994) répondent au problème en faisant une régression locale pondérée (*LWR*, ATKESON et al. 1997). Contrairement à d'autres approches, le modèle n'est pas directement appris : les expériences de l'agent sont gardées en mémoire et un modèle local est calculé à la demande dans la sous-partie de l'espace considérée. Si leur approche leur permet de répondre au problème de montée de la colline, ils indiquent la nécessité d'explorer de manière non-aléatoire l'espace d'état pour accélérer l'acquisition de données pertinentes.

Les travaux présentés par HESTER et STONE (2012) confirment ce problème, et de manière plus général, montrent la nécessité de combiner la planification, la décision et l'apprentissage en ligne au sein d'une architecture. Le problème est déjà pointé par SUTTON et BARTO (1998), la réponse proposée étant les architectures Dyna (SUTTON 1990). Ces algorithmes sont à mi-chemin entre les approches directe et indirecte : l'agent met à jour ses valeurs directement en fonction de sa dernière expérience, ce qui assure une adaptation de la politique en ligne, et met en parallèle à jour son modèle pour un certain nombre de couples (s, a) connus et choisis aléatoirement (voir algorithme 3.3).

Des versions améliorées de Dyna comme *Prioritized Sweeping* (MOORE et ATKESON 1993) ou *Queue-Dyna-Q* (PENG et WILLIAMS 1993) améliorent les étapes de mise à jour du modèle en sélectionnant selon un certain ordre les couples (s, a) à ré-évaluer. La durée

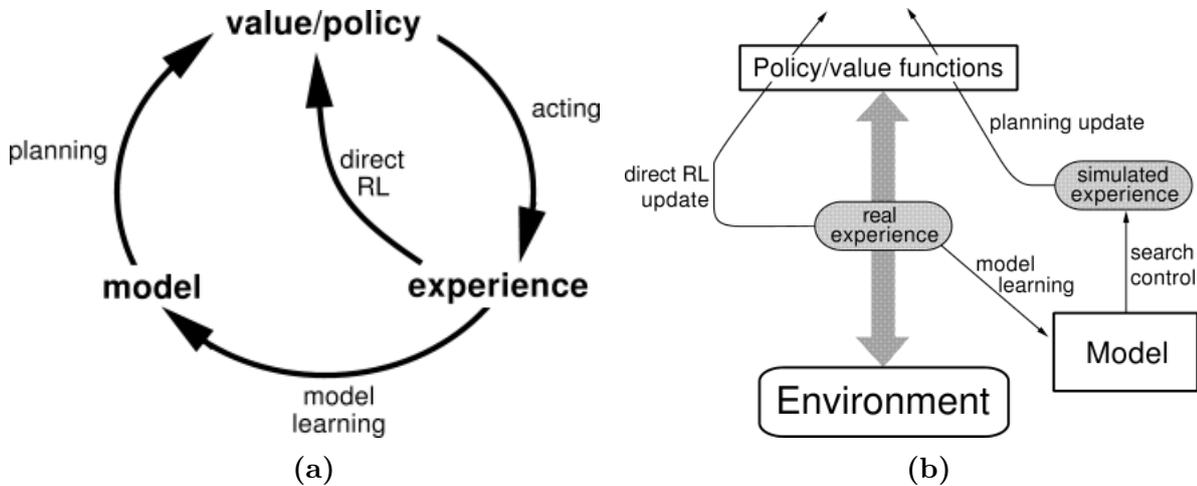


Figure 3.3 – 3.3a : Relations entre planification, action et apprentissage; 3.3b : Architecture Dyna; figures adaptées de SUTTON et BARTO (1998)

Algorithme 3.3 : Dyna-Q (adapté de SUTTON et BARTO (1998))

$\forall (s, a) \in \mathcal{S} \times \mathcal{A}(s)$, initialiser $Q(s, a)$ et $\text{Modele}(s, a)$ arbitrairement

tant que vrai faire

- (a) $s \leftarrow$ état courant (non-terminal)
- (b) $a \leftarrow \epsilon$ -greedy(s, Q)
- (c) Effectuer a ; observer le nouvel état s' et la récompense r
- (d) $Q(s, a) \leftarrow Q(s, a) + \alpha [(r + \gamma \max_b Q(s', b) - Q_k(s, a))]$ (voir (3.11))
- (e) $\text{Modele}(s, a) \leftarrow s', r$
- (f) **pour** N éléments **faire**
 - 1. $s \leftarrow$ aléatoirement tiré parmi les états observés
 - 2. $a \leftarrow$ aléatoirement tiré parmi les actions faites dans s
 - 3. $s', r \leftarrow \text{Modele}(s, a)$
 - 4. $Q(s, a) \leftarrow Q(s, a) + \alpha [(r + \gamma \max_b Q(s', b) - Q_k(s, a))]$

fin

fin

d'apprentissage en est réduite, ce qui permet à l'agent d'être plus performant sur les tâches discrètes évaluées.

3.4.3 Compromis exploration/exploitation et sélection de l'action

Dans les algorithmes que nous avons présentés précédemment, le calcul de la politique, et donc la sélection de l'action à faire dans un état s est en général déterministe, en prenant l'action a qui maximise le retour. Lorsque le MDP est entièrement connu, cette pratique garantit de trouver la politique optimale, mais quand le MDP est estimé ou que l'on s'appuie sur des méthodes directes, ce n'est plus le cas. En effet, les connaissances de l'agent sont formées par l'expérience de ses interactions avec l'environnement, et ne représentent donc potentiellement qu'un sous-espace du problème.

Dans ce cas, l'agent a parfois intérêt à choisir non pas l'action qui semble optimale a^* mais une action *a priori* sous-optimale qui lui permettra d'affiner sa connaissance du problème. On parle alors de « compromis exploration/exploitation ». Cette variabilité dans le choix est d'autant plus importante si l'agent est confronté à un problème non-stationnaire, dont la structure peut évoluer au cours du temps, et pour lequel ses connaissances passées (modèle comme fonction de valeur) deviennent fausses.

Répondre à ce compromis revient à sélectionner stochastiquement l'action que l'agent effectue dans un état donné. Une écriture plus générale du choix déterministe est la règle de sélection ϵ -greedy, décrite par l'équation (3.15). Pour ϵ une faible probabilité, et X une variable aléatoire suivant une loi uniforme sur $[0, 1]$ on a :

$$a = \begin{cases} \text{action aléatoire} & \text{si } X \leq \epsilon \\ \operatorname{argmax}_a Q(s, a) & \text{sinon} \end{cases} \quad (3.15)$$

Le problème de cette méthode est qu'elle choisit de manière équiprobable l'action lorsque l'agent explore. Une action connue comme de faible valeur (parce qu'elle amène vers des récompenses négatives, par exemple) pourra autant être sélectionnée qu'une action quasi-optimale. Afin de prendre en compte l'estimation de la valeur des actions connue par l'agent tout en conservant la possibilité d'explorer, la règle de sélection *softmax* transforme la distribution des valeurs en une distribution de probabilités P . Ces dernières sont obtenues en transformant les valeurs à l'aide d'une exponentielle normalisée ou *fonction softmax* (3.16) :

$$P(a|s) = \frac{\exp(Q(s, a)/\tau)}{\sum_{b \in \mathcal{A}} \exp(Q(s, b)/\tau)} \quad (3.16)$$

τ est la température de sélection, plus sa valeur est faible, plus l'agent tend à sélectionner l'action qu'il pense optimale, plus elle est grande, plus l'action sera choisie selon une distribution quasi-uniforme. On peut également voir τ comme un réglage de la sensibilité de la sélection au contraste des valeurs d'actions. Dans le reste du manuscrit, nous utiliserons la règle de sélection du softmax avec une valeur de τ fixée, mais des travaux s'intéressent à une évolution dynamique de sa valeur (KHAMASSI et al. 2011; AKLIL et al. 2014; TOKIC 2010), ou des autres paramètres en général (SCHWEIGHOFER et DOYA 2003), possibilité que nous évoquerons pour notre architecture (voir chapitre 8).

3.5 Méthodes d'ensemble pour l'apprentissage par renforcement

Les méthodes d'AR que nous avons présentées jusqu'à présent permettent à un agent d'apprendre une solution au problème modélisé en interagissant avec ce problème. Cependant, pour des problèmes de grande dimension (par exemple, le contrôle d'un bras robot pour le jonglage à partir d'un état composé des angle et vitesse de chaque articulation, décrit dans (KOBBER et al. 2013)) ou des domaines plus complexes, la performance de ces méthodes chute à cause de la difficulté d'estimer correctement la fonction de valeur. Certaines méthodes sont également plus sensibles que d'autres selon la structure du MDP (VAN HASSELT 2011, chap. 6).

3.5.1 Méthodes d'ensemble

Une solution consiste à mettre en parallèle plusieurs *experts* – chacun étant un algorithme qui cherche à répondre au problème – et à combiner leurs propositions pour décider du comportement final de l'agent. Ces méthodes sont appelées *méthodes d'ensemble* ou *mélange d'experts*. Elles sont appliquées à la fois en apprentissage supervisé (JACOBS et al. 1991 ; WOLPERT et KAWATO 1998 ; FREUND et SCHAPIRE 1999) pour lequel il existe une littérature conséquente (DIETTERICH 2000 ; SEWELL 2008) et dans une moindre mesure, en apprentissage par renforcement (SINGH 1992 ; DOYA et al. 2002 ; BALDASSARRE 2002 ; JIANG et KAMEL 2006 ; KHAMASSI et al. 2006). Dans le cadre de cette thèse, nous nous concentrerons sur les travaux qui s'appliquent à l'apprentissage par renforcement.

Les méthodes d'ensemble appliquées à l'AR ont reçu un regain d'intérêt récemment, tant pour l'apprentissage en ligne (WIERING et VAN HASSELT 2008) qu'hors-ligne (HANS et UDLUFT 2010). Elles sont également pointées comme une direction future intéressante pour utiliser l'AR de manière « hautement parallèle » (WIERING et OTTERLO 2012, p623), la structure des algorithmes s'adaptant bien à la multiplication des processeurs dans les ordinateurs.

Les algorithmes combinés peuvent être de même nature : DOYA et al. (2002) combine plusieurs algorithmes d'apprentissage par renforcement direct, FAUSSER et SCHWENKER (2011, 2013) des réseaux de neurones apprenant par différences temporelles. WIERING et VAN HASSELT (2008) combinent eux des algorithmes d'AR direct différents : Q-learning, SARSA, Acteur-Critique, QV-learning et ACLA (voir WIERING et VAN HASSELT 2007 pour les deux derniers). Dans ce dernier cas, les algorithmes ayant des règles d'apprentissage différentes, le biais de chaque algorithme vis-à-vis du problème sera différent (par exemple, étant donné les couples (s_1, a_1) et (s_2, a_2) successivement expérimentés par un agent et r_1 la récompense reçue après avoir fait (s_1, a_1) , SARSA calcule l'erreur aux différences temporelles en considérant la valeur de a_2 dans s_2 tandis que Q-learning prend la valeur maximale sur les actions possibles en s_2). HANS et UDLUFT 2010 combinent des réseaux de neurones NFQ (RIEDMILLER 2005) avec deux catégories de topologie. Le choix des algorithmes influence la manière de combiner leur résultat : si on peut combiner directement les fonctions de valeurs d'algorithmes de même nature, il est nécessaire de

raisonner au niveau de la politique quand elles sont calculées par des méthodes différentes.

3.5.2 Techniques d'agrégation

Une problématique essentielle de cette approche est donc la recherche de méthodes de combinaison ou d'agrégation des propositions des experts. L'objectif est d'obtenir une politique finale à partir de laquelle l'agent peut décider de sa prochaine action à faire. Dans un cadre hors-ligne, il est possible d'entraîner les agents indépendamment puis de les combiner une fois qu'ils ont convergé. En ligne, particulièrement dans un problème de robotique, il est plus difficile d'évaluer la convergence de la politique (une partie du MDP pouvant être inconnue de l'agent jusqu'à son exploration), on préférera donc prendre en compte la décision combinée des experts au cours de l'apprentissage.

WIERING et VAN HASSELT (2008) proposent quatre techniques d'agrégation pour combiner les politiques de N experts qui calculent un score de préférence p pour chaque action :

- *Majority Voting* : la préférence est calculée comme suit (a_t^e étant l'action la plus probable de l'expert e dans s_t) :

$$p_t(s_t, a[i]) = \sum_{e=1}^N I(a[i], a_t^e) \quad (3.17)$$

$I(x, y)$ est la fonction indicatrice qui vaut 1 quand $a[i] = a_t^e$ et 0 sinon. L'action la plus probable est donc l'action considérée comme meilleure par une majorité d'experts.

- *Rank Voting* : soit C_e l'ensemble des actions classées par ordre décroissant de probabilité de sélection pour l'expert e . On affecte à chaque action un poids $w_t^e(a[i])$ d'autant plus fort que l'action est bien classée, par exemple $|\mathcal{A}|$ pour la première, $|\mathcal{A}| - 1$ pour la seconde, etc. La préférence finale est la suivante :

$$p_t(s_t, a[i]) = \sum_{e=1}^N w_t^e(a[i]) \quad (3.18)$$

- *Boltzmann Multiplication* : la préférence d'une action est définie comme le produit de ses probabilités de sélection par chaque expert. Avec cette méthode, si l'une des actions a une probabilité nulle pour l'un des experts, sa préférence sera nulle indépendamment des valeurs calculées par les autres experts, mais les auteurs pointent que ce problème peut être évité si les experts calculent leur politique avec une fonction softmax.

$$p_t(s_t, a[i]) = \prod_{e=1}^N \pi_t^e(s_t, a[i]) \quad (3.19)$$

- *Boltzmann Addition* : cette méthode est une variante du *Rank Voting*, mais qui préserve les écarts relatifs entre actions : la préférence de chaque action est la somme de sa probabilité de sélection

$$p_t(s_t, a[i]) = \sum_{e=1}^N \pi_t^e(s_t, a[i]) \quad (3.20)$$

Pour (3.17) et (3.18), la politique finale est calculée en transformant les préférences en probabilités avec la fonction exponentielle normalisée (voir équation (3.16)). Pour (3.19) et (3.20), les valeurs sont simplement normalisées, la politique de chaque expert étant déjà générée par l'application de la fonction exponentielle normalisée sur les valeurs d'action. WIERING et VAN HASSELT (2008) évaluent leur architecture sur des problèmes de labyrinthes discrets (*Grid World*). Leur résultats montrent que *Majority Voting* et *Boltzmann Multiplication* permettent d'obtenir les meilleures performances d'ensemble sur les problèmes considérés, et surpassent les algorithmes testés individuellement.

HANS et UDLUFT (2010) confirment l'intérêt de *Majority Vote* pour des combinaisons de réseaux de neurones de topologies différentes dans le problème du pendule inversé 3.2. Ils évaluent également une méthode de moyennage des fonctions de valeur qui est moins performante que le *Majority Vote* mais meilleure que les agents pris individuellement.

Enfin, dans FAUSSER et SCHWENKER (2011, 2013) les auteurs testent également *Majority Vote* et le moyennage de la fonction de valeur, pour des ensembles de réseaux de neurones, dans les jeux *Tic-Tac-Toe*, *SZ-Tetris* et dans plusieurs labyrinthes discrets. Ils montrent également que les combinaisons d'experts sont plus performantes que les experts pris individuellement dans tous les cas. Leur travail récent suggère cependant qu'une étape d'amélioration supplémentaire consiste à faire de l'AR d'ensembles sélectionnés : plutôt que de fusionner les politiques de tous les experts, ils proposent de prendre un sous-ensemble des experts de sorte à optimiser un critère de qualité (dans leur cas, la qualité de chaque expert est inversement proportionnelle à la somme des erreurs de l'estimation de la fonction de valeur pondérées par la fréquence d'apparition du couple (s, a)) FAUSSER et SCHWENKER (2015). Si cette approche permet d'obtenir de meilleures performances qu'un ensemble entier, elle se limite pour l'instant à des problèmes discrets et le critère choisi la rend dépendante d'un modèle.

3.6 Conclusion

Nous avons décrit dans ce chapitre comment modéliser un problème de décision séquentielle à l'aide des Processus de Décision Markoviens, présenté deux catégories de méthodes de recherche de solution. La Programmation Dynamique permet, étant donné le modèle du problème, d'évaluer et d'améliorer la politique d'un agent. Lorsque ce modèle n'est pas explicitement disponible, il est possible soit d'estimer la valeur de la politique, soit d'estimer le modèle à l'aide des algorithmes d'Apprentissage par Renforcement et en interagissant directement avec l'environnement du problème. Dans tous les cas, le but de l'agent est de trouver la solution optimale au problème, ce qui dans ce cadre correspond à celle qui maximise la récompense reçue.

Nous avons également étendu notre état de l'art aux méthodes d'ensemble dans le cadre de l'apprentissage par renforcement. En combinant les politiques ou les fonctions de valeur de plusieurs algorithmes, la politique finale de l'agent est plus robuste et plus performante sur des problèmes plus complexes. Dans le chapitre suivant, nous nous intéresserons aux travaux de modélisation du comportement des mammifères basés sur l'apprentissage par renforcement et les méthodes d'ensemble.

Enfin, nous n'avons pas abordé directement la question de l'apprentissage par renforcement en robotique. S'il existe effectivement de nombreux travaux qui essaient d'appliquer l'AR dans ce domaine, ils emploient des méthodes continues, et se concentrent sur des problèmes bas-niveau spécifiques (e.g. marche multipode) (KOBBER et al. 2013). Nous nous intéressons plutôt à l'AR comme outil de décision séquentiel discret, décidant d'un comportement abstrait, et où le lien avec le monde réel est délégué aux autres éléments de l'architecture de contrôle robotique.

Chapitre 4

Étude et modèles du comportement des mammifères, inspiration pour la robotique

4.1	Étude du comportement	44
4.1.1	Conditionnements classique et instrumental	45
4.1.2	Premières hypothèses sur l'origine du comportement instrumental	45
4.1.3	Caractérisation des types de comportement	46
4.1.4	Liens entre apprentissage par renforcement et comportement instrumental.	47
4.2	Modèles du comportement instrumental	48
4.2.1	Axes de catégorisation et domaines considérés.	50
4.2.2	Modèles horizontaux	51
4.2.3	Modèles hiérarchiques	53
4.3	Conclusion	56

Dans ce chapitre, nous nous tournons vers l'étude du comportement des mammifères en psychologie et en neuroscience, et les modélisations proposées pour les comprendre. Nous nous intéressons d'abord aux premiers travaux qui ont mis en lumière la notion d'apprentissage par essai-erreur, puis comment ces travaux ont évolué vers les idées de comportements dirigés vers un but et habituels. Nous pointerons le lien fort entre l'apprentissage par renforcement présenté précédemment et les modèles du comportement, puis nous intéresserons à ces modèles qui adressent la question de la coordination des types de comportements. Nous examinerons pour cela aussi bien des modèles du comportement instrumental que des modèles de navigation, ainsi que les travaux de robotique qui s'inspirent de cette idée de coordination de différents systèmes.

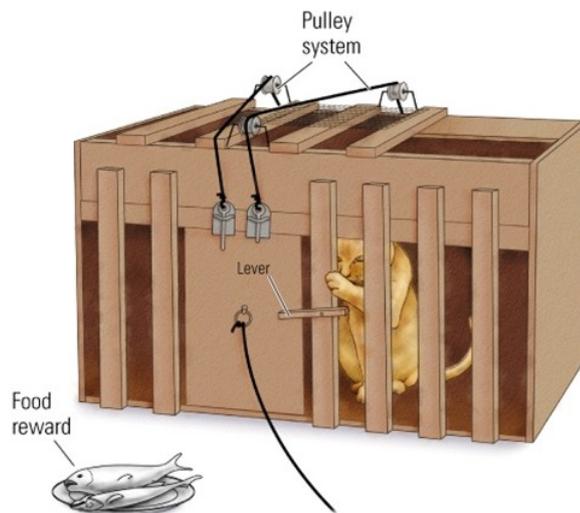


Figure 4.1 – Illustration de la boîte utilisée dans l'expérience de THORNDIKE (1911)

4.1 Étude du comportement

Les travaux fondateurs de l'étude du comportement des êtres vivants datent de la fin du XIX^{me} – début du XX^{me} siècle, avec de nombreuses expériences sur les mammifères. Ces expériences visent à comprendre comment ils se comportent et apprennent dans certaines situations bien contraintes. Ainsi, l'expérience du chat dans la boîte, de THORNDIKE (1911), met en lumière un aspect important du processus d'apprentissage : la notion d'essai-erreur. Dans cette expérience, un animal ayant faim est placé dans une boîte dont la porte peut être ouverte en tirant sur une chaîne puis en pressant un levier (figure 4.1). De la nourriture est placée à l'extérieur de la boîte, visible de l'animal. Initialement, ce dernier exhibe une variété de comportements, comme griffer la porte, donner des coups de patte mais également miauler ou se toiletter. Lorsqu'il réussit, au début par chance, à tirer la chaîne puis à presser le levier, la porte s'ouvre et il peut ainsi accéder à la nourriture. Il est ensuite remis dans la boîte dès lors qu'il a faim. Au fur et à mesure des répétitions de cette expérience avec le même animal, il est observé qu'il met de moins en moins de temps pour sortir de la cage.

Thorndike en formule l'hypothèse que la récompense que représente la nourriture renforce le comportement qui a permis d'y accéder ; il l'exprime comme la « Loi de l'Effet » :

Parmi les différentes réponses à une même situation, celles qui permettent (...) de satisfaire les désirs de l'animal seront, toutes choses égales par ailleurs, connectées plus fermement avec cette situation, de sorte que, lorsqu'elle se présente de nouveau, ces réponses réapparaissent plus probablement ; celles qui (...) sont suivies par une gêne pour l'animal (...) verront leur connexion réduite, de sorte à ce qu'il soit moins probable qu'elles réapparaissent. Plus la satisfaction ou la gêne expérimentées seront importantes, plus le renforcement ou l'affaiblissement sera important. (THORNDIKE 1911, p244)

En parallèle, Pavlov montre qu'il est possible de faire apprendre, avec suffisamment de répétitions, l'association entre un stimulus et une récompense à un animal. Ses travaux sont connus sous le nom de conditionnement Pavlovien, et avec ceux de Thorndike, posent les bases des travaux sur le comportement.

4.1.1 Conditionnements classique et instrumental

Dans l'approche comportementaliste (qui s'intéresse au comportement observable des individus), on fait la distinction entre conditionnement classique, ou Pavlovien (PAVLOV 1927) et conditionnement instrumental, ou opérant (SKINNER 1938). Le premier représente l'apprentissage d'une réponse automatique et involontaire à un stimulus initialement neutre. L'exemple célèbre du chien de Pavlov illustre comment la salivation à la vue de la nourriture (réponse inconditionnelle à un stimulus inconditionnel) peut être transférée à un stimulus initialement neutre (le son d'une cloche). La présence du stimulus neutre avant chaque présentation du stimulus inconditionnel, provoquant la réponse inconditionnelle, crée une association entre le stimulus neutre (qui devient un stimulus conditionnel) et la réponse. Ainsi, en l'absence de nourriture, la salivation est tout de même provoquée par le son de cloche, l'association cloche – salivation formant un *réflexe conditionnel*.

Le conditionnement instrumental diffère du conditionnement classique par le fait que les réponses comportementales de l'animal à un stimulus sont volontaires plutôt que réflexes : elles sont apprises à l'aide de renforcements positif ou négatif. L'action de l'animal se lie à son utilité, ses effets dans le contexte actuel. À l'aide d'une boîte de Skinner¹, il est possible de tester différents cas de conditionnement instrumental, selon le type de stimulus renforçateur qui est donné à l'animal : il peut recevoir un renforcement positif (par exemple, s'il obtient de la nourriture lorsqu'il fait l'action "appuyer sur un levier"), un renforcement négatif (l'animal reçoit des chocs électriques par le plancher de la boîte, qui sont stoppés lorsqu'il appuie sur le levier), une punition positive (il reçoit une décharge électrique lors de l'appui) ou une punition négative (la nourriture dont il disposait lui est retiré lors de l'appui). Dans les deux premiers cas, l'action considérée voit sa fréquence augmenter, tandis qu'elle diminue dans les deux derniers, de manière cohérente avec son effet et son intérêt pour l'animal.

Dans la suite, nous concentrons sur le conditionnement instrumental et les modèles computationnels qui visent à le décrire et l'expliquer.

4.1.2 Premières hypothèses sur l'origine du comportement instrumental

Les premiers travaux, dans le cadre de la navigation, opposent les théories héritées de la psychologie – où le comportement instrumental est supposé contrôlé par des associations apprises entre stimulus perceptuel et réponse motrice – et la vision de TOLMAN (1948) et des théoriciens des cartes cognitives – pour lesquels l'animal apprend une carte de

1. la boîte contient des sources de stimuli conditionnés - haut-parleur, lumières - ainsi que des leviers que l'animal peut actionner pour recevoir ou non une récompense

son environnement, hypothèse supportée par les expériences de BLODGETT (1929), et l'exploite lorsqu'il souhaite atteindre un endroit précis (par exemple, la localisation de nourriture).

Pour Tolman, l'observation de comportements VTE (*Vicarious Trial and Error*, ou « essai-erreur simulé ») est également une indication de l'utilisation d'une carte cognitive. Le comportement VTE correspond à un animal qui hésite, va et vient entre plusieurs chemins ; il est interprété comme l'indication que l'animal évalue les conséquences de plusieurs options. Cette hypothèse est confirmée par ADAMS et DICKINSON (1981) qui montrent qu'un rat ayant appris à presser un levier, motivé par l'obtention de sucrose, stoppe dès lors que ce sucrose rend l'animal malade. Cependant, le degré auquel l'animal est sensible à la conséquence de son action dépend de son degré d'entraînement (ADAMS 1982), le surentraînement le rendant complètement insensible.

Cela conduit à l'hypothèse que les deux manières de créer le comportement existent en parallèle, et qu'il y a un transfert de contrôle entre elles, ce qui est confirmé par DICKINSON (1985) d'un point de vue comportemental et renforcé par des études de lésions (BALLEINE et DICKINSON 1998).

4.1.3 Caractérisation des types de comportement

L'animal est considéré comme ayant un comportement *dirigé vers un but* à deux conditions : d'une part, ses actions reflètent un choix informé de leurs conséquences ; d'autre part, l'effet de la réponse doit être désirable pour l'animal au moment où il choisit son action. Le contrôle est qualifié de réponse-conséquence (*response-outcome*) et les actions sont réévaluées en fonction de leurs conséquences connues. Le comportement résultant est flexible et s'adapte aux changements qui surviennent dans l'environnement. Cette flexibilité se fait avec une réflexion active de l'agent, et en contrepartie d'un coût mental à évaluer les conséquences à un certain horizon (DAYAN 2009).

À l'inverse, l'animal a un comportement *habituel* lorsqu'il agit indépendamment des conséquences courantes de l'action, mais plutôt à cause d'un stimulus perceptuel. Au fur et à mesure de l'expérience de l'agent, les conséquences des actions ont été intégrées dans le processus de décision, et un changement dans ces conséquences (e.g. presser un levier ne rapporte plus de nourriture) ne produit pas immédiatement un changement dans le comportement de l'animal. Le comportement *habituel* est caractérisé par son automatisation, sa robustesse au changement, mais également son faible coût mental à évaluer les différentes options.

Deux manières d'identifier le type du comportement observé sont principalement appliquées dans la littérature : la dévaluation de la conséquence (*outcome devaluation*) et la dégradation de contingence (*contingency degradation*). La dévaluation de la conséquence consiste à rendre neutre ou répulsif l'élément qui était précédemment considéré comme une récompense (donc valué positivement). Cela est fait en nourrissant l'animal à satiété ou en le rendant malade lorsqu'il consomme la nourriture précédemment donnée comme récompense. Après une phase d'apprentissage sur une tâche récompensée, on applique une phase de dévaluation, puis on teste l'animal en *extinction*, c'est-à-dire sans donner la récompense associée précédemment aux actions. Si l'animal a un comportement *dirigé*

vers un but, le taux de réalisation des actions qu'il effectuait pour obtenir la récompense diminue puisque l'obtention de la nourriture n'est plus intéressante pour l'animal. Si au contraire, il persiste dans les mêmes actions, le comportement est considéré comme habituel, puisque les actions ne sont plus liées par leur conséquence, mais choisies dans des situations familières pour l'agent. La dégradation de contingence consiste à modifier les conséquences d'une action après apprentissage par l'animal, par exemple en supprimant la délivrance de nourriture ou en la rendant non-conditionnée par l'action. Comme pour la dévaluation de la conséquence, si l'animal persiste à faire les actions qu'il a apprises précédemment, son comportement est considéré comme *habituel*, s'il adapte son comportement aux nouvelles conditions de délivrance de la récompense, son comportement est considéré comme *dirigé vers un but*.

Ces résultats sont également étendus à l'humain : VALENTIN et al. (2007) montre que l'on retrouve l'effet de la dévaluation de la récompense dans le cortex orbitofrontal chez l'homme, tandis que TRICOMI et al. (2009) parvient à montrer la même désensibilisation à la conséquence de l'action que chez l'animal chez des sujets humains après un entraînement intensif.

Nous ne rentrerons pas dans les détails des substrats neuronaux qui supportent ces comportements, cependant, les études de lésions suggèrent que le comportement *dirigé vers un but* est supporté notamment par le *striatum dorsomédial* (DMS) tandis que le comportement *habituel* est supporté notamment par le *striatum dorsolatéral* (DLS), sous-parties des ganglions de la base (YIN et KNOWLTON (2006), voir LIÉNARD (2013, chap 1) pour une présentation détaillée de l'anatomie des ganglions de la base). Par ailleurs, la partie ventromédiale du Cortex Préfrontal (vmPFC) semble supporter les connaissances de l'animal sur la tâche, soit l'équivalent d'un modèle (HAMPTON et al. 2006). De la même manière, l'activité est transférée entre DMS et DLS avec l'expérience (YIN et al. 2009 ; THORN et al. 2010), ce qui peut être rapproché d'observations similaires dans des tâches spatiales (PACKARD et MCGAUGH 1996).

Des résultats plus récents montrent également que s'il y a bien une évaluation des valeurs dans les deux parties du striatum, le cortex préfrontal joue également un rôle d'intégration des informations pour la décision (DAW et al. 2011 ; GLÄSCHER et al. 2010 ; WUNDERLICH et al. 2012). Nous renvoyons le lecteur vers BALLEINE et O'DOHERTY (2010) pour une revue des différentes hypothèses à ce sujet.

4.1.4 Liens entre apprentissage par renforcement et comportement instrumental.

Afin de comprendre ces comportements et leurs interactions, des modèles computationnels ont été proposés pour tenter d'expliquer et de reproduire les observations faites *in vivo*. Ces modèles s'appuient principalement sur le paradigme de l'apprentissage par renforcement. On note effectivement le parallèle entre la formalisation présentée dans le chapitre 3 et l'idée d'une évolution du comportement motivée par l'obtention d'une récompense.

En pratique, le développement de l'AR a également été fortement influencé par l'étude

du comportement animal. Il s'inspire directement de cette idée d'apprentissage par essai-erreur évoquée au début de ce chapitre, et de la notion de renforcement présente en psychologie (SUTTON et BARTO 1998, chap1). L'apprentissage TD (voir chap. 3) est initialement proposé, en conditionnement classique, comme une extension du modèle de RESCORLA et WAGNER (1972) par SUTTON et BARTO (1981) (notamment pour prendre en compte l'apprentissage au cours d'un essai, alors que RESCORLA et WAGNER proposent un modèle de l'apprentissage d'un essai à l'autre).

En neurophysiologie, il a été montré que la dopamine a un rôle important dans l'évaluation de la récompense dans les tâches de conditionnement (LJUNGBERG et al. 1992). Or, ce neuromodulateur, sécrété par les neurones dopaminergiques, est essentiel dans le processus d'apprentissage (MONTAGUE et al. 2004 ; WISE 2004). Il est également rapproché du signal d'erreur en apprentissage par renforcement : SCHULTZ et al. montre une corrélation forte entre l'activité phasique des neurones dopaminergiques et l'erreur de prédiction calculée dans les algorithmes d'AR direct (SCHULTZ et al. (1997) ; HOLLERMAN et SCHULTZ (1998) ; SCHULTZ (1998) et voir figure 4.2). Un pic apparaît dans l'activité des neurones dopaminergiques d'un animal qui reçoit une récompense non prédite (ce qui correspond algorithmiquement à un retour reçu supérieur à la valeur prédite, voir chap. 3, eq. 3.9). Lorsqu'il a appris le lien entre un stimulus particulier et cette récompense, le pic est présent lors de la présentation du stimulus, mais pas au moment de la récompense (la prédiction est équivalente au retour reçu, l'erreur est quasi-nulle). Enfin, lorsque la prédiction est fautive, l'activité des neurones est inhibée au moment où la récompense aurait dû être délivrée (soit un retour reçu inférieur à la valeur prédite). Si cette hypothèse est discutée par des travaux plus récents (BERRIDGE 2006 ; SCHULTZ 2013 ; BELLOT et al. 2012 ; KISHIDA et al. 2016), l'AR reste très présent dans les modèles proposés.

Enfin, DOYA (1999) associe les différents types d'apprentissage à une zone du cerveau à partir des informations anatomiques et physiologiques connues. L'apprentissage non-supervisé est associé au cortex cérébral, tandis que le cervelet supporterait un apprentissage supervisé (bien que cette hypothèse soit remise en cause au profit de l'apprentissage par renforcement (OHMAE et MEDINA 2015)). L'apprentissage par renforcement est associé aux ganglions de la base (dont le principal noyau d'entrée est le striatum mentionné précédemment), ces derniers étant impliqués dans l'apprentissage de comportements séquentiels (GRAYBIEL 1995). L'activité des neurones dopaminergiques présentée précédemment ainsi que la capacité des modèles d'AR à reproduire les comportements observés (DOMINEY et al. 1995 ; BERNS et SEJNOWSKI 1998) viennent appuyer cette hypothèse.

4.2 Modèles du comportement instrumental

Ces rapprochements entre apprentissage par renforcement et études du vivant conduisent à un rapprochement entre la dichotomie observée entre les comportement dirigé vers un but et comportement habituel de celle qui existe entre AR indirect et direct. Une grande majorité des modèles de l'apprentissage du comportement instrumental dans le vivant s'appuient donc sur ce paradigme, et modélisent dans leurs études le comportement global d'un individu ou d'un animal comme résultant d'un partage du contrôle entre un

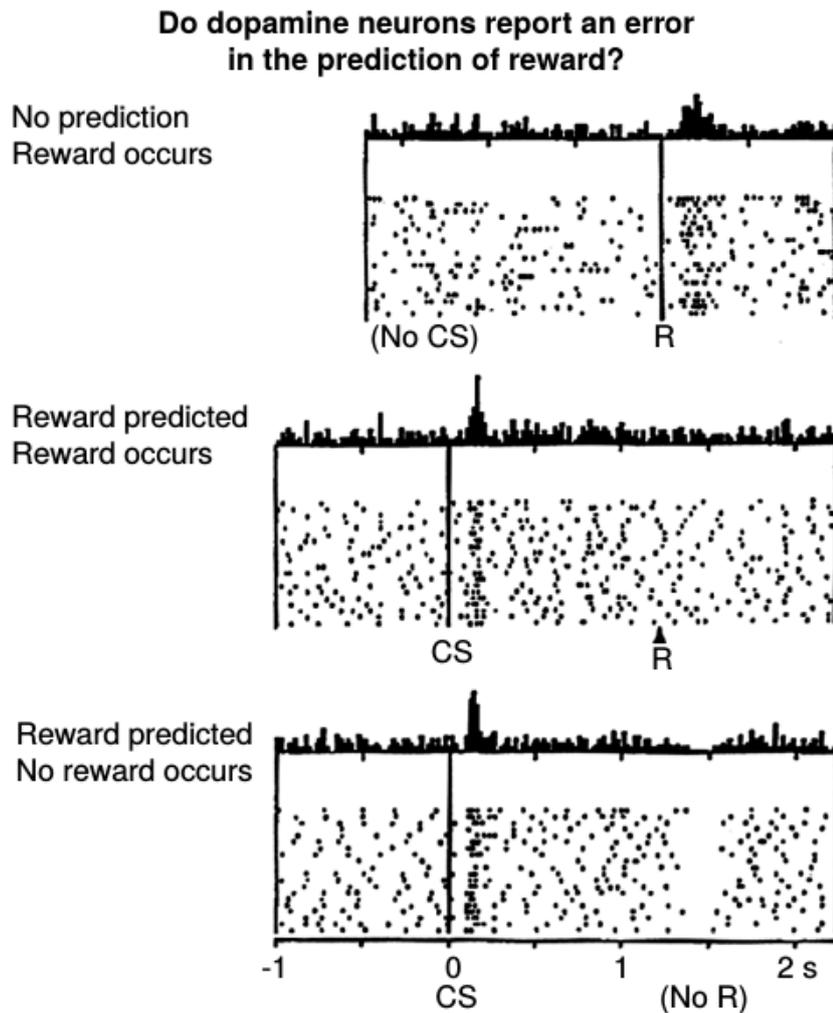


Figure 4.2 – Lien entre dopamine et erreur de prédiction de la récompense. *Haut* : la délivrance d'une récompense inattendue provoque une activité des neurones dopaminergiques. *Milieu* : le stimulus conditionnel a été associé à la récompense ; l'activité des neurones dopaminergiques se produit après perception du stimulus, en anticipation de la récompense, mais n'est pas présente lorsqu'il est délivré. *Bas* : lorsque la récompense n'est pas délivrée comme prévue, l'activité des neurones dopaminergiques subit une inhibition au moment où la récompense aurait dû l'être. Image extraite de SCHULTZ et al. (1997)

algorithme indirect (modèle du comportement dirigé vers un but) et un modèle direct (modèle du comportement habituel). L'hypothèse consiste à dire que l'AR indirect (ou *model-based RL*) est un bon mécanisme pour décrire les comportements dirigés vers un but : une fois le modèle acquis, prendre une décision par planification prend du temps (d'où un temps de réaction plus long), mais permet de s'adapter en quelques essais et erreurs à tout changement de la fonction de récompense. À l'inverse, l'AR direct (ou *model-free RL*) est un bon mécanisme pour décrire les comportements habituels : ils sont lents à acquérir, lents à s'adapter à des changements de la fonction de récompense, mais ils permettent de décider sans planifier (donc d'avoir des temps de réaction très courts). Cette hypothèse a été proposée notamment par DAW et al. (2005). Le travail de DAW et al. (2005) étant à la base de nombreux travaux suivants, nous détaillons un peu plus loin leur modèle. Nous verrons également que le même formalisme, et en particulier la même dichotomie entre AR indirect et direct, permet à la fois d'expliquer la variété des comportements dans les tâches de conditionnement instrumental mais également dans les tâches de navigation (KHAMASSI et HUMPHRIES 2012). Ce formalisme semble donc particulièrement pertinent pour notre objectif de coordonner différents apprentissages dans une architecture robotique qui permette à un robot de s'adapter dans une variété de tâches (notamment une tâche de navigation). L'objectif de cette partie va donc être d'analyser les critères et principes proposés en neurosciences computationnelles pour coordonner ces différents mécanismes (ou algorithmes) d'apprentissage afin de s'en inspirer pour leur coordination au sein de notre architecture robotique.

Nous désignerons dans la suite sous le nom d'*expert* un algorithme qui modélise un de ces comportements, la plupart des modèles présentés combinant un expert faisant de l'AR indirect et un expert faisant de l'AR direct.

4.2.1 Axes de catégorisation et domaines considérés.

À partir du rapprochement entre comportements et algorithmes, on trouve différentes manières de réaliser la combinaison. Nous les examinerons d'une part sous l'angle de leur organisation, d'autre part sous celui de leur méthode d'arbitrage. Le premier point correspond à la distinction entre modèles horizontaux (ou « plats ») et hiérarchiques (principalement évoquée par DEZFOULI et BALLEINE (2012)). Dans les premiers, les experts sont mis au même niveau : les deux évaluent en parallèle les actions disponibles et un troisième système est chargé de l'arbitrage. Dans les seconds, le modèle global s'appuie principalement sur l'un des deux experts qui est capable, en plus de son fonctionnement normal, de faire appel à l'autre lorsque les conditions le requièrent. Le second point s'appuie sur la manière dont la combinaison des informations issues des experts est faite pour décider de l'action finale exécutée par l'agent. Si chaque méthode possède ses spécificités, nous pouvons différencier les catégories suivantes : celles qui *fusionnent* les informations fournies par les experts pour ensuite prendre une décision, et celles qui *sélectionnent* l'un des deux experts pour prendre la décision.

