Architecture cognitive générique pour la coordination de stratégies d'apprentissage en robotique

Rémi Dromnelle

Sorbonne Université
Institut des Systèmes Intelligents et de Robotique

Thèse

présentée en première version en vu d'obtenir le grade de Docteur,
spécialité « Informatique »

par

Dromnelle Rémi

Architecture cognitive générique pour la coordination de stratégies d'apprentissage en robotique.

Thèse soutenue le 01/07/2021 devant le jury composé de :

M.	Nicolas Rougier	LaBRI, INRIA Bordeaux	(Rapporteur)
M.	Olivier Simonin	INSA de Lyon / INRIA	(Rapporteur)
Mme	Aurelie Clodic	LAAS, CNRS, Toulouse	(Examinatrice)
Mme	Céline Teuliere	Institut Pascal, CNRS	(Examinatrice)
M.	Mohamed Chetouani	ISIR, CNRS, Paris	(Examinateur)
M.	Mehdi Khamassi	ISIR, CNRS, Paris	(Co-directeur de thèse)
M.	Raja Chatila	ISIR, CNRS, Paris	(Co-directeur de thèse)
M.	Benoit Girard	ISIR, CNRS, Paris	(Invité)

À mes grands-pères.

Résumé

L'objectif principal de cette thèse est de proposer une nouvelle méthode d'adaptation en ligne de l'apprentissage robotique, permettant aux robots d'adapter dynamiquement et de manière autonome leur comportement en fonction des variations de leur propre performance. Cette performance est elle-même dépendante de la non-stationarité (variations de structures, volatilité de la récompense) de l'environnement (qu'il soit social ou non social). La méthode élaborée est suffisamment générale et tâche-indépendante pour qu'un robot l'utilisant puisse effectuer différentes tâches dynamiques de nature variée sans ajustement des algorithmes ou des paramètres par le programmeur/concepteur. Les algorithmes qui sous-tendent cette méthode ont été élaborés à partir d'une architecture cognitive robotique précédemment conçue à l'ISIR (Renaudo 2016). Ils consistent en un système de méta-contrôle permettant au robot de faire appel à deux experts décisionnels suivant une stratégie comportementale différente. L'expert MB, qui suit une stratégie dite model-based, construit un modèle des effets des actions à long-terme et utilise ce modèle pour décider ; cette stratégie est coûteuse en termes de ressources calculatoires, mais converge rapidement vers la solution. L'expert MF suit quant à lui une stratégie dite model-free, sans modèle, peu coûteuse en termes de ressources calculatoires, mais mettant du temps à converger vers la solution optimale. Ici, l'inspiration vient des neurosciences et de la modélisation du fonctionnement du cerveau, et plus particulièrement de la mise en évidence de différentes catégories de stratégies chez l'animal et de corrélations avec l'activité de certains substrats neuronaux (Daw et al. 2005). Sur la base de ces travaux, nous avons élaboré un nouveau critère de coordination des experts MB et MF permettant au robot de changer dynamiquement de stratégie au cours des tâches qu'il réalise. La méthode consiste à calculer pour chacun des experts décisionnels une valeur de compromis entre la qualité de l'apprentissage, que l'on cherche à maximiser, et le coût de l'inférence, que l'on cherche à minimiser. Les valeurs de compromis des deux experts sont ensuite comparées et une stratégie comportementale est choisie pour guider le comportement du robot. Pour éprouver notre architecture robotique et notre méthode de coordination de comportements, nous avons élaboré trois tâches expérimentales différentes :

- une tâche de navigation dans un environnement réel qui est amené au cours de la tâche à changer (ajouts d'obstacles, changements d'objectifs) (Dromnelle et al. 2020b),
- une tâche simulée de rangement d'objets, où le robot peut potentiellement être aidé par un humain capable d'interagir avec lui de plusieurs manières différentes non connues à l'avance (Dromnelle et al. 2020a),
- une tâche simulée de rangement d'objets, où le robot doit coopérer activement avec un humain pour réaliser l'objectif assigné, et où l'objectif peut changer en cours de tâche.

Nous montrons dans ce travail que notre méthode de coordination de comportements permet au robot de maintenir une performance optimale en termes de performance et de calcul par rapport à un expert MB seul qui

atteint la meilleure performance mais en ayant un grand coût en calcul. Nous montrons aussi que la méthode permet de faire face à des changements brusques de l'environnement, des changements d'objectifs ou de comportements du partenaire humain dans le cas des tâches d'interaction. Les robots ayant réalisé ces expériences, qu'ils soient réels ou virtuels, ont tous utilisé le même jeu de paramètres, montrant ainsi la généralité de notre méthode.

Remerciements

Ce travail a été financé par une bourse de la Délégation Générale de l'Armement, par le Labex SMART ainsi que par le projet européen HumanE AI Network.

J'aimerais tout d'abord remercier les membres de jury de m'avoir honoré de leur présence. Merci à Nicolas Rougier et Olivier Simonin d'avoir pris le temps de remettre leurs rapports. Merci à Aurelie Clodic, Céline Teuliere et Mohamed Chetouani d'avoir accepté de faire partie du jury malgré leurs nombreuses charges de travail.

Je remercie mes encadrants Benoit Girard, Mehdi Khamassi et Raja Chatila. Merci pour votre confiance. Merci d'avoir cru en moi et d'avoir su valoriser mon travail. Merci de m'avoir aidé à naviguer dans le vaste océan de la Recherche afin que je puisse trouver mon îlot. En vérité, je me suis souvent demandé ce que je faisais là, un peu paumé sur cette jolie plage. Mais rapidement, l'îlot rocheux s'est transformé en île luxuriante. Merci à Benoit pour son énergie communicative et son franc-parlé malicieux. Merci à Mehdi pour son optimisme déraisonnable et sa bonne humeur contagieuse. Merci à Raja et ses conseils mûris, fruits de son savoir arborescent. Je n'oublie bien sûr pas Erwan, à qui j'ai succédé, et qui a toujours su être présent depuis l'Autriche pour répondre à mes questions.

Je remercie les doctorants, les post-doc, les stagiaires, et toutes les personnes de l'équipe AMAC (et plus encore) que j'ai pu côtoyer au laboratoire ces dernières années. Merci à Paul et Mathieu, qui m'ont aidé à injecter un peu de sciences-cognitives dans cet environnement d'ingénieurs, et avec qui j'ai avancé jour après jour, crêpe après crêpe, pepsi après coca. Merci à Pierre. Pour son bureau, déjà, sur lequel je me suis jeté après son départ. Mais surtout pour les discussions animées et les séances d'escalade matinales, qui se transformaient systématiquement en escapades intersidérales. Encore aujourd'hui, nous avons les yeux plein d'étoiles. Merci à Alex et David pour les discussions passionnantes et les quelques parties de jeux de rôle. Quel plaisir ce fut d'apprendre que je n'étais pas le seul rôliste de l'équipe ! D'ailleurs, merci à Alex de m'avoir fait découvrir les GN virtuels qui m'ont bien occupé durant ces semaines confinées. Plus généralement, merci à Maud, Elisa, Giuseppe, Ahmed, Alexandre, Elias et Nicolas qui ont définitivement su reprendre le flambeau malgré les complications actuelles. En vrac, je remercie aussi Astrid, Stépagne Gourichon, Leni, François, Marwen, Oussama et toutes les personnes que j'ai eu la chance de côtoyer ces dernières années.