Enfin, nous nous intéresserons à la fois aux modèles du comportement instrumental et aux modèles de navigation animale. On retrouve dans l'étude de la navigation animale et dans les modèles résultants la même idée de type de comportements différents, combinés

selon les contextes (on parle alors de *stratégies de navigation*, O'KEEFE et NADEL (1978) ; RESTLE (1957)). En revanche, les études s'appuient principalement sur une classification selon la nature des informations utilisées (sensorielles, proprioceptives, internes), le point de vue (égocentrique ou allocentrique), le type de mémoire et le temps d'acquisition de chaque stratégie. Il en résulte une catégorisation en stratégies *réactives* (*response strategies*), où le comportement résulte d'associations sensorimotrices, et en stratégies *de lieu* (*place strategies*), qui repose sur des informations de localisation spatiale. La répétition d'une tâche de navigation entraîne un transfert d'une stratégie de lieu à une stratégie réactive (PACKARD 1999), ce qui rappelle le transfert entre comportement dirigé vers un but et habituel. Des rapprochements entre navigation et comportement instrumental ont été faits par plusieurs travaux montrant que la dichotomie entre AR direct et indirect s'applique similairement dans ces deux domaines (KHAMASSI et HUMPHRIES 2012 ; REDISH 1999), mais ces domaines d'étude restent majoritairement séparés. Il nous semble cependant que considérer la navigation comme un type spécialisé de comportement instrumental permet d'adopter une vision plus globale sur le comportement, c'est pourquoi nous ne nous limitons pas à des contributions « purement comportement instrumental ». Enfin, devant la quantité conséquente de modèles de navigation robotique bio-inspirée (FILLIAT et MEYER 2003 ; MEYER et FILLIAT 2003 ; CUPERLIER et al. 2007), nous nous concentrerons sur les propositions qui adressent plus particulièrement la manière de combiner des stratégies.

4.2.2 Modèles horizontaux

Comme évoqué plus haut, le travail de DAW et al. (2005) a été un des premiers à faire le rapprochement entre les types d'AR et les types de comportement instrumental ; en l'occurrence, les experts sont des versions bayésiennes des algorithmes VI (DEARDEN et al. 1999) et Q-learning (DEARDEN et al. 1998). Dans sa version directe, l'agent n'estime plus l'intérêt d'une action comme une valeur unique mais comme une distribution de probabilités de la valeur $Q_{s,a}(q) = P(Q(s, a) = q)$, la variance de la distribution représentant l'incertitude de l'agent sur sa connaissance de $Q(s, a)$. Dans sa version indirecte, cette distribution est maintenue sur les probabilités de transitions et de récompense, pour en déduire les distributions sur les probabilités de valeur $Q_{s,a}$. Ainsi, pour un couple (s, a) donné, si la distribution est proche d'une loi uniforme, l'agent est très incertain de la valeur exacte de l'action ; à l'inverse, si la distribution est piquée sur une valeur particulière, l'agent a une bonne estimation de la valeur de l'action. La valeur de chaque action est ensuite estimée en prenant la valeur moyenne de la distribution de l'expert le plus confiant, donc celui dont la variance est la plus faible (sélection selon le signal d'incertitude au niveau de chaque action).

De nombreux modèles horizontaux proposent de fusionner les valeurs estimées par chaque expert en les sommant : en navigation, c'est l'approche adoptée par GUAZZELLI et al. (1998). Les auteurs combinent deux stratégies de navigation modélisées par AR direct, et exploitant des informations de nature différente : un modèle *Taxon-Affordances*, qui s'appuie sur les stimuli perceptibles (éléments du labyrinthe dans lequel se trouve l'agent : couloir, mur, etc.) et un modèle *World-Graph* qui intègre ces stimuli en un lieu

de l'environnement. Le modèle intégré (*TAM-WG*) somme directement la valeur estimée pour chaque action par chacun des modules avant d'effectuer la sélection. Un principe similaire est utilisé dans HANOUNE (2015) où une stratégie basée sur une carte cognitive et une stratégie basée sur des associations sensorimotrices évaluent chacun les actions disponibles à partir d'une information commune formée par les transitions entre lieux. Ces estimations sont sommées avant compétition entre les valeurs résultantes. Si ces modèles proposent une manière simple de mettre en commun les informations issues de différents experts, ou stratégies, elles donnent la même importance aux propositions de chacun. Une amélioration directe est celle effectuée dans GIRARD et al. (2005) : une stratégie de navigation *taxon* est combinée avec une stratégie de navigation basée sur une carte topologique de l'environnement dans un robot confronté à une tâche de survie. La stratégie basée sur une carte, en apprenant la position de ressources de valeur pour l'agent, est capable de calculer un chemin vers la ressource désirée. La stratégie *taxon* correspond à une stratégie d'approche visuelle. Les *saliences* estimées par chaque stratégie sont fusionnées pour chaque action en faisant une somme pondérée, la pondération étant un paramètre fixé du modèle. Cette pondération permet de favoriser l'une ou l'autre des stratégies lorsqu'elles sont en désaccord sur l'action à suivre. Les auteurs montrent que la performance des systèmes combinés est meilleure que sans la stratégie basée carte. Dans LESAINTE et al. (2014), les différentes pondérations (fixées) attribuées à l'estimation des valeurs d'action par les algorithmes d'AR direct et indirect du modèle permet de reproduire différents types de comportements instrumentaux chez le rat, tandis que dans le modèle de COLLINS et FRANK (2012), qui combine modèle de la mémoire de travail et algorithme d'AR direct, cette pondération est fonction du remplissage de la mémoire de travail. Plus la mémoire de travail est pleine, moins elle est capable de prédire correctement la bonne action, sa pondération diminue donc.

VIEJO et al. (2015) fusionne les valeurs d'action par une simple somme. Cependant, si les valeurs de l'expert habituel sont classiquement apprises par un Q-learning, les valeurs de l'expert dirigé vers un but sont données par un modèle de mémoire de travail qui, entre chaque étape de décision, choisit entre affiner ses estimations ou lancer le processus de sélection de l'action (en sommant les estimations courantes). Tant que les entropies des distributions de probabilités des experts sont hautes, la probabilité de décider est faible, et elle augmente au fur et à mesure que ces entropies diminuent ainsi que du nombre de fois où on a choisi de raffiner les valeurs.

Enfin, pour la coordination de stratégies de navigation en robotique, HASSON et al. (2012) introduit la notion de frustration : cette dernière correspond à une mesure de l'évolution de la distance au but, et augmente si la distance augmente ou stagne. Le dépassement d'un seuil par le niveau de frustration entraîne un changement de stratégie. Le mécanisme se veut générique, et applicable de la même manière aux buts suivis par le robot ou à ses motivations. Il nécessite cependant que le but soit connu ou perceptible, ce qui n'est pas directement applicable à d'autres contextes que la navigation. JAUFFRET et al. (2013) étend ce modèle en ne basant pas la notion de frustration sur l'évolution de la distance au but mais sur celle de l'erreur de prédiction sensorimotrice. L'intérêt de ces approches est illustré dans des expériences robotiques simulées et réelles où sont coordonnées une stratégie de navigation basée sur une carte et une stratégie de suivi de

route. Ces expériences montrent que la coordination de stratégies permet une meilleure performance que la seule navigation basée carte.

En dehors des travaux de DAW et al. (2005) évoqués au début de cette section, les méthodes d'arbitrage présentées jusqu'ici sont des méthodes de fusion. Pour les méthodes de sélection dans les modèles horizontaux, une autre approche consiste à apprendre à sélectionner à chaque étape de décision l'un des experts. On retrouve cette approche en navigation dans CHAVARRIAGA et al. (2005), qui propose un modèle combinant deux stratégies de navigation (*locale*, basée sur la position de l'agent dans l'espace et *taxon*, basée sur des indices perceptifs), modélisées toutes deux par des algorithmes d'AR direct. L'association entre état et stratégie est apprise par un troisième algorithme d'AR direct (*gating network*). La sélection d'une stratégie s'effectue en fonction d'une part de la valeur de l'action proposée par la stratégie, et d'autre part de l'association apprise entre l'état courant et la stratégie. Ainsi, une stratégie qui aurait été peu associée à l'état mais qui prédirait une forte récompense a une chance d'être sélectionnée. Chaque stratégie apprend en fonction de son erreur de prédiction, mais également de sa probabilité d'avoir été sélectionnée, les deux étant agrégés en un facteur h . Ainsi, une stratégie qui prédit bien la récompense et qui avait une forte probabilité d'être sélectionnée dans un état apprend plus vite. Le *gating network* renforce la sélection d'une stratégie dans un état pour faire tendre son poids vers la valeur h . Cette valeur h dépendant de l'erreur de prédiction, ce type de sélection impose des stratégies qui puissent fournir cette information, donc à base d'AR direct.

DOLLÉ et al. (2010) propose un modèle inspiré de celui de CHAVARRIAGA et al. (2005) : la combinaison entre stratégies de navigation est apprise par renforcement par un mécanisme de sélection de stratégies. Cependant, trois stratégies de navigation de nature différente sont utilisées : exploration, guidage visuel et planification de chemin dans une carte, modélisées respectivement par un choix purement aléatoire, un algorithme d'AR direct (Q-learning) et par un algorithme de recherche de chemin dans un graphe (gradient de valeur de maximum dans l'état but). La sélection de stratégies est faite par un *gating network* sous forme de Q-learning glouton, qui apprend à associer un état (composée de l'activité des entrées perceptives et de l'estimation de position de l'agent) à la meilleure stratégie à adopter pour atteindre le but. L'intérêt de cette méthode de sélection réside dans le fait qu'elle ne s'appuie que sur la capacité des experts à faire obtenir de la récompense à l'agent. CALUWAERTS et al. (2012) étend ce modèle à la robotique en l'implémentant dans un animat. Leur *gating network* manipule des états qui ne sont composés que de l'estimation de position, donc la même information que la stratégie basée carte. Ils montrent que leur architecture parvient à apprendre à utiliser la meilleure stratégie dans les différents points de l'espace, en fonction de l'information perceptive disponible, ainsi qu'à ignorer la stratégie d'exploration dans les cas où elle est couplée avec une stratégie plus évoluée qui a appris suffisamment.

4.2.3 Modèles hiérarchiques

Les modèles hiérarchiques, comme nous l'avons dit plus haut, donnent un rôle prépondérant à l'un des deux experts. C'est une vision formalisée par DEZFOULI (2015), mais

dont on trouve les prémisses dans des modèles antérieurs.

Ainsi, KERAMATI et al. (2011) étend le modèle proposé par DAW et al. (2005) sous une telle forme. Si ces derniers ne se préoccupaient que de l'incertitude de chaque agent, KERAMATI et al. (2011) propose un arbitrage par sélection qui optimise le compromis entre précision de l'information et rapidité de décision. Le modèle s'appuie sur l'hypothèse que l'expert dirigé vers un but peut obtenir une *information parfaite* sur l'intérêt de chaque action en planifiant dans ses modèles internes. Leur expert habituel maintient un suivi de son incertitude sur les valeurs Q à l'aide d'un filtre de Kalman (méthode proposée par GEIST et al. (2009)). Ce suivi permet de calculer la *Valeur de Parfaite Information* (VPI) qui représente l'intérêt de l'agent à raffiner son estimation de la valeur d'une action. Cette valeur est comparée au coût estimée à raffiner cette estimation, calculée en multipliant le temps passé à délibérer par la récompense moyenne reçue jusqu'ici. Il s'agit du manque à gagner en terme de récompense à dédier ce temps à raffiner les estimations de valeur d'action. Cette comparaison est faite pour chaque action, produisant une sélection de la valeur maintenue par l'expert habituel si $VPI < \text{coût}$ et sélection de la valeur calculée par l'expert dirigé vers un but sinon. Cette approche offre l'avantage de limiter l'utilisation du processus de planification aux cas où il est réellement nécessaire, mais l'hypothèse d'information parfaite de ce processus est reconnue par les auteurs comme très forte. Ce principe est également utilisé dans le modèle de PEZZULO et al. (2013). Pour chaque action, ils calculent une *Value of Information* (VoI) simplement estimée par le ratio de l'incertitude de l'action et de la différence entre la valeur de l'action considérée et celle de son alternative, qui sert à déclencher le processus de planification. Aucune hypothèse n'étant faite sur la perfection de l'expert dirigé vers un but, l'estimation de valeur d'action de l'expert habituel n'est pas remplacée, mais mise à jour à l'aide de l'estimation de l'expert dirigé vers un but. Le modèle est testé en simulation dans une tâche à choix binaires, et les auteurs vérifient bien que cette méthode leur permet d'exploiter la capacité à simuler les conséquences des actions de l'expert dirigé vers un but en début d'expérience, lorsque l'incertitude sur les valeurs d'action est élevée, ou après un changement de condition dans l'environnement.

Enfin, ces modèles hiérarchiques correspondent à une autre vision que celle des habitudes comme une algorithme d'apprentissage direct, qui associe état et meilleure action en terme de récompense. DEZFOULI et BALLEINE (2012) conservent un algorithme d'AR indirect comme modèle du comportement dirigé vers un but mais pointent que les algorithmes d'AR direct reproduisent imparfaitement les résultats sur les animaux testés dans des situations de dégradation de contingence. Expérimentalement, les animaux qui ont été entraînés suffisamment pour former des comportements habituels sont insensibles à ces situations où le taux de récompense associé à une action varie brutalement au cours de l'expérience et n'adaptent pas leur comportement. Or, les algorithmes d'AR direct classiques présentent une erreur de prédiction importante dans ce cas, et tendent à modifier la valeur, puis le comportement de l'agent au fur et à mesure que les différents états sont expérimentés. Plutôt que de proposer une variante de cette algorithme pour s'adapter à ce défaut de modélisation, ils proposent de modéliser les habitudes comme des séquences d'action. Ils considèrent que le processus de décision est initialement *dirigé vers un but* et s'appuie sur l'algorithme d'AR indirect, mais que certaines contraintes de l'environnement

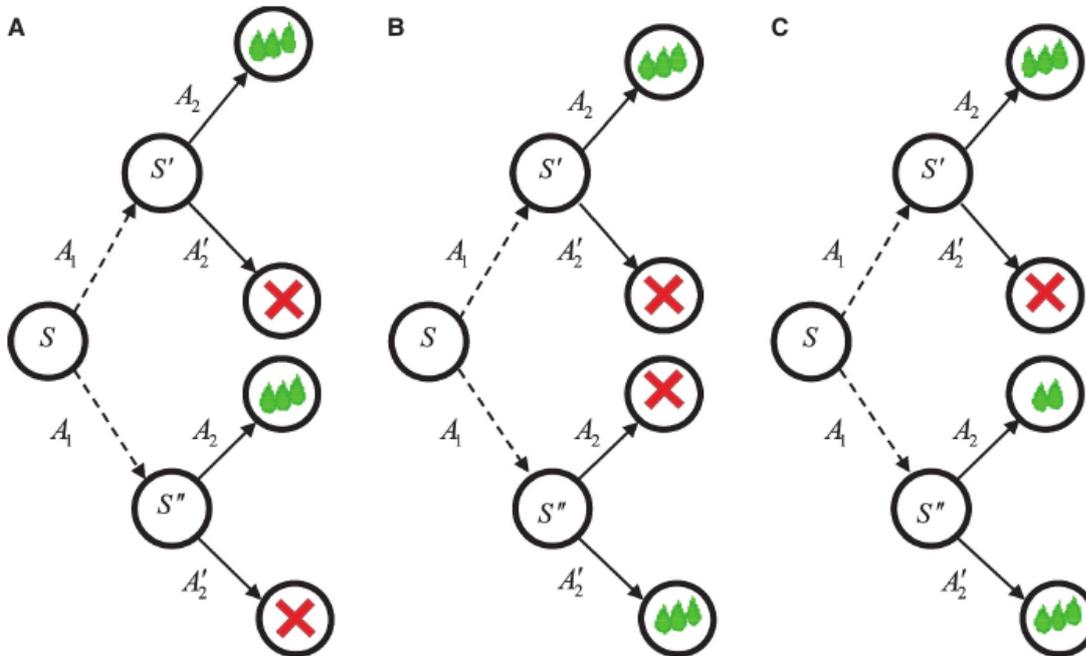


Figure 4.3 – Différentes tâches permettant ou non l'utilisation de séquences d'action. (A) L'action A_1 dans l'état S mène stochastiquement l'agent aux états S' et S'' , dans lesquelles l'action la plus intéressante est A_2 . Quelque soit l'état conséquence de A_1 , l'action A_2 permet la délivrance de récompense, la séquence d'actions $\{A_1, A_2\}$ est donc intéressante à former. (B) Dans ce cas, la récompense amenée par A_2 et A_2' dépend de l'état dans lequel l'agent se trouve. Il n'est pas intéressant de former une séquence d'action, puisque A_1 amène stochastiquement dans S' ou S'' . (C) Cas non-trivial : former la séquence $\{A_1, A_2\}$ produit un comportement sous-optimal quand l'agent est en S'' mais est la meilleure option en S' . La pertinence d'associer les deux actions doit être évaluée selon un critère. Figure extraite de DEZFOULI et BALLEINE (2012)

(et la structure résultante du MDP) ne nécessitent pas de vérifier explicitement la conséquence de l'action et la prise d'une décision : la fin d'une action suffit à entraîner le début de la suivante (voir figure 4.3).

Ainsi, ils définissent une mesure de l'avantage de choisir une action a plutôt que l'action optimale a^* dans l'état s :

$$A(s, a) = Q(s, a) - V(s) \quad (4.1)$$

Et le coût d'exécuter successivement a et a' à partir de l'état s :

$$C(s, a, a') = \sum_{s'} P_{sa}(s') \cdot A(s', a') \quad (4.2)$$

Ce coût est évalué à l'aide du modèle de transitions connu de l'agent au prix du coût computationnel d'évaluer les valeurs d'action dans tout le modèle, mais peut également

l'être incrémentalement en assimilant l'erreur TD à une évaluation de l'avantage.

Ici, le coût est comparé au bénéfice d'utiliser la séquence et on forme l'habitude (sélectionner a dans s entraînera la sélection de a' lorsque a finit) s'il est inférieur au bénéfice, sinon, on sépare en ses actions élémentaires la séquence d'actions existante (voir algorithme 4.1).

Algorithme 4.1 : Critère de formation d'habitudes d'actions

```
si  $-C(s, a, a') < \hat{R}\tau$  alors
  | Formation d'une habitude
  |  $(s, a) \leftarrow (s, \{a, a'\})$ 
sinon
  | Séparation d'une habitude en ses constituants
  |  $(s, \{a, a'\}) \leftarrow (s, a)$ 
fin
```

Enfin, une séquence formée est vue par le processus de décision de la même manière qu'une action élémentaire : sa valeur est estimée et elle peut être sélectionnée. L'état d'arrivée n'est évalué qu'après exécution de toutes les actions de la séquence, et le coût de la séquence n'est pas réévalué. Seule une diminution du bénéfice peut conduire à défaire une habitude établie. Cette approche est par ailleurs appuyée par des données expérimentales chez l'humain (DEZFOULI et BALLEINE 2013).

Une autre hypothèse est proposée dans TOPALIDOU et al. (2015) pour expliquer la formation des habitudes. Selon la proposition que les ganglions de la base guident initialement l'apprentissage du cortex, le comportement habituel n'est plus modélisé comme un algorithme d'apprentissage habituel, qui apprend en fonction de la récompense, mais comme des associations stimulus-action résultant d'un apprentissage Hebbien. Le comportement étant initialement guidé par l'obtention de la récompense, les associations stimulus-action qui y mènent sont visitées plus souvent et les associations ainsi renforcées deviennent des habitudes, formées dans le cortex. D'après les auteurs, ce modèle a permis de reproduire des données (non-publiées à ce jour) de comportement chez le singe.

4.3 Conclusion

Dans ce chapitre, nous avons présenté d'une part un bref historique et les résultats principaux sur l'étude du comportement instrumental chez les mammifères, d'autre part, les modèles qui, soit cherchent à expliquer ces résultats, soit s'en inspirent pour contrôler des robots. Les modèles présentés adoptent une approche qui consiste à doter l'agent de plusieurs types de comportements (ou stratégies quand il s'agit de navigation) et à les combiner selon une méthode qui permette la meilleure performance (qu'il s'agisse d'une performance à accomplir une tâche ou d'une performance à modéliser des données biologiques). Dans ces modèles, on retrouve une vision classique, qui fait un parallèle entre types de comportements instrumentaux (dirigé vers un but, habituel) et types d'algorithmes

d'AR (indirect, direct), la modélisation des premiers par les seconds venant naturellement. Pour les stratégies de navigation, on retrouve également une modélisation à l'aide de l'AR, mais où la différence entre stratégies est plutôt faite selon le type d'information utilisé (bien que plusieurs tentatives d'unification entre navigation et comportement instrumental aient été faites (KHAMASSI et HUMPHRIES 2012)). La majorité de ces modèles organisent les comportements de manière horizontale : aucun n'est actif par défaut, c'est la méthode d'arbitrage qui en décide. Il existe également un ensemble de modèles hiérarchiques, où l'un des comportements est prépondérant. Ces organisations expriment un biais dans l'arbitrage entre comportements, mais également une vision différente de la modélisation des habitudes.

Il ne semble pas y avoir de consensus définitif sur ces questions : les connaissances sur la biologie évoluent régulièrement, amenant à réviser les hypothèses. La contribution des roboticiens à ce débat apporte un point de vue plus fonctionnel sur ces mécanismes, mais avec sa propre motivation d'améliorer les capacités des robots, créatures relativement différentes des humains.

Chapitre 5

Résumé de l'état de l'art

Au cours de l'introduction et des deux chapitres précédents, nous avons exploré plusieurs domaines différents qui cependant s'intéressent à des problématiques similaires : comprendre et créer les mécanismes de décision d'agents autonomes. Nous avons d'abord vu comment la robotique étend les travaux de planification du raisonnement automatisé pour permettre aux robots de calculer leur comportement. Un pur raisonnement basé sur la logique apparaît le problème de raisonner dans le monde réel et ses contraintes. La solution proposée est d'organiser les capacités du robot, en architectures de contrôle robotiques. L'organisation en trois couches, *décisionnelle*, *exécutive* et *fonctionnelle* s'est imposé dans une majorité de modèles, bien que tous les travaux possèdent leur particularité. Ces architectures se sont peu concentrées sur l'apprentissage des capacités, utilisant principalement la logique du premier ordre des planificateurs. Leur évolution en architectures robotiques cognitives est récente mais s'oriente dans la direction de ces questions (LANGLEY et al. 2008 ; KHAMASSI et al. 2016)

Nous avons ensuite étudié les méthodes discrètes d'apprentissage par renforcement, comment elles modélisent un problème de décision séquentiel, analogue à ceux auxquels est confrontée la robotique, et permettent l'adaptation du comportement de l'agent. Nous avons présenté les deux catégories principales des algorithmes de résolution de ces problèmes, par apprentissage direct du comportement ou par apprentissage d'un modèle du problème et adaptation du comportement à partir de ce dernier. Nous avons également évoqué les approches d'ensemble pour l'apprentissage par renforcement, qui consistent à décider du comportement de l'agent par synthèse des avis de plusieurs méthodes de décision sur le problème.

Nous avons enfin présenté les résultats de l'étude du comportement des mammifères en psychologie et en neurosciences qui sert d'inspiration à nos travaux. Ces résultats tendent à renforcer l'hypothèse de l'existence de deux catégories de comportements chez les mammifères. Selon le degré d'entraînement d'un animal ou d'un humain sur une tâche séquentielle, son comportement exhibe des propriétés différentes. L'animal est supposé passer d'un comportement intentionnel, déterminé en anticipant les événements futurs, qualifié de « dirigé vers un but », à un comportement automatique, réactif, qualifié « d'habituel ». Nous avons présenté les travaux majeurs qui proposent, pour le comportement instru-

mental et la navigation animale, des modèles de coordination, basés principalement sur l'apprentissage par renforcement.

Dans les chapitres suivants, nous présentons notre contribution proposant une architecture robotique qui vise à intégrer ces notions de comportements dirigé vers un but et habituel dans le processus de délibération du robot. Nous nous posons d'une part la question de l'intérêt d'un tel principe pour la robotique, d'autre part de la manière dont de tels comportements pourraient être mis en oeuvre et coordonnées dans des tâches plus réalistes que celles des modèles du vivant.

Chapitre 6

Proposition d'une architecture robotique combinant comportement habituel et dirigé vers un but

6.1	Architecture proposée	62
6.1.1	Vue d'ensemble	62
6.1.2	Expert habituel	62
6.1.3	Expert dirigé vers un but	65
6.1.4	Meta-Contrôleur	68
6.1.5	Module de perception	68
6.1.6	Module d'action	69
6.2	Illustration expérimentale	70
6.2.1	Expérience	70
6.2.2	Résultats	74
6.3	Conclusion	77

Dans ce chapitre, nous proposons une architecture de contrôle robotique simplifiée, dans laquelle la couche délibérative présente à la fois des capacités de planification, d'apprentissage d'une habitude et du modèle de l'environnement, inspirée des modèles présentés dans le chapitre précédent. Cette couche interagit avec les couches inférieures qui abstraient les perceptions en états et interprètent les actions en ordres moteurs.

Comme nous l'avons vu, des modèles différents de la notion de comportement habituel et de son interaction avec le comportement dirigé vers un but existent dans la littérature, et il n'y a pas de consensus définitif. Comme nous étudions principalement l'intérêt de ce type de modèle (et de la notion d'habitude) pour la robotique, nous nous inspirerons principalement dans cette thèse des modèles *horizontaux*, où le comportement habituel est modélisé comme un algorithme d'AR direct et le comportement dirigé vers un but comme un algorithme d'AR indirect, pour nous concentrer sur les méthodes d'arbitrage et l'organisation de la couche décisionnelle. Nous examinerons plus en détail le modèle du comportement dirigé vers un but dans le chapitre 7 et en discussion (chap 9).

Le travail présenté dans ce chapitre correspond à la publication suivante :

Erwan RENAUDO, Benoît GIRARD, Raja CHATILA et Mehdi KHAMASSI (2014).
« Design of a Control Architecture for Habit Learning in Robots ». Dans :
Biomimetic and Biohybrid Systems, LNAI Proceedings, p. 249–260

6.1 Architecture proposée

6.1.1 Vue d'ensemble

L'architecture que nous proposons sera construite sur la base d'une architecture de contrôle robotique à trois couches telle que celles présentées au chapitre 2. Nous nous concentrons sur les capacités de décision de la couche délibérative, les couches inférieures étant simplifiées. Pour cette couche de décision, nous nous appuyons sur une organisation *horizontale* et un meta-contrôleur qui a le rôle d'arbitre entre les propositions faites par les experts. Nous souhaitons étendre les travaux initiés par CALUWAERTS et al. (2012) en dehors du contexte de la navigation, plus généralement pour la prise de décision en robotique. L'architecture complète du robot est présentée figure 6.1.

Les experts sont des agents autonomes : à partir d'un état donné, ils apprennent à leur manière en fonction de la récompense reçue, et décident d'une action que nous appellerons *proposition*. Selon que le meta-contrôleur a donné le contrôle à l'un ou l'autre des experts, l'une des actions est exécutée par l'agent et l'autre oubliée. L'action exécutée est également renvoyée vers chaque expert pour qu'il puisse apprendre correctement sur la base des conséquences de l'action effectuée par le robot. Cette action génère potentiellement des changements dans l'environnement, qui sont perçus et mettent à jour l'état. Dans la suite, nous utiliserons la taxonomie suivante :

- *L'apprentissage* désigne pour l'expert habituel la mise à jour des associations entre état et action, et pour l'expert dirigé vers un but la mise à jour de ses modèles internes.
- *La planification* désigne pour les deux experts l'évaluation des valeurs d'action $Q(s, a)$ selon leur propre algorithme.
- *La décision* désigne le tirage d'une action dans la distribution des probabilités d'action P . Sauf mention contraire, tous nos experts déduisent leur distribution de probabilités P comme la distribution Q à laquelle est appliquée une fonction *softmax*.

Dans ce type d'architecture, les experts sont en compétition selon les règles définies par la méthode d'arbitrage du meta-contrôleur : un seul verra sa proposition exécutée. Ils sont également en coopération dans l'apprentissage : le nouvel état et l'éventuelle récompense reçue permettent aux deux experts de mettre à jour leurs connaissances, même si leur proposition n'a pas été retenue.

6.1.2 Expert habituel

L'expert habituel est implémenté sous la forme d'un réseau de neurones sans couche cachée (figure 6.2). Les neurones d'entrée encodent l'état envoyé par le module de percep-

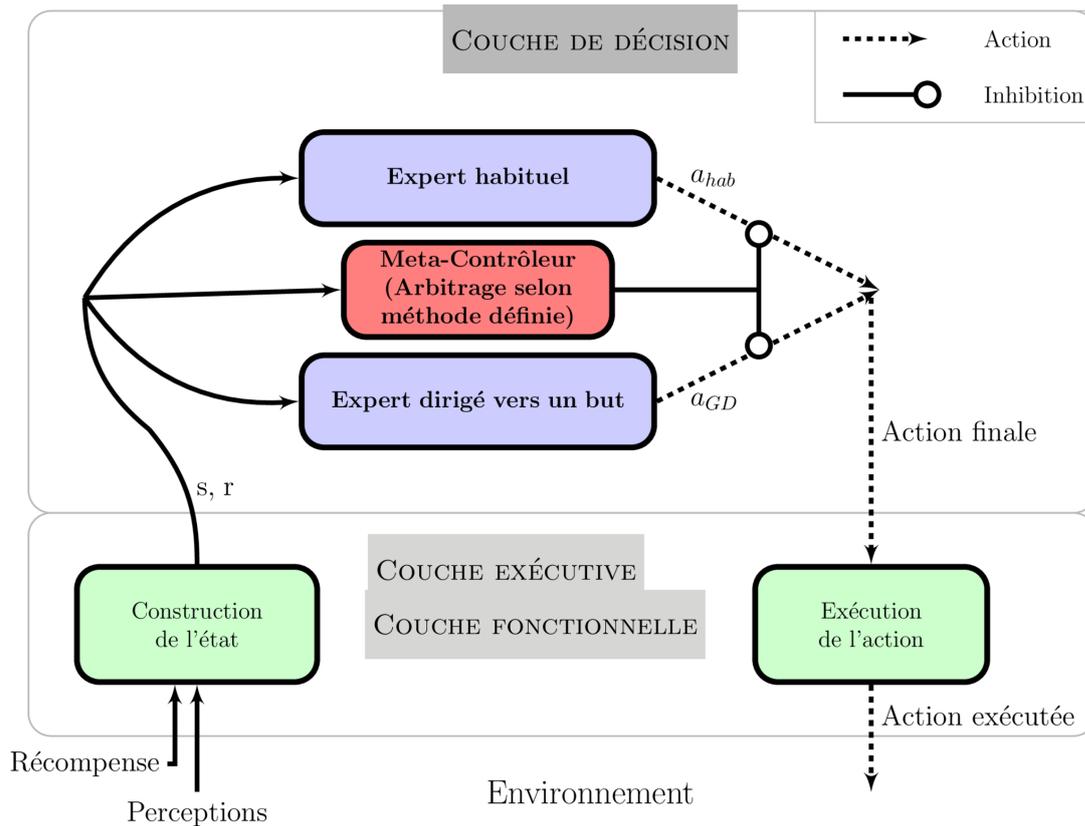


Figure 6.1 – L'architecture de contrôle du robot : l'environnement renvoie des perceptions qui sont transformées en état s . La récompense reçue r est associée à cet état et le tout est envoyé vers la couche de décision. Les deux experts utilisent ces informations pour apprendre conformément à leur nature, le meta-contrôleur utilise ou non ces informations en fonction de la méthode d'arbitrage choisie. L'action finale, déterminée à partir des propositions a_{GD} et a_{hab} parmi les actions connues de l'agent, est exécutée par le module correspondant, qui contient les différentes compétences du robot.

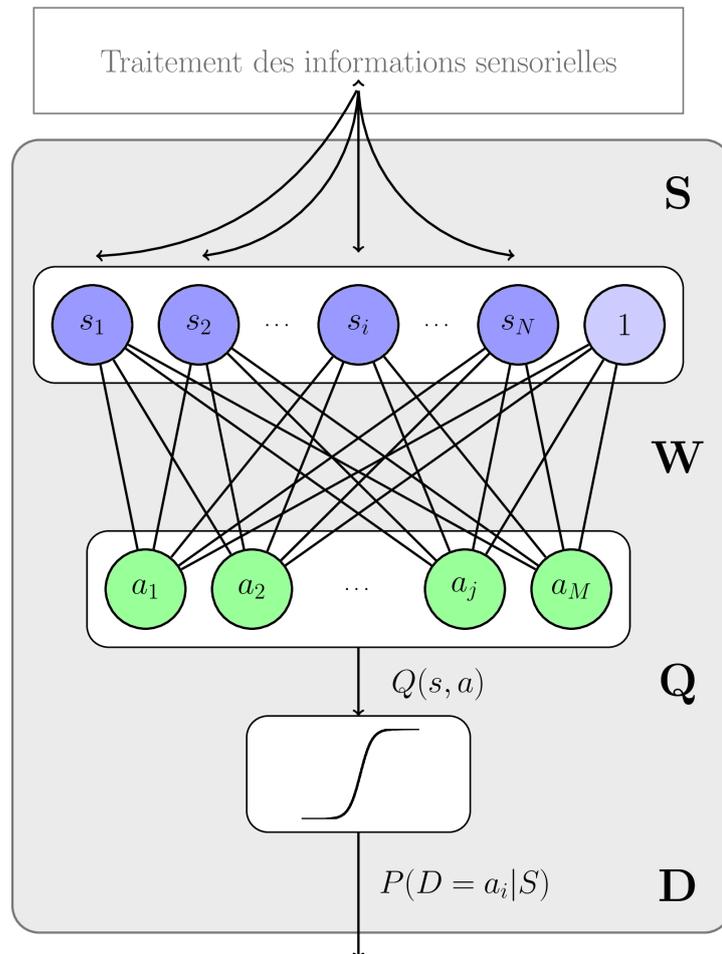


Figure 6.2 – *Expert habituel, modélisé par l’algorithme d’AR direct Q-learning, sous forme de réseau de neurones.*

L’expert reçoit un état S qui est projeté sur les neurones d’entrée s_i , définissant ainsi une activité d’entrée. Cette activité est propagée dans les poids de connexion W pour générer l’activité de la couche d’action. L’activité de cette couche donne l’ensemble des valeurs $Q(S, a_j)$, chaque neurone codant pour une action distincte.

Cette distribution de valeurs est convertie en distribution de probabilités à l’aide d’une fonction softmax, ce qui permet à l’expert de prendre une décision D sur la prochaine action à effectuer.

tion. Chaque neurone de sortie code pour une action, l'activité du neurone représentant la valeur $Q(s, a)$ associée dans l'état courant s . Ces valeurs sont converties en probabilités par une fonction *softmax*, dont la température τ permet de régler le degré d'exploration. La politique est donc apprise directement, sans utilisation d'un modèle décrivant l'effet des actions.

Estimer la valeur d'une action a_j dans l'état S_t se fait au coût d'un produit scalaire entre le vecteur d'entrée augmenté d'un biais et les poids $W_j = (w_{0j}, \dots, w_{Nj})$ qui le relie à l'action :

$$Q_t(S_t, a_j) = a_j^t = W_j^t \cdot (S_t, 1) \quad (6.1)$$

Les poids de connexion W^t sont mis à jour selon l'algorithme Q-learning (WATKINS 1989) : l'erreur de prédiction est ainsi répartie équitablement entre les poids de connexion entre entrées ayant une activité non-nulle et action a effectuée :

$$\delta = r_t + \gamma_{Hab} \cdot \max_b (W_b^{t-1} \cdot S_t) - (W_a^{t-1} \cdot S_{t-1}) \quad (6.2)$$

$$W_a^t = W_a^{t-1} + \alpha_{Hab} \cdot \delta / \sum_n s_n \quad (6.3)$$

avec r_t la récompense instantanée reçue en faisant a dans S_{t-1} α_{Hab} le taux d'apprentissage des poids, γ_{Hab} le facteur d'atténuation de la récompense distance obtainable. Cette mise à jour est effectuée après chaque expérience d'un état, ce qui rend la modification des poids lente.

6.1.3 Expert dirigé vers un but

L'expert dirigé vers un but modélise le comportement éponyme par un algorithme d'AR indirect. Notre expert apprend les relations (S, a, S') sous la forme d'une *force de transition* T , qui quantifie la relation entre S , a et S' . La probabilité de transition S, a, S' du modèle est déduite en normalisant la force $T(S, a, S')$ par la somme des forces pour tous les états finaux du couple (S, a) . La force T est mise à jour incrémentalement (6.4). L'ensemble de ces relations peut être vu comme un graphe orienté, où les noeuds sont les états connus de l'agent et les arêtes les actions qu'il peut faire, constituant le modèle de la fonction de transition. Il mémorise également la dernière récompense obtenue associée au couple (S, a) , ce qui correspond au modèle de la fonction de récompense (6.5).

$$T_t(S, a, S') = T_{t-1}(S, a, S') + \alpha_{MB} \cdot (1 - T_{t-1}(S, a, S')) \quad (6.4)$$

$$R_t(S, a) = r_t \quad (6.5)$$

Contrairement à la plupart des modèles décrits dans le chapitre 4, l'agent ne connaît pas ces fonctions *a priori* : il les construit au fur et à mesure de ses interactions avec l'environnement.

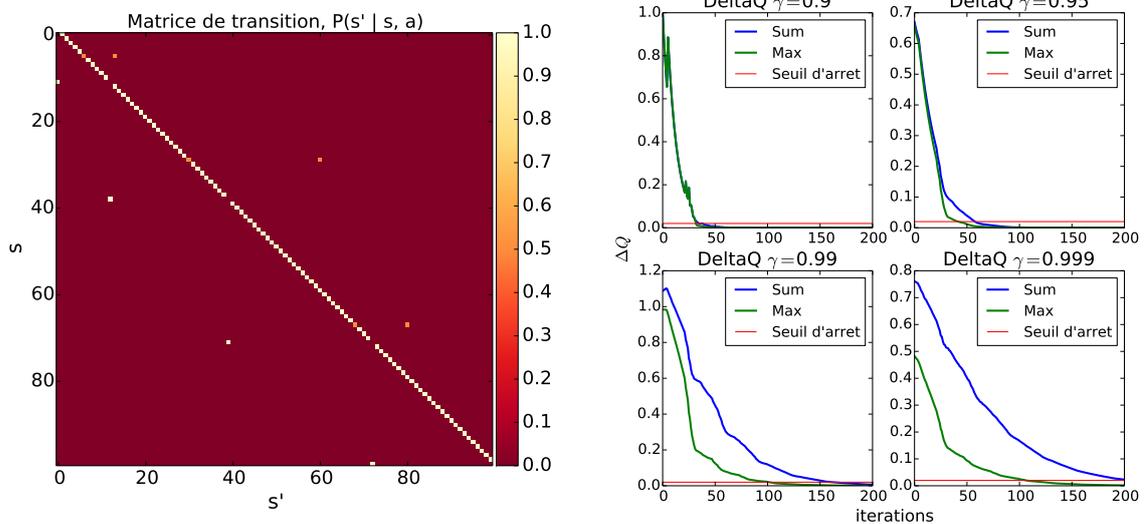


Figure 6.3 – Illustration des différences entre l'équation de mise à jour des valeurs Q classique (sum) et celle utilisée dans notre expert (max). **Gauche** : Matrice de transition correspondant à un MDP cyclique et majoritairement déterministe, à 100 états et 1 action, avec une récompense dans l'état 64. **Droite** : Variation des valeurs Q lors de la planification selon les deux méthodes et pour des valeurs de γ différentes. Le seuil (ligne rouge) correspond à la valeur ϵ choisi « petit », en dessous de laquelle la planification est stoppée. Chaque itération est un parcours de chaque état du problème.

Notre expert planifie dans l'espace d'états : lorsqu'un nouvel état est reçu, les récompenses connues sont propagées dans le graphe de transition pour évaluer la valeur des actions $Q(s, a)$ et prendre une décision, cette décision étant ou non exécutée par l'agent selon le choix du meta-contrôleur. La valeur de l'action est incrémentalement mise à jour par l'algorithme VI selon l'équation (6.6) :

$$Q_t(s, a) = \max \left(r_t(s, a), \gamma_{MB} \cdot \sum_{s'} T_{t-1}(s, a, s') \cdot \max_{a'} Q_t(s', a') \right) \quad (6.6)$$

Nous utilisons cette forme de la mise à jour des valeurs plutôt que celle décrite au chapitre 3 afin de réduire le nombre d'étapes de mise à jour. En effet, en présence de cycles dans la tâche, une fois que la récompense a été propagée une fois dans tout le graphe, la valeur de l'état récompensé augmente pour prendre en compte la nouvelle valeur de l'état suivant, ce qui provoque une nouvelle étape de mise à jour des valeurs (voir figure 6.3). Avec l'équation (6.6), l'algorithme ne prend pas en compte la valeur du prochain état dans l'état récompensé, ce qui rend la convergence des valeurs Q plus rapide (voir figure 6.3 pour une illustration).

Malgré l'utilisation de cette équation de mise à jour des valeurs, au fur et à mesure que l'agent rencontre de nouveaux états, son modèle de transition se complexifie et le temps et le coût de la planification croissent, ce qui rend l'expert dirigé vers un but de moins en

moins réactif. Notre architecture décide de l'action à faire à partir des avis des deux experts dans l'état courant. Dans les modèles du comportement instrumental, cette hypothèse est valide : les tâches sont généralement synchrones au comportement de l'agent, la taille de l'espace d'état reste faible. Mais dans une tâche robotique, nous nous attendons à ce que l'environnement puisse évoluer indépendamment du robot (autres agents, parties dynamiques) ainsi qu'à un grand nombre d'états expérimentés, et ce manque de réactivité peut conduire à effectuer des actions inadaptées, calculées pour un état du monde qui a changé depuis.

Pour compenser ces problèmes et conserver une certaine réactivité de l'expert dirigé vers un but, nous implémentons les propriétés suivantes :

1. Planification budgétée : l'expert dirigé vers un but a un temps limité pour planifier. Nous considérons qu'il vaut mieux que l'expert reste réactif plutôt qu'il ait une estimation exacte des valeurs. Une fois le temps dépassé, la planification est interrompue et l'estimation actuelle des valeurs Q est utilisée pour proposer une action dans l'état courant. Le budget de temps est fixé a priori de sorte à ce que l'expert reste réactif à son environnement.
2. Le mécanisme précédent réduisant le nombre d'états traités lors de la planification, il est nécessaire d'adopter une politique de mise à jour qui puisse se concentrer sur l'estimation des valeurs d'action dans les états que le robot rencontre effectivement. Pour cela, nous faisons l'hypothèse que les états les plus visités par le robot sont ceux dans lesquels il est nécessaire d'avoir une estimation correcte des valeurs d'actions. En effet, au fur et à mesure qu'une politique performante est trouvée, l'agent explore de moins en moins et se concentre sur un sous-ensemble d'états qui l'amènent à la récompense. Il est ainsi plus important qu'il ait des valeurs plus précises dans ces états que dans les états qui auront été rencontrés une ou deux fois et qui seront potentiellement le résultat d'un aliasing perceptuel.

Un nombre réduit d'états à évaluer est sélectionné ; ce nombre est déterminé selon le contraste dans la distribution des visites d'états : si la distribution est proche d'une distribution uniforme, aucune information ne désigne un état à favoriser par rapport à un autre. Le nombre d'états choisis doit être comparable au nombre d'état connus. À l'inverse, si quelques états ont été plus visités que les autres, il est possible de se concentrer sur l'estimation de leurs valeurs.

Soit V_S le nombre de visites de l'état S et V le nombre de visites de l'ensemble des états connus du robot. $P(S)$ est la probabilité de visiter l'état S et se calcule par leur ratio (6.7). L'entropie de la distribution de probabilité (6.8) ainsi générée sur l'ensemble des états donne une mesure du contraste de cette distribution. Cette entropie $H(V_S)$ est comparée à l'entropie maximale H_{max} de la distribution pour donner le ratio R_c qui décrit à quel point la distribution est piquée sur un sous-ensemble d'états (6.9).

$$P(S) = V_S/V \tag{6.7}$$

$$H(V_S) = - \sum_{S \in \mathcal{S}} P(S) \cdot \log_2(P(S)) \quad (6.8)$$

$$R_c = \frac{H(V_S)}{H_{max}} \text{ avec } H_{max} = \log_2|\mathcal{S}| \quad (6.9)$$

Ce mécanisme vise à faciliter la planification quand le nombre d'états augmente, mais il ne doit pas pour autant perturber le processus lorsque le nombre d'états reste faible. Le ratio R_c est transformé en un ratio R_n de sorte à ce que le nombre d'états choisis soit l'ensemble des états connus pour une taille de l'espace d'état faible, et corresponde au ratio du nombre total d'états au delà (6.10) :

$$R_n = (1 - \omega) + \omega \cdot R_c \text{ avec le poids } \omega = \frac{1}{1 + e^{-\sigma|\mathcal{S} - \mathcal{S}_0|}} \quad (6.10)$$

Les valeurs pour lesquelles on considère que le nombre d'états est *faible ou élevé* sont déterminées spécifiquement à la tâche : dans la tâche de poussée de blocs présentée plus bas (figure 6.4), le nombre d'états à partir duquel ce mécanisme entre en jeu est $\mathcal{S}_0 = 50$ ($\sigma = 0.25$)

Le nombre d'états N sur lequel aura finalement lieu la planification est le suivant (6.11) :

$$N = R_n \cdot |\mathcal{S}| \quad (6.11)$$

6.1.4 Meta-Contrôleur

Le meta-contrôleur arbitre entre les propositions des Experts, lors de la réception d'un état, selon une *méthode d'arbitrage* définie, et en parallèle de la décision des Experts. Nous détaillerons dans le chapitre 7 les différentes méthodes que nous avons étudiées pour effectuer cet arbitrage. Nous définissons également une méthode de référence, qui sert de démonstration de principe pour vérifier la faisabilité et la pertinence de notre architecture, et dont la performance servira de point de comparaison pour juger de l'intérêt des autres méthodes développées.

Cette méthode de référence est une *sélection aléatoire et équiprobable* de l'une des propositions fournies par les Experts. Dans ce chapitre, nous comparons deux versions simplifiées de l'architecture, où seul un des Experts contrôle le comportement durant toute l'expérience, à l'architecture complète et utilisant cette méthode de référence, afin de vérifier que la combinaison des deux experts est plus performante que chaque expert testé isolément.

6.1.5 Module de perception

Parce qu'un robot plongé dans un environnement réel risque toujours d'être confronté à de nouvelles informations, nous ne pouvons et ne souhaitons fournir un MDP explicite représentant notre problème réel pour les tâches considérées : c'est à l'agent de se faire sa

propre représentation en fonction de ses expériences. Pour cette raison, nous avons vu que l'expert dirigé vers un but construit son propre modèle de la tâche à partir des états qu'il rencontre. Comme il est difficile, voir impossible d'énumérer tous les états d'une tâche complexe, nous choisissons de les générer automatiquement à partir des informations perceptuelles pertinentes pour la tâche, ce qui *ancree* l'information utilisée à un niveau décisionnel à des dimensions pertinentes de l'environnement.

Ce rôle est dévolu au module de perception, qui transforme dynamiquement les informations perçues et éventuellement mémorisées en un état abstrait. Cette transformation est spécifique à chaque tâche étudiée, et sera présentée dans la description de l'expérience.

Par ailleurs, nous ne chercherons pas à trouver une méthode générale pour réaliser cette transformation, mais nous discuterons au chapitre 9 de son influence sur l'apprentissage et sur la qualité de l'information disponible à plus haut niveau.

Ce module de perception reçoit également un signal de récompense qui permet aux experts d'apprendre. Nous considérons que cette récompense est donnée directement par l'environnement ou par un autre système du robot extérieur à notre architecture de contrôle : un système de motivation qui aurait appris à valuer les éléments de l'environnement (e.g. un point de rechargement ou un être humain qui exprime de la satisfaction à l'égard du robot).

6.1.6 Module d'action

Comme nous nous intéressons au mécanisme général de transfert du contrôle vers l'un ou l'autre expert, nous souhaitons rendre la couche de décision agnostique de la nature de l'action effectuée. Ainsi, les experts ne conçoivent les actions que par leurs conséquences ou leur valeur, et le choix d'une action dans un état particulier n'est basé que sur son intérêt dans la tâche courante. Il est cependant nécessaire pour l'agent de transformer l'action discrète qu'il a décidé d'effectuer en ordres moteurs continus effectifs sur le robot. Cette tâche est accomplie par le module d'action, qui met à disposition de la couche de décision un certain nombre d'actions (assurant ainsi une partie du rôle de la couche fonctionnelle) et surveille leur exécution dans le monde réel (couche exécutive). Contrairement aux architectures robotiques classiques, il n'y a pas de notion d'échec d'une action : la conséquence de l'action modifie ou non les signaux perceptuels, mais la notion de succès ou d'échec n'a de sens que par rapport à une tâche définie. Par conséquent, c'est au niveau décisionnel que le comportement dirigé vers un but apprend (dans son modèle de transition) les conséquences de l'action (états d'arrivée et récompense éventuelle) et leur nature possiblement probabiliste (un couple état, action mène à plusieurs états).

Comme pour le module de perception, nous donnons à l'agent la capacité de faire des actions ad hoc à la tâche considérée ; toutes ces actions forment l'ensemble des compétences de l'agent, que les experts peuvent ou non recruter pour atteindre un but particulier.

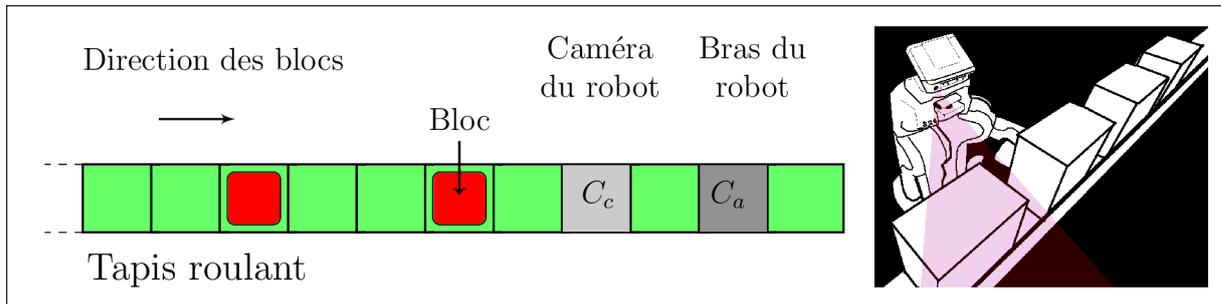


Figure 6.4 – Expérience de poussée de blocs. Gauche : schématisation de l'environnement du robot ; un tapis roulant discret transporte des blocs en direction du robot. Ce dernier peut vérifier la présence d'un bloc en C_c à l'aide de sa caméra, utiliser son bras en C_a pour pousser un éventuel bloc présent. Droite : illustration de la tâche.

6.2 Illustration expérimentale

6.2.1 Expérience

Notre architecture est implémentée avec ROS (QUIGLEY et al. 2009), ce qui la rend théoriquement directement transférable entre une simulation et un robot réel. Nous simulons un robot simplifié contrôlé par notre architecture afin d'évaluer son efficacité dans une tâche jouet de poussée de blocs (figure 6.4).

Le robot simulé est placé devant un tapis roulant discret sur lequel sont placés des blocs. Ces blocs sont entraînés à une vitesse v_b et à une distance d_{ib} l'un de l'autre. Dans cette thèse, nous garderons d_{ib} constante (4 blocs), mais nous nous autoriserons à faire changer v_{bs} pendant l'expérience, ce qui permet de définir deux conditions :

1. Condition régulière (RC) : v_b ne varie pas, l'environnement évolue de manière prédictible. La méthode d'arbitrage doit préférentiellement donner la main à l'expert habituel après qu'il ait appris, cet expert étant supposé capable de contrôler une politique performante pour un bas coût computationnel.
2. Condition changement de vitesse (SS) : v_b change de valeur environ à mi-expérience après une phase constante. La méthode d'arbitrage doit redonner la main à l'expert dirigé vers un but, supposé plus rapide à se réadapter à de nouvelles conditions en prenant en compte le nouveau modèle du monde.

Perception et construction de l'état

Comme nous l'avons précisé plus haut, l'agent construit les états manipulés par la couche de décision à partir des perceptions à sa disposition. Dans cette expérience, l'agent dispose de deux modalités : une modalité visuelle et une modalité tactile. La première apporte une information binaire sur la présence d'un bloc en C_c , la seconde sur la présence d'un bloc en C_a . De ces modalités sont respectivement produits les signaux binaires p_{bs} (pour *block seen*) et p_{bt} (pour *block touched*). Ces signaux alimentent une mémoire discrète par modalité (M^{bs} et M^{bt}) où chaque case représente l'évolution de l'occupation de la case

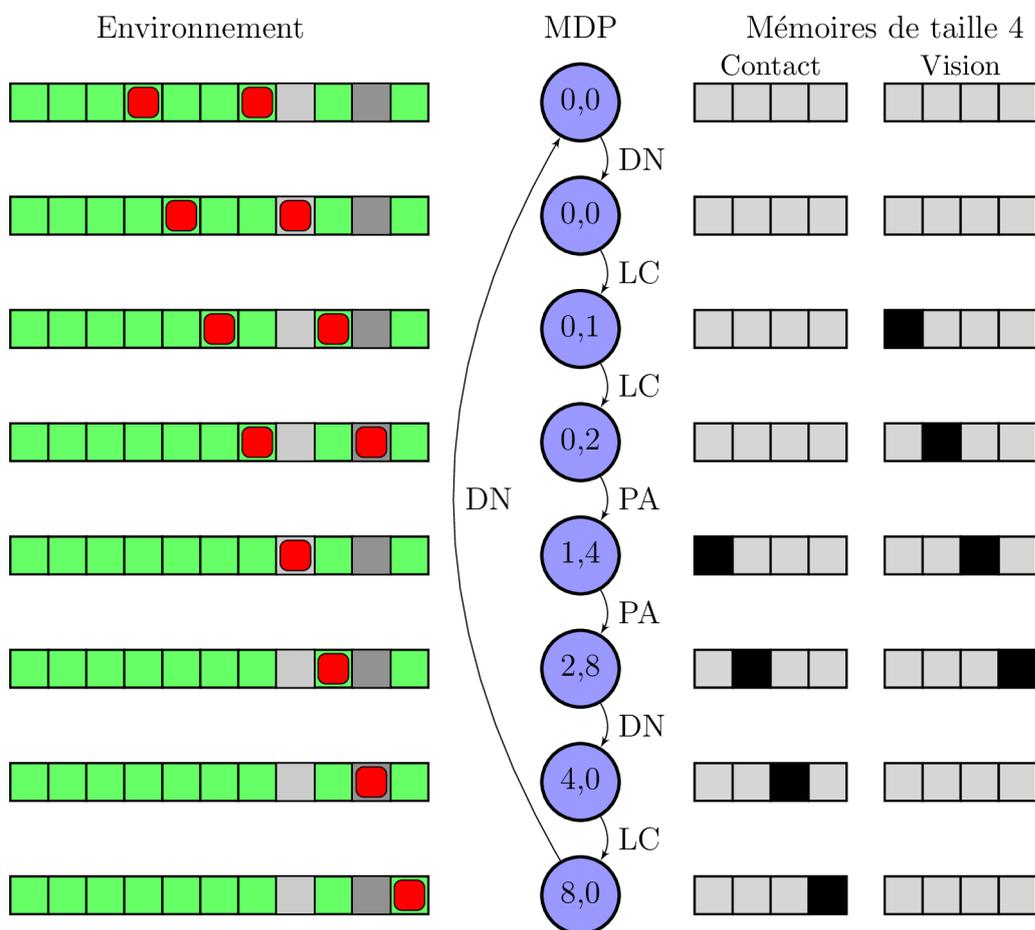


Figure 6.5 – Gauche : Évolution de l’environnement ; Centre : MDP théorique obtenu en suivant la politique indiquée et étant donné l’environnement décrit à gauche. Les chiffres représentent l’identifiant de l’état et sont directement la valeur décimale de la valeur binaire encodée par chaque mémoire, avec le bit de poids faible à gauche ; Droite : évolution des mémoires en fonction de la politique et de l’environnement. État (0,1) : l’action LC faite dans l’état précédent, quand un bloc est présent en C_c permet de mettre à jour la mémoire correspondante. Cette action répétée en (0,1) indique l’absence de bloc. L’état évolue naturellement vers (0,2) du fait de la mise à jour temporelle de la mémoire. En (0,2), PA permet de toucher et enlever le bloc précédemment vu, et l’état évolue vers (1,4). Les actions suivantes ne montrent la présence d’aucun bloc et les mémoires se vident progressivement pour un retour à l’état (0,0).

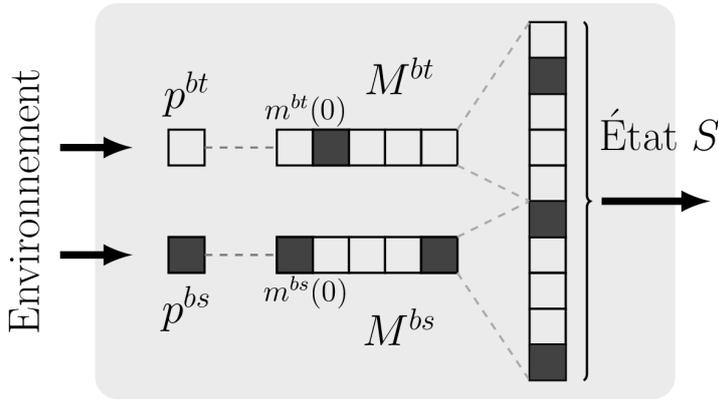


Figure 6.6 – *Module de perception pour la tâche de poussée de blocs. La couleur des cases correspond à l'information binaire de la présence d'un bloc (noir : présence, blanc : absence). p^{bt}, p^{bs} sont les signaux perceptuels instantanés, qui alimentent le premier élément de leur mémoire respective ($m^{bt}(0), m^{bs}(0)$). Les mémoires M^{bt}, M^{bs} sont concaténées pour former un vecteur de variables binaires qui définit l'état actuel expérimenté par l'agent.*

perçue par la modalité dans le passé. La configuration des mémoires à un instant t forme l'état s_t manipulé par les algorithmes d'AR (voir figure 6.6).

Ces signaux sont calculés en fonction de la présence d'un bloc à l'emplacement mesuré par leur modalité, et en fonction de l'action faite (6.12).

$$p^{bt} = C_a \cdot PA, p^{bs} = C_c \cdot LC \quad (6.12)$$

Les mémoires ont une taille finie (fixée à 8 éléments dans l'expérience), l'information la plus ancienne étant oubliée au fur et à mesure. Ces mémoires sont mise à jour temporellement : si une limite de temps est dépassée ($t_{max} = 0.1s$) dans l'expérience), les éléments connus en mémoire sont décalés vers le passé (équation (6.13)). Ce décalage se produit également lorsqu'un signal perceptuel apporte de l'information, afin de mettre à jour le premier élément de la mémoire avec la nouvelle information perçue.

Un exemple de l'évolution des mémoires pendant la tâche est donnée dans la figure 6.5.

$$\begin{cases} m_t^{bs,bt}(i) = m_{t-1}^{bs,bt}(i-1) & \forall i \in |M| \\ m_t^{bs,bt}(0) = p^{bs,bt} \end{cases} \quad (6.13)$$

Il est à noter que du fait de l'asynchronie entre le comportement du robot et l'environnement, des erreurs perceptuelles peuvent se produire, générant des états n'ayant pas de correspondance dans l'environnement. Par exemple, faire LC au moment où un bloc est présent en C_c permet de mettre à 1 le premier élément de la mémoire correspondante. Mais si cette action est répétée avant que le bloc n'ait été déplacé dans la simulation, deux éléments consécutifs de la mémoire retiendront la présence d'un bloc. Ainsi, deux états correspondent à la même configuration de l'environnement. Ces imperfections du système de perception participent à évaluer l'intérêt du principe de combiner des comportements dirigé vers un but et habituel dans des tâches plus proches de la robotique, où l'incertitude sensorielle est permanente.

Table 6.1 – Valeurs testées et choisies des paramètres des experts dans la première version de l’architecture.

Param	Valeurs	Hab	GD
α	0.01, 0.05, 0.1, 0.5, 0.9	0.1	0.5
γ	0.1, 0.5, 0.98, 0.9999	0.9999	0.5
τ	0.01, 0.1, 0.5, 1.0, 5.0	0.05	0.1

Actions

Le robot dispose de 3 actions fournies par son module d’action ; ces actions permettent de modifier l’environnement mais également de mettre à jour les perceptions du robot (voir 6.2.1). Ces actions sont les suivantes :

1. **Ne rien faire (DN pour *Do Nothing*)** : Le robot n’agit pas sur son environnement et attend la réception du prochain état. Cette action ne modifie pas l’environnement et ne met pas à jour les perceptions, mais n’a pas de coût en terme de récompense ($R_t = 0$).
2. **Regarder (LC pour *Look Cam*)** : Cette action permet au robot d’accéder à sa caméra et de mesurer l’état de la case C_c du tapis roulant. Elle met à jour p^{bs} , ne change pas l’état du tapis roulant, et a un coût $R_t = -0.03$ qui représente la dépense énergétique à faire cette action.
3. **Pousser (PA pour *Push Arm*)** : Cette action met à jour la perception p^{bt} et modifie le tapis roulant en enlevant un bloc présent en C_a . Elle a également un coût en énergie $R_t = -0.03$. Cependant, dans la tâche considérée, elle rapporte une récompense instantanée positive si un bloc est touché. Cette récompense est sommée au coût pour un total de $R_t = 0.97$ points de récompense reçu par le robot.

L’intérêt de ces actions est le suivant : la tâche étant répétitive et totalement prédictible dans la condition RC, on souhaite que l’agent découvre qu’il n’a pas besoin de l’action *LC* : en effet, en répétant *DN* puis en faisant *PA*, il est possible de recevoir la récompense en la réduisant au minimum par les coûts d’action. En condition SS, le nombre de répétitions de *DN* doit être réduit, puisque la fréquence des blocs augmente ; juste après le changement, *LC* retrouve un intérêt dans la mesure où percevoir un bloc en C_c permet de prédire qu’il sera en C_a deux cases plus tard. Après suffisamment d’essais à cette nouvelle vitesse, l’agent devrait découvrir à nouveau qu’il peut se passer de *LC*. En condition RC et en début de condition SS, $v_b = 8$ bloc/s, $v_b = 13.2$ bloc/s après le changement qui survient à $t = 1250$ décisions. Ces valeurs correspondent respectivement aux politiques optimales suivantes : DN-DN-DN-DN-PA, DN-DN-PA.

Enfin, les experts sont paramétrés en les évaluant sur la condition RC de la tâche : nous choisissons les jeux de paramètres qui maximisent la récompense accumulée durant l’expérience et à égalité, ceux qui assurent un écart type minimal à la récompense accumulée. Les valeurs testées et choisies sont présentées dans la table 6.1.

6.2.2 Résultats

Nous évaluons d'une part chaque expert individuellement, d'autre part l'architecture complète avec sélection aléatoire. Cela nous permet de définir la performance de référence face à laquelle nous comparerons nos autres méthodes d'arbitrage. Afin de valider l'intérêt de combiner des experts de nature différente au sein de la couche de décision, nous souhaitons d'une part que ces combinaisons améliorent la performance de l'agent comparée à un agent contrôlé par un expert seul ; d'autre part que les différentes méthodes d'arbitrage permettent de dépasser la performance de la sélection aléatoire, méthode d'arbitrage la plus naïve.

Pour estimer la qualité de la politique suivie par le robot, nous mesurons la récompense moyenne reçue par le système (plutôt qu'une estimation de la pente de la récompense cumulée (comme dans RENAUDO et al. (2014)). Cette méthode nous semble plus pertinente qu'une estimation de la pente faite uniquement en fin d'expérience, pour laquelle le choix d'une fenêtre de taille fixée influe sur l'estimation. La récompense moyenne est calculée incrémentalement par pondération exponentiellement décroissante des échantillons les plus anciens (filtre passe-bas numérique de paramètre $a = 0.01$), ce qui donne une estimation tout au long de l'expérience, le paramètre réglant la sensibilité à la dynamique mais pas la moyenne elle-même. Lorsque la vitesse est à 8 bloc/s (répétition de la séquence DN-DN-DN-DN-PA simultanément à la présence des blocs en C_a), la qualité de la politique optimale rapporte 0.194 pt/action, une politique complètement aléatoire rapportant 0.046 pt/action. Pour une vitesse de 13.2 bloc/s (DN-DN-PA), ces valeurs sont de 0.323 bloc/s pour la politique optimale et 0.091 bloc/s pour la politique purement aléatoire. La performance globale du robot est définie comme la récompense cumulée au cours de l'expérience (CR pour *Cumulative Reward*).

Dans la condition RC, on observe que l'expert habituel converge vers une meilleure politique en moyenne que l'expert dirigé vers un but (figure 6.7, haut). La combinaison trouve une politique de qualité intermédiaire, ce qui est cohérent avec le fait que la combinaison des propositions des experts est aléatoire. Les valeurs non nulles de la moyenne en début d'expérience s'expliquent par les choix « chanceux » du robot lors des premières actions : l'état exact du tapis roulant au moment où le robot agit pour la première fois est variable et il est possible qu'un bloc se trouve à la position atteignable par le bras et que le robot décide de faire PA, ce qui lui rapporte de récompense rapidement. Les configurations trouvent des politiques non-optimales mais meilleures qu'un comportement purement aléatoire, confirmant l'apprentissage du robot sur la tâche.

Ces meilleures politiques trouvées par l'expert habituel conduisent à une meilleure performance globale du robot en moyenne (figure 6.8, haut). Cependant, l'écart-type de la performance des politiques trouvées par l'expert habituel est plus importante que celle de l'expert dirigé vers un but, illustrant une plus grande capacité de ce dernier à reproduire la même performance au travers des répétitions de l'expérience. La combinaison obtient même un écart-type encore plus faible, suggérant une plus grande capacité à garder une politique de même qualité tout au long de l'expérience. En effet, quelque soit l'expert qui propose une action récompensée, les deux apprennent. L'expert habituel étant meilleur sur cette tâche, il permet à l'expert dirigé vers un but d'apprendre de meilleurs modèles

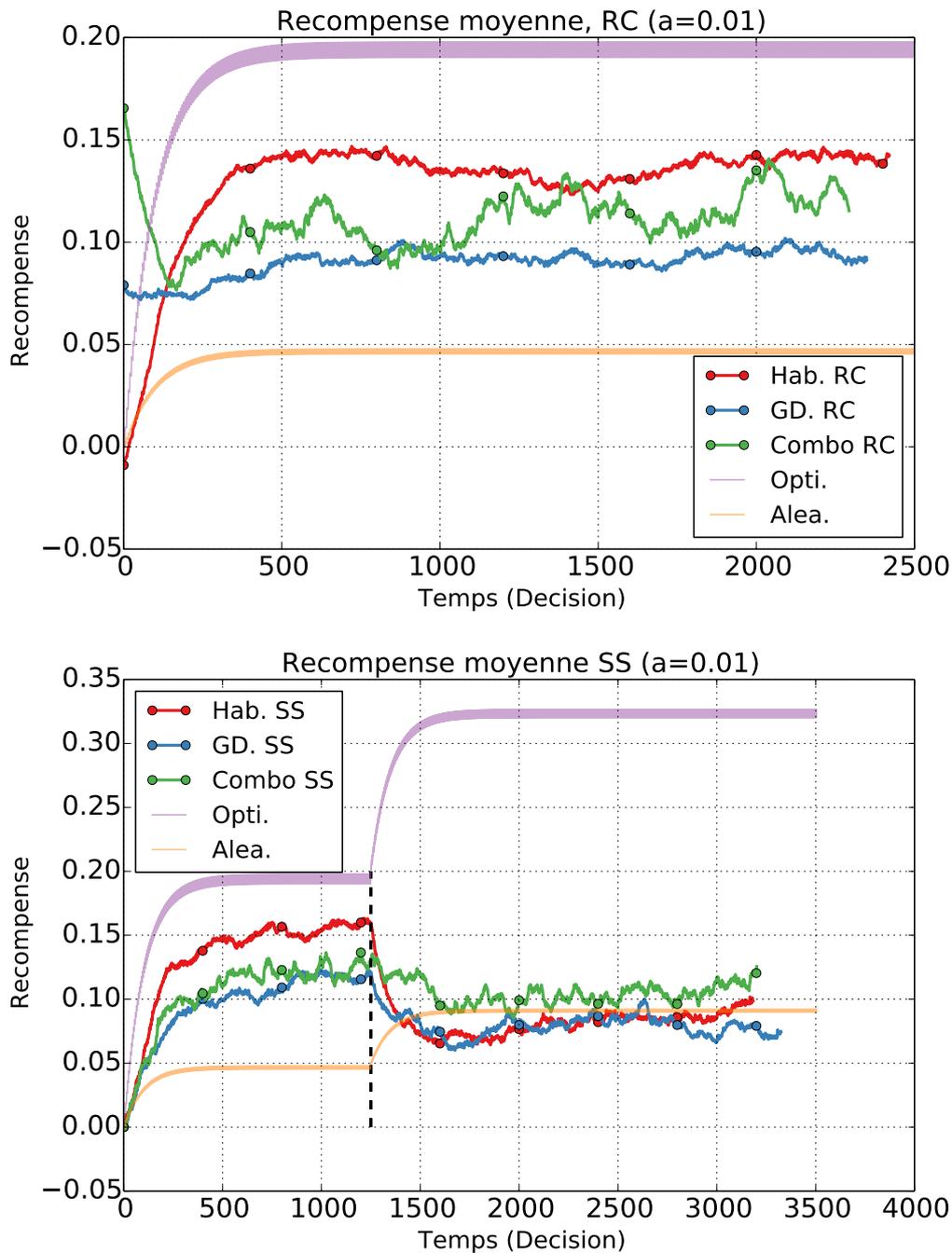


Figure 6.7 – Récompense moyenne (calculée incrémentalement par une moyenne glissante exponentielle de paramètres a) reçue dans les deux conditions et les différentes configurations (Hab : expert habituel seul; GD : expert dirigé vers un but seul; Combo : experts supervisés par le meta-contrôleur, Opti. : politique optimale; Aléa : politique aléatoire). **Haut** : les deux experts seuls convergent vers une politique de qualité stable, non-optimale, mais meilleure qu'un comportement purement aléatoire. La politique trouvée par la combinaison est intermédiaire, ce qui est principalement dû à la méthode d'arbitrage. **Bas** : lors du changement de vitesse, les récompenses moyennes chutent pour les experts habituel et dirigé vers un but. La combinaison est légèrement affectée par le changement de vitesse mais parvient à maintenir une politique plus performante que les experts pris individuellement qui font moins bien que l'aléatoire.

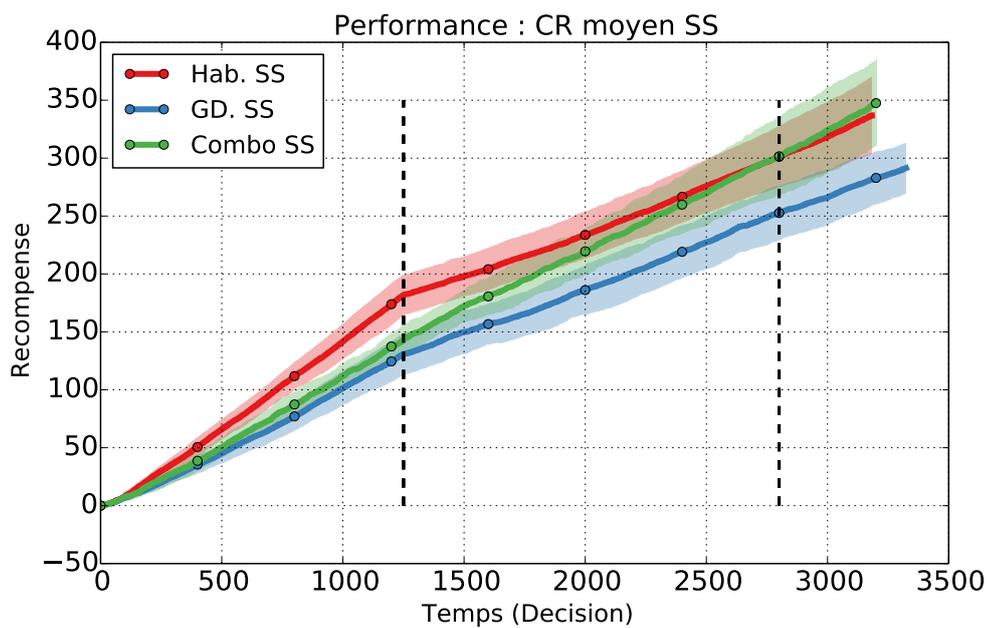
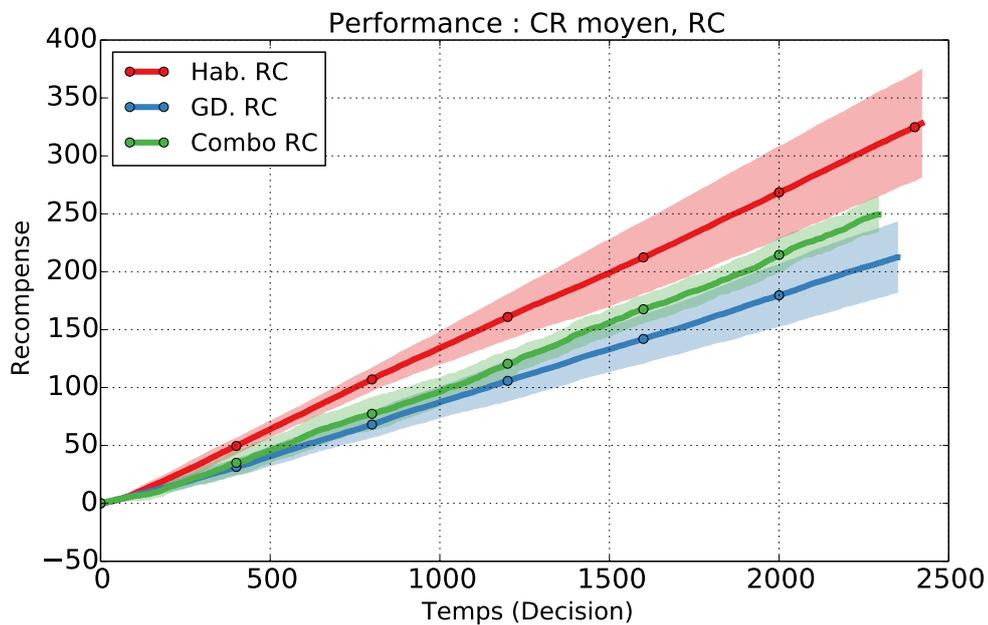


Figure 6.8 – Performances de l'architecture dans les deux conditions, et les différentes configurations de contrôle (Hab : habituel; GD : dirigé vers un but; Combo : experts supervisés par le meta-contrôleur). **Haut** : dans cette condition, l'expert habituel seul obtient la meilleure performance, comparé à l'expert dirigé vers un but. La combinaison avec sélection aléatoire obtient une performance intermédiaire, ce qui est cohérent avec ce type de sélection; **Bas** : les performances sont comparables à la condition régulière. La ligne pointillée de gauche indique l'instant du changement de vitesse, celle de droite l'instant où la combinaison rattrape la performance de l'expert habituel seul.

et de faire bénéficier la combinaison de sa moindre variance.

Dans la condition SS, le changement de condition provoque une chute de la qualité de la politique (figure 6.7, bas) et une cassure dans la performance du robot (figure 6.8, bas). La chute de qualité est moins importante pour l'expert dirigé vers un but que pour l'expert habituel, notamment parce que la qualité de sa politique était en moyenne moins élevée, les deux se retrouvant avec une politique de qualité comparable, mais également moins bonnes qu'une politique purement aléatoire. Pour la combinaison, la chute de performance est également moindre que pour les deux experts seuls. Elle conserve ainsi une qualité de politique plus élevée après le changement de condition et meilleure que l'aléatoire. En conséquence, la performance du robot contrôlé par la combinaison rattrape celle de l'expert habituel en fin d'expérience (figure 6.8, bas).

Cette évolution de la récompense s'explique par les propriétés de chaque expert : après avoir appris une politique et confronté à un changement, l'expert habituel a besoin de temps pour désapprendre la politique précédente et s'adapter à la nouvelle ; or, sa politique n'est plus adaptée aux nouvelles conditions (la nouvelle séquence d'action est plus courte et pas en phase avec la politique apprise). Il doit pour cela expérimenter à nouveau les différents états pour mettre à jour ses associations état - action, ce qui est rendu difficile par sa tendance à persévérer dans la même politique tant que trop peu de mises à jour ont été faites dans les différents états. L'expert dirigé vers un but doit également mettre à jour son modèle, mais peut exploiter sa connaissance des conséquences des actions pour trouver une nouvelle politique dès qu'il a un modèle suffisant. Dans cette expérience, la performance de ce dernier reste cependant limité par son approximation des valeurs d'action pour converger vers une politique de meilleure qualité.

6.3 Conclusion

Dans ce chapitre, nous avons proposé et implémenté une architecture de contrôle robotique simplifiée capable d'exhiber des comportements habituel et dirigé vers un but tels que classiquement modélisés par le courant principal de la littérature des neurosciences computationnelles. Nous avons évalué cette architecture dans une tâche simulée de poussée de blocs. Cette tâche met en jeu un environnement dynamique avec lequel l'agent doit agir en synchronie. Nous avons défini deux conditions expérimentales : dans la première (RC), les paramètres de la dynamique sont constants, et l'environnement prédictible, dans la seconde (SS), les paramètres changent à un instant donné et le robot doit réapprendre la bonne politique en fonction de la nouvelle dynamique.

Pour des experts paramétrés au mieux de leur performance dans la condition RC, nous avons étudié le comportement de chaque expert dans chaque condition et de leur combinaison aléatoire, comme démonstration de faisabilité de notre architecture. Nous avons montré que dans la condition RC, il est tout à fait possible d'obtenir une politique de bonne qualité en laissant le seul expert habituel contrôler l'agent. Dans la condition SS, la combinaison permet d'atténuer la perte de performance due à l'inadéquation de politique de la même manière que l'expert dirigé vers un but est moins affecté par le changement que l'expert habituel.

Ces résultats sont encourageants pour le principe de combiner plusieurs experts aux propriétés différentes. Dès lors que l'expert habituel a trouvé une politique pertinente, et à condition que l'environnement soit prédictible, il peut à moindre coût computationnel contrôler le robot efficacement pour accomplir la tâche apprise. Lorsqu'un changement de condition se produit dans l'environnement, la performance de l'expert dirigé vers un but et de la combinaison sont plus robustes que celle de l'expert habituel.

La performance inférieure de l'expert dirigé vers un but vient contredire l'hypothèse qu'il a une information parfaite évoquée dans KERAMATI et al. (2011). Plusieurs raisons viennent expliquer cette contradiction : premièrement, la différence entre les MDP considérés dans les tâches de neurosciences et ceux résultant d'une construction dans un problème robotique. Dans le premier cas, la plupart des tâches sont modélisées avec moins d'une dizaine d'états (voir par exemple DEZFOULI et BALLEINE (2012) ; KERAMATI et al. (2011) ; LESAIN et al. (2014)), et n'évoluent que de manière synchrone avec les actions de l'agent. Lorsque l'AR est appliqué à la robotique, la « malédiction de la dimensionalité » apparaît (KAELBLING et al. 1996 ; KOBER et al. 2013), rendant l'estimation des valeurs d'action par l'expert dirigé vers un but d'autant plus coûteuse. Comme l'agent doit en plus rester réactif à son environnement, l'évaluation des actions se fait au prix de leur précision. C'est le problème auquel nous sommes confrontés dans la mesure où notre robot rencontre plusieurs centaines d'états au cours d'une expérience (certains étant le résultat d'erreur de perception).

Si ce problème peut être contrecarré en travaillant sur la manière de construire les états, il peut également l'être en adaptant le processus de planification de l'expert dirigé vers un but. C'est l'approche proposée par HUYS et al. (2012) qui montre que des humains, confrontés à un raisonnement dont le nombre d'options croît rapidement, élaguent les possibilités en fonction de la valeur estimée des transitions. Dans ce travail, nous avons d'abord étudié la possibilité d'oublier progressivement les transitions du modèle trop peu visitées sans qu'une influence concrète puisse être constatée sur la performance. Nous avons ensuite adopté l'approche par planification budgétée présentée plus haut qui permet à l'expert dirigé vers un but de proposer une action dans un temps raisonnable indépendamment de la taille de son modèle, mais au prix de valeurs plus précises. Nous examinons au chapitre 7 une manière de rendre cet expert plus performant face à des environnements complexes.

Chapitre 7

Étude de critères pour le changement de comportement

7.1 Critères	80
7.2 Architecture	82
7.3 Résultats expérimentaux	84
7.3.1 Expérience et paramètres	84
7.3.2 Résultats	84
7.3.3 Conservation ou réinitialisation du plan de l'expert dirigé vers un but	87
7.4 Discussion	92

Nous avons vu précédemment que combiner un expert dirigé vers un but et un expert habituel pour décider du comportement global d'un robot offrait une bonne performance à moindre coût dans un contexte prédictible, et une robustesse de cette performance lorsque des changements surviennent dans les conditions de la tâche. Dans ce chapitre, nous nous intéressons aux méthodes d'arbitrage entre experts afin d'exploiter au mieux les propriétés intrinsèques de ces derniers.

Comme évoqué dans le chapitre 4, les littératures du comportement instrumental, de la navigation animale ainsi que de leurs contreparties robotiques proposent différentes méthodes pour combiner les estimations des deux experts. Ici, nous proposons de tester deux méthodes de sélection basées sur les signaux mesurables dans notre architecture, et quatre méthodes de fusion des valeurs d'action, dans la tâche de poussée de blocs décrite au chapitre précédent. Nous ne nous intéressons pas, dans cette thèse, à des méthodes de sélection par apprentissage. La raison est que ces méthodes associent un expert à un état du problème. Étant donné que nous associons à chaque expert un type de comportement dirigé vers un but ou habituel, les a priori sur ces comportements suggèrent qu'un contexte stable suffisamment expérimenté devrait voir un transfert du comportement de l'agent vers un comportement habituel, et un changement de conditions un retour vers le comportement dirigé vers un but. Une association état - expert ne nous semble donc pas



adaptée dans un cadre général du comportement.

Le travail présenté dans ce chapitre correspond aux publications suivantes :

Erwan RENAUDO, Benoît GIRARD, Raja CHATILA et Mehdi KHAMASSI (2015c).
« Which criteria for autonomously shifting between goal-directed and habitual behaviors in robots? » Dans : *5th International Conference on Development and Learning and on Epigenetic Robotics (ICDL-EPIROB)*. Providence, RI, USA, p. 254–260

Erwan RENAUDO, Benoît GIRARD, Raja CHATILA et Mehdi KHAMASSI (2015b).
« Respective advantages and disadvantages of model-based and model-free reinforcement learning in a robotics neuro-inspired cognitive architecture ». Dans : *Biologically Inspired Cognitive Architectures BICA 2015*. Lyon, France, p. 178–184

7.1 Critères

Afin d'étendre le modèle présenté dans le chapitre précédent, nous nous intéressons à des méthodes d'arbitrage pour tirer le meilleur parti des experts qui composent notre couche de décision.