Bien entendu je remercie mes ami.e.s. Léa et Romain, particulièrement, avec qui j'ai cohabité à Cachan durant de longs mois. Cette coloc de doctorants était aussi chouette que motivante. Je pense qu'assister à vos longues semaines de rédaction a aidé à m'y prendre plus tôt, afin de vivre plus se-

reinement cette période. Merci à Théo d'avoir repris le flambeau et d'avoir transformé notre appartement en véritable ludothèque. Je remercie plus généralement la Crowmunauté, de Montpelier à Boston, en passant par Bordeaux et Paris. Aucune frontière ne semble pouvoir nous séparer. Je pourrais vous citer un à une en évoquant une anecdote personnelle, mais il n'est pas d'usage que les remerciements excèdent dix pages. Merci aussi aux nombreuses personnes que j'ai rencontrées ces derniers mois, et qui m'ont aidé à traverser cette année compliquée. Une pensée particulière à Anaïs et à nos nombreux échanges. Dédicace attendrie au Champêtre Crew. Et merci à Émeline pour ces belles années d'amour.

Il serait mal venu d'oublier ceux qui n'existent pas réellement, et qui pourtant, ont été très présents pour moi ces dernières années. Alors merci à Alan, Xavier, Max, Florence, Betrand, William, Victor et Al'Boubakour. Techniquement parlant, je suis responsable de la mort de chacun d'entre vous (d'un, particulièrement). Désolé pour ça. Mais c'était là la seule manière de vous faire découvrir les ténèbres de ce monde. J'espère que l'immortalité acquise suffira à mon pardon. Merci à François de les avoir pris sous son aile, et d'avoir toujours su les considérer avec confiance, respect et bienveillance (ou pas). Merci aussi à Pü, qui situé à quelques années-lumière de cette réalité, a vu son destin être tragiquement bouleversé par ma faute. Sache que si les épreuves ne font que commencer, et que ton futur s'annonce aussi pénible que douloureux, tu auras malgré tout l'occasion de goûter aux Jours Heureux. Tu retrouveras une famille, découvriras l'amour et arracheras ta liberté.

Enfin, je souhaite remercier les membres de ma famille. Merci à ma mère Dolores et à mon père Thierry, sans qui je ne serais jamais arrivé là. Si cela est techniquement très vrai, je fais ici avant tout allusion à mon éducation. Je me suis toujours senti particulièrement libre et détendu dans mes études (parfois un peu trop), et cela grâce à eux. Merci à Andy et Stéphanie d'être entrés dans ma vie, et à cette dernière de m'avoir offert un petit frère. Merci à Malo donc. Quel plaisir de te voir grandir, quel plaisir de me voir en toi, par moment. Je suis très pressé de connaître la suite de ton histoire. Merci à ma grande-soeur Marie, une indéfectible alliée de vie. Notre impérissable attachement est réconfortant. Merci à mes grands-mères Marie-Claire et Isabelle, que j'aime particulièrement fort, et à mes grands-pères Michel et Gabriel, à qui ce manuscrit est dédié. Merci à mes tantes Véroniques et Sophie, merci à mes cousins et cousines Arthur, Justine et Lucie. Finalement, merci à mon oncle Didier, qui m'a, probablement sans le savoir, motivé en partie à faire un doctorat.

À Cachan, le 16 juin 2021.

Table des matières

Liste des figures

Introduction

1

Sommaire

1.1 Contexte général

Les organismes vivants évoluent dans des milieux complexes, ouverts et incertains. Pour qu'un individu puisse se maintenir et se reproduire dans de tels environnements, il doit nécessairement être capable de mettre en place des mécanismes de régulation lui permettant de s'adapter de manière flexible aux contingences extérieures. Il est intéressant d'observer que chez les organismes complexes, de tels mécanismes de régulation sont présents à la fois à l'échelle moléculaire (allostérie), métabolique (cycle de Krebs), cellulaire (régulation des quatre phases du cycle cellulaire), motrice (copie efférente) et cognitive (plasticité neuronale, métacognition). Dans les champs de l'ingénierie et de la recherche, les capacités d'auto-régulation du monde vivant ont de tout temps représenté un modèle d'inspiration pour le développement d'agents artificiels toujours plus autonomes (Pfeifer et al. 2007, Guillot et Meyer 2008).

En robotique notamment, que de chemin parcouru depuis le début du 20^e siècle, où *Electric dog* (Missner 1912), un robot capable de se diriger vers des sources lumineuses, fût développé en s'inspirant du phototropisme des papillons de nuit. Aujourd'hui, *BigDog*, son lointain cousin, affiche des capacités d'équilibrioception impressionnantes, lui permettant de se déplacer en forêt, sur des monticules de gravats, ou encore sur du verglas (Raibert et al. 2008). *Spot*, son successeur à la couleur jaune, a rejoint il y a quelques mois l'équipement de laboratoires de recherche et d'industriels. Plus léger, moins énergivore, et capable d'être associé à divers modules tel qu'un bras robotique, il s'est notamment illustré en effectuant des inspections sur des plates-formes pétrolières dans la Mer du Nord, ou en dansant avec son camarade *Atlas* sur le tube "Do You Love Me" de *The Contours*. Désormais, *Spot* s'est rendu accessible au grand public, ainsi qu'à la police de New-York, où le robot a servi d'éclaireur durant une prise

d'otage, à l'école militaire française de Saint-Cyr, dans le cadre d'exercices d'entraînement, ou encore à la ville de Singapour, afin de surveiller la distanciation physique des usagers d'un parc en cette période de pandémie. Oui, sans surprise, la robotique autonome intéresse beaucoup les armées et les États du monde. Dans le spot publicitaire du lancement de *Spot*, les mentions "inspect" et "protect" sont d'ailleurs mises en avant. Pourtant, les robots peuvent s'illustrer dans d'autres domaines profitables à l'humanité et nécessitant une grande autonomie, telle que l'exploration de ruines ou de zones radioactives après une catastrophe, où des êtres humains attendraient d'être sauvés, ou encore l'exploration d'endroits inaccessibles à l'homme, tels que les abysses ou Mars (Wong et al. 2018).

En terme d'équilibrioception, *Spot* est donc actuellement ce qui se fait de mieux dans le domaine de la robotique. Mais si ce robot possède effectivement une multitude de capteurs et de moteurs lui permettant d'exhiber des capacités **réactives** de bas-niveau exceptionnelles, il n'est en revanche pas doté de capacités **délibératives** de haut-niveau. En effet, il ne sait pas prendre de décision, au sens où il n'est pas capable d'effectuer un choix entre plusieurs solutions susceptibles de résoudre un problème auquel il serait confronté. Pour évoluer dans son environnement, *Spot* est soit piloté à distance, soit programmé pour suivre une cible ou parcourir un itinéraire routinier. En cela, il est bien peu autonome.

Or, progresser vers l'autonomie comportementale d'un robot nécessite de le doter de capacités de délibération adaptative, c'est-à-dire des capacités lui permettant d'agir correctement dans des situations qu'il connaît déjà, mais également de trouver automatiquement des solutions dans des situations nouvelles. Pour ce faire, le robot doit être capable d'intégrer ses expériences quotidiennes pour pouvoir progresser et apprendre à changer dynamiquement de stratégies comportementales lorsque nécessaire. Par stratégies comportementales, nous entendons par exemple la capacité de construire une carte cognitive (un modèle mental de l'environnement) et de l'utiliser pour planifier des déplacements, ou la capacité d'associer des directions de déplacement à des amers visuels reconnus autour de soi.

1.2 Problématique

Dans ce travail de recherche, nous adressons donc la question de la délibération adaptative en robotique, au travers de l'exploitation des connaissances acquises précédemment et de la coordination en ligne de stratégies comportementales. Pour cela, nous prenons inspiration de l'étude du comportement animal, qui montre que les animaux sont capables, à force d'apprentissage, de changer de stratégies comportementales en fonction du contexte environnemental (modification de la structure de l'environnement, changement des zones d'intérêt, etc.). D'un point de vue évolutif, être capable d'alterner dynamiquement entre plusieurs types de comportements, c'est pouvoir profiter des avantages de chacun d'entre eux au moment le plus opportun. C'est aussi pouvoir économiser des ressources énergétiques, dans le cas où certains de ces comportements demanderaient des calculs mentaux plus complexes. Or, la problématique de l'économie de ressources énergétiques est centrale pour un animal, celle-ci étant intimement liée à sa survie.

Parmi les études cherchant à modéliser le comportement animal, nombreuses sont celles qui s'appuient sur la dichotomie entre les *habitudes comportementales*, des comportements causés par un stimulus perceptuel et ne requérant aucune estimation anticipative de l'effet du comportement, et les *comportements dirigés vers un but*, des comportements impliquant une délibération active sur les conséquences attendues du comportement, et caractérisés par leur grande flexibilité (Skinner 2019, Dickinson 1985). Cette dichotomie n'est pas sans rappeler celle du psychologue et économiste Daniel Kahneman, qui oppose deux systèmes de la prise de décision chez l'humain : un premier système rapide, intuitif et émotionnel, et un second système plus lent, plus réfléchi, plus contrôlé et plus logique (Kahneman 2012). Nous pourrions aussi faire allusion à la dichotomie séculaire entre inconscient et conscient, mais ici, c'est avant tout de l'étude du comportement animal dont nous tirons notre inspiration.

À partir de ces connaissances et des travaux de modélisation effectués ces dernières années (Daw et al. 2005, Renaudo 2016), nous proposons d'étudier une architecture robotique générique dont la couche décisionnelle est capable d'apprendre des *habitudes comportementales*, d'apprendre et de planifier des *comportements dirigés vers un but*, et de coordonner dynamiquement ces deux types de comportements. Plus spécifiquement, nous cherchons à répondre à deux questions principales :

— Un tel mécanisme d'apprentissage et de méta-contrôle a-t-il un intérêt pour la robotique, et permet-il, par exemple, d'éviter de consommer trop de ressources calculatoires à planifier lorsque les *habitudes comportementales* sont suffisamment efficaces ?
— Un tel modèle de méta-contrôle permet-il de résoudre des tâches de nature et de complexité variées, et si oui, quelles sont ses limites ?

1.3 Plan de la thèse

Ce manuscrit est composé de neuf chapitres. Les quatre premiers dressent un état de l'art des domaines de recherche dans lesquels notre travail puise son inspiration. Les quatre suivants présentent notre contribution méthodologique et expérimentale. Le dernier discute notre contribution et propose des perspectives.

Dans le second chapitre, nous introduisons l'apprentissage par renforcement, le type d'apprentissage automatique au cœur des modèles présentés dans ce manuscrit, dont celui que nous avons conceptualisé. Dans le troisième, nous faisons un détour par le domaine de l'apprentissage interactif, et présentons des méthodes permettant à un robot faisant de l'apprentissage par renforcement d'intégrer les retours évaluatifs d'un enseignant humain. Dans le chapitre qui suit, nous nous intéressons au comportement des mammifères et montrons que la dichotomie introduite précédemment entre les *habitudes comportementales* et les *comportements dirigés vers un but* présente des analogies avec différents types d'apprentissage par renforcement, à l'origine de travaux de modélisation dont nous nous inspirons pour répondre à notre problématique. Nous terminons l'état de l'art avec le chapitre cinq en nous intéressant aux travaux passés ayant permis le développement d'outils de planification automatique en robotique,

avant de décrire l'architecture que nous avons développé sur la base des connaissances présentées précédemment.

Dans le sixième chapitre, nous évaluons notre architecture robotique dans une tâche de navigation simulée et réelle non-stationnaire (environnement probabiliste, ajouts d'obstacles, changement de la zone récompensée en cours de tâche), afin de répondre à la première des deux questions de notre problématique. Nous continuons cette évaluation dans le chapitre suivant, avec cette fois-ci une tâche simulée d'interaction humain-robot, où le robot interagit avec un humain au comportement plus ou moins profitable à la résolution de son objectif (mettre des cubes dans des boîtes). Ici, nous confrontons le robot à un nouvel environnement complexe et changeant, afin de confirmer les observations faites dans l'expérience de navigation, et afin de vérifier le caractère générique de notre architecture. Enfin, le chapitre huit nous permet de confronter notre architecture à une tâche simulée de coopération humain-robot, où le robot doit nécessairement collaborer avec son partenaire afin de progresser dans la résolution de la tâche. Dans ce chapitre, le fait que le partenaire humain puisse rapidement se muer en adversaire nous a permis de mettre le doigt sur une limite de notre modèle de coordination, à laquelle nous avons apporté une solution.

Nous clôturerons ce manuscrit en discutant des différents choix que nous avons faits au cours de ce travail de recherche, et des pistes à explorer pour améliorer l'architecture robotique et son modèle de coordination.

Apprentissage automatique

2

Sommaire

Dans ce chapitre, nous nous intéressons à l'apprentissage automatique comme d'un outil pour la prise de décision d'agents artificiels. Si différents formalismes existent pour l'apprentissage automatique en robotique, tels que les approches évolutionnaires (Doncieux et al. 2015) ou les systèmes de classification d'apprentissage (Urbanowicz et Moore 2009), nous mettons ici l'emphase sur l'apprentissage par renforcement (Sutton et Barto 1998, Kober et al. 2013), qui est à la base de la plupart des modèles du comportement opérant chez l'animal (Sutton 1992). Les problèmes abordés par ces formalismes diffèrent des problèmes d'apprentissage supervisé du fait qu'ici, personne ne donne à l'agent les bonnes réponses concernant ce qu'il devrait faire dans chaque situation. Au lieu de ça, l'agent doit optimiser son comportement de manière autonome en fonction d'un critère de performance. La principale différence entre ces différents formalismes réside justement dans la définition de la fonction

d'évaluation utilisée pour calculer la performance, et dans les mécanismes utilisés pour optimiser le comportement. Dans ce travail, nous considérons donc le cadre de l'apprentissage par renforcement, qui se situe à michemin entre l'apprentissage supervisé et l'apprentissage non-supervisé. Dans un premier temps, nous commençons par définir le formalisme sur lequel l'apprentissage par renforcement s'appuie afin de présenter dans un second temps une revue des méthodes associées. Pour finir, nous étendons notre état de l'art à l'apprentissage par renforcement ensembliste en mettant l'accent sur les méthodes de fusion. À noter que si nous nous intéressons dans ce chapitre à l'application de l'apprentissage par renforcement à la robotique, nous considérons plutôt l'apprentissage par renforcement comme un outil permettant de résoudre des problèmes de prise décision symboliques de haut niveau. D'autres études s'intéressent à l'application de l'apprentissage par renforcement via des méthodes continues, en se concentrant non pas sur le système de prise de décision du robot, mais sur des problèmes de plus bas-niveau tels que le contrôle des moteurs d'un bras mécanique (Kober et al. 2013). Dans ce travail de recherche, l'enjeu est avant tout d'améliorer l'autonomie décisionnelle des robots. Nous montrons dans ce chapitre en quoi l'apprentissage par renforcement nous semble être un outil adapté pour atteindre cet objectif, un outil que nous aurons l'occasion d'évaluer sur des problèmes réels plus loin dans ce manuscrit, grâce à l'architecture de contrôle robotique que nous avons développée.

2.1 Définitions

2.1.1 Types d'apprentissage automatique

Les algorithmes d'apprentissage automatique (connu sous l'appellation *machine learning* en anglais) sont communément catégorisés en cinq grandes familles d'algorithmes : l'apprentissage supervisé, l'apprentissage non-supervisé, l'apprentissage semi-supervisé et l'apprentissage par renforcement.