Parmi l'ensemble des informations mesurables dans nos experts lorsqu'ils contrôlent seuls le robot, nous en retenons deux dont l'évolution au cours de la tâche est informative : la *récompense moyenne* (MR) reçue par l'agent et *l'entropie de la distribution de probabilité d'action* (E_{hab} et E_{GD}) de chaque expert. À partir de ces informations, nous proposons deux méthodes que nous appellerons « méthodes basées signal ».

La récompense moyenne est, comme nous l'avons évoqué précédemment, une mesure de la qualité de la politique suivie par l'agent (voir figure 7.1, haut). Plus cette valeur est haute, plus la politique est valable. À l'inverse, une moyenne négative indique une politique qui ramène plus de punition que de récompense. La décroissance de la récompense moyenne est un indice de l'inadaptation croissante de la politique à la tâche, et indique potentiellement un changement dans l'environnement. La méthode de sélection du meta-contrôleur est donc la suivante : la récompense moyenne à l'instant t , \bar{r}_t , est évaluée selon une moyenne glissante exponentielle (7.1) de paramètre λ . Sa variation est directement estimée par la différence temporelle entre deux mesures de la récompense moyenne : $\Delta\bar{r}_t = \bar{r}_t - \bar{r}_{t-1}$

$$\bar{r}_t = (1 - \lambda) \cdot \bar{r}_{t-1} + \lambda \cdot r_t \quad (7.1)$$

- Si $\Delta\bar{r}_t$ est négatif, le meta-contrôleur suit les propositions de l'expert dirigé vers un but : ce dernier permet théoriquement une meilleure réadaptation grâce à son modèle, ses propositions doivent mener à trouver une nouvelle politique efficace. Dans le cas où elle est relativement constante, c'est également l'expert dirigé vers un but qui a le contrôle par défaut, mais nous évoquons en discussion de ce chapitre d'autres propositions.

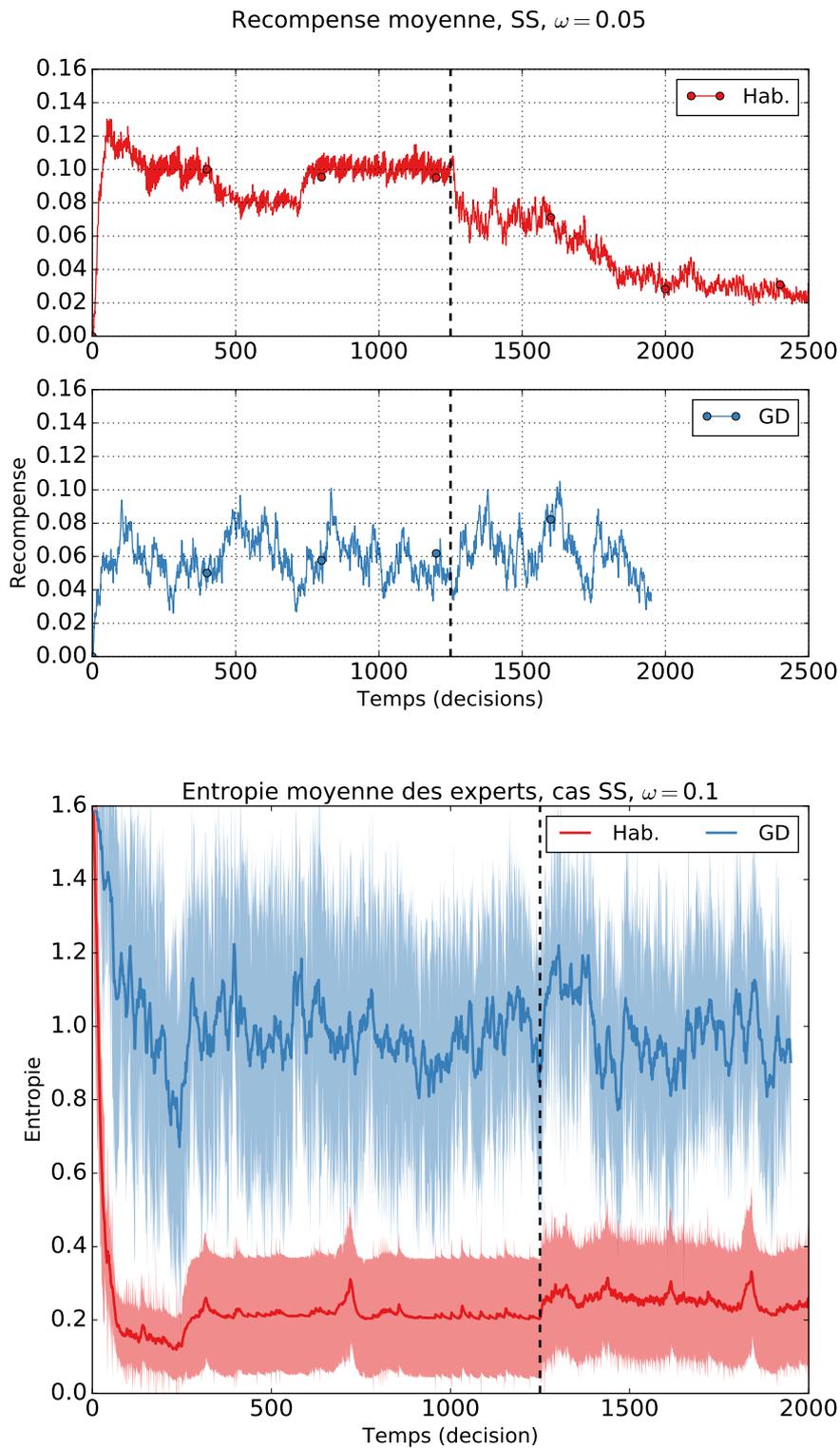


Figure 7.1 – Signaux internes moyens sur 10 répétitions de l'expérience de poussée de blocs dans la condition SS, lissée avec un filtre passe-bas de paramètre ω . **Haut** : Récompense moyenne. La chute de la qualité de la politique suivie est clairement visible pour l'expert habituel. **Bas** : Entropie moyenne. On observe une décroissance puis une stabilisation de l'entropie tant que le contexte est stable, puis une croissance lors du changement de vitesse.

- Si $\Delta\bar{r}_t$ est positif, le meta-contrôleur suit les propositions de l’expert habituel : en effet, si la politique s’améliore, les deux experts apprenant en parallèle, le coût computationnel de l’expert habituel est plus intéressant que celui de l’expert dirigé vers un but.

Dans une architecture où les experts renvoient une action choisie stochastiquement au meta-contrôleur, la distribution de probabilités qui a servi à prendre la décision donne une information sur le degré de connaissance qui supporte cette décision : une décision prise dans une distribution uniforme n’est ni réellement exploratoire, ni réellement exploitatrice : il n’y a simplement pas assez d’information pour faire mieux qu’un choix aléatoire. Au contraire, une distribution bien contrastée (i.e. piquée) reflète une décision informée, et une « volonté » de l’expert (pour autant qu’on puisse parler de volonté pour un mécanisme de choix aléatoire) d’explorer ou d’exploiter. L’entropie $H_t^{Hab,GD}$ de cette distribution (voir equation 7.2), qui évalue le contraste des probabilités, mesure une forme d’incertitude dans la décision de l’agent. Cette incertitude baisse au fur et à mesure que l’agent reçoit de la récompense, donc de l’information sur quelle action choisir (voir figure 7.1, bas). La méthode de sélection consiste donc à choisir la proposition de l’expert le plus confiant, donc celui dont l’entropie de la distribution d’action est la plus faible.

$$H_t^E(x) = - \sum_{i=0}^{|\mathcal{A}|} P_i * \log_2(P_i) \quad (7.2)$$

avec E l’expert considéré et $P_i = p(a = a_i | s)$.

Enfin, nous avons vu dans le chapitre 3 comme dans le chapitre 4 qu’il existe des méthodes, que nous appellerons « méthodes basées fusion », qui synthétisent les estimations de valeur ou de probabilités d’actions fournies par chaque expert pour prendre ensuite la décision. Plutôt que la décision d’un expert, ce type de méthode d’arbitrage favorise les actions qui font le consensus de plusieurs experts.

Étant donné notre architecture et l’organisation décrite plus bas, ces méthodes de fusion peuvent être utilisées directement : au lieu de prendre une décision par expert, l’action est décidée par le meta-contrôleur. Les experts transmettent leurs estimations des probabilités d’action, et ces dernières sont fusionnées selon la méthode choisie. Dans ce chapitre, nous évaluons les quatre méthodes proposées par WIERING et VAN HASSELT (2008) : *Majority Voting*, *Rank Voting*, *Boltzmann Addition*, *Boltzmann Multiplication*. Nous ne les détaillons pas ici et renvoyons le lecteur à la partie 3.5.2 du chapitre 3 pour leur fonctionnement exact.

7.2 Architecture

L’architecture considérée dans ce chapitre est semblable à celle du chapitre précédent, à l’exception d’une modification dans la relation entre experts et meta-contrôleur (voir figure 7.2). En effet, les informations utilisées par les méthodes d’arbitrage décrites plus haut ne peuvent être obtenues qu’après planification de chaque expert. Si la récompense moyenne peut être mesurée globalement, cela n’est pas le cas des distributions de pro-

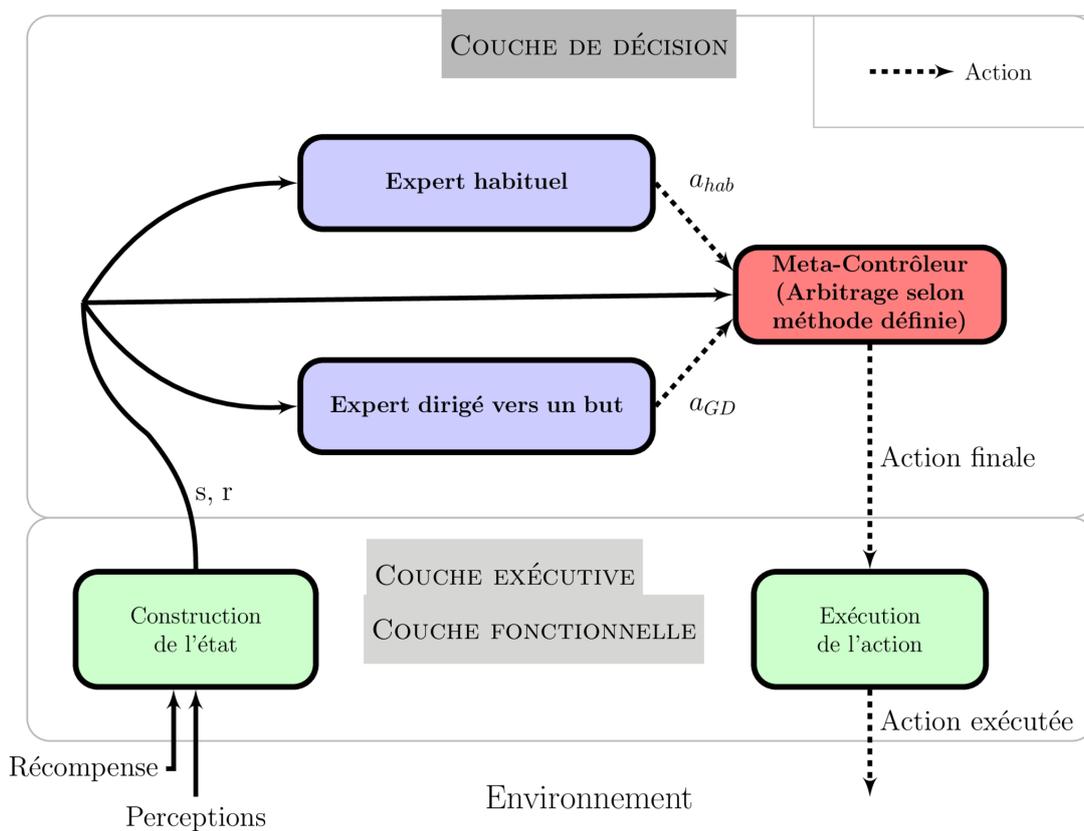


Figure 7.2 – Architecture modifiée : le meta-contrôleur reçoit les propositions des experts et arbitre en fonction de la méthode choisie : soit il fait suivre l'une des deux propositions (méthode signal), soit il forme une distribution de probabilités sur les actions pour prendre lui-même une décision.

Table 7.1 – Valeurs testées et choisies des paramètres des experts dans la seconde version de l'architecture.

Param	Valeurs	Hab	GD
α	0.0001, 0.001, 0.01, 0.1, 0.5, 0.9	0.5	0.01
γ	0.1, 0.5, 0.98, 0.9999	0.9999	0.98
τ	0.01, 0.1, 0.5, 1.0, 5.0	0.1	0.01

babilités d'actions. Le meta-contrôleur est donc placé en sortie des experts et attend de recevoir les informations nécessaires à la méthode choisie pour arbitrer et transmettre une action finale pour exécution au reste du système. Cette organisation, comme celle du chapitre précédent, implique cependant que chaque expert doit faire cette planification en parallèle. Par ailleurs, l'arbitrage a posteriori offre un meilleur contrôle sur la gestion des propositions des experts : en effet, avec l'arbitrage a priori, le temps pris par le meta-contrôleur pour sélectionner l'un des experts et leur envoyer sa décision est comparable à celui pris par l'expert habituel pour choisir une action. Si cet expert avait le contrôle précédemment, il peut envoyer sa proposition avant d'avoir reçu le nouveau choix du meta-contrôleur, et outrepasser l'arbitrage.

7.3 Résultats expérimentaux

7.3.1 Expérience et paramètres

Nous évaluons les méthodes d'arbitrage dans l'expérience de poussée de blocs décrite au chapitre 6. Les paramètres de l'expérience ne changent pas, mais nous optimisons comme au chapitre précédent ceux des experts, suite au changement structurel de l'architecture.

λ (le taux de mise à jour dans le calcul de la récompense moyenne) est fixé à $\lambda = 0.1$, ce qui donne une durée de vie de 10 états dans le passé ; les cycles de notre tâche entre deux récompenses étant de l'ordre de la demie douzaine d'états, cette valeur rend le système sensible aux variations de récompense de l'ordre de deux poussées réussies de blocs, donc des variations relativement rapides devant la durée de la tâche complète.

7.3.2 Résultats

Comme au chapitre précédent, nous examinons la performance de nos différentes configurations - experts seuls ou combinés par l'une des méthodes présentées plus haut - dans les différentes conditions - régulière et changement de vitesse - sous forme de la récompense cumulée au cours de l'expérience (figure 7.3 pour les méthodes de sélection, 7.4 pour celles de fusion).

Dans les deux conditions, l'expert dirigé vers un but est moins performant que l'expert habituel seul, tandis que la méthode d'arbitrage aléatoire permet une meilleure récompense en moyenne. Ces performances diffèrent du chapitre précédent où nous trouvions que la performance de cette méthode était intermédiaire à celles des deux experts seuls.

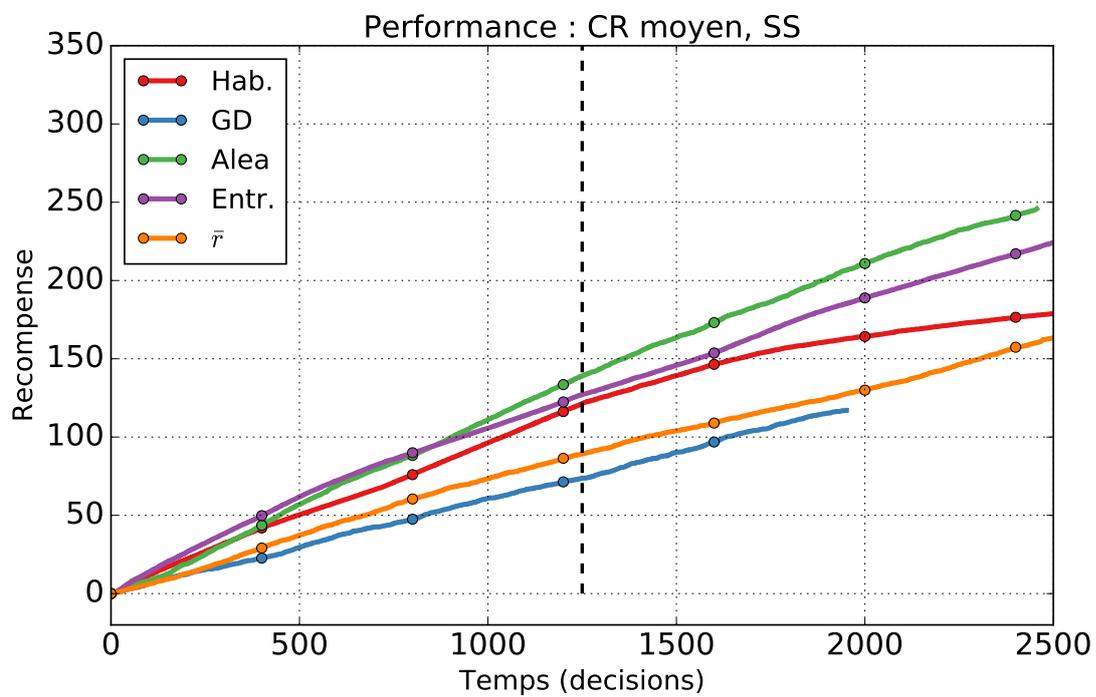
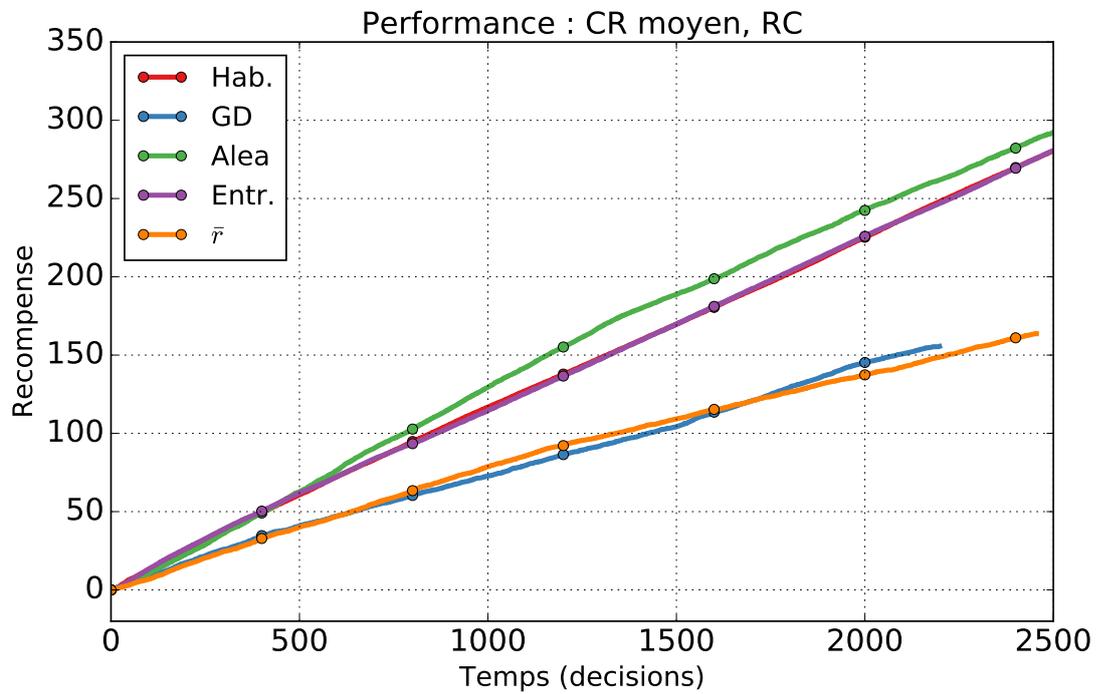


Figure 7.3 – Performance des méthodes de sélection dans les deux conditions, pour 10 itérations de l'expérience. **Haut** : condition RC. **Bas** : condition SS.

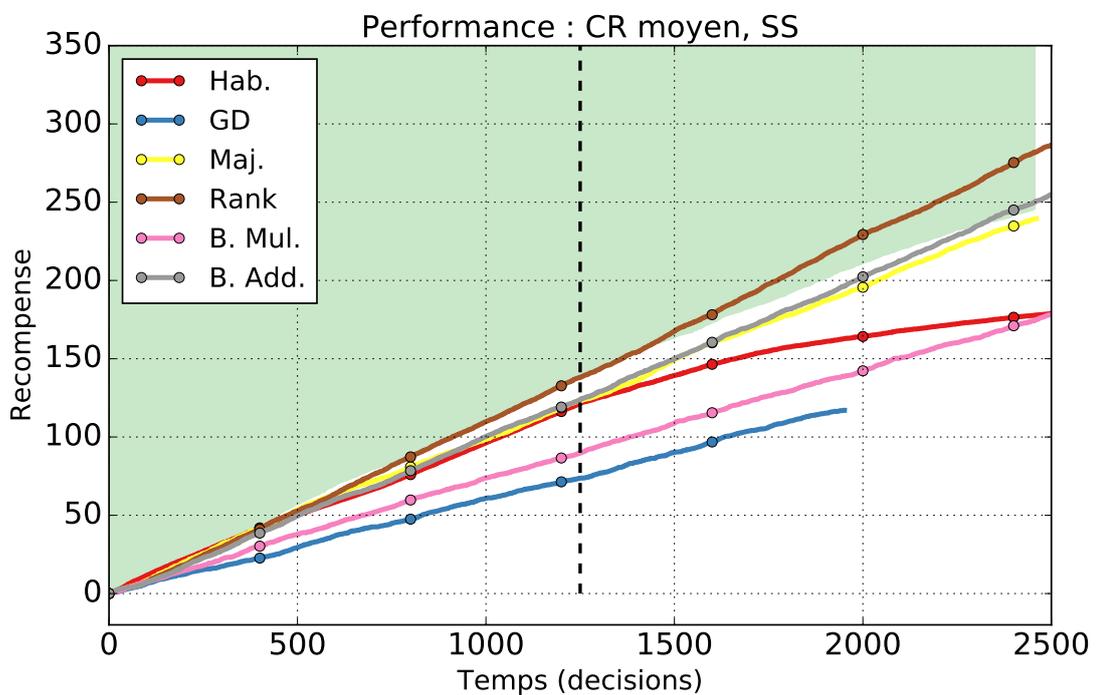
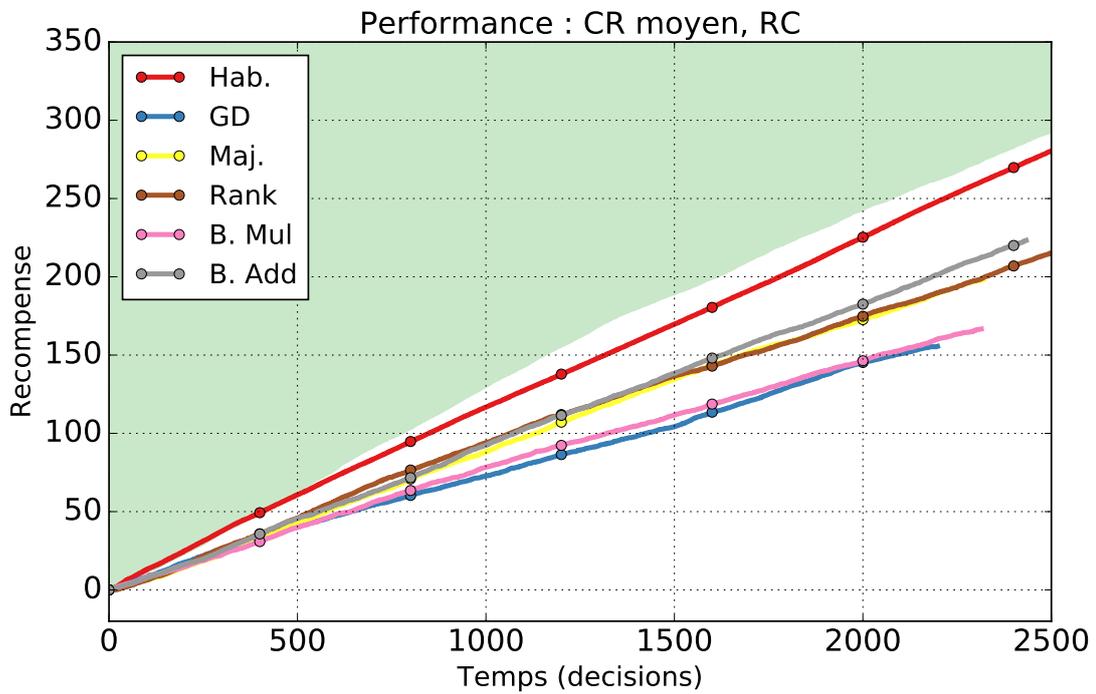


Figure 7.4 – Performance des méthodes de fusion dans les deux conditions, pour 10 itérations de l'expérience. Les aires vertes indiquent les performances supérieures à la sélection aléatoire. **Haut** : condition RC. **Bas** : condition SS.

Cela s'explique par le paramétrage différent : la performance obtenue par l'expert habituel est bimodale (figure 7.5) ; certaines expériences ont permis d'accumuler une forte récompense tandis que d'autres restent proches d'une performance nulle ou négative. Dans ces dernières, il peut arriver que le robot se retrouve bloqué dans l'état initial ou qu'il persiste à exploiter une action non optimale mais qui a mené vers un état de meilleure valeur (par exemple, continuer à faire LC bien qu'il n'y ait pas de blocs à voir et que DN suffirait).

Dans la condition RC, les méthodes d'arbitrage par sélection (figure 7.3, haut) ont une performance moyenne proche de celles des experts seuls : la méthode basée sur la récompense moyenne suit la performance de l'expert dirigé vers un but et celle basée sur l'entropie suit la performance de l'expert habituel. Les taux de sélection de chaque expert pour chaque méthode 7.6 montrent de manière cohérente que l'un des deux experts est majoritairement en contrôle du robot. Ces performances sont dues aux biais introduits par les méthodes : l'arbitrage basé sur la récompense moyenne tend à donner la main préférentiellement à l'expert dirigé vers un but par construction ; c'est également le cas quand la politique permet une récompense stable. Ainsi, la performance de la combinaison est similaire à celle de cet expert. La méthode basée sur l'entropie fait principalement confiance à l'expert habituel : en cachant les récompenses reçues dans ses valeurs Q, il obtient de bonnes estimations et sa distribution de probabilité d'action est très contrastée. L'expert dirigé vers un but au contraire est contraint en temps pour planifier, ce qui rend ses estimations de valeurs moins précises et sa distribution de probabilités moins contrastée. L'entropie résultante est donc plus élevée que celle de l'expert habituel.

Les méthodes de fusion ne permettent pas non plus de dépasser la performance de l'expert habituel ou de la combinaison aléatoire dans la condition RC, mais font mieux que l'expert dirigé vers un but, à l'exception de la fusion par multiplication de Boltzmann, dont la performance est rattrapée par celle de l'expert dirigé vers un but seul.

Dans la condition SS, la performance de l'expert habituel seul décroît après le changement de vitesse, tandis que les performances atteintes sont inférieures à celles de la condition RC. Les méthodes de fusion et de sélection ne sont pas affectées de manière importante par le changement de conditions, ce qui est une nouvelle démonstration que l'on gagne par rapport à un expert habituel à le combiner d'une manière ou d'une autre avec un expert dirigé vers un but.

En revanche, dans la condition SS, la méthode *Rank Voting* arrive à dépasser la performance de la sélection aléatoire, et les méthodes *Majority Voting* et *Boltzmann Addition* à s'en rapprocher. Si dans le cas RC, *Rank Voting* avait une performance relativement variable, celles obtenues dans la condition SS ont une variance plus faible 7.5. En dehors de l'expert dirigé vers un but, c'est la seule méthode qui exhibe un tel changement entre les deux conditions.

7.3.3 Conservation ou réinitialisation du plan de l'expert dirigé vers un but

Les expériences présentées dans ce chapitre ont été l'occasion pour nous d'analyser plus profondément les propriétés du système et les raisons expliquant ses performances dans

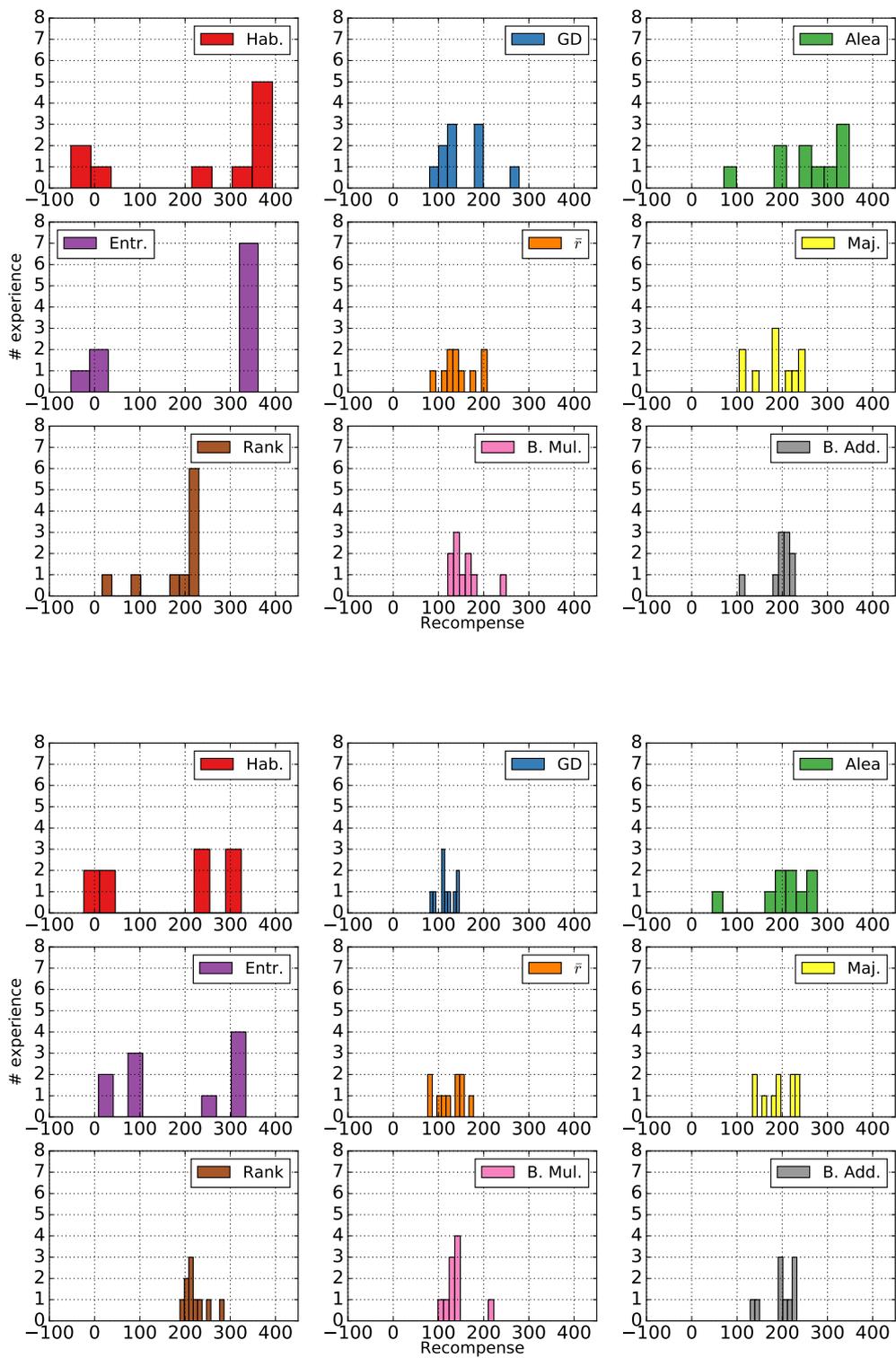


Figure 7.5 – Histogrammes de performance des méthodes d'arbitrage. **Haut** : condition RC. **Bas** : condition SS.

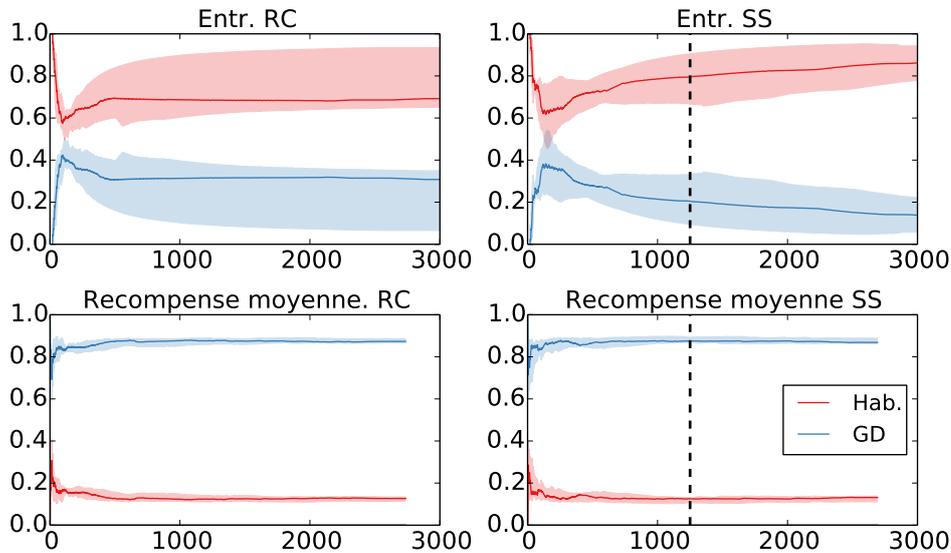


Figure 7.6 – Taux de sélection pour les méthodes d’arbitrage basées signal, dans les deux conditions. **Haut** : sélection selon l’entropie, RC et SS. **Bas** : sélection selon la récompense moyenne.

les différentes conditions. En particulier, comme nous avons vu dans le chapitre précédent et confirmé avec les résultats ci-dessus, l’expert dirigé vers un but a une performance relativement faible dans cette tâche, ce qui contraste fortement avec les hypothèses en neurosciences computationnelles selon lesquelles l’expert dirigé vers un but a accès à une information parfaite. Cela est principalement dû au fait que, contrairement aux simulations en neurosciences où la tâche est modélisée par une dizaine d’états maximum, dans nos expériences robotiques l’expert construit souvent de l’ordre de 200 états. Cet expert disposant d’un budget et d’un temps limité pour planifier à chaque décision, il ne peut faire qu’une propagation partielle de l’information dans le modèle pendant la planification, ce qui résulte difficilement en une bonne estimation des valeurs. Ainsi, la plupart des combinaisons testées avec différentes méthodes d’arbitrage obtiennent des performances comprises entre celles des experts dirigé vers un but et habituel plutôt que des performances meilleures ou au moins égales à celles de l’expert habituel.

Or, les modèles du comportement sont en général proposés et évalués dans des tâches non dynamiques, où seules les actions de l’agent font évoluer l’état. Il leur est donc possible d’évaluer, uniquement à partir des récompenses connues, les valeurs d’action dans chaque état DAW et al. (2005) ; DEZFOULI et BALLEINE (2012). Au contraire, en intelligence artificielle, les algorithmes *anytime* (ZILBERSTEIN 1993) adressent la question du compromis solution exacte / temps de calcul, et la recherche d’une solution se fait incrémentalement au cours des décisions et des actions de l’agent.

Nous proposons donc ici une nouvelle version de notre expert dirigé vers un but et le comparons avec la version précédente en le testant seul dans notre tâche de poussée de bloc. La version *RaZ*, pour remise à zéro, correspond à la version utilisée jusqu’à présent et directement inspirée des modèles de neurosciences : une planification contrainte en

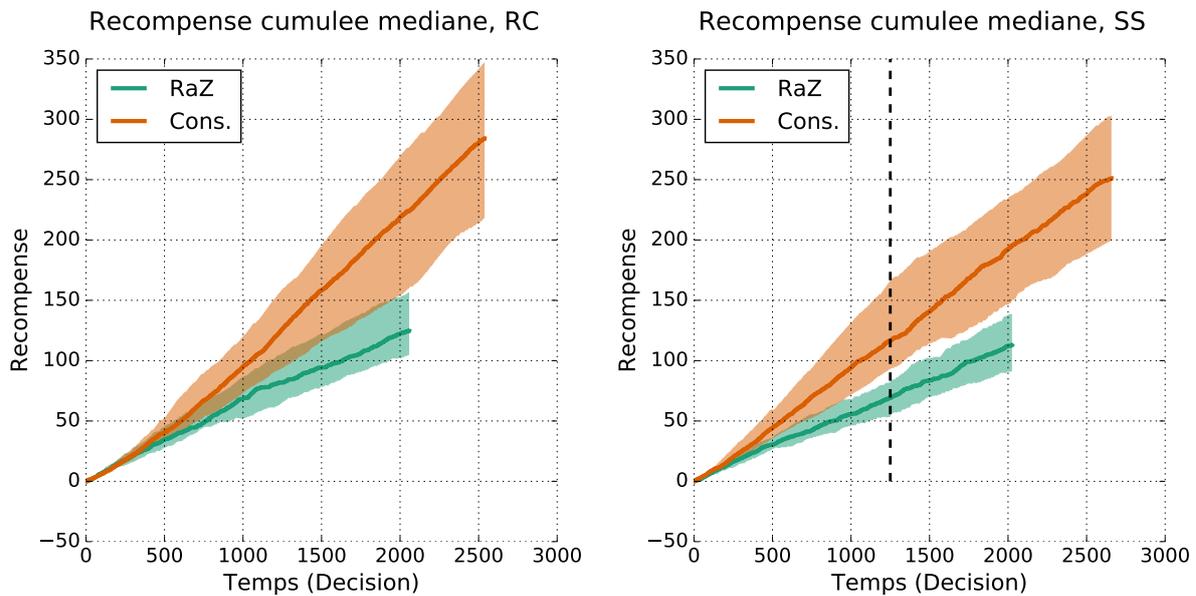


Figure 7.7 – Performance médiane et 1^{er} et 3^{me} quartiles des deux versions de l'expert dirigé vers un but, pour 50 répétitions de l'expérience dans chaque condition. **Gauche** : Condition RC. **Droite** : condition SS. La ligne en pointillés indique l'instant du changement de vitesse.

temps, pour laquelle après chaque décision, les estimations de valeurs sont oubliées. La nouvelle version *Cons.* que nous proposons ici, pour conservation des valeurs, repart de l'estimation précédente pour ajuster ses estimations aux nouvelles informations reçues sur le monde suite à la dernière action effectuée.

Résultats

Comme attendu, nous trouvons que conserver les estimations de valeur améliore grandement la performance de l'expert dans notre tâche. Ce résultat se vérifie dans les deux conditions (figure 7.7). Garder le plan n'altère pas la performance dans la condition SS (ce qui pourrait être le cas si le problème changeait drastiquement, les estimations devenant un a priori erroné). Le temps de calcul nécessaire à la planification est également réduit pour la version *Cons.* (figure 7.8 ; les temps affichés représentent le temps de planification et le temps de décision, mais ce dernier ne dépend pas des connaissances de l'expert : il s'agit simplement d'un tirage dans la distribution de probabilités d'action ; ce temps ne change donc pas la relation entre les temps de planification des versions des experts ; de plus, le dépassement comparé au budget de temps alloué - 0.1 s - s'explique par le fait qu'une passe d'évaluation des états peut démarrer alors que le budget est quasiment épuisé et prendre plus de temps que restant pour se terminer, la vérification étant faite à la fin de la passe). En effet, les informations initiales permettent de réduire le nombre de passes nécessaires sur les états avant convergence des valeurs. Dans une tâche de dimensionnalité importante, dynamique, il est donc important d'exploiter au mieux l'information acquise au prix des calculs effectués.

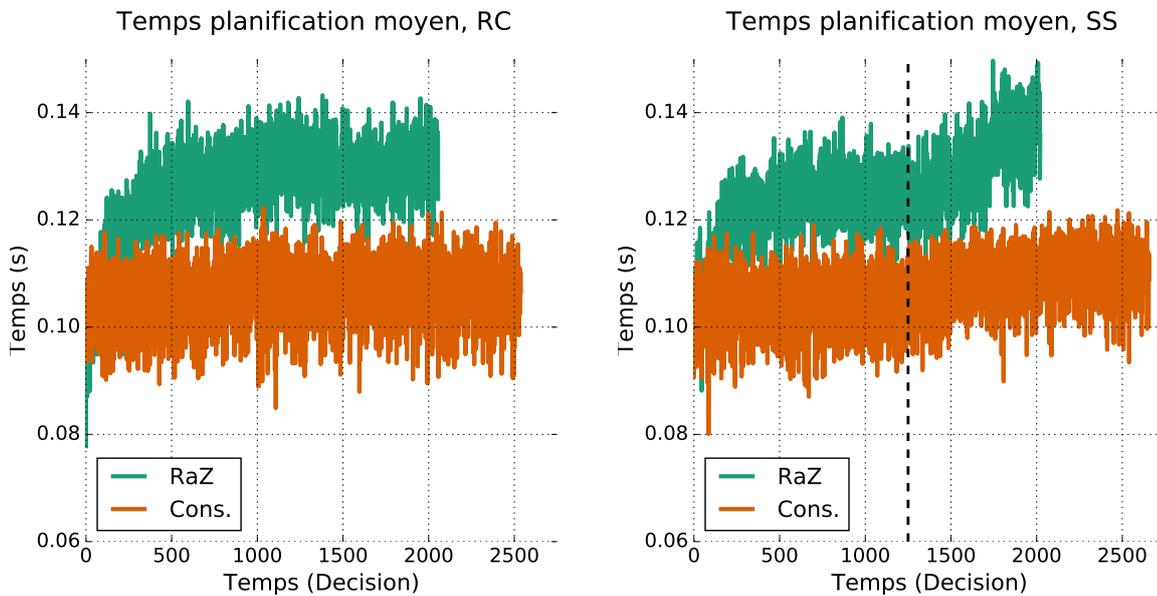


Figure 7.8 – Temps de planification et décision des deux versions de l’expert dirigé vers un but.