- Dans l'apprentissage supervisé, des classes sont prédéfinies et des exemples sont étiquetés par l'expérimentateur. L'objectif de l'algorithme est d'apprendre à classer automatiquement des données en suivant la classification définie. On distingue généralement deux phases : une première dite "d'apprentissage", où l'algorithme va établir un modèle à partir des données étiquetées, et une seconde dite "de test", où l'algorithme va apprendre à prédire l'étiquette d'une nouvelle donnée à partir du modèle crée.
- Dans l'apprentissage non-supervisé, seuls des exemples choisis par l'expérimentateur sont fournis à l'algorithme, mais sans aucune étiquette associée. L'objectif de l'algorithme est d'apprendre à découvrir la structure des données en les classant selon les attributs dont il dispose, afin de les regrouper en groupes homogènes. Ici, les étiquettes sont définies par l'algorithme. Les algorithmes d'apprentissage non supervisé utilisent généralement des outils de calcul de similarité pour établir leur classification.
- Dans l'apprentissage semi-supervisé, l'algorithme à accès à des

données étiquetées et non étiquetées. Utiliser des données non-étiquetées en combinaison avec des données étiquetées permet d'améliorer significativement la qualité de l'apprentissage. De plus, lorsque les jeux de données sont grands, étiqueter chacune des données devient rapidement fastidieux pour l'expérimentateur. L'apprentissage semi-supervisé revêt donc un intérêt pratique évident.

— Dans l'apprentissage par transfert, l'algorithme est capable de reconnaître des connaissances apprises à partir de tâches sources et de les restituer dans la tâche cible afin d'améliorer son apprentissage. Tout l'enjeu de l'apprentissage par transfert est de savoir comment identifier les similitudes entre les tâches sources et cibles et de comment transférer les connaissances associées.

— Dans l'apprentissage par renforcement (abrégé AR par la suite), l'algorithme a accès seulement à une indication de l'intérêt de sa réponse, donnée sous forme d'un scalaire par une fonction de récompense. Son objectif est d'apprendre un comportement étant donnée une observation en interagissant avec l'environnement. L'AR se situe à mi-chemin entre l'apprentissage supervisé et l'apprentissage non-supervisé : certes, l'algorithme ne connaît pas la réponse à apporter dans une certaine situation comme dans l'apprentissage supervisé, mais en interagissant avec son environnement par essai-erreur, il obtient tout de même une indication sur l'intérêt de la réponse qu'il vient de fournir. Dans ce manuscrit, nous nous intéressons exclusivement à l'apprentissage par renforcement.

2.1.2 Processus de Décision Markoviens

Un Processus de Décision Markovien (abrégé MDP par la suite) est un modèle stochastique permettant de modéliser la prise de décision d'un agent dans des situations où les résultats obtenus sont en partie aléatoires (Howard 1960). On appelle un *agent* une entité autonome capable de percevoir son environnement grâce à des capteurs et d'agir sur celui-ci via des effecteurs afin de réaliser des buts. Classiquement, lorsque l'on souhaite étudier des problèmes d'optimisation à l'aide d'un algorithme d'AR, on modélise ces problèmes sous la forme d'un MDP (Sigaud et Buffet 2013), c'est-à-dire à l'aide du tuple <S,A,T,R> où :

— S représente l'ensemble des états $s \in S$ qui définissent l'environnement et donc toutes les situations dans lesquelles l'agent peut se trouver,

— A représente l'ensemble des actions $a \in A$ que l'agent peut réaliser dans l'environnement,

— $T : S \times A \times S \rightarrow [0, 1]$ est une fonction de transition de probabilité qui décrit la probabilité qu'a l'agent d'arriver dans l'état s' lorsqu'il réalise l'action a dans l'état s. Cette fonction représente l'effet des actions de l'agent dans l'environnement.

— $R : S \times A \rightarrow \mathbb{R}$ est une fonction de récompense qui indique la récompense que l'agent obtient lorsqu'il effectue une action a dans un état s. Cette fonction représente le but que l'agent doit atteindre, où les transitions désirables sont associées à des récompenses positives, et les transitions indésirables à des récompenses négatives.

Finalement, le comportement de l'agent est représenté par une politique notée π définie comme $\pi : S \rightarrow A$ si la politique est déterministe ou $\pi : S \times A \rightarrow [0, 1]$ dans le cas stochastique. Généralement, on suppose que la *Propriété de Markov* est respectée lorsque dans un état $s \in S$, l'agent possède toute l'information lui permettant de choisir l'action $a \in A$ qui lui permettra de résoudre le problème (i.e., accumuler des récompenses au cours du temps). La plupart du temps, l'environnement est considéré comme stationnaire, c'est-à-dire que les fonctions T et R ne changent pas durant le problème, ce qui facilite sa résolution. Pourtant, dans la réalité, l'environnement peut évoluer au cours du temps (modification de la fonction T), et son objectif peut être mis à jour (modification de la fonction R). Notre travail de recherche s'intéresse à des problèmes mettant en scènes des environnements non-stationnaires, comme nous le verrons dans les chapitres suivants.

Les MDPs sont une extension des chaînes de Markov. La différence étant l'addition des actions choisies par l'agent et des récompenses gagnées par celui-ci. S'il n'y a qu'une seule action à tirer dans chaque état et que les récompenses sont égales, le processus de décision markovien devient alors une chaîne de Markov. Généralement, le terme « chaîne de Markov » désigne les processus de Markov à temps et à espace d'états S discret. Dans ce manuscrit, nous nous placerons exclusivement dans le cas discret.

2.1.3 Fonctions de valeur et politique optimale

La politique d'un agent décrit les choix des actions que celui-ci joue dans chacun des états. Cette politique est définie selon le critère d'intérêt que l'agent cherche à maximiser, qui est lié à sa fonction de récompense R, puisque l'agent cherche à accumuler des récompenses au cours du temps. Le plus souvent, le critère choisi est le critère de récompense actualisé :

$$R_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i+1} \tag{2.1}$$

La qualité d'une politique est souvent mesurée par la quantité de récompenses que l'agent est capable d'accumuler au cours du temps. Pour accumuler des récompenses, il est important que l'agent ne prenne pas seulement en compte la récompense instantanée r_t qu'il reçoit à un instant t, mais aussi les récompenses futures qu'il s'attend à recevoir plus tard. Ainsi, il sera capable de faire la différence entre deux décisions apportant immédiatement une unité de récompense, mais dont l'une permet d'obtenir plus tard plusieurs unités de récompenses supplémentaires, et dont l'autre emmène indubitablement vers l'obtention de malus (récompense négative). Le facteur d'actualisation $\gamma \in [0, 1]$ nous permet de prendre en compte cet horizon temporel et représente le taux d'oubli des récompenses futures.

Lorsqu'un critère d'arrêt et une politique sont définis, la quantité de récompenses cumulées que l'agent espère recevoir lorsqu'il suit la politique π depuis un état s, aussi appelée *valeur d'état*, peut être calculée via

la fonction de valeur des états $V^{\pi}(s)$ (Sutton et Barto 1998) :

$$V^{\pi}(s) = \sum_{s' \in S} T(s, \pi(s), s')(R(s, \pi(s), s') + \gamma V^{\pi}(s')) \tag{2.2}$$

Avec cette expression, nous comprenons que la valeur d'un état correspond à la somme pondérée de la récompense immédiate que l'agent peut obtenir en effectuant l'action proposée par la politique π, additionnée à la valeur des états voisins atteignables depuis celui-ci. Lorsque $\gamma = 0$, les politiques considérées comme optimales sont celles qui rapportent immédiatement de la récompense à l'agent. Plus γ tend vers 1, plus les politiques qui mènent à des récompenses de plus en plus lointaines sont considérées comme optimales.

Puisque la fonction $V^{\pi}(s)$ nous permet d'évaluer une politique, il est désormais possible de chercher la politique optimale, c'est-à-dire la politique π^* dont le retour attendu est plus grand que toutes les autres politiques π. À la politique optimale π^* correspond donc une fonction de valeur des états optimale notée $V^*(s)$:

$$V^*(s) = \max_a \sum_{s' \in S} T(s, a, s')(R(s, a, s') + \gamma V^*(s')) \tag{2.3}$$

Respectivement, nous pouvons mesurer la quantité de récompenses cumulées que l'agent espère recevoir lorsqu'il effectue l'action a depuis l'état s avant de suivre la politique π, aussi appelée *valeur d'état-action* ou encore Q-valeur, grâce à la fonction de valeur des états-actions $Q^{\pi}(s, a)$:

$$Q^{\pi}(s) = \sum_{s' \in S} T(s, a, s')(R(s, a, s') + \gamma V^{\pi}(s')) \tag{2.4}$$

ainsi que la fonction de valeur des états-action optimale notée $Q^*(s, a)$:

$$Q^*(s) = \max_a \sum_{s' \in S} T(s, a, s')(R(s, a, s') + \gamma V^*(s')) \tag{2.5}$$

Sachant que les deux fonctions de valeur sont intimement liées, telles que $V^{\pi}(s) = Q^{\pi}(s, \pi(s))$ et donc que $V^*(s) = \max_b Q^*(s, b)$, nous pouvons aussi écrire la fonction 2.5 sous la forme :

$$Q^*(s) = \max_a \sum_{s' \in S} T(s, a, s')(R(s, a, s') + \max_b \gamma Q^*(s', b)) \tag{2.6}$$

Pour l'agent, l'enjeu est finalement d'estimer une politique optimale (et donc la fonction de valeur optimale) en interagissant avec son environnement. Dans les deux sous-parties suivantes, nous présenterons différentes méthodes permettant d'estimer ces fonctions de valeur en fonction des a priori faits sur l'information dont dispose l'agent

2.2 Programmation dynamique

Dès lors que l'agent connaît les fonctions de récompense R et de transition T du problème, il dispose de toutes les informations dont il a besoin pour déduire la politique optimale. On dit que l'agent a accès au

modèle de la tâche, qui associe le modèle de transition au modèle de récompense. Pour calculer la politique optimale, il peut utiliser la programmation dynamique, une méthode algorithmique permettant de résoudre des problèmes d'optimisation en les décomposant itérativement en sous-problèmes. Cette méthode est portée par les deux algorithmes principaux : *Value Iteration* (Bellman 1966) et *Policy Iteration* (Howard 1960)

2.2.1 Value Iteration

À partir d'une fonction de valeur définie aléatoirement, l'algorithme *Value Iteration* (abrégé value *VI* par la suite) 2.2 permet d'améliorer cette fonction de valeur au cours d'un processus itératif, jusqu'à atteindre la fonction de valeur optimale. Suite à ça, la politique optimale peut être déterminée. Ici, l'étape d'amélioration est effectuée directement après avoir partiellement évalué la politique, et non pas après son évaluation complète. On considère en effet qu'après quelques itérations de l'évaluation, les estimations des valeurs deviennent suffisantes pour fixer la nouvelle politique. C'est tout le principe des algorithmes dits *gloutons*, qui consistent à miser sur des optmima locaux dans l'espoir d'obtenir plus tard l'optimum global. VI pousse cette idée à l'extrême en effectuant une seule étape d'évaluation avant l'amélioration de la politique. Finalement,

l'algorithme VI est donc divisé en deux phases distinctes : la recherche de la fonction de valeur optimale et l'extraction de la politique associée.

Algorithme 2.1 : Value Iteration (adapté de Sutton et Barto (1998))

Initialiser V^π aléatoirement

mise-à-jour des valeurs
répéter
  (a) $\Delta \leftarrow 0$
  (b) **pour chaque** $s \in S$ **faire**
    1. $v \leftarrow V^\pi(s)$
    2. $V^\pi(s) \leftarrow \max_a \sum_{s'} T(s,a,s')[R^\pi(s) + \gamma V^\pi(s')]$
    3. $\Delta \leftarrow max(\Delta | v - V^\pi(s))$
  fin
jusqu'à $\Delta < \epsilon$ *(pour ϵ petit et positif)*

Calcul d'une politique déterministe
pour chaque $s \in S$ **faire**
  $\pi(s) \leftarrow argmax_a \sum_{s'} T(s,a,s')[R^\pi(s) + \gamma V(s')]$
fin
retourner π

Algorithme 2.2 : Q-learning (adapté de Sutton et Barto (1998))

Initialiser $Q(s,a)$ pour chaque $s \in S$ et $a \in A$. $Q(terminal,.) = 0$
Choisir l'état initial s_0

répéter
  1. Choisir a depuis s en utilisant une politique dérivée de Q ($\epsilon - greddy$)
  2. Effectuer l'action a, observer r et s'
  3. Mettre à jour $Q(s,a)$:

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha \underbrace{\left(\overbrace{r(s) + \gamma max_b Q_t(s',b)}^{\text{retour reçu}} - \overbrace{Q_t(s,a)}^{\text{prédiction}} \right)}_{\delta_t \text{ : erreur de prédiction}}$$

  4. $s \leftarrow s'$
jusqu'à *Jusqu'à ce que s soit terminal*

2.2.2 Policy Iteration

À partir d'une politique définie aléatoirement, l'algorithme *Policy Iteration* 2.3 va chercher la fonction de valeur de cette politique. Cette fonction de valeur lui permettra d'améliorer la politique, de définir une nouvelle fonction de valeur plus optimale, et ainsi d'améliorer à nouveau la politique. Ici, chaque politique trouvée est garantie comme étant une amélioration stricte de la précédente. Si ce n'est pas le cas, c'est que la politique optimale a été trouvée et que l'algorithme peut prendre fin. Finalement, l'algorithme PI est donc divisé en deux phases distinctes qui seront répétées jusqu'à convergence de la politique : l'évaluation de la politique et l'amélioration de la politique. À l'inverse de PI, VI n'a pas besoin de répéter ces deux phases de manière itérative, puisqu'elle cherche d'une traite la fonction de valeur optimale, et donc la politique optimale associée (ce

qui explique la présence du *max* en b.2 de l'algorithme 2.2, et son absence dans l'algorithme 2.3).

Algorithme 2.3 : Policy Iteration (adapté de Sutton et Barto (1998))

Initialiser π aléatoirement

1 Évaluation de la politique
répéter
  (a) $\Delta \leftarrow 0$
  (b) **pour chaque** $s \in S$ **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 $\Delta < \epsilon$ *(pour ϵ petit et positif)*

2 Amélioration de la politique
stable $\leftarrow$ *vrai*
pour chaque $s \in S$ **faire**
  (a) $b \leftarrow \pi(s)$
  (b) $\pi(s) \leftarrow argmax_a \sum_{s'} T(s, a, s')[R^{\pi}(s) + \gamma V(s')]$
  (c) **si** $b \neq \pi(s)$ **alors** $stable \leftarrow faux$
fin
si *non stable* **alors** aller en 1
sinon retourner V^{π} et π

À noter que ces deux algorithmes peuvent être formulés en utilisant la fonction de valeur des états-actions $Q^{\pi}(s, a)$ plutôt que la fonction de valeur des états $V^{\pi}(s)$.

Bien que très efficaces lorsqu'il s'agit de trouver une solution optimale dans le cas d'un problème formalisé par un MDP, ces deux algorithmes sont très dépendants de l'accès de l'agent au modèle de la tache (aux modèles de transition T et de récompense R), et de la stationnarité de ce modèle. Dans un environnement réaliste, l'agent pourrait en effet seulement avoir accès à un modèle partiel, erroné ou changeant de la tâche. Dans le cas extrême, où l'agent n'a accès à aucun modèle de la tâche, ces méthodes deviennent alors totalement obsolètes, et l'agent ne pouvant plus simuler en avance le résultat de sa politique doit désormais interagir avec son environnement pour obtenir de l'information sur celui-ci. Pour finir, l'efficacité de ces méthodes se fait au prix d'un temps de calcul (et donc d'un coût calculatoire) très dépendant de la taille des espaces d'état S et d'action A de la tâche, puisque toutes deux itèrent jusqu'à convergence. Si la problématique du temps de calcul (du coût calculatoire) peut être à peine gênante pour des agents virtuels évoluant dans des environnements simulés sur des ordinateurs possédant des ressources importantes, elle devient rapidement très handicapante lorsque l'agent est incarné dans un robot physique évoluant dans le monde réel. Alors, le coût calculatoire impacte directement l'autonomie du robot, et donc la résolution du problème elle-même.