Dans la condition SS, le temps pour la version *RaZ* augmente de manière importante après le changement de vitesse, alors qu’il n’augmente que peu en moyenne pour la version *Cons.*. En effet, entre chaque décision, le modèle est mis à jour, et bien que le changement de vitesse provoque l’apparition de nouveaux états, ceux-ci sont intégrés au fur et à mesure dans le modèle. L’estimation des valeurs précédentes ne nécessite donc que peu de mises à jour puisque les modèles de transition et de récompense sont suffisamment proches de leur version précédente. La version *Cons.* nécessite ainsi un temps plus faible pour planifier.

Bien que ce mécanisme améliore les performances de l’expert dirigé vers un but, il ne lui permet pas réellement de faire mieux que les meilleures instances de l’expert habituel, celles-ci correspondant au plus haut des deux modes de performance entre lesquels cet expert alterne (figure 7.9). Dans la condition RC, leurs performances sont comparables, mais dans la condition SS, l’expert habituel est parfois encore meilleur qu’en condition RC (s’il découvre que le cycle pour obtenir de la récompense est plus court, ou qu’il suit une politique qui consiste à répéter PA, sa performance augmente naturellement : répéter PA rapporte en moyenne 0.3 pt/action de récompense, comparé à 0.194 pt/action dans le cas optimal RC). Ses performances sont bimodales, ce qui est dû à sa paramétrisation : en optimisant selon la récompense moyenne, la variance des performances reste haute. À l’inverse, si l’on cherche des paramétrisations qui réduisent cette variance, alors la performance moyenne de l’expert est plus faible (figure 7.10). Ces résultats montrent que face à un environnement dynamique et changeant, il est plus intéressant de s’appuyer sur un expert habituel peu coûteux et tout de même capable de trouver une bonne politique, plutôt qu’un expert dirigé vers un but dont le modèle, construit au fur et à mesure des expériences du robot, ne peut être exploité correctement sous les contraintes imposées. Ceci contredit à première vue les hypothèses venant des neurosciences que nous avons déjà

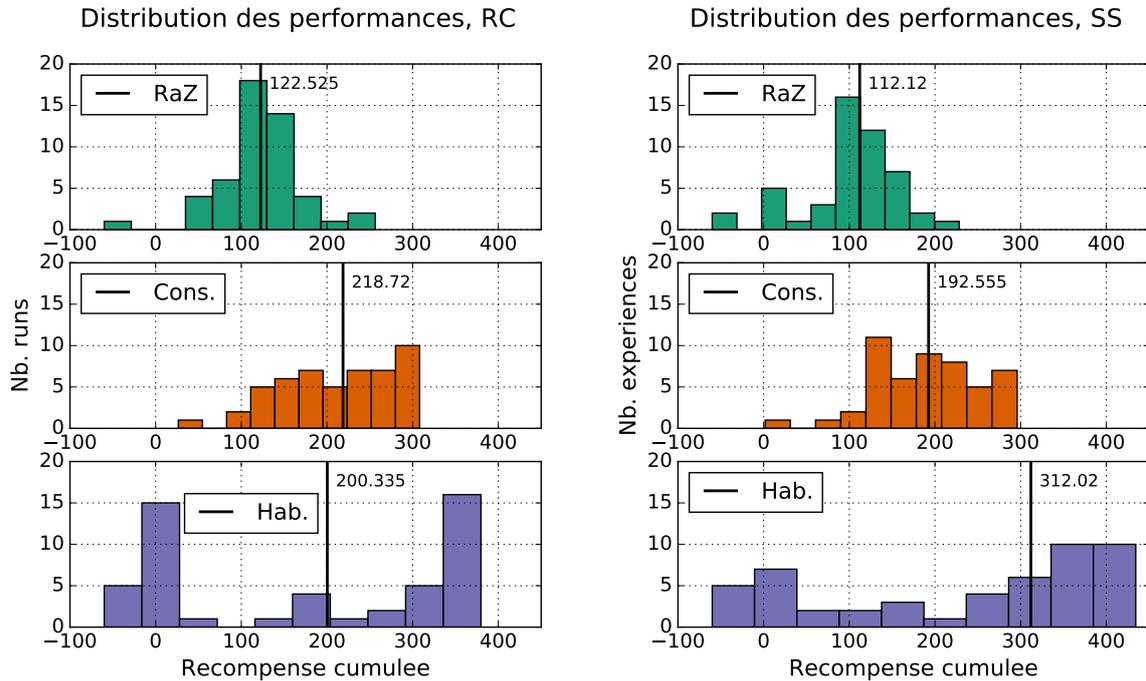


Figure 7.9 – Histogrammes de la performance à 2000 décisions, pour 50 répétitions de l’expérience. **Haut** : expert dirigé vers un but avec remise à zéro des valeurs. **Milieu** : expert dirigé vers un but avec conservation des valeurs. **Bas** : expert habituel. Les lignes verticales indiquent la valeur médiane de la distribution.

mentionnées selon lesquelles l’expert dirigé vers un but est sensé être systématiquement plus performant que l’expert habituel (e.g. KERAMATI et al. (2011)). Néanmoins, ceci semble renforcer l’idée qu’il est utile de combiner plusieurs méthodes d’apprentissage si l’on veut pouvoir généraliser à plusieurs tâches, notre meta-contrôleur devant être capable dans chaque tâche rencontrée de discriminer automatiquement quel expert est le plus performant.

7.4 Discussion

Dans ce chapitre, nous avons étudié différentes méthodes pour arbitrer entre nos experts en fonction des informations disponibles au sein de l’architecture. Nous avons évalué deux méthodes de sélection, basées sur la récompense moyenne et l’incertitude de chaque expert sur sa décision, et quatre méthodes de fusion, qui combinent les estimations de probabilité d’action de chaque expert selon le principe qu’une décision communautaire rend le choix d’action final plus pertinent.

Nous avons montré que dans notre tâche dynamique, aucune de ces méthodes ne surpasse nettement une sélection aléatoire des propositions des experts, bien que la méthode de fusion *Rank* obtienne une meilleure performance dans la condition SS. Ces résultats sont principalement dus à la difficulté pour l’expert dirigé vers un but, implémenté comme un

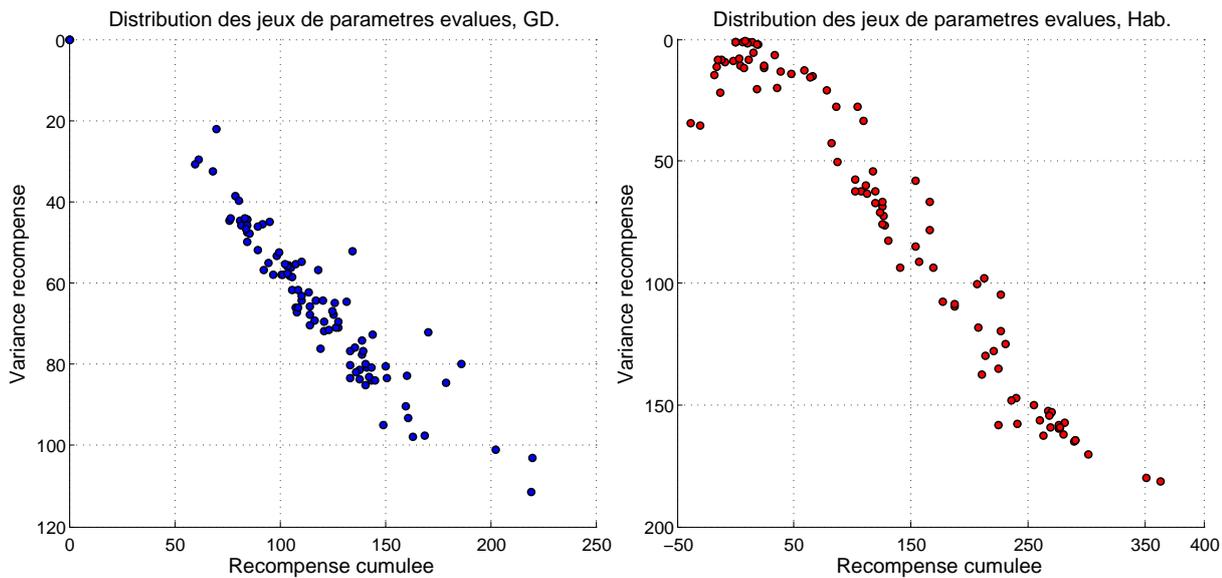


Figure 7.10 – *Distributions des jeux de paramètres selon leur performance moyenne et leur variance (l'origine de l'axe des ordonnées est en haut). Chaque point représente 10 répétitions de l'expérience pour un jeu de paramètres. Les paramètres qui optimisent la performance moyenne tendent également à faire augmenter la variance dans la performance de l'expert. **Gauche** : expert dirigé vers un but. **Droite** : expert habituel*

algorithme d'AR indirect, d'obtenir une bonne performance et des estimations correctes des valeurs d'action, lorsqu'il est contraint à planifier avec un budget limité.

Dans les modèles du comportement en neurosciences, l'utilisation de l'expert dirigé vers un but suppose qu'il planifie à partir de zéro : aucun a priori n'est conservé sur la valeur des actions. C'est d'ailleurs l'explication avancée pour les temps de réaction supérieurs des animaux (DAW et al. 2005 ; KERAMATI et al. 2011) après changement de condition. Or nous avons montré que la performance de cet expert peut être améliorée en effectuant cette planification au fur et à mesure des décisions du robot. C'est l'approche des algorithmes *anytime* évoqués plus haut, et c'est également la piste suivie par HESTER et al. (2012). Dans ce cas, la recherche de solution est effectuée non pas depuis la récompense mais depuis l'état courant du robot, jusqu'aussi loin que le temps le permet. Cette approche permet de se concentrer sur les états les plus probablement visités dans le futur, à partir des informations de transition connues plutôt que selon une heuristique, comme nous l'avons fait (« les états les plus visités sont les plus intéressants »). KERAMATI et al. (2011) exploite effectivement le modèle de son expert dirigé vers un but de cette manière mais s'arrête après une profondeur de recherche fixe.

Les méthodes basées signaux que nous avons évalué présentent un certain nombre de limites ; si nos résultats sur la conservation du plan nous incitent à ne pas conclure définitivement sur les résultats des méthodes d'arbitrage, quelques points sont tout de même améliorables :

Pour la récompense moyenne, le paramètre λ règle à quel degré de dynamique de la tâche le système est sensible. Bien estimer cette dynamique pour régler correctement ce

paramètre n'est pas trivial : une valeur trop faible rend le meta-contrôleur aveugle à des changements plus rapides, tandis qu'une valeur trop forte fait changer d'expert quasiment à chaque action. Dans notre tâche, cette dynamique change également d'une condition à l'autre ; il est donc nécessaire de faire varier ce paramètre en fonction de la dynamique perçue de l'environnement.

En l'état, la méthode sélectionne par défaut les propositions de l'expert dirigé vers un but lorsque la récompense moyenne est stable. Dans la mesure où une récompense moyenne stable indique que la politique suivie rapporte régulièrement de la récompense, il serait plus logique de s'appuyer sur les décisions de l'expert habituel, qui doit pouvoir suivre cette politique efficacement sans évaluer les actions dans l'ensemble du modèle de transitions.

Le critère d'entropie se rapproche de la notion d'incertitude des experts proposée par DAW et al. (2005). Dans leur travail, l'incertitude est calculée sur les connaissances de l'expert vis-à-vis de l'action considérée. Dans notre proposition, cette incertitude est liée à la décision d'une action, et ne nécessite pas de maintenir une distribution de valeurs pour chaque action et dans chaque état. Une limitation est que cette incertitude peut être erronée vis-à-vis de la tâche : l'expert habituel tend à persister dans la même politique le temps que soient propagés les changements dans la récompense. Le risque est donc que l'expert reste très certain dans sa proposition alors qu'il n'a pas pris en compte les nouvelles conditions de l'environnement. Il serait également plus pertinent de mesurer cette entropie sur les valeurs Q que sur les probabilités résultantes : la transformation réalisée par le *softmax* introduit un biais à cause de la température choisie (qui peut être différente pour chaque expert). Dans notre cas, ça ne change pas le fait que l'expert dirigé vers un but est plus incertain que l'expert habituel : sa valeur de τ est 10 fois supérieure à celle de l'expert habituel. Cette mesure permet cependant d'estimer une notion d'incertitude des experts sans un coût mémoire important (maintenir une distribution pour chaque action et chaque état, DAW et al. (2005)) ou un coût computationnel inévitable (mise à jour du Kalman Q -learning, KERAMATI et al. (2011)).

Pour les méthodes de fusion, nous obtenons des résultats différents de WIERING et VAN HASSELT (2008) : pour eux, les méthodes *Boltzmann multiplication* et *Majority Voting* dépassent les performances des autres méthodes, alors que la première a une performance faible dans nos résultats. En effet, multiplier les valeurs d'action tend à réduire le contraste, d'autant plus si l'une des distributions est proche de l'uniforme, comme celle de l'expert dirigé vers un but. À l'inverse, *Rank Voting* exacerbe les contrastes faibles entre deux actions en leur donnant la même importance qu'un contraste fort entre deux actions. Ainsi, la moindre variation entre les valeurs d'action est exploitée, et même un faible retour de récompense permet de faire ressortir l'action qui l'a rapporté. Cette différence s'explique également par la nature des experts mis en jeu. Dans leur approche, les algorithmes d'AR combinés sont tous directs, ils ne sont donc pas confrontés à la difficulté d'exploiter un modèle dans une tâche contrainte en temps. Ces résultats plutôt contradictoires limitent l'importance des conclusions que nous pouvons tirer de cette étude, mais les résultats sur la conservation ou la remise à zéro du plan de l'expert dirigé vers un but suggèrent qu'il est nécessaire d'évaluer plus profondément ces méthodes d'arbitrage.

Comme évoqué plus haut, la performance de l'expert dirigé vers un but est liée directe-

ment à sa capacité à exploiter son modèle, et par conséquent à avoir un modèle adéquat du problème. Dans l'optique d'une méthode d'arbitrage par sélection, donner au meta-contrôleur des informations sur le degré d'apprentissage du modèle semble pertinent pour que la sélection de cet expert puisse se faire lorsqu'il est efficace. Plus généralement, le meta-contrôleur devrait pouvoir estimer l'état de l'apprentissage dans chaque expert pour effectuer son arbitrage. Nous étudierons cette idée dans la seconde partie du chapitre 8.

Chapitre 8

Extension à d'autres tâches

8.1	Expérience d'interaction	98
8.1.1	Architecture et experts	98
8.1.2	Description de la tâche d'interaction	101
8.1.3	Résultats	102
8.1.4	Discussion intermédiaire	106
8.2	Expérience de navigation	107
8.2.1	Discussion	118
8.3	Conclusion	121

Dans ce chapitre, nous nous concentrons sur l'évaluation de notre architecture dans des tâches différentes, afin d'étudier de manière plus générale le principe de combiner différents systèmes d'apprentissage à un niveau décisionnel pour la robotique. Nous présentons des résultats préliminaires sur deux tâches non dynamiques : une tâche de coopération homme-robot, où deux agents agissent dans le même environnement, et une tâche de navigation, où le robot doit apprendre à rejoindre un lieu but.

Nous nous intéressons d'abord à une tâche d'interaction homme-robot où la coopération de l'humain est nécessaire pour atteindre le but. Nous posons les bases d'une architecture qui combine un expert dirigé vers un but capable de prendre en compte l'humain dans son processus de planification et un expert habituel apprenant les associations état-action. Nous souhaitons étudier le problème sous l'angle de la prise de décision dans un contexte spécifique, afin de comprendre comment notre architecture peut contribuer à plus long terme aux problèmes liés à l'interaction homme-robot (GOODRICH et SCHULTZ 2007).

Dans la tâche de navigation, nous nous intéressons à une variation de notre architecture où la méthode d'arbitrage donne le contrôle du robot à l'un des deux experts, qui peut planifier, tandis que l'autre reste observateur et ne fait qu'apprendre du comportement global. Nous proposons de baser cette sélection sur le coût à planifier mesuré directement dans chaque expert, ainsi qu'une mesure de leur apprentissage. Nous présentons des premiers résultats sur l'étude de cette variation dans une tâche de navigation sur robot réel.

Le travail sur la tâche d'interaction correspond à la publication suivante :



Erwan RENAUDO, Sandra DEVIN, Benoît GIRARD, Raja CHATILA, Rachid ALAMI, Mehdi KHAMASSI et Aurélie CLODIC (2015a). « Learning to Interact with Humans Using Goal-Directed and Habitual Behaviors ». Dans : *RoMan 2015, Workshop on Learning for Human-Robot Collaboration*.

8.1 Expérience d'interaction

Le développement de la robotique que nous évoquons en introduction conduit à un nombre croissant d'applications où le robot n'est pas isolé mais fonctionne parmi les êtres humains. L'interaction homme-robot (HRI), qui cherche à adresser les questions relatives à ces situations a émergé comme un domaine à part entière (GOODRICH et SCHULTZ 2007, chapitre 3). En HRI, l'autonomie du robot n'est pas vue comme un objectif en soi mais comme un moyen d'améliorer l'interaction. Nous adoptons un point de vue différent mais compatible dans ce travail (et cette thèse en général) : l'autonomie est une fin en soi, et les mécanismes généraux qui la supportent doivent permettre d'adresser n'importe quel problème, donc en particulier un problème d'interaction.

8.1.1 Architecture et experts

Nous proposons donc de repartir de l'architecture, présentée dans les chapitres précédents. Plus particulièrement, nous interfaçons notre couche de décision aux couches exécutives et fonctionnelles de l'architecture de contrôle robotique de ALAMI et al. (1998), l'idée étant ici de partir d'un système qui est utilisé couramment au LAAS pour la planification lors de tâches d'interaction homme-robot, et d'étudier dans quelle mesure notre coordination d'apprentissages peut aider à affiner les performances, notamment en permettant d'éviter le coût et le temps de la planification lorsqu'une solution moins coûteuse a été apprise par notre expert habituel. Dans ce travail la gestion des états et l'exécution des actions sont assurés par PRS (*Procedural Reasoning System*, INGRAND et al. (1996)) : ce dernier maintient la base de données des faits, qui est mise à jour en fonction des informations perçues par le robot ; il exécute et supervise les actions du plan reçu. Dans notre cas, les actions sont reçues une à une plutôt que sous forme de plan, du fait de l'arbitrage réalisé par le meta-contrôleur.

De la couche de décision, nous conservons l'organisation en experts monitorés par le meta-contrôleur, mais nous changeons l'expert dirigé vers un but, pour utiliser le planificateur HATP plutôt qu'un algorithme d'AR indirect (figure 8.1). Le meta-contrôleur reçoit donc une proposition issue d'une part de l'algorithme Q-learning, qui apprend les associations état-action, et d'autre part de HATP, dont le plan prend en compte le comportement que doivent avoir les autres agents pour progresser vers le but.

Ce travail étant pour l'instant exploratoire, comme au chapitre 6, le meta-contrôleur se contente d'un choix aléatoire parmi les propositions des experts. Parmi les six méthodes proposées au chapitre 7, cinq exploitent la distribution de probabilités qui est estimée par chaque expert. Or, la nature du planificateur fait que cette information n'est pas disponible (la planification utilisant un domaine déterministe). Nous évoquons en discussion quelques

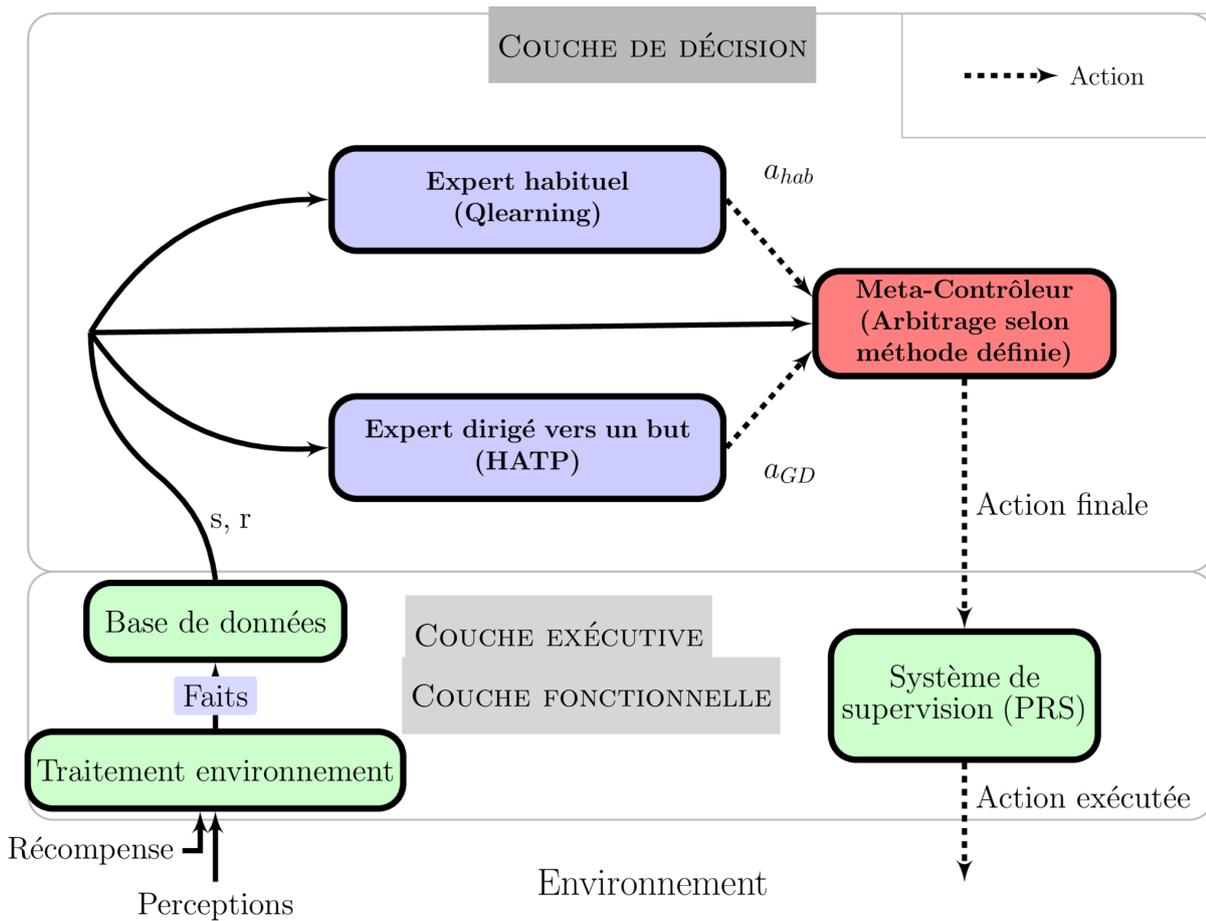


Figure 8.1 – Architecture de contrôle du robot. Les couches fonctionnelle et exécutive sont réalisées par PRS. L'état est déduit des valeurs des faits stockées dans la base de données, et associé à la récompense reçue de l'environnement.

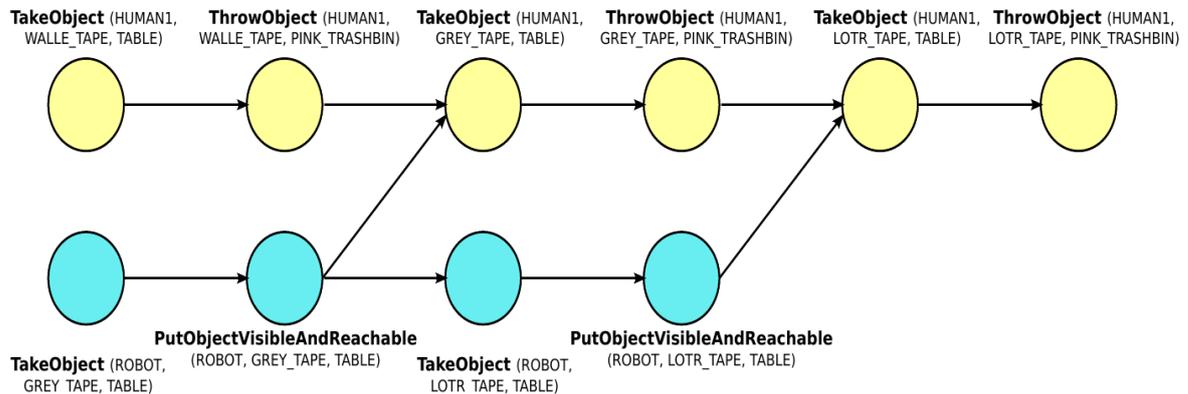


Figure 8.2 – Exemple de plan produit par HATP. Chaque couleur correspond au flux de décision d'un agent. Pour progresser dans le plan, certaines actions nécessitent l'intervention de plusieurs agents et demandent une synchronisation.

possibilités pour un critère pertinent.

HATP

HATP (pour *Human-Aware Task Planner*, LALLEMENT et al. (2014)) est un planificateur HTN qui intègre la notion d'agents multiples dans la recherche de ses plans. Il s'appuie sur une description de domaine, qui représente les connaissances transmises par un concepteur expert. Ce domaine est comparable au modèle du MDP d'un algorithme d'AR indirect. À partir de ce domaine, HATP cherche une solution de coût minimal, en terme de temps d'exécution mais également de coûts sociaux (e.g. éviter une action brusque pour ne pas effrayer l'être humain, ou équilibrer la répartition des actions pour ne pas surcharger le collaborateur). Un exemple de plan « Human-Aware » est donné figure 8.2.

Les deux experts ont des rôles complémentaires :

- HATP possède des connaissances a priori, qui ne sont pas apprises, mais donne une base pour planifier immédiatement, réduisant le besoin d'explorer. Comme la plupart des planificateurs, et plus globalement des processus de recherche de solution à long terme, il est sensible à la malédiction de la dimensionnalité, et met plus de temps à trouver un plan lorsque la complexité du problème augmente. Par ailleurs, il est possible qu'aucun plan ne soit trouvé à partir des connaissances fournies dans le domaine
- L'expert habituel sous la forme d'un algorithme Q-learning a besoin d'apprendre, ce qui prend du temps, réparti entre les différentes actions du robot, mais contrairement au planificateur, il ne vérifie par la faisabilité d'une action : il l'essaye. L'expert habituel est donc un bon moyen d'explorer des solutions que le planificateur ne peut concevoir à cause des limites de son domaine.

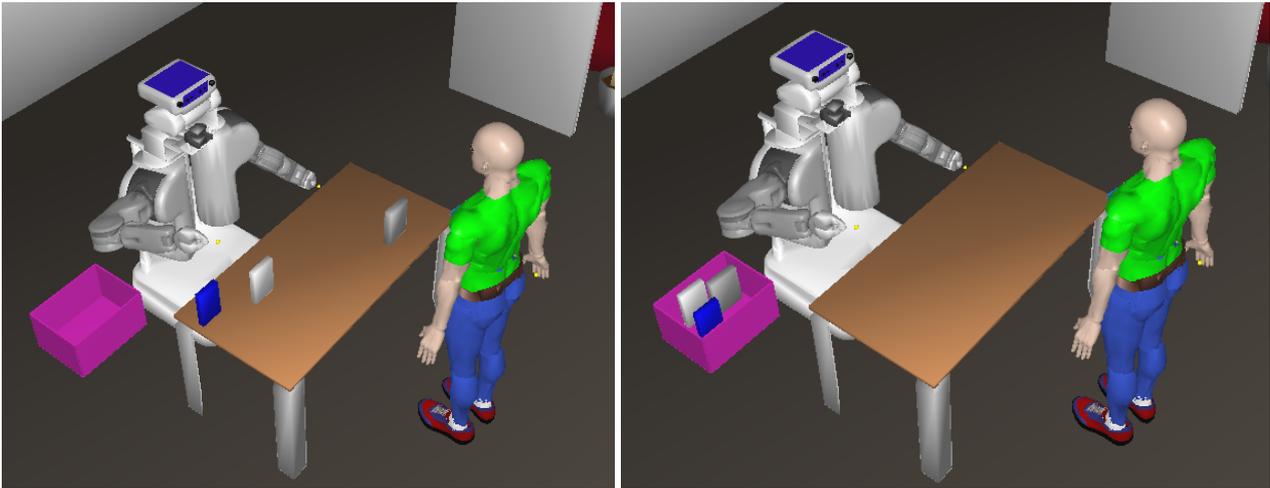


Figure 8.3 – Le robot et l’humain sont face à la table, où une boîte est atteignable uniquement par l’humain, les deux autres et la poubelle sont atteignables uniquement par le robot. *Gauche* : État initial. *Droite* : État final.

8.1.2 Description de la tâche d’interaction

La tâche d’interaction est inspirée de la tâche décrite dans ALAMI et al. (2011). Le but commun au robot et à l’humain est de nettoyer la table, i.e. mettre toutes les boîtes dans une poubelle (figure 8.3). La tâche est simulée en calculant les interactions physiques, mais les capteurs sont simplifiés en extrayant directement les informations géométriques depuis le simulateur (move3D, SIMÉON et al. (2001)). L’architecture logicielle du robot est directement adaptable sur robot réel (MALLET et al. 2000 ; CLODIC et al. 2009 ; LEMAIGNAN et al. 2012).

L’état est construit à partir d’un vecteur de faits booléens. Ces faits sont mis à jour automatiquement en fonction des relations géométriques perçues (SISBOT et al. 2011).

- Object isReachableBy Robot (including Trashbin)
- Object isReachableBy Human (including Trashbin)
- Object isIn Trashbin
- Human isReachableBy Robot
- Robot hasInHand Object
- Human hasInHand Object

Dans le vecteur, chaque fait est calculé pour les différents objets de la tâche et les différents agents : les trois boîtes et la poubelle pour *IsReachableBy*, les trois cassettes pour les autres faits. Ce vecteur comporte donc 18 informations sur l’environnement.

L’ensemble des actions accessibles aux deux agents de l’interaction est le suivant : PickObject, ThrowObject, GiveToHuman = (give(from robot), take(from human)), TakeFromHuman =(give(from human), take(from robot)), Wait, Goto (Object, Trashbin, Human).

L’action *Wait* permet à un agent d’attendre pendant que l’autre agit ou qu’une limite de temps est atteinte. Cette action permet au robot de se resynchroniser avec son partenaire

s'il ne peut lui-même faire évoluer le problème vers le but (typiquement s'il ne reste qu'une seule cassette, hors de sa portée). Dans ce travail, nous avons simulé un humain coopératif; nous envisageons à plus long terme des situations où les réactions de l'humain sont moins conciliantes, ou des scénarii qui alternent entre humain coopératif et non coopératif. Ce dernier cas peut être rapproché du changement de vitesse dans la tâche de poussée de blocs : un contexte stable, qui permettait une politique bien définie et un contrôle habituel, change et rend cette politique inadaptée. Les actions *GiveToHuman* et *TakeFromHuman* sont des méthodes décomposables en actions élémentaires impliquant les deux agents : pour que l'action se fasse, le *give* de l'un doit coïncider avec le *take* de l'autre. L'action *GoTo* permet de rejoindre la position d'un objet ou de l'autre agent et potentiellement de rendre vraies les faits *IsReachableBy*. De la même manière que pour les faits, ces descriptions génèrent 18 actions qui correspondent aux différentes valeurs que peuvent prendre leurs arguments.

8.1.3 Résultats

La performance (récompense cumulée) est analysée pour les configurations suivantes : expert habituel et dirigé vers un but seuls, combinaison aléatoire des experts. Le robot reçoit 1 pt de récompense lorsque la table est vidée et qu'il attend. À ce moment, l'environnement reprend sa configuration initiale et la tâche peut être accomplie à nouveau. La performance est donc directement une image du nombre de fois où la table a été nettoyée avec succès. Le temps total des expériences est fixé, cependant le nombre de décisions prises peut varier.

Contrairement aux résultats obtenus dans les chapitres précédents, la performance de l'expert habituel est relativement faible dans le temps imparti (figure 8.4), le robot ne réussissant pas à accomplir la tâche plus de deux fois en moyenne (soit une politique rapportant en moyenne 2.10^{-3} pt/action de récompense). L'information dont dispose HATP seul lui suffit à trouver un plan efficace (pour environ 0.15 pt/action, soit 7 actions entre deux récompenses). La combinaison aléatoire fait moins bien que HATP seul, du fait du bruit rajouté par l'expert habituel dans les actions proposées par HATP (0.08 pt/action, 12 actions par point de récompense). Les moyennes de récompense par action données entre parenthèses sont calculées sur l'ensemble de l'expérience, l'influence de l'apprentissage de l'expert habituel étant négligeable devant les écarts entre chaque configuration. La performance de l'expert habituel est due à la dimensionnalité importante de l'espace d'états : les 18 faits génèrent 2^{18} états potentiels dans lesquels sont possibles 18 actions, et l'algorithme n'apprenant qu'en fonction de l'expérience juste passée, la tâche est très difficile pour lui.

En revanche, on voit ici un intérêt d'utiliser un expert habituel dans cette tâche d'interaction homme-robot dans le temps de calcul plus court pris par cet expert avant chaque décision, comparé au temps plus long nécessaire à HATP. La figure 8.5 (gauche) montre le nombre d'actions proposées au système de supervision durant les répétitions de l'expérience. On peut voir que l'expert habituel a le temps de prendre beaucoup plus de décisions qu'HATP dans la durée impartie par l'expérience. C'est une conséquence des processus différents de décision dans chaque expert : d'une part, la planification est plus longue que la simple comparaison d'un nombre fini de Q-valeurs pour décider; d'autre

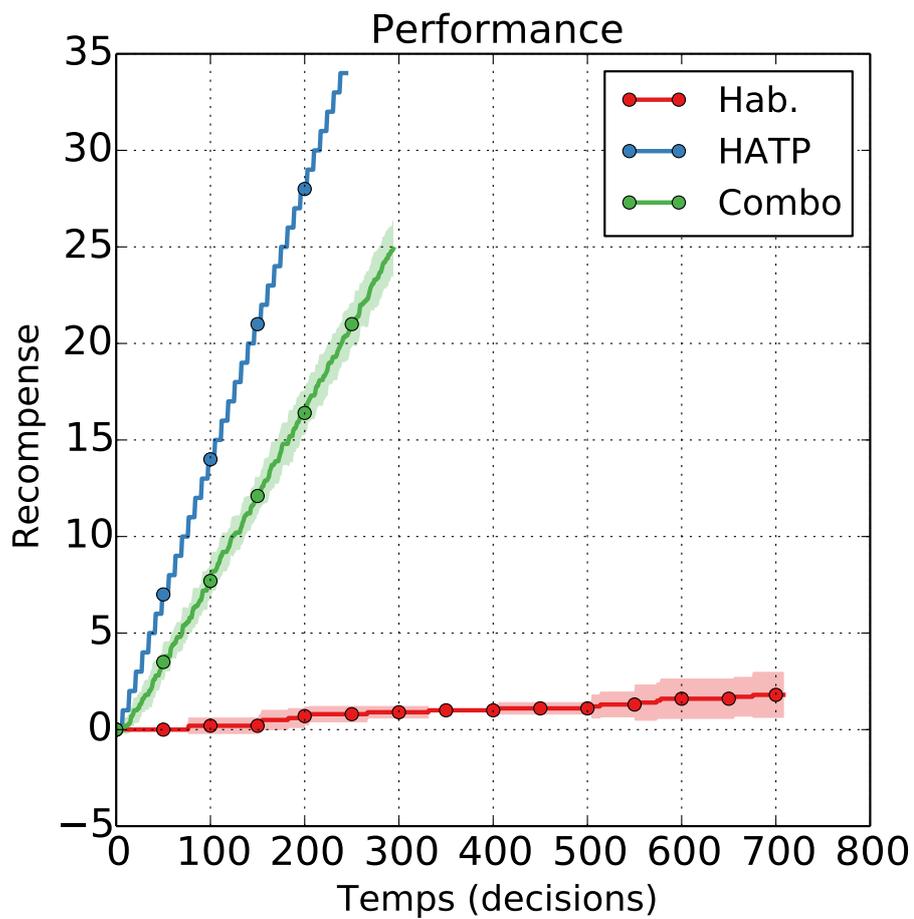


Figure 8.4 – Performance pour 10 simulations de la tâche. Le nombre de décisions prises est très variable, mais chaque simulation dure le même temps.

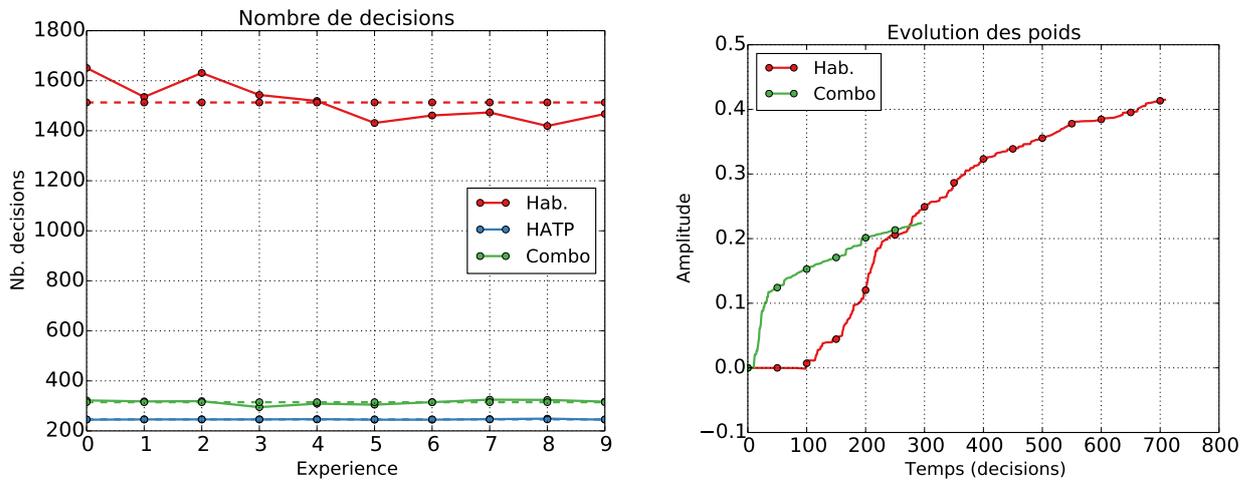


Figure 8.5 – Gauche : Nombre d’actions proposées dans chaque répétition de l’expérience, pour chaque configuration. La ligne pointillée représente la valeur moyenne de ce nombre. **Droite :** Mesure de l’évolution des poids dans le réseau de neurones qui implémente l’expert habituel, pour ce dernier seul ou combiné avec HATP. Cette mesure est la somme des valeurs absolues des poids. Les poids étant initialisés à zéro, plus cette valeur est grande, plus le réseau a appris l’influence de chaque fait qui compose l’état dans l’intérêt des actions.

part, il y a un autre facteur qui explique des différences aussi importantes dans le nombre de décisions prises par les deux experts : HATP propose des actions qui sont faisables parce que leurs préconditions sont valides. Ses actions sont donc systématiquement exécutées, l’information apportée par le domaine étant suffisamment précise par rapport à la réalité du problème. En revanche, l’expert habituel n’a aucun a priori et essaye ; beaucoup des actions qu’il propose ne sont pas exécutables et le système de supervision renvoie tout de suite le besoin de décider d’une nouvelle action. Pour la combinaison, le nombre d’actions reste proche de celui d’HATP, suggérant que le guidage de l’expert dirigé vers un but permet d’éviter trop d’actions inutiles, mais également que les propositions de l’expert habituel tendent vers des actions exécutables. En effet, si ce dernier n’apprenait pas, ce nombre d’action, comme la performance, serait plus intermédiaire aux valeurs pour les experts seuls. Comme la combinaison est bien équiprobable (figure 8.6), cette tendance de la combinaison à se rapprocher de la performance de l’expert dirigé vers un but est bien due à l’interaction des experts.

Enfin, nous comparons l’apprentissage de l’expert habituel lorsqu’il est seul ou combiné (figure 8.5, droite). Nous estimons le degré d’apprentissage comme la somme des valeurs absolues des poids : ces derniers étant initialisés à zéro, tout écart à cette valeur représente de l’information sur l’intérêt de l’une ou l’autre action. Pour la combinaison, l’apprentissage apparaît beaucoup plus tôt en moyenne dans l’expérience, grâce au guidage des connaissances a priori de HATP. Il y a un transfert entre l’information contenue initialement dans un expert vers l’autre. Cela signifie que tant que les conditions du problème changent à un degré acceptable pour les connaissances de HATP, l’apprentissage d’une *habitude* (selon la modélisation classique des neurosciences) peut se faire rapidement.