2.3 Apprentissage par renforcement

Dès lors que l'agent ne connaît pas le modèle de la tâche (i.e., les fonctions de transition T et de récompense R), les méthodes de programmation dynamique deviennent inefficaces. Pour faire face à ce problème, deux solutions s'offrent à l'agent :

— estimer directement la valeur des états en interagissant avec l'environnement, et en déduire donc la meilleure action à réaliser dans chaque état qu'il découvre. On parle d'*apprentissage par renforcement direct* ou d'*apprentissage par renforcement sans modèle* (plus connu sous l'appellation *model-free reinforcement learning* en anglais) .

— estimer le modèle de la tâche en interagissant avec l'environnement. Avec une estimation du modèle, l'agent peut en déduire le comportement général à adopter en utilisant la programmation dynamique, par exemple. On parle d'*apprentissage par renforcement indirect* ou d'*apprentissage par renforcement basé sur un modèle* (plus connu sous l'appellation *model-based reinforcement learning*) .

Ces deux solutions, dont nous expliciterons les méthodes associées dans les sous-parties suivantes, s'appuient donc sur la nécessité qu'a l'agent d'explorer son environnement afin d'accroître sa connaissance de celui-ci.

2.3.1 Évolution de la connaissance de l'agent

Prenons le cas où l'agent ne connaît que l'état initial de la tâche $s^1 \in S$ et l'entièreté de son espace d'action A. N'ayant aucune autre information sur la tâche autre que cet unique état, il effectue l'action $a^4 \in A$ au hasard et arrive dans un nouvel état, que nous appellerons s^2, et qui ne contient pas de récompense. Ainsi, l'agent obtient trois types informations différentes :

— la valeur du nouvel état $s^2 \in S$ dans lequel il est arrivé ;

— la probabilité de transition $T(s^1, a^4, s^2) = 1.0$, qui indique qu'effectuer l'action a^4 dans l'état s^1 a une probabilité de 100% d'emmener l'agent dans l'état s^2 ;

— la récompense $R(s^1, a^4) = 0$, qui indique qu'effectuer l'action a^4 dans l'état s^1 n'apporte pas de récompense.

Désormais situé dans l'état s^2, l'agent va pouvoir effectuer une nouvelle action et progresser dans sa découverte de l'environnement. Si dans cet exemple, $T(s^1, a^4, s^2) = 1.0$ et $R(s^1, a^4) = 0$, rien ne dit qu'effectuer l'action a^4 dans l'état s^1 pourrait emmener dans un état différent de s^2, ou que s^2 pourrait contenir une unité de récompense dans un futur proche. Tout dépend de la stationnarité et de la stochasticité de l'environnement. Pour calculer la probabilité qu'une transition aie lieu, il suffit de compter pour chaque couple (s, a) le nombre de fois où l'état s' a été atteint $C(s, a, s')$ et de normaliser ensuite sur tous les états atteints depuis s en effectuant l'action a :

$$P(s'|s, a) = T(s, a, s') = \frac{C(s, a, s')}{\sum_{u \in s} C(s, a, u)} \tag{2.7}$$

2.3.2 Apprentissage par renforcement direct

Lorsque l'agent ne possède pas la capacité d'exploiter les informations relatives aux modèles de transition T et de récompense S, il est donc à la fois privé de la connaissance lui permettant de savoir quels états sont liés aux autres, mais aussi de celle lui indiquant où se trouvent les états récompensants et ceux à éviter. Alors, il ne lui reste plus que trois données informatives : la valeur de l'état s qu'il a quitté (appelé *état précédent*), la valeur de l'état s' dans lequel il est arrivé (appelé *état courant*), et bien sûr l'action a qu'il a réalisée pour changer d'état. Ainsi, l'agent pourra mettre à jour localement sa fonction de valeur, et devra répéter cette mise-à-jour au fur et à mesure qu'il explorera l'environnement afin d'obtenir une estimation complète de celle-ci.

En comparaison des méthodes de programmation dynamique, qui itèrent d'une traite jusqu'à convergence de la politique, les méthodes d'apprentissage par renforcement direct mettent donc bien plus de temps à faire converger la politique de l'agent. Ce défaut de l'apprentissage direct, lié au *problème de l'attribution de crédits temporels*, peut être corrigé par des méthodes dites de *reward shaping*. Nous aurons l'occasion de revenir sur ces deux notions dans une sous-partie dédiée. En revanche, si l'apprentissage direct met plus de temps à converger que l'apprentissage indirect, il est aussi bien moins coûteux en terme de ressources calculatoires, dû au fait que ses réponses rapides ne nécessitent pas de planification. Cela est un avantage certain lorsque la problématique de l'économie d'énergie doit être considérée, comme en robotique autonome.

La mise-à-jour itérative effectuée par l'apprentissage direct est appelée *apprentissage par différence temporelle* (abrégé TD par la suite) (Sutton et Barto 1987). Les algorithmes d'apprentissage par différence temporelle sont une extension du modèle de Rescorla (1972), proposant de modéliser le conditionnement classique de Pavlov, comme nous le verrons dans le chapitre 4. À nouveau, il s'agit d'estimer V^π ou Q^π afin d'en déduire la politique π à adopter. *L'apprentissage par différence temporelle* est défini par l'équation suivante :

$$V_{t+1}(s) = V_t(s) + \alpha.\delta_t \tag{2.8}$$

où $\alpha \in [0, 1]$ est le taux d'apprentissage et δ_y l'erreur de prédiction de la valeur de l'état précédent s au temps t. Nous expliciterons le calcul de l'erreur de prédiction dans les sous-parties suivantes, où nous présenterons plusieurs méthodes faisant appel à *l'apprentissage par différence temporelle*.

TD(0)

La méthode *TD(0)* consiste à mettre à jour directement la valeur de l'état précédent $V(s)$ une fois arrivé dans l'état courant s' en fonction de l'erreur entre la prédiction de cette valeur et le retour reçu dans l'état courant s' :

$$V_{t+1}(s) = V_t(s) + \alpha \underbrace{\left(\overbrace{r(s) + \gamma V_t(s')}^{\text{retour reçu}} - \overbrace{V_t(s)}^{\text{prédiction}} \right)}_{\delta_t \text{ : erreur de prédiction}} \tag{2.9}$$

où γ est le facteur d'actualisation que nous avons présenté précédemment, et qui permet de prendre en compte l'horizon temporel des récompenses futures.

Comme nous pouvons le voir, la méthode *TD(0)* ne prend en compte que la transition expérimentée entre l'état s et l'état s'. L'estimation de la politique π qui découle du calcul de $V(s)$ est donc basée sur un horizon temporel extrêmement réduit et ne peut pas être modifiée sans plus d'informations. Puisque nous traitons ici des méthodes *model-free*, il n'est donc pas possible d'estimer le modèle de la tâche. En revanche, plutôt que d'estimer la valeur de $V(s)$, nous pourrions chercher à estimer celle de la fonction de valeur des états-actions $Q(s,a)$, déjà plus informative. En effet, pour une seule valeur d'état $V(s)$, l'état s possède un nombre de valeurs d'état-action $Q(s,a)$ égale à la taille de l'espace d'action A de l'agent. Le choix de la Q-valeur qui sera mise-à-jour, et donc de l'action qui sera favorisée / défavorisée, impliquera aussi directement la politique π.

SARSA

La méthode *SARSA* (abréviation de *State-Action-Reward-State-Action* (Rummery et Niranjan 1994) consiste à mettre à jour la valeur $Q(s,a)$ de l'état précédent une fois arrivé dans l'état courant, en fonction de l'erreur entre la prédiction de cette valeur et le retour reçu dans l'étant courant, qui correspond ici à la récompense et la valeur $Q(s',a')$:

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha \left(r(s) + \underbrace{\gamma Q_t(s',a')}_{\equiv V^{\pi}(s')} - Q_t(s,a) \right) \tag{2.10}$$

où a' est l'action effectuée par l'agent dans l'état courant s'.

Ici, nous pouvons voir que pour mettre à jour les valeurs d'état-action $Q(s,a)$, la méthode *SARSA* à besoin de l'action a', et donc de la décision de l'agent prise dans l'état courant. De ce fait, *SARSA* permet d'évaluer la politique π, mais également de la modifier, via le choix de l'action a'. Pour cette raison, on dit que cette méthode apprend *sur politique* : l'estimation des Q-valeurs est faite en supposant que la politique actuelle continue à être suivie. Si par rapport à la méthode *TD(0)*, *SARSA* permet donc d'influer sur la politique suivie par l'agent, sa performance devient aussi très dépendante de cette politique. Dès lors que la politique suivie n'est pas *gloutonne* (c'est-à-dire que l'action associée à la plus grande Q-valeur n'est pas nécessairement choisie), l'évaluation de l'état suivant peut devenir sous-optimale.

Q-learning

La méthode du *Q-learning* (Watkins 1989, Watkins et Dayan 1992) consiste à mettre à jour la valeur $Q(s,a)$ de l'état précédant une fois arrivé dans l'état courant, en fonction de l'erreur entre la prédiction de cette valeur et le retour reçu dans l'état courant, qui correspond ici à la récompense et à la plus grande valeur $Q(s',a')$ de l'état courant :

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha \left(r(s) + \underbrace{\gamma max_b Q_t(s',b)}_{\equiv V^{\pi}(s')} - Q_t(s,a) \right) \tag{2.11}$$

Ici, nous pouvons voir que pour mettre à jour les valeurs d'état-action $Q(s,a)$, la méthode de *Q-learning* n'a plus besoin de l'action a' associée à la politique actuelle potentiellement sous-optimale. La valeur choisie est toujours celle correspondant à la plus grande Q-valeur, et donc l'action la plus optimale connue à l'itération t. Pour cette raison, on dit que cette méthode apprend *hors politique* : l'estimation des Q-valeurs est indépendante de la politique actuelle.

Deep Q-Network

Jusqu'alors, tous les algorithmes que nous avons présentés sont des algorithmes tabulaires, c'est-à-dire des algorithmes enregistrant dans un tableau les résultats discrets des fonctions de valeur (un tableau V contenant autant de cellules que d'états pour les valeurs d'état $V(s)$, ou un tableau Q contenant autant de cellules que d'états multiplié par le nombre d'actions pour les valeurs d'état-action $Q(s,a)$).

Dans certains problèmes, et surtout lorsque le nombre d'états devient trop important, il devient compliqué de représenter les fonctions de valeur par des tableaux : c'est ce que l'on nomme le Fléau de la Dimension (*Curse of Dimensionality* en anglais) (Bellman 2015). En apprentissage automatique, cela revient souvent à tirer des inférences d'un nombre réduit d'expériences dans un espace de possibilités de dimension élevée. Pour combattre ce fléau, une première solution serait de réduire la dimension de l'environnement, c'est-à-dire de projeter des données issues d'un espace de grande dimension dans un espace de plus petite dimension (Van Der Maaten et al. 2009). Dès lors que les données en sortie de la réduction représentent bien les données d'entrée, cela entraîne automatiquement une amélioration des propriétés de stabilité et de robustesse des algorithmes, et simplifie et accélère la résolution du problème d'optimisation associé, en réduisant l'espace des solutions (Bousquet et Elisseeff 2002). Une autre solution consiste à approximer les tableaux V et Q avec un approximateur de fonction, tel qu'un réseau de neurones artificiel. C'est la solution qui nous intéresse ici.

Le *DQN* (abréviation de *Deep Q-Nework*) (Mnih et al. 2015a) est un réseau de neurones convolutifs permettant d'approximer la fonction de valeur d'état-action $Q(s,a)$ grâce à l'apprentissage profond. Le réseau de neurones, noté Q_θ (où θ sont ses paramètres), prend en entrée les états $s \in S$ et retourne en sortie autant de Q-valeurs qu'il y a d'action $a \in A$. L'action choisie par l'agent est celle associée à la plus grande de ces Q-valeurs. Le fonctionnement du *DQN* est basé sur l'*Algorithme du Gradient* (Lemaréchal 2012), un algorithme d'optimisation permettant de trouver le minimum de n'importe quelle fonction convexe en convergeant progressivement vers celui-ci. Dans le cas de l'apprentissage profond, cela consiste souvent à minimiser l'erreur estimée par une fonction de perte en optimisant les paramètres θ du réseau. L'erreur (appelée aussi perte) est mesurée

comme étant la différence entre le résultat prédit et le résultat réel (le résultat cible). Dans l'algorithme *DQN*, l'erreur à minimiser correspond à l'erreur de prédiction de l'algorithme de *Q-learning* 2.11 :

$$L(\theta) = \left(\overbrace{r + \gamma max_b Q_t(s', b, \theta^{cible})}^{\text{Q-valeur cible}} - \overbrace{Q_t(s, a, \theta^{\text{prédit}})}^{\text{Q-valeur prédite}} \right) \tag{2.12}$$

Du fait que le même réseau calcule la valeur prédite et la valeur cible, des divergences entre ces deux valeurs sont souvent observées. Pour contrer cet effet, le *DQN* utilise deux réseaux distincts mais à l'architecture identique : un premier qui va se charger de calculer la valeur prédite et un second qui va se charger de calculer la valeur cible, mais dont les paramètres initiaux seront de temps à autre mis à jour selon les paramètres du premier réseau.

Pour finir, le *DQN* utilise aussi un mécanisme d'*experience replay*, qui consiste à séparer l'acquisition des données de la phase d'apprentissage. Concrètement, lorsque l'agent arrive dans un nouvel état, il stocke dans un tableau la donnée (s, a, r, s') sans réaliser d'apprentissage. C'est seulement dans un second temps que l'apprentissage sera réalisé sur des données prélevées aléatoirement depuis ce tableau. Le premier avantage de l'*experience replay* est de pouvoir apprendre plusieurs fois les expériences précédentes, ce qui est utile lorsque l'acquisition de données est coûteuse ou que l'apprentissage est lent. Le second avantage est que les données prélevées depuis un tableau ressemblent davantage à des données indépendantes et identiquement distribuées, ce qui est connu pour faciliter la convergence des approximateurs de fonction. Le fonctionnement de l'algorithme *DQN* est illustré dans la figure 2.1.