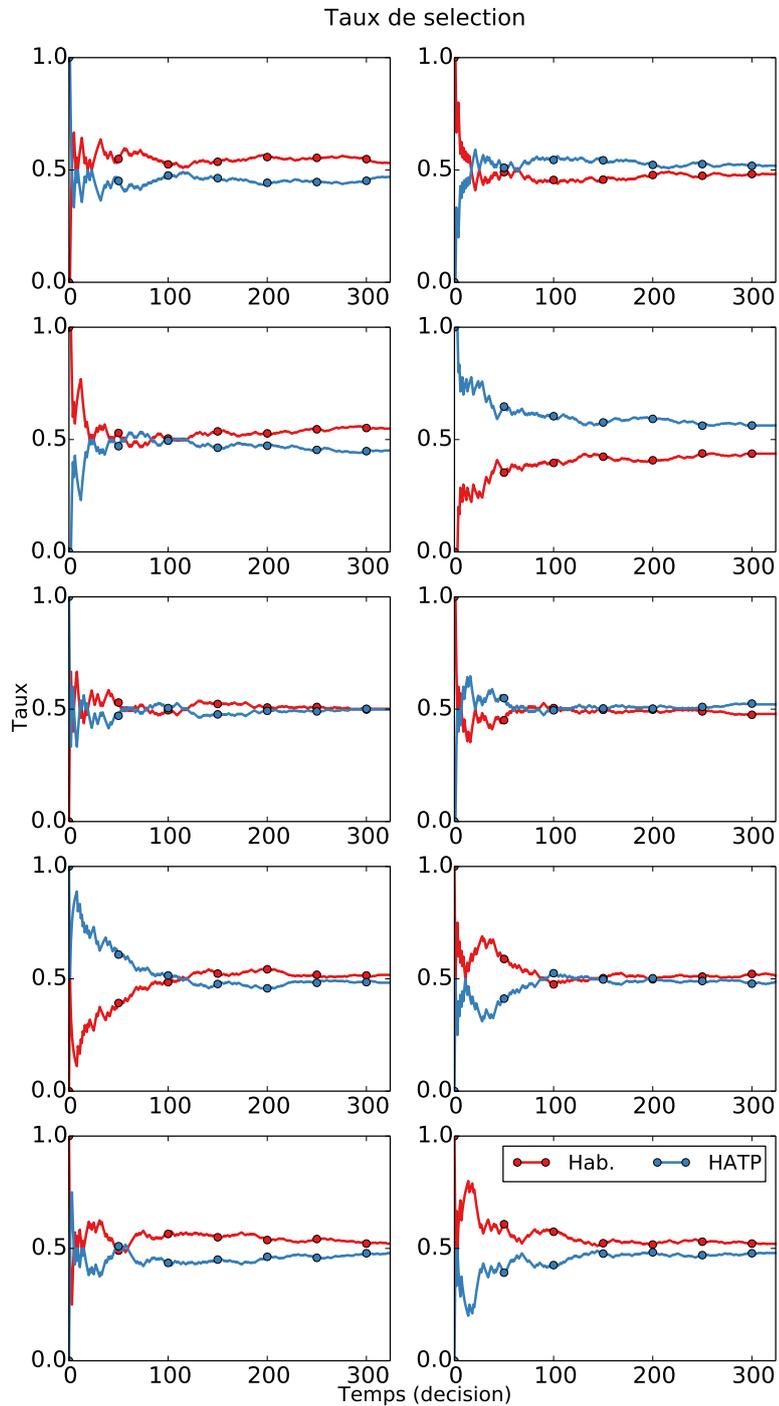


Figure 8.6 – Taux de sélection des experts dans les expériences de la combinaison. Ces courbes montrent que la sélection de chaque expert est bien quasi-équiprobable.

8.1.4 Discussion intermédiaire

Nous avons appliqué, dans cette première partie, notre principe de combinaison d'experts à un contexte d'interaction homme-robot. Nous avons combiné un planificateur qui tient compte des différents agents présent dans l'environnement, et qui dispose de connaissances initiales, et un algorithme d'AR direct et évalué les différentes configurations (comportements seuls, combinaison aléatoire) dans une tâche de « nettoyage de table » collaborative. Nous avons vu que si la combinaison n'égale pas la performance du planificateur spécialisé seul (puisqu'elle rajoute des décisions exploratoires pendant la phase où l'expert habituel apprend la bonne politique comportementale), elle obtient néanmoins de meilleures performances que l'expert qui modélise les habitudes seul. La vitesse d'apprentissage de ce dernier est accrue, montrant un transfert d'information entre les deux experts.

Bien que nous n'ayons utilisé que la méthode de sélection aléatoire, ces résultats montrent que le principe de combinaison d'experts reste intéressant dans une tâche plus réaliste que la tâche de poussée de blocs. Cette combinaison permettrait a minima de réduire le temps de calcul du robot en évitant la planification dans les situations où le meta-contrôleur reconnaît la tâche comme suffisamment stable et connue pour pouvoir y exécuter une habitude comportementale. Il est néanmoins important de noter que dans le cadre d'une interaction homme-robot, un comportement aléatoire exhibé par le robot peut produire une baisse de son acceptation dans un contexte de collaboration à une tâche (GOETZ et al. 2003). Ainsi, il semble que dans un contexte d'interaction homme-robot, la méthode de sélection proposée doit donner préférentiellement le contrôle au planificateur tant que l'expert habituel n'a pas suffisamment appris un comportement stable. Dans un algorithme d'AR direct, il existe une image de cette apprentissage, qui est l'évolution des estimations de valeurs. Le contexte dans lequel nous avons évalué notre architecture est stable ; l'humain agit comme attendu par HATP. Si ça n'est plus le cas, HATP peut se retrouver incapable de trouver une solution d'après son domaine. Détecter une telle « détresse » du système de planification permettrait de donner la main à l'expert habituel et d'explorer des actions précédemment considérées comme utiles, pour ramener le robot dans un état où le planificateur est efficace. Une mesure du temps mis par la planification, ou du nombre de plans évalués permettrait de détecter ce genre de problème. Dans la mesure où le but de la tâche est donné dans le domaine, une mesure analogue à la frustration (HASSON et al. 2012 ; JAUFFRET et al. 2013) pourrait également être utilisée. La limitation de connaître un but peut être simplement levée en ne calculant cette frustration qu'une fois qu'un but est connu. Des essais de méthode de sélection basée sur ces informations (variation de valeurs de l'expert habituel, blocage du planificateur) ont été conduits mais ne sont pas présentés dans ce manuscrit car trop peu développés.

Finalement, dans notre architecture, nos experts démarrent avec une différence dans leurs connaissances initiales. Il serait possible, au prix de quelques efforts, de transcrire la description de domaine en un équivalent compréhensible pour l'algorithme de renforcement. Cependant, si des connaissances bien conçues peuvent améliorer l'efficacité du robot dans une tâche donnée, elles seront toujours incomplètes face à la variabilité des tâches auxquelles peut être confronté un robot autonome. Utiliser ces connaissances pour guider le comportement initial permet d'améliorer l'apprentissage, mais le transfert de

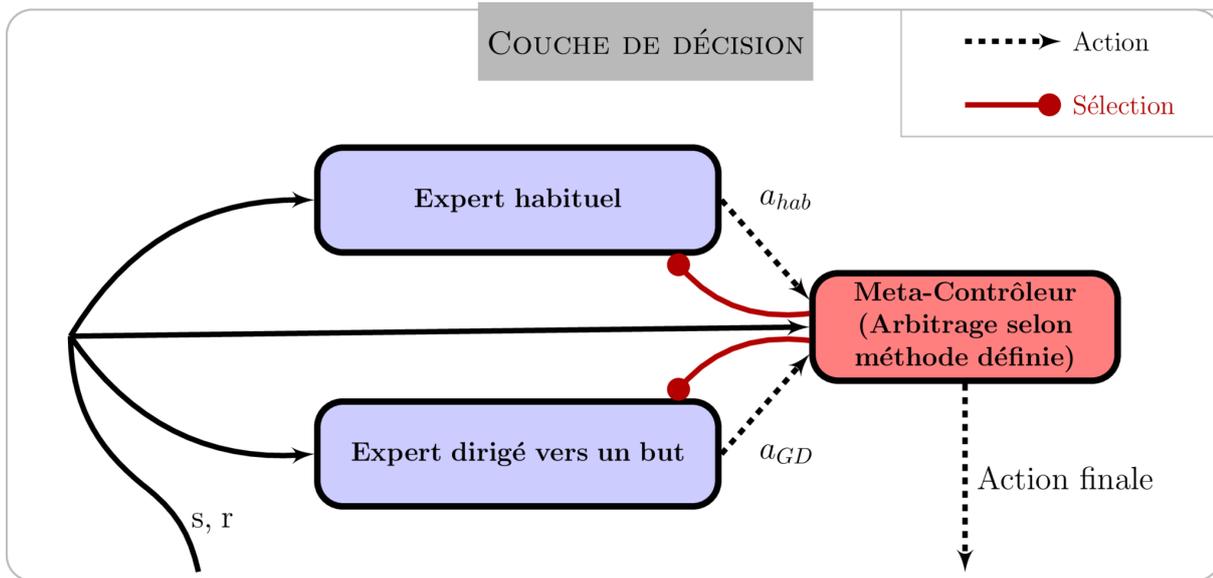


Figure 8.7 – La couche de décision de notre architecture. Les liens rouges permettent au meta-contrôleur de décider lequel des experts planifiera pour la prochaine décision.

connaissances permet de filtrer ces dernières pour n'apprendre que les comportements qui auront effectivement rapporté de la récompense.

8.2 Expérience de navigation

Nous étendons également notre architecture à une tâche de navigation robotique réelle.

Une limitation de notre architecture jusqu'ici réside dans le fait que chaque expert planifie en parallèle, et aucun gain ne résulte réellement d'un point de vue coût de calcul à sélectionner l'expert habituel. Or, un des intérêts à adjoindre une capacité d'habituation au robot est de pouvoir éviter le processus de planification si possible. Nous nous proposons d'étudier une nouvelle variation de l'architecture qui adresse cette question dans le cadre de la navigation d'un robot.

Organisation de la couche de décision

La relation entre les experts et le meta-contrôleur est réorganisée pour que ce dernier puisse inhiber la planification de l'un ou l'autre expert à la prochaine étape de décision, en fonction des informations dont il dispose (voir figure 8.7). Initialement, comme décrit au chapitre 6, chaque expert enchaîne, à la réception d'un nouvel état, les étapes d'apprentissage, de planification et de décision. L'inhibition permet de sauter l'étape de planification, l'expert met donc à jour ses connaissances puis décide de la prochaine action.

Méthode d'arbitrage

Nous choisissons en premier lieu une méthode d'arbitrage par sélection ; elle choisit quel expert planifie et lui donne le contrôle pour la prochaine décision. Il est en effet inutile de différencier l'expert qui planifie et celui qui contrôle : celui qui a sauté le processus de planification a une estimation uniforme de la valeur des actions dans l'état actuel. Cette affirmation ne se vérifie pas dans le cas d'un expert dirigé vers un but qui conserve ses estimations, nous détaillons ce point dans la discussion.

Par ailleurs, nous avons vu dans le chapitre précédent que les méthodes d'arbitrage basée sur un seul signal tendent à coller au comportement d'un expert en particulier. Nous proposons donc d'utiliser une mesure plus riche pour arbitrer de manière plus cohérente avec les propriétés des algorithmes d'AR direct et indirect implémentés dans nos experts.

Deux aspects nous semblent importants pour donner le contrôle à l'un ou l'autre des experts : son degré d'apprentissage et son coût intrinsèque à évaluer les actions. Le premier indique à quel point les connaissances du robot sont à jour, et donc si sa décision est informée ou résulte de la chance : c'est une information comparable à l'entropie de la distribution d'action du chapitre précédent (à quel point peut-on faire confiance à l'expert ?), mais directement image du processus d'apprentissage, là où l'entropie donne une information liée au processus de planification. Le coût intrinsèque est lié à ce dernier et dépend de la nature de l'expert. Nous choisissons comme mesure du coût de planification les temps moyens mesurés dans chaque expert pour accomplir ce processus (T_{Hab}, T_{GD}). Pour l'expert habituel, la mesure d'apprentissage correspond à la variation moyenne des estimations de valeur d'action δQ (ou directement des poids). Pour l'expert dirigé vers un but, l'apprentissage consiste à mettre à jour le modèle de transitions manipulé par l'expert. La mesure de son apprentissage est donc la variation moyenne des probabilités de transitions δP . Ces mesures (apprentissage et coût) sont moyennées au cours du temps pour réduire la variation due au passage d'un état à un autre (filtres passe bas de paramètre $\alpha = 0.2$ (5 décisions) pour les mesures de l'apprentissage et $\alpha = 0.02$ (50 décisions) pour les temps de planification).

À apprentissage égal, et ayant convergé après suffisamment de temps dans un contexte stable, nous souhaitons nous reposer sur l'expert habituel, dont le coût de décision est faible devant celui de l'expert dirigé vers un but. En cas de changement de la position du but (donc de la récompense), l'expert dirigé vers un but doit être sollicité préférentiellement : il peut retrouver un nouveau chemin vers d'éventuelles autres récompenses connues, et contrairement à l'expert habituel, recommence tout de suite à explorer en mettant à jour ses valeurs d'action.

La méthode d'arbitrage que nous proposons calcule donc de manière statique deux valeurs de sélection V_E pour chaque expert E , basées sur les mesures évoquées précédemment (équation (8.1)).

$$\begin{aligned} V_{Hab} &= -(\alpha_{Hab} \cdot \delta Q + \beta_{Hab} \cdot T_{Hab}) \\ V_{GD} &= -(\alpha_{GD} \cdot \delta P + \beta_{GD} \cdot T_{GD}) \end{aligned} \tag{8.1}$$

Les coefficients α et β nous permettent de régler l'importance de chaque terme dans



Figure 8.8 – Arène de l'expérience dans laquelle évolue le robot. Les points verts au sol indiquent approximativement les positions initiales. **Bas, gauche** : le robot rentre dans l'état 30 (zone but rouge) et reçoit une récompense.

la valeur. Comme les modifications du modèle et celles des poids ne sont pas du même ordre, $\alpha_{GD} = 1$ et $\alpha_{Hab} = 12$, pour ne transférer le contrôle vers l'expert habituel que sa convergence est bien établie. Les valeurs β sont fixées à 5 pour les deux experts, afin que seul la différence naturelle de valeurs entre les temps de planification fasse la différence, à apprentissage égal. Ces valeurs sont transformées en probabilités par la fonction *softmax*, et la sélection de l'expert est effectuée à chaque décision d'action. Le temps de planification d'un expert étant régulièrement nul (s'il n'est pas choisi par le meta-contrôleur), T_{Hab} et T_{GD} ne sont mis à jour que pour l'expert qui a effectivement planifié à cette étape de décision.

Protocole expérimental

Un robot Turtlebot est placé dans une arène d'environ $7,5 \text{ m} \times 3,5 \text{ m}$, dans laquelle sont disposés des obstacles (figure 8.8). Le robot est une base mobile contrôlée par un ordinateur embarqué. L'ordinateur utilise ROS pour traiter les signaux de ses capteurs, commander la base mobile et s'interfacer avec notre architecture. Il dispose d'une Kinect qui renvoie une estimation de distance aux obstacles dans son champ de vision, et de capteurs de contact à l'avant et sur les côtés de la base mobile.

Le robot doit apprendre à rejoindre une zone précise de l'environnement (située dans le couloir central). Lorsqu'il y parvient, il reçoit une récompense unitaire et est ramené aléatoirement à une des quatre positions initiales, situées dans les quatre coins de l'arène, pour recommencer.

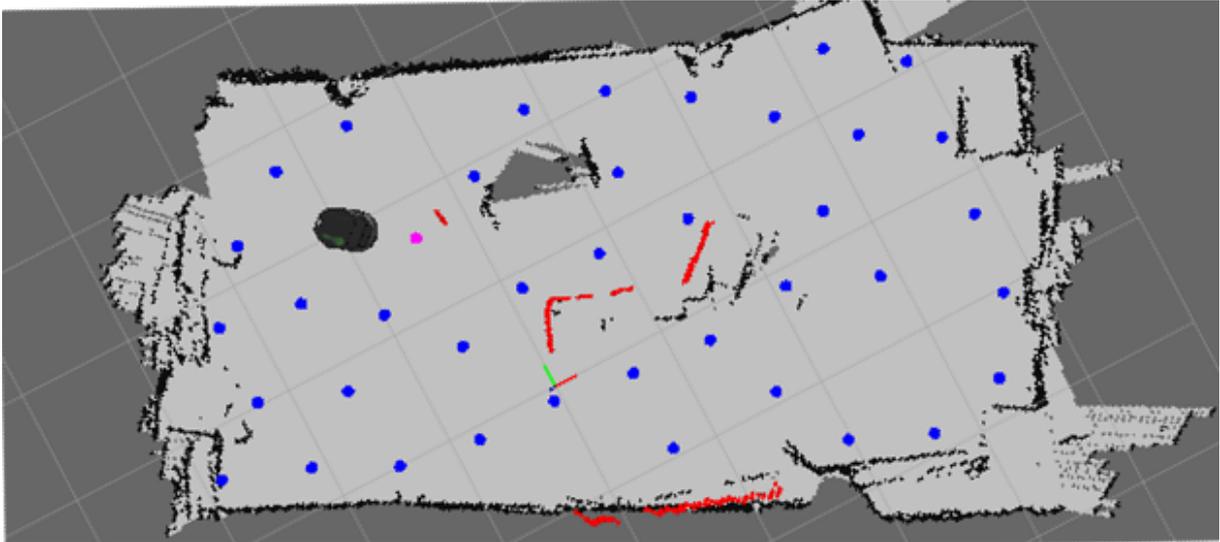


Figure 8.9 – Visualisation des données manipulées par le robot. Les points bleus sont les centres des états, le point rose est l'état actif. La zone gris clair représente l'espace libre dans la carte du robot, les lignes noires bruitées la position des obstacles connus dans la carte. Les lignes rouges au centre et en bas de l'image sont les obstacles que le robot perçoit à l'aide de son capteur Kinect. Les segments orthogonaux indique l'origine et le repère de la carte (rouge : x ; vert : y). Le modèle 3D du robot est placé à son estimation de position dans la carte.

Perception du robot Le robot se localise dans l'environnement selon un algorithme de Localisation et de Cartographie Simultanée (*SLAM*, GRISSETTI et al. (2007)) par filtrage particulaire. Cet algorithme maintient un nombre N de particules, chacun étant une estimation de la position du robot, associée à une probabilité que le robot y soit effectivement. Ces estimations sont mises à jour en fonction de l'odométrie du robot et d'une observation réalisée par un capteur extéroceptif. Les observations sont issues de la mesure de profondeur de la Kinect limitée à une hauteur fixe. À partir de ces informations, le robot construit une carte d'occupation qui positionne les obstacles dans un référentiel allocentrique (figure 8.9). Nous discrétisons cette carte métrique en un ensemble de centres ajoutés incrémentalement pendant une phase d'exploration de l'environnement. Ces centres définissent les états au sens de l'apprentissage par renforcement (voir figure 8.10). L'état courant est le centre dont le robot est le plus proche lorsque son action précédente est terminée et qu'il en évalue les conséquences.

Ce processus de création de carte et d'états est théoriquement faisable simultanément, et pendant une navigation motivée par la récompense du robot. Cependant, nous séparons ces différentes phases afin de ne créer des états que lorsque la carte issue du SLAM est assez stable. Les états présentés sur la figure 8.10 sont ceux utilisés pour le reste des expériences. La récompense est obtenue quand le robot est dans l'état 30.

Actions Le robot peut se déplacer dans 8 directions allocentriques équitablement réparties, représentées sur la figure 8.10. Lorsqu'une direction est sélectionnée, le robot s'oriente puis avance d'une distance fixe (0.5m dans nos expériences). Une fois cette distance ac-

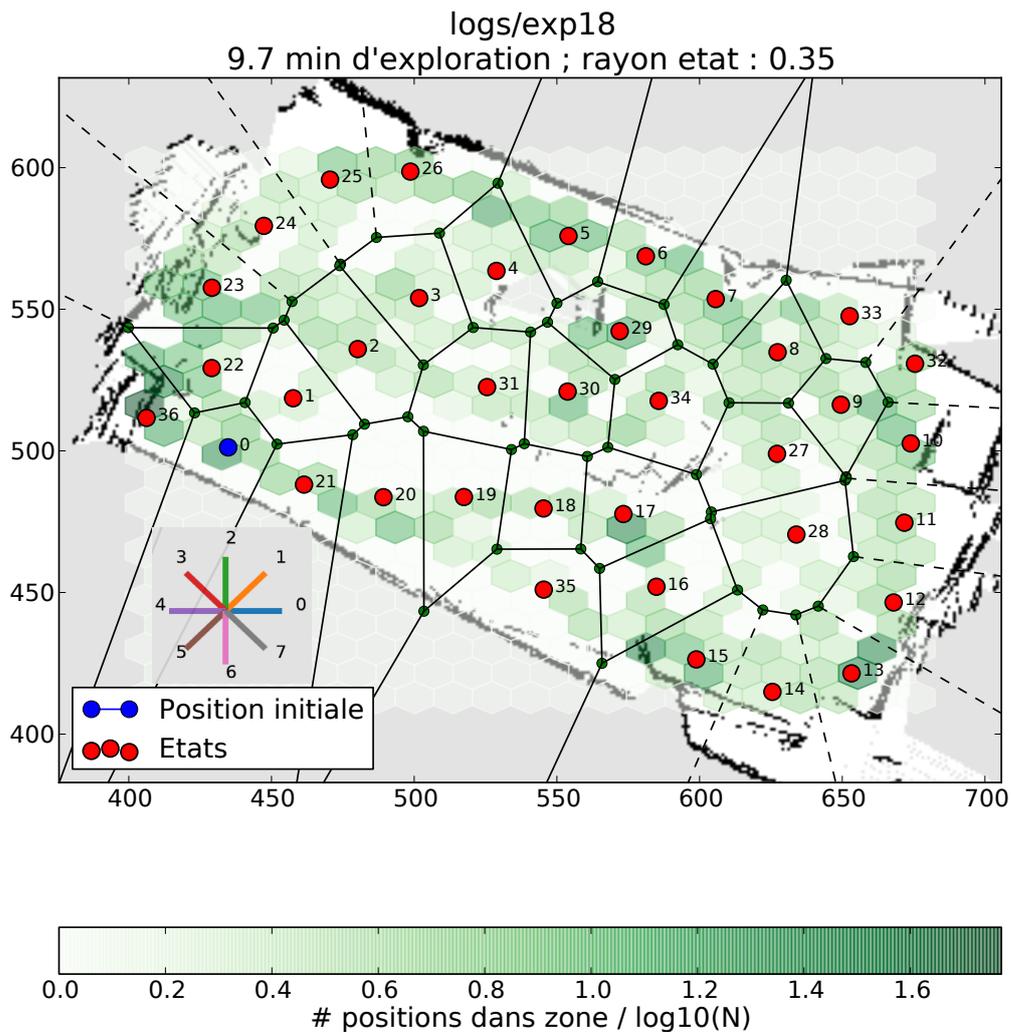


Figure 8.10 – Carte construite après environ 10 minutes d'exploration, et positions des états incrémentalement créés après 10 autres minutes (le temps indiqué est le second temps). Le pavage coloré vert indique le temps passé dans chaque endroit. rayon_etat est la distance d'influence d'un état, un nouvel état étant créé lorsque le robot s'est éloigné de deux fois cette valeur d'un état connu. L'étoile à 8 branches indique les directions correspondant à chaque action du robot.

complie, l'action est considérée comme finie et une nouvelle décision peut être prise dans l'état courant. Une machine à trois états permet de gérer l'évitement d'obstacle : tant que l'obstacle est assez loin, il n'a pas d'influence sur la navigation qui se fait en ligne droite ; si l'obstacle se rapproche à distance intermédiaire, le robot s'oriente progressivement vers le côté de plus grande ouverture perçue (à une vitesse angulaire de 0.2 rad/set diminue sa vitesse ; si l'obstacle est trop proche, le robot s'arrête et tourne du côté de plus grande ouverture. Enfin, deux cas spéciaux viennent compléter ces comportements : si la Kinect ne reçoit aucune donnée (parce qu'elle est trop proche d'un obstacle) ou que les pare-chocs détectent un contact, le robot recule d'une distance fixe et met fin à l'action courante.

Particularités des experts

Nous appliquons également quelques modifications aux experts :

– Expert habituel :

1. L'implémentation sous forme de réseau de neurones que nous avons utilisée jusqu'à présent inclut un biais dans la couche d'entrée, pour mieux approximer la fonction des valeurs d'action. Dans cette tâche épisodique, la couche d'entrée code en position l'état reconnu. Dans ce cas, et sans autre information, le biais vient favoriser l'action précédemment récompensée. Bien que ce soit un comportement normal du réseau, il nuit cependant à l'exploration puisqu'il empêche le robot de tester d'autres actions intéressantes, et ce, alors qu'il n'y a pas lieu de généraliser dans tous les états l'intérêt de l'action précédemment récompensée. Nous désactivons donc le biais dans cette expérience.
2. Sans biais, le robot est libre d'explorer de manière purement aléatoire chaque action. Or, sans récompense reçue pour guider le comportement vers des actions intéressantes dans la tâche, ou sans état voisin contenant de l'information, l'expert n'apprend pas et perd du temps à faire une action et son contraire. Pour pousser l'expert à choisir préférentiellement des actions qu'il n'a pas déjà essayé dans un état s de l'environnement, nous initialisons ses valeurs d'actions à 0.5, soit la moitié de la valeur maximale de récompense possible, approche inspirée par l'algorithme RMAX (BRAFMAN et TENNENHOLTZ 2002). Ainsi, les attentes pour chaque action sont initialement hautes, et celles qui sont déçues dévaluent l'action, poussant l'expert à choisir préférentiellement les autres et à explorer de manière moins aléatoire (mais biaisée par une information pertinente).

– Expert dirigé vers un but

1. La mise à jour des valeurs se fait selon l'équation classique de VI plutôt que la forme présentée au chapitre 6.
2. Sa fonction de récompense est initialisée à 0.01 pt pour favoriser également l'exploration.
3. L'équation de mise à jour des transitions initialement utilisée souffre d'un problème de dilution des probabilités de transition au fur et à mesure que la force tend vers 1 : lorsque d'une transition stochastique a été peu visitée, la différence entre les valeurs permet d'avoir des probabilités bien distinctes. Lorsque le

Table 8.1 – Valeurs choisies des paramètres des experts dans l’expérience de navigation.

Param	Hab	GD
α	0.6	n.a.
γ	0.9	0.95
τ	0.025	0.1

nombre de visites de cette transition augmente dans les différents états d’arrivée, les probabilités de transition tendent vers une loi uniforme, écrasant les différences statistiques. Nous adoptons ici une méthode directement proportionnelle au nombre de visites pendant une fenêtre de temps (de taille 6 décisions) : pour évaluer la probabilité d’une transition (s, a, s') , on considère les n derniers passages par le couple s, a , et la probabilité de transition est directement la proportion de passages vers s' .

Les valeurs de paramètres choisies sont listés dans la table 8.1.

Résultats préliminaires

Nous réalisons deux expériences de contrôle avec chaque expert seul en contrôle du robot. Dans tous les cas, le robot démarre avec la carte métrique et la connaissance des états :

- l’expert habituel est lancé environ 120 minutes et dispose de ce temps pour trouver puis apprendre à atteindre la position but.
- l’expert dirigé vers un but est lancé une fois environ 60 minutes sans récompense dans l’environnement, pour un apprentissage latent de son modèle de transitions. Ce modèle est ensuite réutilisé comme connaissances initiales pour relancer l’expert dirigé vers un but à nouveau 60 minutes en présence de la récompense et doit atteindre le but.

Un aperçu du modèle sous la forme de ses matrices de transition est donnée en fin de chapitre (figure 8.17).

En deux heures, l’expert habituel arrive à atteindre entre 3 et 7 fois le but (figure 8.11). C’est relativement peu pour que l’expert apprenne correctement, bien qu’il tombe sur la récompense en moins d’une heure à chaque fois. En conséquence, l’apprentissage de l’expert est assez faible : seules les valeurs d’action dans les états directement voisins de l’état 30 (29, 31, 34) sont les seules à dépasser la valeur initiale. Dans les autres états, les valeurs d’action n’ont été dévaluées que du fait de l’exploration (figure 8.12). L’expert dirigé vers un but a une performance plus variable mais également bien meilleure : une fois que la récompense a été trouvée, le modèle permet au robot d’effectuer les bonnes actions pour retourner en 30, moyennant quelques errances due à l’aspect stochastique des actions (par exemple pour l’expérience 11, après 20 min, où un plateau est expérimenté).

Malgré cette faible performance, l’expert confirme son faible coût : son processus de planification est d’un ordre de grandeur constant de 10^{-3} secondes, et ne dépend pas de l’obtention ou non de la récompense (figure 8.13). Au contraire, pour l’expert dirigé vers un but, on observe des pics de l’ordre d’une seconde lorsqu’une récompense est reçue. Par

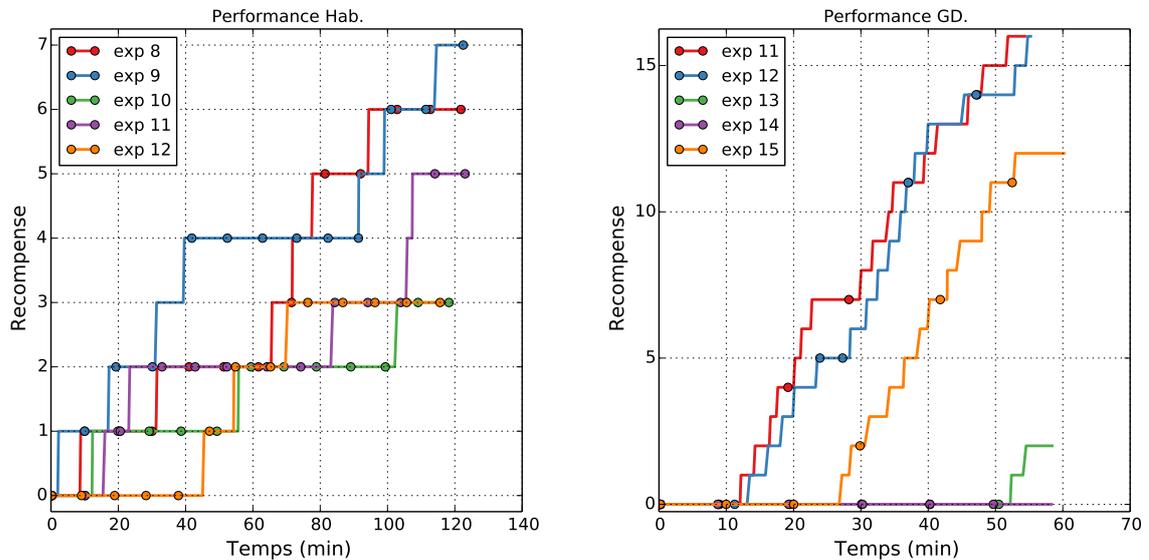


Figure 8.11 – Performances en récompense cumulée des experts seuls, pour 5 répétitions de l’expérience. *Gauche* : expert habituel ; *Droite* : Expert dirigé vers un but

exemple pour l’expérience 11 où entre 10 et 20 minutes, le temps de planification présente une série de pics entre 0.1 et 1 seconde, qui correspondent aux récompenses reçues avant le plateau évoqué plus haut. En dehors de ces épisodiques, le temps mis par le processus est de l’ordre de 10^{-2} secondes, soit 10 fois plus que l’expert habituel (voir l’expérience 14 où aucune récompense n’est reçue).

Pour la combinaison d’experts, le temps de découverte de la première récompense est assez variable. Par contre, son obtention, comme pour l’expert dirigé vers un but, permet d’atteindre rapidement de bonnes performances (figure 8.14). La combinaison semble augmenter le temps nécessaire à l’exploration, ce qui demande à être confirmé par des expériences supplémentaires, mais s’explique par l’interaction des deux processus de sélection de l’action : si l’exploration d’un seul expert est bien guidée par les mises à jour qu’il fait dans ses valeurs ou sa fonction de récompense, les différences dans le paramétrage des experts peuvent conduire à des décisions moins performantes (l’expert habituel voulant explorer localement les actions qu’il n’a pas essayé et l’expert dirigé vers un but voulant aller vers des zones distantes pour lesquelles il pense encore qu’il y a de la récompense).

La figure 8.15 montre (à droite) les probabilités de sélection d’un expert calculée par notre méthode, et (à gauche) le taux de sélection durant l’expérience. Ces résultats montrent que notre critère donne favorablement la main à l’expert dirigé vers un but. La valeur que nous avons choisi pour pondérer la variation des poids ne permet pas d’observer de transfert net, indice également du fait que l’expert habituel est loin d’avoir complètement appris une bonne politique. Cependant, l’apprentissage de l’expert habituel au bout de deux heures est plus avancé que lorsqu’il est seul (figure 8.16) : les valeurs dans les états voisins du but (29, 31, 24) sont plus importantes que celles de l’expert seul, et une partie de la valeur commence à se propager vers les états suivants. Comme nous l’avons vu

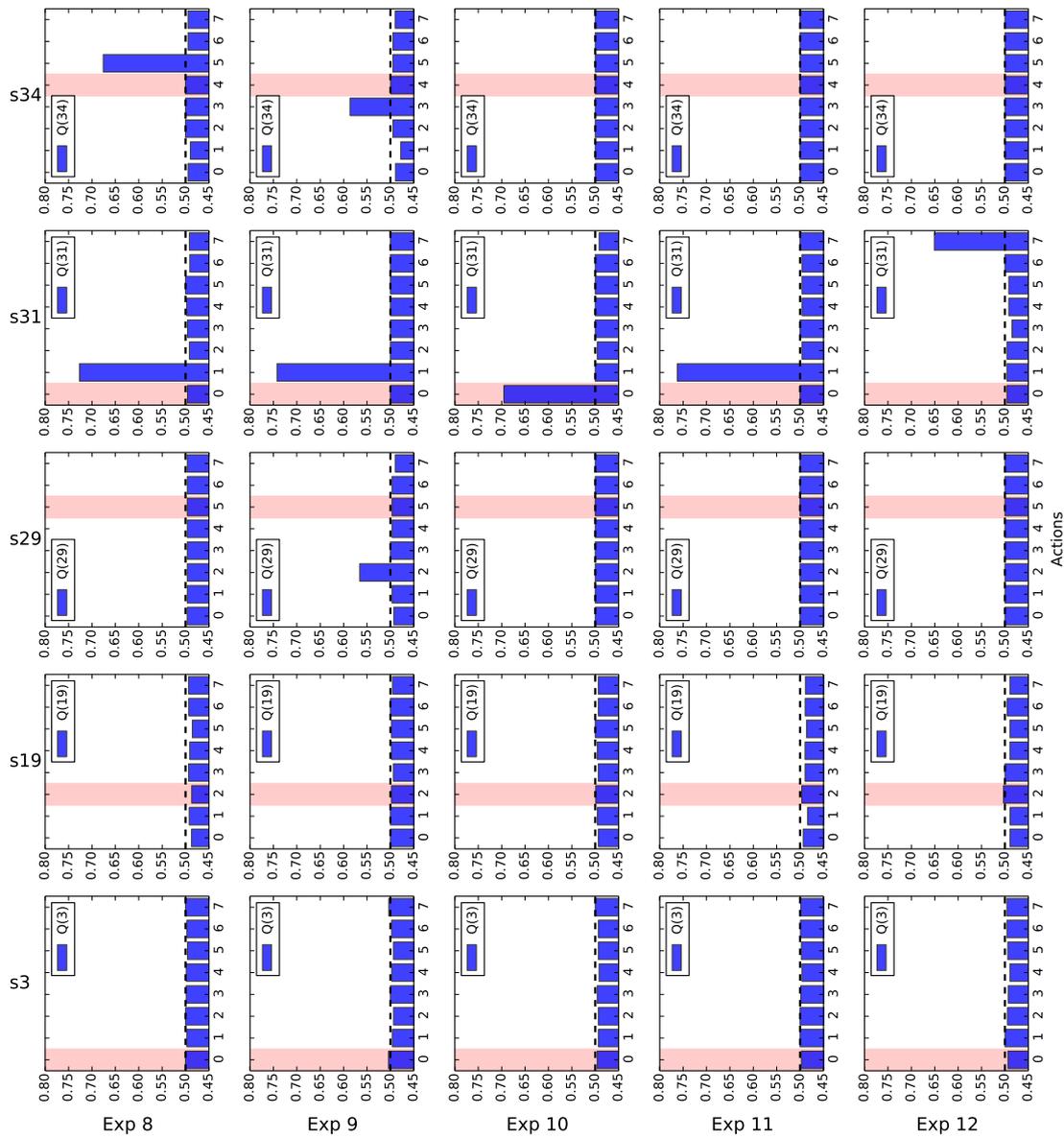


Figure 8.12 – États des valeurs d'action dans les états voisins de l'état but (voir figure 8.10) pour chaque expérience de l'expert habituel. Les barres rouges indiquent la meilleure action pour progresser vers l'état récompensé, d'un point de vue humain, les actions voisines étant également efficaces.

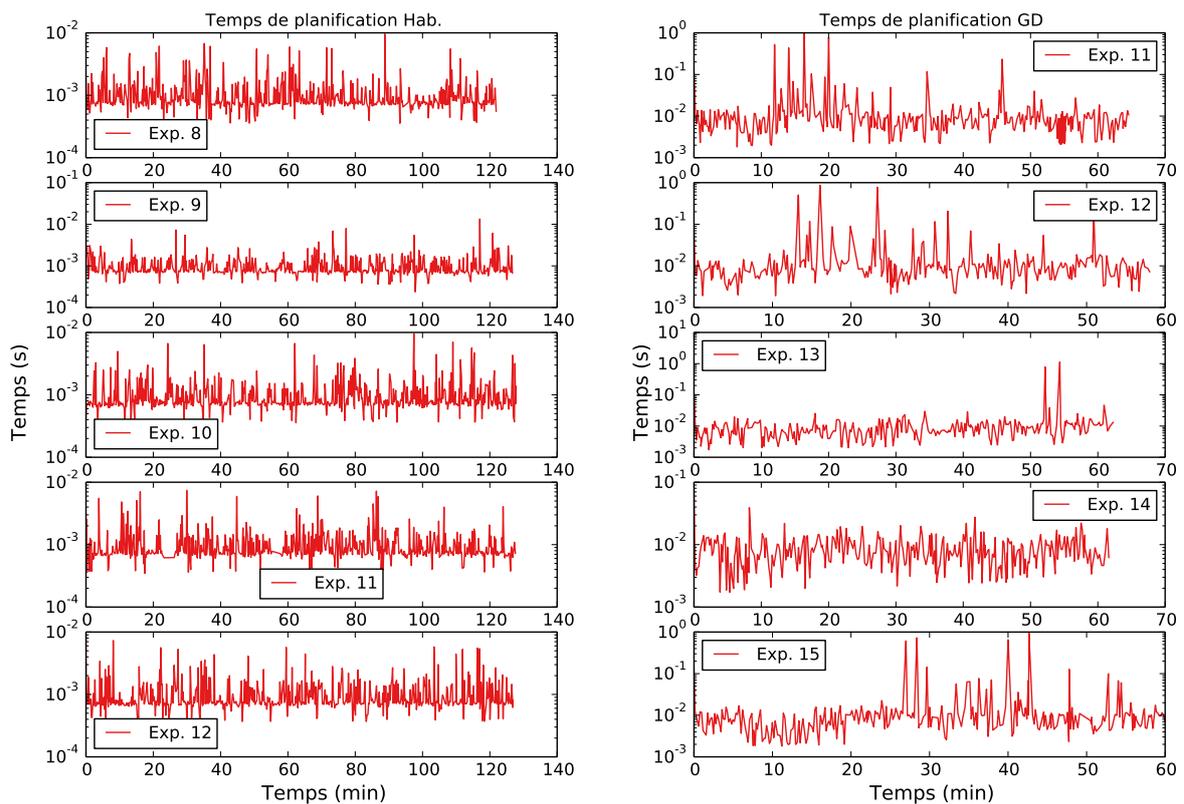


Figure 8.13 – *Temps de planification des experts seuls dans chaque expérience. Gauche : expert habituel ; l'ordre de temps reste constant, d'environ 10^{-3} secondes. Droite : Expert dirigé vers un but ; des pics de l'ordre d'une seconde dans le temps de planification sont expérimentés lorsqu'une récompense est reçue, et provoque de nombreuses mises à jour dans les valeurs. Par ailleurs, le temps mis par le processus est de l'ordre de 10^{-2} secondes, soit 10 fois plus que l'expert habituel.*

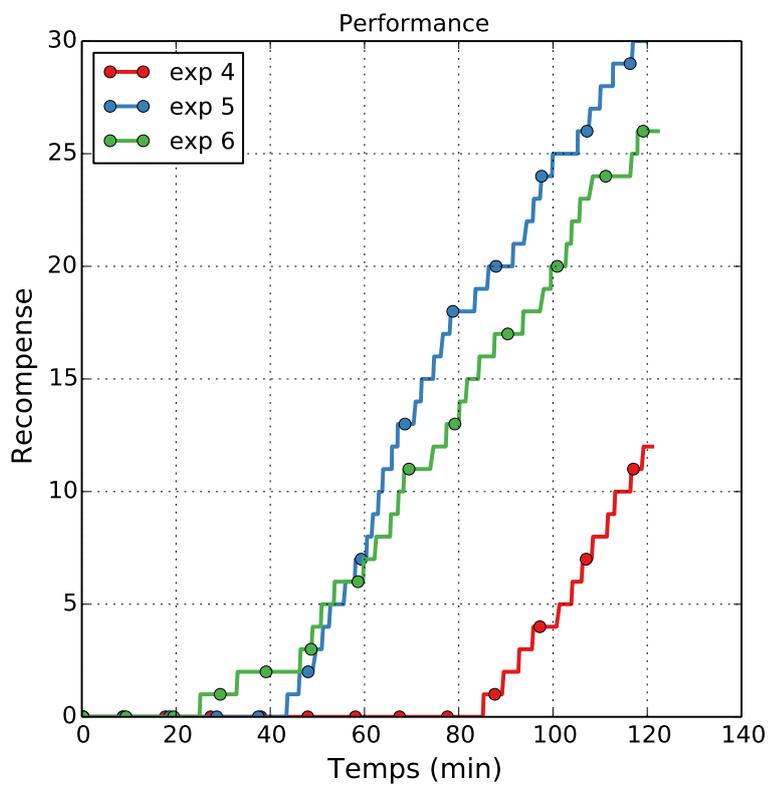


Figure 8.14 – Performance de la combinaison d’experts pour 3 répétitions de l’expérience.

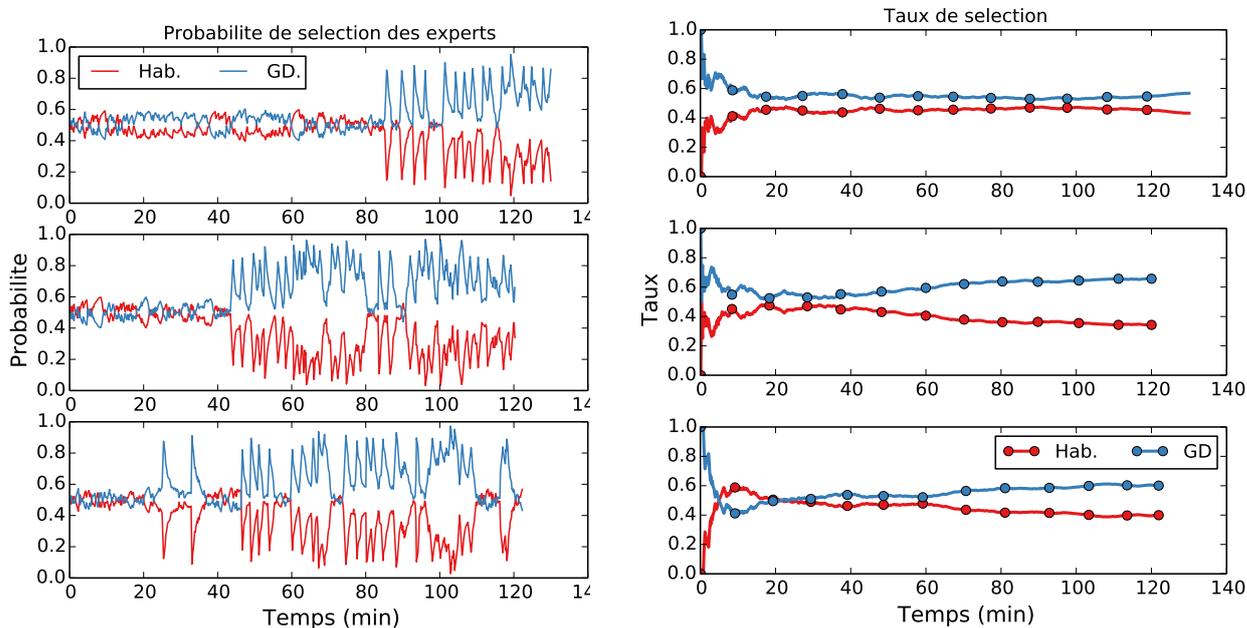


Figure 8.15 – *Gauche* : Probabilités de sélection des experts calculée selon la méthode d’arbitrage. *Droite* : Sélection effective des experts durant l’expérience. De haut en bas : expériences 4, 5 et 6

pour la tâche d’interaction, la combinaison, même avec un expert dirigé vers un but qui n’a pas une information aussi déterministe que celle de HATP, améliore l’apprentissage de l’expert habituel.

8.2.1 Discussion

Dans cette seconde partie, nous avons proposé une méthode d’arbitrage qui vise à répondre aux problématiques soulevées dans les chapitres précédents. Elle favorise l’expert habituel - qui est moins coûteux dans son processus de planification - lorsque les deux experts ont appris et l’expert ayant le meilleur degré d’apprentissage sinon. Nous avons évalué notre architecture dans une tâche de navigation robotique, et nos résultats préliminaires montrent que la combinaison améliore la performance du robot comparé aux experts seuls, et confirme l’effet sur l’apprentissage déjà évoqué dans la première partie de ce chapitre.

Nous avons rapidement mentionné ce point dans la description de l’architecture, mais nous y revenons ici : introduire un mécanisme qui permet de sauter le processus de planification questionne la manière dont on choisit l’action finale (sélection de la proposition d’un expert ou fusion de leurs informations respectives). La question qui se pose est de savoir si l’expert qui ne planifie pas doit tout de même décider dans l’estimation de valeurs dont il dispose. Plusieurs cas se présentent :

- l’expert habituel a une estimation non uniforme des valeurs d’action uniquement s’il planifie : ainsi, dans le cas où il ne le fait pas, il peut prendre au mieux une décision

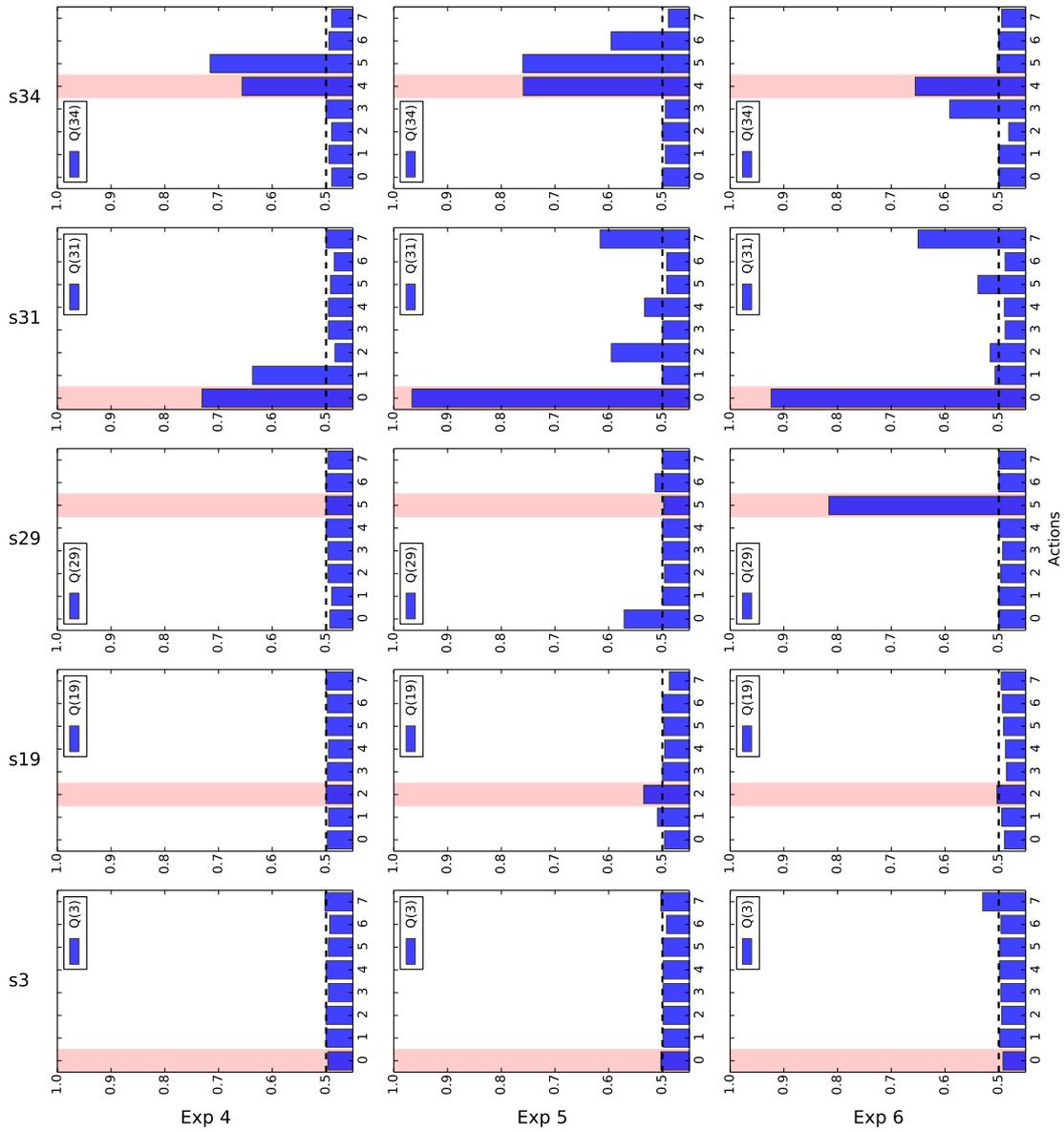


Figure 8.16 – États des valeurs d'action de l'expert habituel dans les états voisins de l'état but pour l'architecture complète. Les barres rouges indiquent la meilleure action pour progresser vers l'état récompensé, d'un point de vue humain, les actions voisines étant également efficaces.

aléatoire.

- L'expert dirigé vers un but a une estimation non uniforme des valeurs d'action s'il planifie, mais également s'il ne planifie pas mais qu'il conserve les estimations précédentes, comme présenté au chapitre précédent.

Ces différents cas influent sur la méthode d'arbitrage ; dans le cas d'une sélection, que nous avons testé, le robot a toujours l'estimation de l'un ou l'autre expert pour décider. Il exploite donc l'information accumulée jusque là sur la valeur des actions ; dans le cas d'une fusion (par somme pondérée par exemple), le résultat est similaire si l'expert dirigé vers un but ne conserve pas le plan : la décision est prise (à l'aide d'une fonction *softmax*) selon l'information d'un seul des experts. Mais s'il conserve le plan, l'estimation des valeurs fournie par l'expert habituel est biaisée par celle de l'expert dirigé vers un but. Tant que le robot n'a pas découvert le lieu de la récompense, le comportement est exploratoire, mais dès qu'elle est trouvée, il est possible au prix d'une seule planification complète puis de mises à jour au fur et à mesure que le modèle s'affine, de bénéficier des connaissances de l'expert dirigé vers un but tout en contrôlant le robot au coût de l'expert habituel. Nous étudierons cette idée dans des travaux futurs pour en saisir les limites, définir précisément la méthode de fusion et vérifier l'intérêt pour le comportement d'un robot.

Les taux de sélection de notre méthode d'arbitrage soulèvent également un questionnement sur la problématique de l'exploration : le transfert entre expert ne se produit-il pas dans le temps imparti à cause du paramétrage, ou parce que la phase d'exploration est longue, et ne permet pas au robot d'obtenir assez de récompenses pour apprendre suffisamment ? Si les initialisations non-nulles des poids et de la fonction de récompense de chaque expert est une manière de pousser le robot à explorer, nous avons vu qu'il pouvait s'agir d'une stratégie à part entière (DOLLÉ et al. 2010 ; CALUWAERTS et al. 2012), qui perd de son intérêt au fur et à mesure que le robot obtient de la récompense. La notion de curiosité artificielle (OUDEYER et al. 2007), ou de nouveauté (MAESTRE et al. 2015) peuvent également être des mécanismes explicites chargés de guider le robot vers l'acquisition de connaissances. S'il est probable que ce genre de mécanismes s'intègre à un niveau délibératif, sa prise en compte dans le processus de planification en lui-même ou comme un autre système interagissant avec lui demande à être étudiée. De notre point de vue lié au domaine de l'apprentissage par renforcement, on retrouve classiquement l'exploration comme le compromis à l'exploitation des connaissances sur la valeur des actions. Cependant, il est tout à fait possible de consacrer un expert à l'exploration, ce qui permet sa sélection active par le meta-contrôleur (DOLLÉ et al. 2010 ; CALUWAERTS et al. 2012). Explorer devient donc un choix à part entière et les experts peuvent se consacrer à exploiter au mieux l'information sur les valeurs d'action en leur possession.

Une dernière question sur laquelle nous reviendrons dans le chapitre suivant est celle de la nature de la couche fonctionnelle dans une architecture robotique, et de l'origine des actions. Nous avons ici proposé un contrôleur basé sur des actions discrètes, et simplifié l'environnement en « cases » grossières, néanmoins créées automatiquement par l'algorithme. Or, ces choix de conception influent directement sur la capacité de la couche de décision à apprendre une politique reproductible dans le monde. S'il est possible de chercher à concevoir des systèmes les plus prédictibles dans leur exécution d'action, doter les robots de la capacité de les former eux-même semble une étape nécessaire vers plus

d'autonomie.

8.3 Conclusion

Dans ce chapitre, nous avons mis en œuvre notre architecture dans deux tâches différentes réalistes d'un point de vue robotique. Nous avons pu illustrer les bénéfices de la combinaison de systèmes dans ces deux tâches, pas tant au niveau de la performance brute (même si les résultats de navigation le suggèrent, ils demandent plus de données et d'analyse pour l'affirmer) qu'au niveau de ses effets pour l'apprentissage. Nous discutons plus généralement de ces résultats mis en perspective par rapport à ceux des chapitres 6 et 7 dans le chapitre suivant.

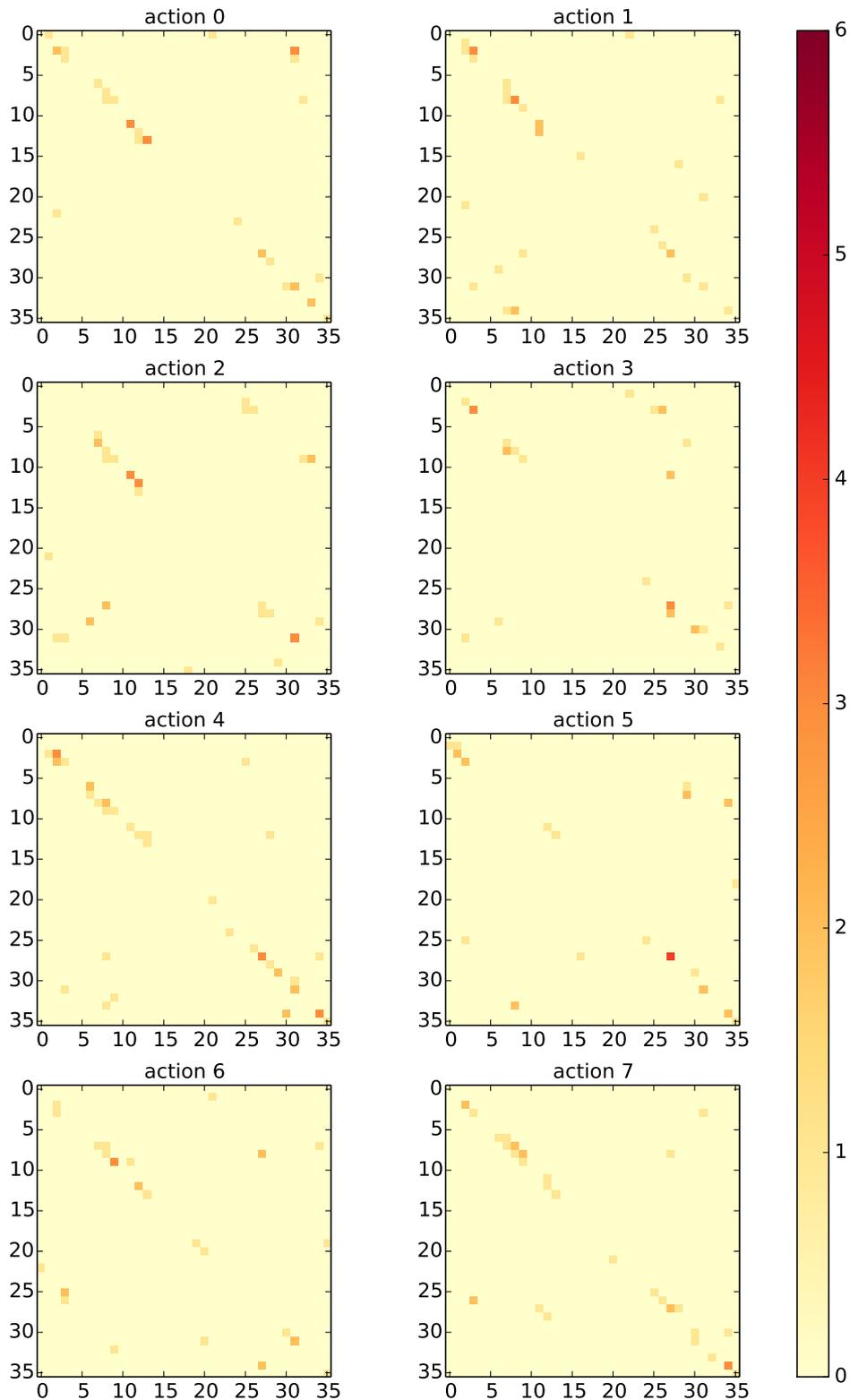


Figure 8.17 – Matrices de transition du modèle construit au bout d'environ une heure d'exploration par l'expert dirigé vers un but. En ordonnée, l'état s de départ, en abscisse, l'état s' d'arrivée. La couleur représente la force de transition : le nombre de fois où la transition a été réalisée, dans la limite des 6 dernières expériences de s, a .

Chapitre 9

Discussion et perspectives

9.1 Résumé des contributions

Dans cette thèse, nous nous sommes intéressés à la manière dont nous pouvions doter un robot de la capacité à exprimer un comportement habituel, complémentaire de sa capacité à planifier pour agir. Ce comportement habituel vise à remplacer l'utilisation de la planification dans les contextes suffisamment connus du robot, et gagner ainsi le coût de cette planification, tout en agissant correctement. Nous avons proposé, en nous inspirant des modèles de l'étude du comportement chez les mammifères, une architecture de contrôle robotique dont la couche de décision combine un algorithme d'apprentissage par renforcement indirect (expert dirigé vers un but) et un algorithme d'apprentissage par renforcement direct (expert habituel), algorithmes classiquement utilisés pour modéliser l'automatisation du comportement chez l'animal.

Une première étude a montré que, d'un point de vue robotique, la combinaison de ces algorithmes offrait un avantage face à un changement de condition dans la tâche, en terme de robustesse de la performance. En revanche, contrairement à l'hypothèse des neurosciences computationnelles que l'algorithme indirect a une meilleure information sur les valeurs d'action que l'algorithme direct (KERAMATI et al. 2011), nos résultats montrent que l'algorithme direct atteint dans cette tâche une meilleure performance que l'algorithme indirect. Cette différence s'explique principalement par le fait que dans notre architecture, le modèle n'est pas donné a priori mais construit au cours des expériences, et la tâche dynamique, ce qui rend le processus de planification peu efficace.

Nous avons ensuite proposé une série de méthodes, inspirées de la littérature des neurosciences computationnelles et de l'apprentissage par renforcement, pour combiner le plus efficacement possible ces algorithmes et bénéficier des interactions qui se produisent entre eux. Les résultats que nous avons obtenus n'améliorent pas la performance du robot en soi, mais nous ont permis de mieux identifier les difficultés de l'algorithme indirect. Nous avons par conséquent proposé un mécanisme de conservation du plan trouvé par cet algorithme, idée relativement ignorée à notre connaissance dans les modèles du comportement instrumental.

En nous éloignant de la modélisation classique des comportements, nous avons évalué notre architecture dans une tâche d'interaction homme-robot. Ce travail nous a permis de constater l'effet bénéfique de la combinaison de systèmes de décision pour la robotique. Nous avons pu observer le transfert entre les connaissances données a priori au système de planification vers l'algorithme d'AR direct. Ces résultats ont été renforcés par l'étude d'une tâche de navigation sur robot réel, dans laquelle nous avons également proposé une nouvelle méthode d'arbitrage. Si la méthode d'arbitrage ne permet pas un réel transfert de contrôle entre comportements, elle demande à être plus étudiée pour juger de son intérêt.

9.2 Discussions et perspectives

Nous discutons ici des limites de notre approche et des perspectives ouvertes à sa suite.

9.2.1 Incarnation des processus de décision

Donner le contrôle d'un agent dans un environnement réaliste à un algorithme de décision pose de nombreux problèmes, bien connus en robotique, mais moins voire pas considérés dans les modèles du comportement des mammifères. Une différence importante entre les deux domaines réside dans le point de vue pris face au problème : le modélisateur formalise la tâche expérimentée par l'animal sous forme de MDP ; les observations du comportement de l'animal sont simplifiées à la lumière de ce MDP ; il peut ensuite définir un modèle de l'animal dans le cadre du modèle du problème, et chercher à reproduire les données observées dans le modèle du problème. Or, cette transformation des observations du monde en un MDP plus ou moins fidèle est loin d'être triviale pour un robot qui, si l'on souhaite se confronter à des problèmes réels, ne dispose que des informations de ses capteurs. Du point de vue de la délibération abstraite, l'état est un symbole manipulé par le robot, et le problème de l'ancrage des symboles s'y applique (HARNAD 1990). De plus, l'application des algorithmes d'apprentissage par renforcement suppose que la propriété de Markov tient : toute l'information pour prendre une décision est contenue dans l'état présent. Même dans des représentations d'états pour lesquelles la transition entre le monde et le MDP qui modélise le problème est simple, e.g. la discrétisation de l'espace métrique de notre tâche de navigation (chapitre 8), cette propriété n'est pas forcément vérifiée : dans l'état 29 de notre tâche, faire l'action 5 peut effectivement mener dans l'état récompensé 30, mais aussi nous faire heurter le mur si le robot est en fait vers le haut de la zone couverte par l'état. Une définition d'état rendant cette hypothèse plus vraie pourrait être de s'appuyer sur les transitions entre lieux, mais il est probable que cette définition soit-elle même insuffisante dans un autre endroit de l'espace. En robotique, ce rôle d'abstraction est dédié aux couches inférieures de l'architecture. Il s'appuie en général sur de nombreuses hypothèses (SISBOT et al. 2011). Une manière plus adaptable consiste à apprendre incrémentalement des classes sur les perceptions. PIATER et al. (2011) propose un mécanisme de décomposition incrémentale sur des descripteurs de perception, en fonction de l'erreur de prédiction commise par le robot, une forte erreur étant indicative du besoin d'affiner la représentation d'état. YASUDA et OHKURA (2008) comparent l'expérience sensorielle

courante à celles vécues précédemment pour décider de créer un nouvel état. Les progrès récents de l'apprentissage profond (MNIH et al. 2015 ; SCHMIDHUBER 2014 ; LECUN et al. 2015) offrent également une piste intéressante : la profondeur des réseaux de neurones considérés en fait des approximateurs de fonctions capables d'apprendre des descripteurs très abstraits de leur flux perceptif. Cette puissance d'abstraction se fait cependant au prix de l'entraînement du réseau avec suffisamment d'exemples, ce qui n'est pas forcément facile à obtenir dans une tâche robotique réelle.

Une autre question, soulevée par le côté abstrait du niveau de délibération où nous nous plaçons, est la question des actions. Comme pour les états, définir les actions pertinentes pour le robot, qui lui permettront d'agir dans l'ensemble des situations auxquelles il sera confronté n'est pas évident. Plutôt que de les définir a priori, il nous semble pertinent de donner au robot la capacité propre de former ses propres actions : le robot peut ainsi utiliser dans son processus de décision des actions qui n'auront pas été prévues initialement, voire des actions plus complexes, dont il aurait cependant appris la pertinence. KONIDARIS et BARTO (2009) ; KONIDARIS et al. (2010) proposent ainsi une approche d'apprentissage par renforcement hiérarchique (BARTO et MAHADEVAN 2003) qui peut apprendre par démonstration ou à partir de la politique suivie par le robot. Ils évaluent cette approche sur robot réel (KONIDARIS et al. 2011) et montrent d'une part, qu'il est possible d'acquérir des actions pertinentes avec peu de connaissances données a priori, et d'autre part, que certaines de ces actions sont transférables à d'autres tâches (les actions de manipulation, indépendantes de la pièce dans lequel le robot apprend). Ce type d'approche offre une piste très intéressante pour effectuer le processus d'abstraction, KONIDARIS et al. (2014) étend d'ailleurs ce travail à la génération d'actions en langage PDDL pour leur intégration dans le processus de planification.

Ces travaux font écho à l'acquisition d'habitudes comportementales dans le vivant. Si la vision classique (DAW et al. 2005) modélise les habitudes comme des associations état - action motivée par l'obtention de récompense, elle ne détaille pas le processus d'apprentissage de ces habitudes, qui est simplement fait en parallèle du comportement dirigé vers un but et selon un apprentissage par renforcement direct. Cette vision se concentre plutôt sur la manière de coordonner les modèles du comportement. Cependant, le travail de DEZFOULI (2015), en remettant en question ce modèle classique des habitudes, montre que la question des habitudes n'est pas tant celle de comment coordonner des systèmes multiples d'apprentissage (puisque la sélection d'une habitude comportementale se fait de la même manière qu'une autre action) mais plutôt celle de former une représentation plus abstraite d'un ensemble d'actions élémentaires. Acquérir des habitudes revient donc à former des actions plus complexes, elles-mêmes pouvant être intégrées au sein d'autres actions plus complexes. Cette idée présente une manière relativement élégante de construire incrémentalement le comportement d'un robot, depuis des actions élémentaires, et représente à notre avis une piste à explorer pour que la notion d'habitude contribue à accroître l'autonomie des robots.

9.2.2 Intérêt des habitudes pour la robotique

L'intérêt de notre approche réside dans les propriétés complémentaires des systèmes coordonnés. Nous avons illustré dans le chapitre 8 les différents coûts de planification, élevés et proportionnels à la quantité d'états à évaluer pour le système de planification, faibles et indépendants de la quantité d'états pour le système habituel. L'interaction des deux systèmes accroît la vitesse d'apprentissage du système habituel. Une problématique que nous n'avons pas adressé est la question de l'apprentissage de plusieurs habitudes. En restant dans une vision horizontale de la coordination des systèmes, la nature actuelle du système habituel écrase une habitude acquise par une éventuelle nouvelle habitude qui se formerait dans un contexte stable. Pour remédier à ce problème, il est nécessaire de détecter quand le contexte a changé, et nous avons proposé un ensemble de mesures (entropie de décision, mesure de l'apprentissage, etc.) qui peuvent aider à cela. La notion de nouveauté, proposée par JAUFFRET et al. (2013) pour rendre la notion de frustration indépendante d'un but, est également à considérer : elle mesure une incapacité à prédire, ce qui est aussi le cas de l'erreur de prédiction de notre système habituel, mais elle a l'avantage de fonctionner sur les perceptions du robot plutôt que sur un signal peu riche comme la récompense. Cette nouveauté peut servir à associer le contexte sensoriel une fois qu'il est stable à la politique apprise par le système habituel. Un tel mécanisme se rapproche de l'idée de changement de contexte développée dans CALUWAERTS et al. (2012), mais là encore sans dépendre de l'information de récompense.

Dans notre travail, nous nous sommes principalement concentrés sur l'évaluation de notre architecture en comparant les conditions de contrôle, où un seul système décide des actions du robot, et la combinaison de systèmes selon différents critères. Pour conclure sur l'intérêt de la notion d'habitude en robotique, il sera nécessaire de comparer d'autres modèles de la littérature à notre architecture, dans les tâches présentées ici. Nous avons notamment commencé à étudier le modèle de KERAMATI et al. (2011) dans la tâche de poussée de blocs du chapitre 6 dans le cadre du stage de Master 1 de Rémi Dromnelle. Le modèle est adapté pour apprendre la fonction de transition au fur et à mesure. Les résultats préliminaires montrent que si le modèle apprend bien (figure 9.1, gauche) et accumule de la récompense, la découverte d'une politique est plus lente que les méthodes proposées ici (9.1, droite). Il est cependant nécessaire d'étudier plus systématiquement l'espace des paramètres du modèle et notamment de regarder plus finement son apprentissage de la fonction de transition comparé à notre expert dirigé vers un but.

9.2.3 Performance et récompense

Nous avons majoritairement analysé notre architecture sous l'angle de sa performance à effectuer la tâche auquel le robot est confronté. Si l'accumulation de récompense est bien ce qui définit l'optimalité dans le paradigme de l'apprentissage par renforcement, d'un point de vue robotique, l'évaluation de l'autonomie ne devrait pas se limiter à cette notion de récompense.

Plus que sa capacité à atteindre un but, c'est la capacité du robot à agir « de son propre chef » qui le rend autonome. L'architecture que nous avons proposée est un système de

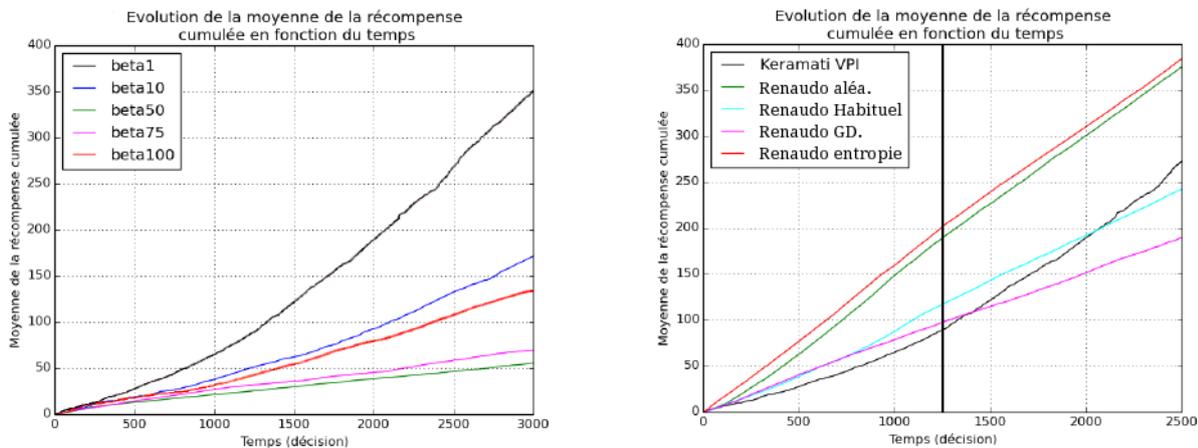


Figure 9.1 – Comparaison du modèle de KERAMATI *et al.* (2011) avec quelques méthodes de sélection de RENAUDO *et al.* (2015c). **Gauche** : Performance dans la condition régulière du modèle seul pour différentes valeurs de température du softmax ($\beta = 1/\tau$). **Droite** : Performance dans la condition SS. La ligne verticale est l’instant du changement de vitesse.

décision, avec la particularité de combiner un système qui décide à long terme (planificateur) et un système qui décide en fonction de son expérience passée (système habituel). Ce système de décision détermine le comportement en fonction des retours qui lui sont faits. Mais si ces retours viennent, par exemple, d’un autre humain, ce que le robot décidera de faire dépend de la volonté de l’humain. Dans ce cadre, le seul facteur d’autonomie est que lorsque la décision est prise de manière probabiliste, le robot peut « choisir » de ne pas suivre la meilleure action connue vers la source de récompense.

Il est donc nécessaire de doter le robot d’un système de motivation qui le pousse à agir, et qui lui soit propre. Il existe une littérature relativement importante sur la question des motivations (RYAN et DECI 2000 ; BALDASSARRE et MIROLI 2013b), qui différencie *motivation extrinsèque* et *motivation intrinsèque*. Ces concepts viennent de la psychologie (HARLOW 1950), et désignent respectivement une motivation dont la source est extérieure à l’individu de celle dont la source est l’individu lui-même. La première regroupe par exemple l’acquisition de ressources (nourriture, satisfaction sociale), tandis que la seconde concerne l’acquisition d’information. La curiosité, l’attirance vers le nouveau sont des exemples de motivations intrinsèques. L’intérêt d’un tel mécanisme dans le cadre de notre architecture est qu’elle peut fournir à l’architecture de décision un moyen d’apprendre en l’absence d’un but explicite. La notion de nouveauté peut pousser le robot à explorer les limites de son modèle de transitions en plaçant automatiquement de la récompense qui va guider la planification. Si ce processus incite le robot à passer plusieurs fois par les mêmes transitions, le système habituel peut former une habitude qui d’un point de vue d’un observateur extérieur, serait uniquement la conséquence de la statistique des transitions visitées, mais serait en fait dirigé par une récompense interne au robot. Développer un tel modèle ici est en dehors du propos, mais le concept de motivation intrinsèque a effectivement été appliqué à la robotique, dans le cadre d’une approche développementale (voir OUDEYER *et al.* (2013) ; HART et GRUPEN (2013) dans BALDAS-

SARRE et MIROLLI (2013a)). La notion de performance devient alors liée à l'adaptabilité du robot plutôt que sa capacité à trouver le chemin le plus court et sûr dans son espace d'états.

9.3 Conclusion

Nous n'avons fait qu'effleurer l'ensemble des questions qui se posent à une architecture robotique réellement efficace, qui bénéficie de la notion d'habitude. Nous avons essayé de dresser un parallèle entre le comportement des mammifères et le comportement des robots, qui soulève de nombreuses questions sur l'un comme l'autre domaine. Les perspectives qui nous semblent importantes sont les suivantes :

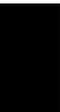
- Étudier le lien du monde avec les états et les actions du robot ; plus généralement, la question des représentations sensorimotrices qui sont nécessaires à un robot autonome.
- Étudier le comportement observé, dans le vivant ou pour les robots, comme une expression de la formation de ces représentations.
- Intégrer la notion de motivation dans l'architecture, pour rendre le robot réellement autonome plutôt que simplement attiré par les « sources de plaisir » de l'environnement.

Ce travail représente une petite étape dans l'étude de ce parallèle et nous espérons que les études futures pourront apporter des réponses à ces questions.

Publications

Articles

- RENAUDO, Erwan, Benoît GIRARD, Raja CHATILA et Mehdi KHAMASSI (2014). « Design of a Control Architecture for Habit Learning in Robots ». Dans : *Biomimetic and Biohybrid Systems, LNAI Proceedings*, p. 249–260.
- RENAUDO, Erwan, Sandra DEVIN, Benoît GIRARD, Raja CHATILA, Rachid ALAMI, Mehdi KHAMASSI et Aurélie CLODIC (2015a). « Learning to Interact with Humans Using Goal-Directed and Habitual Behaviors ». Dans : *RoMan 2015, Workshop on Learning for Human-Robot Collaboration*.
- RENAUDO, Erwan, Benoît GIRARD, Raja CHATILA et Mehdi KHAMASSI (2015b). « Respective advantages and disadvantages of model-based and model-free reinforcement learning in a robotics neuro-inspired cognitive architecture ». Dans : *Biologically Inspired Cognitive Architectures BICA 2015*. Lyon, France, p. 178–184.
- (2015c). « Which criteria for autonomously shifting between goal-directed and habitual behaviors in robots? » Dans : *5th International Conference on Development and Learning and on Epigenetic Robotics (ICDL-EPIROB)*. Providence, RI, USA, p. 254–260.



Bibliographie

- ADAMS, Christopher D (1982). « Variations in the sensitivity of instrumental responding to reinforcer devaluation ». Dans : *The Quarterly Journal of Experimental Psychology* 34.2, p. 77–98.
- ADAMS, Christopher D et Anthony DICKINSON (1981). « Instrumental responding following reinforcer devaluation ». Dans : *The Quarterly journal of experimental psychology* 33.2, p. 109–121.
- AKLIL, Nassim, Alain MARCHAND, Virginie FRESNO, Étienne COUTUREAU, Ludovic DENOYER, Benoît GIRARD et Mehdi KHAMASSI (2014). « Modelling rat learning behavior under uncertainty in a non-stationary multi-armed bandit task ». Dans : *Fourth Symposium on Biology of Decision Making (SBDM 2014)*. Paris.
- ALAMI, R., M. WARNIER, J. GUITTON, S. LEMAIGNAN et E. A. SISBOT (2011). « When the robot considers the human... » Dans : *Proceedings of the 15th International Symposium on Robotics Research*.
- ALAMI, Rachid, Raja CHATILA, Sara FLEURY, Malik GHALLAB et Félix INGRAND (1998). « An architecture for autonomy ». Dans : *The International Journal of Robotics Research* 17.4, p. 315–337.
- ARKIN, Ronald C. (1989). « Motor schema-based mobile robot navigation ». Dans : *The International Journal of Robotics Research* 8.4, p. 92–112.
- (1990). « Integrating Behavioral, Perceptual, and World Knowledge in Reactive Navigation ». Dans : *Robot. Auton. Syst.* 6.1-2, p. 105–122. ISSN : 0921-8890.
- ARKIN, Ronald C et Tucker BALCH (1997). « AuRA : Principles and practice in review ». Dans : *Journal of Experimental and Theoretical Artificial Intelligence* 9.2-3, p. 175–189.
- ARKIN, Ronald C., Edward M. RISEMAN et Allen R. HANSON (1988). *AuRA : An Architecture for Vision-Based Robot Navigation*. Rap. tech.
- ATKESON, Christopher G., Andrew W. MOORE et Stefan SCHAAL (fév. 1997). « Locally Weighted Learning ». Dans : *Artif. Intell. Rev.* 11.1-5, p. 11–73. ISSN : 0269-2821.
- BALDASSARRE, Gianluca (2002). « A modular neural-network model of the basal ganglia's role in learning and selecting motor behaviours ». Dans : *Cognitive Systems Research* 3.1, p. 5–13.
- BALDASSARRE, Gianluca et Marco MIROLI (2013a). *Intrinsically Motivated Learning in Natural and Artificial Systems*. Springer Publishing Company, Incorporated. ISBN : 364232374X, 9783642323744.
- (2013b). « Intrinsically motivated learning systems : an overview ». Dans : *Intrinsically Motivated Learning in Natural and Artificial Systems*. Springer, p. 1–14.
- BALLEINE, Bernard W et Anthony DICKINSON (1998). « Goal-directed instrumental action : contingency and incentive learning and their cortical substrates ». Dans : *Neuropharmacology* 37.4–5, p. 407–419. ISSN : 0028-3908.

- BALLEINE, Bernard W et John P O'DOHERTY (2010). « Human and rodent homologies in action control : corticostriatal determinants of goal-directed and habitual action ». Dans : *Neuropsychopharmacology* 35.1, p. 48–69.
- BARTO, Andrew G. et Sridhar MAHADEVAN (2003). « Recent Advances in Hierarchical Reinforcement Learning ». Dans : *Discrete Event Dynamic Systems* 13.4, p. 341–379.
- BARTO, Andrew G., Richard S. SUTTON et Charles W. ANDERSON (1983). « Neuronlike adaptive elements that can solve difficult learning control problems ». Dans : *IEEE Transactions on Systems, Man, and Cybernetics* 13.5, p. 835–846.
- BELLMAN, Richard (1957). *Dynamic Programming*. 1^{re} éd. Princeton, NJ, USA : Princeton University Press.
- BELLOT, J., O. SIGAUD et M. KHAMASSI (2012). « Which Temporal Difference Learning algorithm best reproduces dopamine activity in a multi-choice task? » Dans : *From Animals to Animats : Proceedings of the 12th International Conference on Adaptive Behaviour (SAB 2012)*, Ziemke, T., Balkenius, C., Hallam, J. (Eds). T. 7426/2012. Lecture Notes in Computer Science. Odense, Denmark : Springer, p. 289–298.
- BERGMANN, Ralph, Héctor MUÑOZ-AVILA, Manuela VELOSO et Erica MELIS (1998). *Case-Based Reasoning Applied to Planning*.
- BERNS, G. S. et T. J. SEJNOWSKI (1998). « A computational model of how basal ganglia produce sequences ». Dans : *Journal of Cognitive Neuroscience* 10, p. 108–121.
- BERRIDGE, Kent C. (2006). « The debate over dopamine's role in reward : the case for incentive salience ». Dans : *Psychopharmacology* 191.3, p. 391–431. ISSN : 1432-2072.
- BLODGETT, H.C. (1929). *The Effect of the Introduction of Reward Upon the Maze Performance of Rats*. University of California publications in psychology. Kraus Reprint Company.
- BONASSO, R. Peter (1991). « Integrating Reaction Plans and Layered Competences Through Synchronous Control ». Dans : *Proceedings of the 12th International Joint Conference on Artificial Intelligence - Volume 2. IJCAI'91*. Sydney, New South Wales, Australia : Morgan Kaufmann Publishers Inc., p. 1225–1231. ISBN : 1-55860-160-0.
- BONASSO, R. Peter, R. James FIRBY, Erann GAT, David KORTENKAMP, David P. MILLER et Marc G. SLACK (1997). *Experiences with an Architecture for Intelligent, Reactive Agents*.
- BORRELLY, Jean-Jacques, Éve COSTE-MANIERE, Bernard ESPIAU, Konstantinos KAPELLOS, Roger PISSARD-GIBOLLET, Daniel SIMON et Nicolas TURRO (1998). « The ORCCAD architecture ». Dans : *The International Journal of Robotics Research* 17.4, p. 338–359.
- BRAFMAN, Ronen I. et Moshe TENNENHOLTZ (oct. 2002). « R-max - a general polynomial time algorithm for near-optimal reinforcement learning ». Dans : *Journal of Machine Learning Research* 3, p. 213–231. ISSN : 1532-4435.
- BROOKS, Rodney A. (1986). « A robust layered control system for a mobile robot ». Dans : *Robotics and Automation, IEEE Journal of* 2.1, p. 14–23. ISSN : 0882-4967.
- CALUWAERTS, K., M. STAFFA, S. N'GUYEN, C. GRAND, L. DOLLÉ, A. FAVRE-FÉLIX, B. GIRARD et M. KHAMASSI (2012). « A biologically inspired meta-control navigation system for the Psikharpax rat robot ». Dans : *Bioinspiration & Biomimetics*.
- CELIBERTO, Luiz A, Jackson P MATSUURA, Ramón López de MANTARAS, Reinaldo BIANCHI et al. (2011). « Using cases as heuristics in reinforcement learning : A transfer learning application ». Dans :
- CHAVARRIAGA, Ricardo, Thomas STRÖSSLIN, Denis SHEYNIKHOVICH et Wulfram GERSTNER (2005). « A computational model of parallel navigation systems in rodents ». Dans : *Neuroinformatics* 3.3, p. 223–241.

- CHEVALIER, Pauline, Brice ISABLEU, Jean-Claude MARTIN et Adriana TAPUS (2016). « Advances in Robot Design and Intelligent Control : Proceedings of the 24th International Conference on Robotics in Alpe-Adria-Danube Region (RAAD) ». Dans : sous la dir. de Theodor BORANGIU. Cham : Springer International Publishing. Chap. Individuals with Autism : Analysis of the First Interaction with Nao Robot Based on Their Proprioceptive and Kinematic Profiles, p. 225–233. ISBN : 978-3-319-21290-6.
- CHIEN, Steve A, Russell KNIGHT, Andre STECHERT, Rob SHERWOOD et Gregg RABIDEAU (2000). « Using Iterative Repair to Improve the Responsiveness of Planning and Scheduling. » Dans : *AIPS*, p. 300–307.
- CLODIC, Aurélie, Hung CAO, Samir ALILI, Vincent MONTREUIL, Rachid ALAMI et R. CHATILA (2009). « Experimental Robotics : The Eleventh International Symposium ». Dans : sous la dir. d'Oussama KHATIB, Vijay KUMAR et George J. PAPPAS. Berlin, Heidelberg : Springer Berlin Heidelberg. Chap. SHARY : A Supervision System Adapted to Human-Robot Interaction, p. 229–238. ISBN : 978-3-642-00196-3.
- COLLINS, A. et M.J. FRANK (2012). « How much of reinforcement learning is working memory, not reinforcement learning? A behavioral, computational, and neurogenetic analysis ». Dans : *Eur J Neurosci* 35.7, p. 1024–1035.
- CUPERLIER, Nicolas, Mathias QUOY et Philippe GAUSSIER (2007). « Neurobiologically inspired mobile robot navigation and planning ». Dans : *Frontiers in Neurorobotics* 1.3. ISSN : 1662-5218.
- DAUTENHAHN, Kerstin et Iain WERRY (2004). « Towards interactive robots in autism therapy : Background, motivation and challenges ». Dans : *Pragmatics & Cognition* 12.1, p. 1–35.
- DAW, Nathaniel D., Yaël. NIV et Peter DAYAN (2005). « Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. » Dans : *Nat. Neurosci.* 8.12, p. 1704–1711.
- DAW, Nathaniel D., Samuel J. GERSHMAN, Ben SEYMOUR, Peter DAYAN et Raymond J. DOLAN (2011). « Model-Based Influences on Humans' Choices and Striatal Prediction Errors ». Dans : *Neuron* 69.6, p. 1204–1215. ISSN : 0896-6273.
- DAYAN, Peter (2009). « Goal-directed control and its antipodes ». Dans : *Neural Networks* 22.3. Goal-Directed Neural Systems, p. 213–219. ISSN : 0893-6080.
- DE MANTARAS, Ramon Lopez, David MCSHERRY, Derek BRIDGE, David LEAKE, Barry SMYTH, Susan CRAW, Boi FALTINGS, Mary Lou MAHER, Michael T COX, Kenneth FORBUS, Mark KEANE, Agnar AAMODT et Ian WATSON (2005). « Retrieval, reuse, revision and retention in case-based reasoning ». Dans : *Knowledge Engineering Review* 20.3, p. 215.
- DEARDEN, Richard, Nir FRIEDMAN et Stuart RUSSELL (1998). « Bayesian Q-learning ». Dans : *Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*. AAAI '98/IAAI '98. Madison, Wisconsin, USA : American Association for Artificial Intelligence, p. 761–768. ISBN : 0-262-51098-7.
- DEARDEN, Richard, Nir FRIEDMAN et David ANDRE (1999). « Model Based Bayesian Exploration ». Dans : *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*. UAI'99. Stockholm, Sweden : Morgan Kaufmann Publishers Inc., p. 150–159. ISBN : 1-55860-614-9.
- DEGRIS, Thomas, Martha WHITE et Richard S. SUTTON (2012). « Linear Off-Policy Actor-Critic. » Dans : *ICML*. Omnipress.
- DEZFOULI, Amir (2015). « Hierarchical models of goal-directed and automatic actions ». Thèse de doct.

- DEZFOULI, Amir et Bernard W BALLEINE (2012). « Habits, action sequences and reinforcement learning ». Dans : *European Journal of Neuroscience* 35.7, p. 1036–1051.
- (2013). « Actions, action sequences and habits : evidence that goal-directed and habitual action control are hierarchically organized ». Dans : *PLoS computational biology* 9.12.
- DICKINSON, A. (1985). « Actions and Habits : The Development of Behavioural Autonomy ». Dans : *Philosophical Transactions of the Royal Society B : Biological Sciences* 308.1135, p. 67–78.
- DIETTERICH, Thomas G (2000). « Ensemble methods in machine learning ». Dans : *Multiple classifier systems*. Springer, p. 1–15.
- DOLLÉ, L., D. SHEYNIKHOVICH, B. GIRARD, R. CHAVARRIAGA et A. GUILLOT (2010). « Path planning versus cue responding : a bioinspired model of switching between navigation strategies ». Dans : *Biological Cybernetics* 103.4, p. 299–317.
- DOMINEY, Peter F., Michael A. ARBIB et Jean-Paul JOSEPH (1995). « A cortico-subcortical model for generation of spatially accurate sequential saccades ». Dans : *Journal of Cognitive Neuroscience* 7.3, p. 311–336.
- DOYA, Kenji (1999). « What are the computations of the cerebellum, the basal ganglia and the cerebral cortex ? » Dans : *Neural Networks* 12.7-8, p. 961–974.
- DOYA, Kenji, Kazuyuki SAMEJIMA, Ken ichi KATAGIRI et Mitsuo KAWATO (2002). « Multiple model-based reinforcement learning ». Dans : *Neural Computation* 14, p. 1347–1369.
- ESTLIN, Tara, Gregg RABIDEAU, Darren MUTZ et Steve CHIEN (1999). « Using continuous planning techniques to coordinate multiple rovers ». Dans : *In IJCAI Workshop on Scheduling and Planning*, p. 4–45.
- FAUSSER, Stefan et Friedhelm SCHWENKER (2011). « Ensemble Methods for Reinforcement Learning with Function Approximation ». Dans : *Multiple Classifier Systems - 10th International Workshop, MCS 2011, Naples, Italy, June 15-17, 2011. Proceedings*, p. 56–65.
- (2013). « Neural Network Ensembles in Reinforcement Learning ». Dans : *Neural Processing Letters* 41.1, p. 55–69.
- (2015). « Selective neural network ensembles in reinforcement learning : Taking the advantage of many agents ». Dans : *Neurocomputing* 169, p. 350–357.
- FIKES, Richard E. et Nils J. NILSSON (1971). « STRIPS : A New Approach to the Application of Theorem Proving to Problem Solving ». Dans : *Proceedings of the 2Nd International Joint Conference on Artificial Intelligence*. IJCAI'71. London, England : Morgan Kaufmann Publishers Inc., p. 608–620.
- FILLIAT, David et Jean-Arcady MEYER (2003). « Map-based navigation in mobile robots : : I. a review of localization strategies ». Dans : *Cognitive Systems Research* 4.4, p. 243–282.
- FIRBY, Robert James (1989). « Adaptive Execution in Complex Dynamic Worlds ». Thèse de doct. New Haven, CT, USA.
- FREUND, Yoav et Robert E. SCHAPIRE (1999). « A Short Introduction to Boosting ». Dans : *In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, p. 1401–1406.
- GAT, Erann (1998). « Artificial Intelligence and Mobile Robots ». Dans : sous la dir. de David KORTENKAMP, R. Peter BONASSO et Robin MURPHY. Cambridge, MA, USA : MIT Press. Chap. Three-layer Architectures, p. 195–210. ISBN : 0-262-61137-6.
- GEIST, Matthieu, Olivier PIETQUIN et Gabriel FRICOUT (2009). « Kalman Temporal Differences : the deterministic case ». Dans : *Adaptive Dynamic Programming and Reinforcement Learning, 2009. ADPRL'09. IEEE Symposium on*. IEEE, p. 185–192.

- GIRARD, B., D. FILLIAT, J.-A. MEYER, A. BERTHOZ et A. GUILLOT (2005). « Integration of navigation and action selection functionalities in a computational model of cortico-basal ganglia-thalamo-cortical loops ». Dans : *Adaptive Behavior* 13.2, p. 115–130.
- GLÄSCHER, Jan, Nathaniel DAW, Peter DAYAN et John P O'DOHERTY (2010). « States versus rewards : dissociable neural prediction error signals underlying model-based and model-free reinforcement learning ». Dans : *Neuron* 66.4, p. 585–595.
- GOETZ, Jennifer, Sara KIESLER et Aaron POWERS (2003). « Matching robot appearance and behavior to tasks to improve human-robot cooperation ». Dans : *Robot and Human Interactive Communication, 2003. Proceedings. ROMAN 2003. The 12th IEEE International Workshop on*. IEEE, p. 55–60.
- GOODRICH, Michael A. et Alan C. SCHULTZ (jan. 2007). « Human-robot Interaction : A Survey ». Dans : *Found. Trends Hum.-Comput. Interact.* 1.3, p. 203–275. ISSN : 1551-3955.
- GRAYBIEL, Ann M (1995). « Building action repertoires : memory and learning functions of the basal ganglia ». Dans : *Current Opinion in Neurobiology* 5.6, p. 733 –741. ISSN : 0959-4388.
- GRISSETTI, G., C. STACHNISS et W. BURGARD (fév. 2007). « Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters ». Dans : *Trans. Rob.* 23.1, p. 34–46. ISSN : 1552-3098.
- GUAZZELLI, Alex, Mihail BOTA, Fernando J. CORBACHO et Michael A. ARBIB (1998). « Affordances. Motivations, and the World Graph Theory. » Dans : *Adaptive Behaviour* 6.3-4, p. 435–471.
- HAMPTON, Alan N., Peter BOSSAERTS et John P. O'DOHERTY (2006). « The role of the ventromedial prefrontal cortex in abstract state-based inference during decision making in humans ». Dans : *Journal of Neuroscience* 26.32, p. 8360–8367.
- HANOUNE, Souheil (2015). « Towards a biologically plausible model of action selection for a mobile robot ». Thèse de doct. Université de Cergy-Pontoise.
- HANS, Alexander et Steffen UDLUFT (2010). « Ensembles of neural networks for robust reinforcement learning ». Dans : *Machine Learning and Applications (ICMLA), 2010 Ninth International Conference on*. IEEE, p. 401–406.
- HARLOW, Harry F (1950). « Learning and satiation of response in intrinsically motivated complex puzzle performance by monkeys. » Dans : *Journal of Comparative and Physiological Psychology* 43.4, p. 289.
- HARNAD, Stevan (1990). « The Symbol Grounding Problem ». Dans : *Physica D : Nonlinear Phenomena* 42, p. 335–346.
- HART, Stephen et Roderic GRUPEN (2013). « Intrinsically motivated affordance discovery and modeling ». Dans : *Intrinsically Motivated Learning in Natural and Artificial Systems*. Springer, p. 279–300.
- HARTLEY, Ralph et Frank PIPITONE (1991). « Experiments with the subsumption architecture ». Dans : *Proceedings of IEEE International Conference on Robotics and Automation, 1991*. T. 2, p. 1652–1658.
- HASSON, C., P. GAUSSIER et S. BOUCENNA (2012). « Emotions as a dynamical system : the interplay between the meta-control and communication function of emotions ». Dans : *Paladyn* 2.3, p. 111–125. ISSN : 2081-4836.
- HESTER, Todd et Peter STONE (2012). « Learning and Using Models ». Dans : *Reinforcement Learning : State of the Art*. Sous la dir. de Marco WIERING et Martijn van OTTERLO. Berlin, Germany : Springer.

- HESTER, Todd, Michael QUINLAN et Peter STONE (2012). « RTMBA : A Real-Time Model-Based Reinforcement Learning Architecture for Robot Control ». Dans : *IEEE International Conference on Robotics and Automation (ICRA)*.
- HOLLERMAN, J. R. et W. SCHULTZ (1998). « Dopamine neurons report an error in the temporal prediction of reward during learning. » Dans : *Nature Neuroscience* 1, p. 304–309.
- HOWARD, Ronald A. (1960). *Dynamic programming and Markov processes*. Cambridge, MA.
- HÜLLERMEIER, Eyke (2007). *Case-based approximate reasoning*. T. 44. Springer Science & Business Media.
- HUYS, Quentin J. M., Neir ESHEL, Elizabeth O’NIONS, Luke SHERIDAN, Peter DAYAN et Jonathan P. ROISER (mar. 2012). « Bonsai Trees in Your Head : How the Pavlovian System Sculpts Goal-Directed Choices by Pruning Decision Trees ». Dans : *PLoS Comput Biol* 8.3, p. 1–13.
- INGRAND, F. F., R. CHATILA, R. ALAMI et F. ROBERT (1996). « PRS : a high level supervision and control language for autonomous mobile robots ». Dans : *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*. T. 1, 43–49 vol.1.
- INGRAND, Félix et Malik GHALLAB (2014). « Deliberation for autonomous robots : A survey ». Dans : *Artificial Intelligence*, p. 1–40.
- JACOBS, Robert A., Michael I. JORDAN, Steven J. NOWLAN et Geoffrey E. HINTON (mar. 1991). « Adaptive Mixtures of Local Experts ». Dans : *Neural Comput.* 3.1, p. 79–87. ISSN : 0899-7667.
- JAUFFRET, A., M. BELKAID, N. CUPERLIER, P. GAUSSIER et P. TARROUX (2013). « Frustration as a way toward autonomy and self-improvement in robotic navigation ». Dans : *2013 IEEE Third Joint International Conference on Development and Learning and Epigenetic Robotics (ICDL)*, p. 1–7.
- JIANG, Ju et M.S. KAMEL (2006). « Aggregation of Reinforcement Learning Algorithms ». Dans : *Neural Networks, 2006. IJCNN '06. International Joint Conference on*, p. 68–72.
- KAELBLING, Leslie Pack, Michael L. LITTMAN et Andrew W. MOORE (1996). « Reinforcement learning : a survey ». Dans : *Journal of Artificial Intelligence Research* 4, p. 237–285.
- KEARNS, Michael et Satinder SINGH (nov. 2002). « Near-Optimal Reinforcement Learning in Polynomial Time ». Dans : *Mach. Learn.* 49.2-3, p. 209–232. ISSN : 0885-6125.
- KERAMATI, M., A. DEZFOULI et P. PIRAY (2011). « Speed/Accuracy trade-off between the habitual and goal-directed processes. » Dans : *PLoS Computational Biology* 7.5, p. 1–25.
- KHAMASSI, M. et M. D. HUMPHRIES (2012). « Integrating cortico-limbic-basal ganglia architectures for learning model-based and model-free navigation strategies ». Dans : *Frontiers in Behavioral Neuroscience* 6 :79, p. 1–19.
- KHAMASSI, M., B. GIRARD, A. CLODIC, S. DEVIN, E. RENAUDO, E. PACHERIE, R. ALAMI et R. CHATILA (2016). « Integration of Action, Joint Action and Learning in Robot Cognitive Architectures ». Dans : *Intellectica*. to appear.
- KHAMASSI, Mehdi, Louis-Emmanuel MARTINET et Agn e’s GUILLOT (2006). « Combining Self-Organizing Maps with Mixture of Experts : Application to an Actor-Critic Model of Reinforcement Learning in the Basal Ganglia ». Dans : *From Animals to Animats 9 (SAB 2006)*. Sous la dir. de S. NOLFI, G. BALDASSARE, R. CALABRETTA, J.C. HALLAM, D. MAROCCO, J.-A. MEYER, O. MIGLINO et D. PARISI. LNAI 4095. Berlin, Heidelberg : Springer-Verlag, p. 394–405.

- KHAMASSI, Mehdi, Stéphane LALLÉE, Pierre ENEL, Emmanuel PROCYK et Peter F. DOMINEY (2011). « Robot cognitive control with a neurophysiologically inspired reinforcement learning model ». Dans : *Frontiers in Neurorobotics* 5 :1.
- KISHIDA, KT, I SAEZ, T LOHRENZ, MR WITCHER, AW LAXTON, SB TATTER, JP WHITE, TL ELLIS, PE PHILLIPS et PR MONTAGUE (2016). « Subsecond dopamine fluctuations in human striatum encode superposed error signals about actual and counterfactual reward. » Dans : *Proceedings of the National Academy of Sciences of the United States of America* 113.5, p. 200–5.
- KNIGHT, Russel, Steve CHIEN, Erann GAT, Tom STARBIRD, Kim GOSTELOW, Bob KELLER et Weldon SMITH (2000). « Integrating Model-based Artificial Intelligence Planning with Procedural Elaboration for Onboard Spacecraft Autonomy ». Dans : *SpaceOps*.
- KOBER, Jens, J. Andrew BAGNELL et Jan PETERS (2013). « Reinforcement Learning in Robotics : A Survey ». Dans : *International Journal of Robotics Research* 11, p. 1238–1274.
- KONDA, Vijay R. et John N. TSITSIKLIS (2003). « On Actor-Critic Algorithms ». Dans : *SIAM Journal on Control and Optimization* 42.4, p. 1143–1166.
- KONIDARIS, George et Andrew G. BARTO (2009). « Skill Discovery in Continuous Reinforcement Learning Domains using Skill Chaining ». Dans : *Advances in Neural Information Processing Systems 22*. Sous la dir. d'Y. BENGIO, D. SCHUURMANS, J. D. LAFFERTY, C. K. I. WILLIAMS et A. CULOTTA. Curran Associates, Inc., p. 1015–1023.
- KONIDARIS, George, Scott KUINDERSMA, Andrew BARTO et Roderic GRUPEN (2010). « Constructing Skill Trees for Reinforcement Learning Agents from Demonstration Trajectories ». Dans : *Advances in Neural Information Processing Systems 23 (NIPS)*. Vancouver, BC, Canada, p. 1162–1170.
- KONIDARIS, George, Scott KUINDERSMA, Roderic GRUPEN et Andrew BARTO (2011). « Autonomous Skill Acquisition on a Mobile Manipulator ». Dans : *Proceedings of the Twenty-Fifth Conference on Artificial Intelligence (AAAI)*. San Francisco, CA, p. 1468–1473.
- KONIDARIS, George, Leslie Pack KAEHLING et Tomas LOZANO-PEREZ (2014). « Constructing symbolic representations for high-level planning ». Dans :
- KORTENKAMP, David et Reid SIMMONS (2008). « Robotic systems architectures and programming ». Dans : *Springer Handbook of Robotics*. Springer, p. 187–206.
- KURUP, Unmesh et Christian LEBIERE (2012). « What can cognitive architectures do for robotics? » Dans : *Biologically Inspired Cognitive Architectures* 2, p. 88–99. ISSN : 2212-683X.
- LALLEMENT, Raphaël, Lavindra de SILVA et Rachid ALAMI (2014). « HATP : An HTN Planner for Robotics ». Dans : *CoRR* abs/1405.5345.
- LANGLEY, Pat, John E. LAIRD et Seth ROGERS (2008). « Cognitive architectures : Research issues and challenges ». Dans : *Cognitive Systems Research* 10.2, p. 141–160.
- LECUN, Yann, Yoshua BENGIO et Geoffrey HINTON (2015). « Deep learning ». Dans : *Nature* 521.7553, p. 436–444.
- LEMAIGNAN, Séverin, Mamoun GHARBI, Jim MAINPRICE, Matthieu HERRB et Rachid ALAMI (mar. 2012). « Roboscopia : A Theatre Performance for a Human and a Robot ». Dans : *Human-Robot Interaction Conference (HRI 2012)*. Boston, United States, p. 1.
- LESAINT, F., O. SIGAUD, S. B. FLAGEL, T. E. ROBINSON et M. KHAMASSI (fév. 2014). « Modelling Individual Differences in the Form of Pavlovian Conditioned Approach Responses : A Dual Learning Systems Approach with Factored Representations ». Dans : *PLoS Comput Biol* 10.2.

- LIÉNARD, J. (2013). « Modèles des Ganglions de la Base : Étude de l'anatomie fonctionnelle et de la Pathophysiologie à l'aide d'Algorithmes Évolutionnistes Multi-Objectifs ». Thèse de doct. Paris, France : UPMC.
- LJUNGBERG, T., P. APICELLA et W. SCHULTZ (1992). « Responses of monkey dopamine neurons during learning of behavioral reactions ». Dans : *Journal of Neurophysiology* 67.1, p. 145–163.
- MAESTRE, C., A. CULLY, C. GONZALES et S. DONCIEUX (2015). « Bootstrapping interactions with objects from raw sensorimotor data : a Novelty Search based approach ». Dans : *IEEE International Conference on Developmental and Learning and on Epigenetic Robotics*, p. 1–6.
- MALLET, Anthony, Simon LACROIX et Laurent GALLO (2000). « Position Estimation in Outdoor Environments using Pixel Tracking and Stereovision ». Dans : *Proceedings of the 2000 IEEE International Conference on Robotics and Automation, ICRA 2000, April 24-28, 2000, San Francisco, CA, USA*, p. 3519–3524.
- MATARIĆ, Maja J. et François MICHAUD (2008). « Behavior-Based Systems ». Dans : *Springer Handbook of Robotics*. Sous la dir. de Bruno SICILIANO et Oussama KHATIB. Springer Berlin Heidelberg, p. 891–909. ISBN : 978-3-540-23957-4.
- MCDERMOTT, Drew, Malik GHALLAB, Adele HOWE, Craig KNOBLOCK, Ashwin RAM, Manuela VELOSO, Daniel WELD et David WILKINS (1998). *PDDL - The Planning Domain Definition Language*. Rap. tech. CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision et Control.
- MEYER, Jean-Arcady et David FILLIAT (2003). « Map-based navigation in mobile robots : : Ii. a review of map-learning and path-planning strategies ». Dans : *Cognitive Systems Research* 4.4, p. 283–317.
- MNIH, Volodymyr, Koray KAVUKCUOGLU, David SILVER, Andrei A RUSU, Joel VENESS, Marc G BELLEMARE, Alex GRAVES, Martin RIEDMILLER, Andreas K FIDJELAND, Georg OSTROVSKI et al. (2015). « Human-level control through deep reinforcement learning ». Dans : *Nature* 518.7540, p. 529–533.
- MONTAGUE, P Read, Steven E HYMAN et Jonathan D COHEN (2004). « Computational roles for dopamine in behavioural control ». Dans : *Nature* 431.7010, p. 760–767.
- MOORE, Andrew W. et Christopher G. ATKESON (1993). « Prioritized sweeping : Reinforcement learning with less data and less time ». Dans : *Machine Learning*, p. 103–130.
- MUSCETTOLA, N. (1994). « HSTS : Integrating planning and scheduling ». Dans : *Intelligent Scheduling*. Sous la dir. de M. ZWEBEN et M. FOX. Morgan Kaufmann, p. 169–212.
- NAU, Dana, Yue CAO, Amnon LOTEM et Hector MUNOZ-AVILA (1999). « SHOP : Simple Hierarchical Ordered Planner ». Dans : *Proceedings of the 16th International Joint Conference on Artificial Intelligence - Volume 2. IJCAI'99*. Stockholm, Sweden : Morgan Kaufmann Publishers Inc., p. 968–973.
- NILSSON, Nils J. (1969). « A Mobile Automaton : An Application of Artificial Intelligence Techniques ». Dans : *Proceedings of the 1st International Joint Conference on Artificial Intelligence. IJCAI'69*. Washington, DC : Morgan Kaufmann Publishers Inc., p. 509–520.
- OBERTI, Roberto, Massimo MARCHI, Paolo TIRELLI, Aldo CALCANTE, Marcello IRITI, M HOCEVAR et H ULBRICH (2014). « Crops agricultural robot : application to selective spraying of grapevine's diseases' ». Dans : *Proc. RHEA-2014, Madrid, Spain*, p. 21–23.

- OHMAE, Shogo et Javier F. MEDINA (2015). « Climbing fibers encode a temporal-difference prediction error during cerebellar learning in mice ». Dans : *Nat Neurosci* advance online publication.
- O'KEEFE, J. et L. NADEL (1978). *The hippocampus as a cognitive map*. Oxford, United Kingdom : Clarendon Press.
- OUDEYER, Pierre-Yves, Frédéric KAPLAN et Verena V HAFNER (2007). « Intrinsic motivation systems for autonomous mental development ». Dans : *Evolutionary Computation, IEEE Transactions on* 11.2, p. 265–286.
- OUDEYER, Pierre-Yves, Adrien BARANES et Frédéric KAPLAN (2013). « Intrinsically motivated learning of real-world sensorimotor skills with developmental constraints ». Dans : *Intrinsically motivated learning in natural and artificial systems*. Springer, p. 303–365.
- PACKARD, M. G. (oct. 1999). « Glutamate infused posttraining into the hippocampus or caudate-putamen differentially strengthens place and response learning. » Dans : *Proceedings of the National Academy of Sciences of the United States of America* 96.22, p. 12881–12886. ISSN : 0027-8424.
- PACKARD, M. G. et J. L. MCGAUGH (1996). « Inactivation of hippocampus or caudate nucleus with lidocaine differentially affects expression of place and response learning. » Dans : *Neurobiology of Learning and Memory* 65, p. 65–72.
- PAVLOV, Ivan P. (1927). *Conditional Reflex, An Investigation of The Psychological Activity of the Cerebral Cortex*. New York, Oxford University press.
- PENG, Jing et Ronald J. WILLIAMS (1993). « Efficient Learning and Planning Within the Dyna Framework ». Dans : *Adaptive Behavior*, p. 437–454.
- PEZZULO, Giovanni, Francesco RIGOLI et Fabian CHERSI (2013). « The Mixed Instrumental Controller : Using Value of Information to Combine Habitual Choice and Mental Simulation ». Dans : *Frontiers in Psychology* 4.92. ISSN : 1664-1078.
- PIATER, J., S. JODOGNE, R. DETRY, D. KRAFT, N. KRÜGER, O. KROEMER et J. PETERS (fév. 2011). « Learning Visual Representations for Perception-Action Systems ». Dans : *International Journal of Robotics Research* 30.3, p. 294–307.
- PUTERMAN, Martin L. (1994). *Markov Decision Processes : Discrete Stochastic Dynamic Programming*. 1st. New York, NY, USA : John Wiley & Sons, Inc. ISBN : 0471619779.
- QUIGLEY, M., K. CONLEY, B. P. GERKEY, J. FAUST, T. FOOTE, J. LEIBS, R. WHEELER et A. Y. NG (2009). « ROS : an open-source Robot Operating System ». Dans :
- REDISH, A David (1999). « Beyond the cognitive map : from place cells to episodic memory ». Thèse de doct.
- RENAUDO, Erwan, Benoît GIRARD, Raja CHATILA et Mehdi KHAMASSI (2014). « Design of a Control Architecture for Habit Learning in Robots ». Dans : *Biomimetic and Biohybrid Systems, LNAI Proceedings*, p. 249–260.
- RENAUDO, Erwan, Sandra DEVIN, Benoît GIRARD, Raja CHATILA, Rachid ALAMI, Mehdi KHAMASSI et Aurélie CLODIC (2015a). « Learning to Interact with Humans Using Goal-Directed and Habitual Behaviors ». Dans : *RoMan 2015, Workshop on Learning for Human-Robot Collaboration*.
- RENAUDO, Erwan, Benoît GIRARD, Raja CHATILA et Mehdi KHAMASSI (2015b). « Respective advantages and disadvantages of model-based and model-free reinforcement learning in a robotics neuro-inspired cognitive architecture ». Dans : *Biologically Inspired Cognitive Architectures BICA 2015*. Lyon, France, p. 178–184.

- RENAUDO, Erwan, Benoît GIRARD, Raja CHATILA et Mehdi KHAMASSI (2015c). « Which criteria for autonomously shifting between goal-directed and habitual behaviors in robots? » Dans : *5th International Conference on Development and Learning and on Epigenetic Robotics (ICDL-EPIROB)*. Providence, RI, USA, p. 254–260.
- RESCORLA, R. A. et A. W. WAGNER (1972). « A theory of Pavlovian conditioning : Variations in the effectiveness of reinforcement and nonreinforcement ». Dans : *Classical Conditioning II : Current Research and Theory*. Sous la dir. d'A. H. BLACK et W. F. PROKASY. New York : Appleton-Century-Crofts. Chap. 3, p. 64–99.
- RESTLE, Frank (1957). « Discrimination of cues in mazes : A resolution of the "place-vs.-response" question. » Dans : *Psychological review* 64.4, p. 217.
- RIEDMILLER, Martin (2005). « Neural fitted Q iteration – first experiences with a data efficient neural reinforcement learning method ». Dans : *In 16th European Conference on Machine Learning*. Springer, p. 317–328.
- RUMMERY, G. A. et M. NIRANJAN (1994). *On-Line Q-Learning Using Connectionist Systems*. Rap. tech. TR 166. Cambridge, England : Cambridge University Engineering Department.
- RUMMERY, Gavin Adrian (1995). « Problem Solving With Reinforcement Learning ». Thèse de doct. Cambridge University Engineering Department, Cambridge, England.
- RYAN, Richard M et Edward L DECI (2000). « Intrinsic and extrinsic motivations : Classic definitions and new directions ». Dans : *Contemporary educational psychology* 25.1, p. 54–67.
- SCHAAL, S. et C. G. ATKESON (1994). « Robot juggling : An implementation of memory-based learning ». Dans : 1, p. 57–71.
- SCHMIDHUBER, Jürgen (2014). « Deep Learning in Neural Networks : An Overview ». Dans : *CoRR* abs/1404.7828.
- SCHULTZ, W. (1998). « Predictive Reward Signal of Dopamine Neurons. » Dans : *Journal of Neurophysiology* 80, p. 1.
- SCHULTZ, W., Peter DAYAN et Read P. MONTAGUE (1997). « A Neural Substrate of Prediction and Reward ». Dans : *Science* 275, p. 1593–1599.
- SCHULTZ, Wolfram (2013). « Updating dopamine reward signals ». Dans : *Current Opinion in Neurobiology* 23.2. Macrocircuits, p. 229–238. ISSN : 0959-4388.
- SCHWEIGHOFER, Nicolas et Kenji DOYA (2003). « Meta-learning in Reinforcement Learning ». Dans : *Neural Networks* 16.1, p. 5–9.
- SEWELL, Martin (2008). *Ensemble Learning*. Rap. tech.
- SIMÉON, T., J-P. LAUMOND et F. LAMIRAUX (2001). « Move3D : a generic platform for path planning ». Dans : *in 4th Int. Symp. on Assembly and Task Planning*, p. 25–30.
- SINGH, Satinder, Tommi JAAKKOLA, Michael L. LITTMAN et Csaba SZEPESVÁRI (1998). « Convergence Results for Single-Step On-Policy Reinforcement-Learning Algorithms ». Dans : *Machine Learning*, p. 287–308.
- SINGH, Satinder P. (1992). « The Efficient Learning of Multiple Task Sequences ». Dans : *Advances in Neural Information Processing Systems 4*. Morgan Kaufmann, p. 251–258.
- SISBOT, E Akin, Raquel ROS et Rachid ALAMI (2011). « Situation assessment for human-robot interactive object manipulation ». Dans : *RO-MAN, 2011 IEEE*. IEEE, p. 15–20.
- SKINNER, Burrhus Frederic (1938). « The behavior of organisms : An experimental analysis. » Dans :
- SUTTON, R. S. et A. G. BARTO (1987). « A temporal-difference model of classical conditioning ». Dans : *Ninth Annual Conference of the Cognitive Science Society*, p. 355–378.

- (1998). *Introduction to Reinforcement Learning*. 1st. Cambridge, MA, USA : MIT Press. ISBN : 0262193981.
- SUTTON, Richard S. (1990). « Integrated Architectures for Learning, Planning, and Reacting Based on Approximating Dynamic Programming ». Dans : *In Proceedings of the Seventh International Conference on Machine Learning*. Morgan Kaufmann, p. 216–224.
- (1992). « Reinforcement Learning Architectures ». Dans : *Proceedings ISKIT'92 International Symposium on Neural Information Processing*.
- SUTTON, Richard S et Andrew G BARTO (1981). « Toward a modern theory of adaptive networks : Expectation and prediction ». Dans : *Psychological review* 88.2, p. 135.
- SUTTON, R.S., A.G. BARTO et Ronald J. WILLIAMS (1992). « Reinforcement learning is direct adaptive optimal control ». Dans : *Control Systems, IEEE* 12.2, p. 19–22.
- THORN, Catherine A., Hisham ATALLAH, Mark HOWE et Ann M. GRAYBIEL (2010). « Differential Dynamics of Activity Changes in Dorsolateral and Dorsomedial Striatal Loops during Learning ». Dans : *Neuron* 66.5, p. 781–795. ISSN : 0896-6273.
- THORNDIKE, Edward Lee (1911). « Animal intelligence : Experimental studies. » Dans :
- TOKIC, Michel (2010). « Adaptive ϵ -greedy exploration in reinforcement learning based on value differences ». Dans : *KI 2010 : Advances in Artificial Intelligence*. Springer Berlin Heidelberg, p. 203–210.
- TOLMAN, Edward C. (1948). « Cognitive maps in rats and men ». Dans : *Psychological Review* 55, p. 189–208.
- TOPALIDOU, Meropi, Daisuke KASE, Thomas BORAUD et Nicolas ROUGIER (2015). « The formation of habits : a computational model mixing reinforcement and Hebbian learning ». Dans : *The Multi-disciplinary Conference on Reinforcement Learning and Decision Making (RLDM 2015)*. Edmonton, Canada.
- TRICOMI, Elizabeth, Bernard W BALLEINE et John P O'DOHERTY (2009). « A specific role for posterior dorsolateral striatum in human habit learning ». Dans : *European Journal of Neuroscience* 29.11, p. 2225–2232.
- VALENTIN, Vivian V, Anthony DICKINSON et John P O'DOHERTY (2007). « Determining the neural substrates of goal-directed learning in the human brain ». Dans : *The Journal of neuroscience* 27.15, p. 4019–4026.
- VAN HASSELT, Hado (2011). « Insights in Reinforcement Learning : formal analysis and empirical evaluation of temporal-difference learning algorithms ». Thèse de doct. Universiteit Utrecht.
- VERNON, D., G. METTA et G. SANDINI (2007a). « A Survey of Artificial Cognitive Systems : Implications for the Autonomous Development of Mental Capabilities in Computational Agents ». Dans : *Evolutionary Computation, IEEE Transactions on* 11.2, p. 151–180. ISSN : 1089-778X.
- (2007b). « The iCub cognitive architecture : Interactive development in a humanoid robot ». Dans : *2007 IEEE 6th International Conference on Development and Learning*, p. 122–127.
- VIEJO, Guillaume, Mehdi KHAMASSI, Andrea BROVELLI et Benoît GIRARD (2015). « Modelling choice and reaction time during arbitrary visuomotor learning through the coordination of adaptive working memory and reinforcement learning ». Dans : *Frontiers in Behavioral Neuroscience* 9.225. ISSN : 1662-5153.
- VOLPE, Richard, Issa A. D. NESNAS, Tara ESTLIN, Darren MUTZ, Richard PETRAS et Hari DAS (2000). *CLARAty : Coupled Layer Architecture for Robotic Autonomy*. Rap. tech. NASA - JET PROPULSION LABORATORY.

- VOLPE, Richard, Issa NESNAS, Tara ESTLIN, Darren MUTZ, Richard PETRAS et Hari DAS (2001). « The CLARAty architecture for robotic autonomy ». Dans : *Aerospace Conference, 2001, IEEE Proceedings*. T. 1. IEEE, p. 1–121.
- WATKINS, Christopher J. C. H. et Peter DAYAN (mai 1992). « Technical Note : Q-Learning ». Dans : *Mach. Learn.* 8.3-4, p. 279–292. ISSN : 0885-6125.
- WATKINS, Christopher John Cornish Hellaby (1989). « Learning from Delayed Rewards ». Thèse de doct. Cambridge, UK : King’s College.
- WIERING, M. et M. van OTTERLO, éd. (2012). *Reinforcement Learning : State-of-the-Art*. Springer.
- WIERING, Marco et Hado VAN HASSELT (2007). « Two novel on-policy reinforcement learning algorithms based on TD (λ)-methods ». Dans : *Approximate Dynamic Programming and Reinforcement Learning, 2007. ADPRL 2007. IEEE International Symposium on*. IEEE, p. 280–287.
- WIERING, Marco A. et Hado VAN HASSELT (2008). « Ensemble Algorithms in Reinforcement Learning. » Dans : *IEEE Transactions on Systems, Man, and Cybernetics, Part B* 38.4, p. 930–936.
- WISE, Roy A (2004). « Dopamine, learning and motivation ». Dans : *Nature reviews neuroscience* 5.6, p. 483–494.
- WITTEN, Ian H. (1977). « An adaptive optimal controller for discrete-time Markov environments ». Dans : *Information and Control* 34.4, p. 286–295.
- WOLPERT, D.M. et M. KAWATO (1998). « Multiple paired forward and inverse models for motor control ». Dans : *Neural Networks* 11.7–8, p. 1317–1329. ISSN : 0893-6080.
- WOODEN, D., M. MALCHANO, K. BLANKESPOOR, A. HOWARDY, A. A. RIZZI et M. RAIBERT (2010). « Autonomous navigation for BigDog ». Dans : *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, p. 4736–4741.
- WUNDERLICH, Klaus, Peter DAYAN et Raymond J DOLAN (2012). « Mapping value based planning and extensively trained choice in the human brain ». Dans : *Nature neuroscience* 15.5, p. 786–791.
- YASUDA, Toshiyuki et Kazuhiro OHKURA (2008). « From Animals to Animats 10 : 10th International Conference on Simulation of Adaptive Behavior, SAB 2008, Osaka, Japan, July 7-12, 2008. Proceedings ». Dans : sous la dir. de Minoru ASADA, John C. T. HALLAM, Jean-Arcady MEYER et Jun TANI. Berlin, Heidelberg : Springer Berlin Heidelberg. Chap. A Reinforcement Learning Technique with an Adaptive Action Generator for a Multi-robot System, p. 250–259.
- YIN, Henry H. et Barbara J. KNOWLTON (2006). « The role of the basal ganglia in habit formation ». Dans : *Nature Reviews Neuroscience* 7.6, p. 464–476.
- YIN, Henry H., Shweta Prasad MULCARE, Monica R.F. HILÁRIO, Emily CLOUSE, Terrell HOLLOWAY, Margaret I. DAVIS, Anita C. HANSSON, David M. LOVINGER et Rui M. COSTA (2009). « Dynamic reorganization of striatal circuits during the acquisition and consolidation of a skill ». Dans : *Nature Neuroscience* 12.3, p. 333–341.
- ZILBERSTEIN, Shlomo (1993). « Operational Rationality through Compilation of Anytime Algorithms ». Thèse de doct. Berkeley, CA, USA : University of California at Berkeley.

Glossaire

A	ACLA	Algorithme Actor Critic Learning Automaton
	Acteur-Critique	Méthode Acteur-Critique
	AR	Apprentissage par Renforcement
D	dirigé vers un but	Comportement pour expert dirigé vers un but
	Dyna	Algorithmes de type Dyna
	Dyna-Q	Algorithme Dyna-Q
H	habituel	Comportement ou expert habituel
P	PI	Policy Iteration
Q	Q-learning	Algorithme Q-learning
	QV-learning	Algorithme QV-learning
R	ROS	Robot Operatng System
S	SARSA	Algorithme State-Action-Reward-State-Action
T	TD	Algorithmes aux différences temporelles (de l'anglais <i>Temporal-Difference learning</i>)
V	VI	Value Iteration
	VTE	Vicarious Trial and Error

Index

A	Dyna-Q..... 35, 36	R
ACLA..... 39	H	ROS..... 70, 109
Acteur-Critique..... 35, 39	habituel . 15, 48, 50–52, 54,	S
AR .. 15, 25, 26, 39, 41, 42,	56, 60–62, 64, 70, 72, 74–79,	SARSA..... 34, 39
47, 48, 50–54, 57, 61, 64, 65,	81, 82, 84, 87, 89, 91–94, 97,	
72, 78, 93, 94, 98, 100, 106,	98, 100, 102, 104, 106–109,	T
108, 124	112–116, 118–120, 123	TD..... 33, 36, 48, 56
D	P	V
dirigé vers un but .. 15, 48,	PI..... 31, 32	VI..... 31, 32, 51, 66, 112
50–52, 54, 60–62, 65–67, 69,	Q	VTE..... 46
70, 72, 74–80, 82, 84, 87,	Q-learning... 39, 51–53, 64,	
89–94, 97, 98, 102, 104, 108,	65, 94, 98, 100	
112–114, 116, 118, 120, 122,	QV-learning..... 39	
123, 125, 126		
Dyna..... 36, 37		