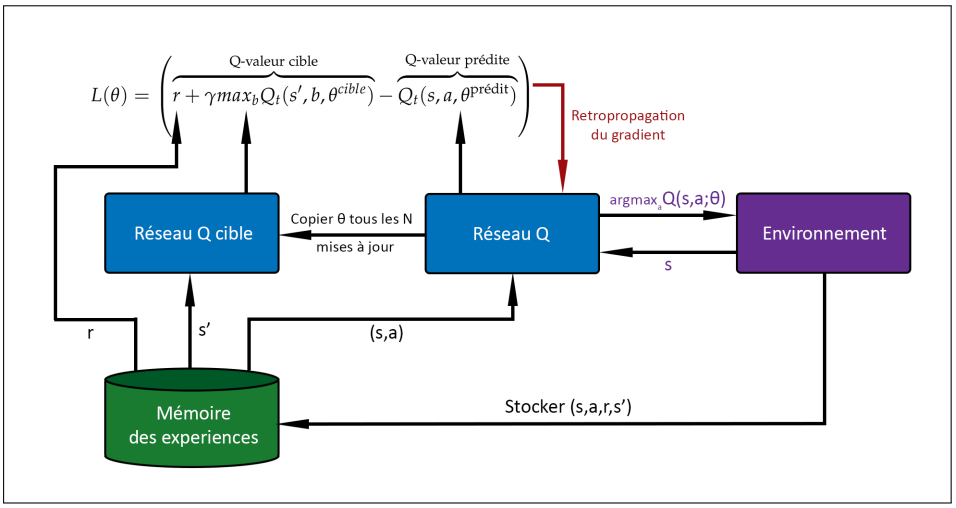

FIGURE 2.1 – *Illustration schématique du fonctionnement de l'algorithme DQN utilisant deux réseaux de neurones jumeaux et une mémoire de replay, tel que décrit dans Mnih et al. (2015a).*

Si nous ne faisons ici qu'effleurer le sujet de l'apprentissage par renforcement profond, et que la méthode mériterait un chapitre à elle toute seule, son utilisation reste très anecdotique dans notre travail de recherche en comparaison des méthodes d'apprentissage par renforcement tabulaires que nous avons présentées précédemment. Rappelons simplement

que ces dernières années, l'AR profond porté par l'algorithme *DQN* et ses déclinaisons, a permis de résoudre de nombreux problèmes d'apprentissage de nature variée et à la complexité élevée, tels que le Jeu du Go (Silver et al. 2016) ou encore la prédiction de la structure de protéines (Senior et al. 2020). Si la notoriété et le succès technique de l'AR profond s'expliquent par un certain nombre d'innovations réalisées ces dix dernières années, la méthode est avant tout née de la convergence de plusieurs champs de recherche scientifique datant de la seconde moitié du 20e siècle, tels que les réseaux de neurones artificiels (Hebb 2005, Minsky et Papert 2017), l'AR (Sutton 1992) ou les arbres de décision (Quinlan 2014), mais aussi de l'explosion de la puissance des machines modernes capables de traiter rapidement des quantités de données massives contenues dans des bases de données tout aussi grandes. Rappelons que pour devenir aussi doué au Jeu du Go, le programme AlphaGo (Silver et al. 2016) a joué plusieurs millions de parties contre lui-même. Dans notre travail de recherche, nous cherchons avant-tout à développer des solutions permettant à des robots d'apprendre à la volée de manière autonome à réaliser efficacement un nombre varié de tâches non-stationnaires dans le monde réel, avec un souci constant d'économie calculatoire, et donc d'autonomie. L'apprentissage profond, de part l'entraînement intensif et très coûteux qu'il demande, s'accorde difficilement avec ce cadre expérimental. Nous montrerons dans le chapitre 6 comment l'algorithme *DQN* parvient difficilement à atteindre une performance satisfaisante après les quelques centaines d'action de navigation suffisantes à notre architecture pour trouver une bonne solution. À l'heure de la crise climatique, il est aussi important de rappeler qu'il est de notre responsabilité de réfléchir au développement d'IA plus économiques, et donc plus écologiques. Or ces dernières années, la recherche en AR profond reposait avant-tout sur de puissantes machines particulièrement énergivores. Aujourd'hui, les choses évoluent doucement, et des réseaux de neurones profonds pensés pour la robotique réelle commencent à être développés (Caballero et al. 2021, Beeching et al. 2021).

Approche acteur-critique

Les algorithmes acteur-critique (Barto et al. 1983, Konda et Tsitsiklis 2000) constituent une approche hybride entre les méthodes basées sur des fonctions de valeur (les méthodes présentées jusqu'alors), et les méthodes basées sur le gradient de la politique (Williams 1992), qui optimisent directement la politique depuis les récompenses perçues sans calculer de fonction de valeur. Concrètement, les algorithmes acteur-critique calculent à la fois la politique grâce à un *acteur*, qui maintient des valeurs de préférences $p(s,a)$, et la fonction de valeur grâce à un *critique* (2.8), qui évalue aussi l'erreur de prédiction (2.9). Cette erreur de prédiction est utilisée afin d'évaluer l'*acteur* en mettant à jour la préférence de la transition expérimentée $p(s,a)$. Nous n'avons pas utilisé d'algorithmes acteur-critique dans ce travail de recherche, et ne rentrerons donc pas plus dans le détail de ces méthodes.

2.3.3 Apprentissage par renforcement indirect

En reprenant l'exemple de la sous-partie *Évolution de la connaissance de l'agent : un exemple*, nous comprenons que dès lors que l'agent possède la capacité d'exploiter les informations relatives aux modèles de transition T et de récompense S (ce qui n'est donc pas possible dans le cas de l'apprentissage direct), il réussira au fur et à mesure de son exploration à construire un modèle partiel de la tâche. Une fois ce modèle construit, il devient alors tout simplement possible d'appliquer les algorithmes de *VI* (2.2) et de *PI* (2.3) afin d'estimer d'une traite la politique à adopter. Comme le faisait remarquer Sutton (1992), l'intérêt d'apprendre un modèle plutôt que des valeurs d'états réside dans le fait que ce modèle contient bien plus d'informations sur l'environnement, permettant par exemple de raisonner sur les causes et les conséquences des actions effectuées par l'agent, alors que l'apprentissage direct ne fait qu'apprendre un comportement et sa valeur pour un problème spécifique considéré.

Une solution intermédiaire à l'utilisation des algorithmes de programmation dynamique, qui ne peuvent de toute manière être utilisés de manière optimale que lorsque l'agent possède un modèle complet de la tâche, serait de faire appel à des méthodes à mi-chemin entre l'apprentissage direct et l'apprentissage indirect. Par exemple, l'agent pourrait mettre à jour ses valeurs d'état en fonction de l'expérience précédente afin d'assurer l'adaptation de sa politique, comme pourrait le faire un algorithme de *Q-learning*, tout en mettant à jour en parallèle son modèle de la tâche pour les couples (s, a) connus, comme pourrait le faire un algorithme de *VI*. C'est ce que propose de faire, avec quelques subtilités, l'algorithme *Dyna-Q* :

Algorithme 2.4 : Dyna-Q (adapté de Sutton et Barto (1998))

$\forall (s, a) \in S \times A(s)$, *initialiser* $Q(s, a)$ *et* $Modele(s, a)$ *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) $Q(s, a) \leftarrow Q(s, a) + \alpha[(r + \gamma max_b Q(s', b) - Q_k(s, a))]$
- (e) $Modele(s, a) \leftarrow s', r$
- (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. $s', r \leftarrow Modele(s, a)$
  4. $Q(s, a) \leftarrow Q(s, a) + \alpha[(r + \gamma max_b Q(s', b) - Q_k(s, a))]$
- **fin**

fin

Bien que l'algorithme *Dyna-Q* soit une grande amélioration par rapport aux méthodes précédentes, il souffre d'être relativement non dirigé, du fait de la sélection aléatoire des couples (s, a). Typiquement, lorsque l'agent atteint un état récompensé ou se coince dans une impasse, nous aimerions que l'algorithme puisse se concentrer sur ces états du MDP plutôt que sur des états choisis aléatoirement. Ce problème est abordé par les algorithmes *Prioritized Sweeping* (Moore et Atkeson 1993, Aubin et al. 2018)

et *Queue-Dyna* (Peng et Williams 1993), qui bien que développés indépendamment, proposent des solutions très similaires :

Algorithme 2.5 : Prioritized Sweeping (adapté de Sutton et Barto (1998))

$\forall(s,a) \in S \times A(s)$, *initialiser* $Q(s,a)$ *et* $Modele(s,a)$ *arbitrairement et* $PQueue$ *comme étant vide*

tant que *vrai* **faire**
- (a) $s \leftarrow$ état courant (non terminal)
- (b) $a \leftarrow politique(s,Q)$
- (c) Effectuer a ; observer le nouvel état s' et la récompense r
- (d) $Modele(s,a) \leftarrow s', r$
- (e) $P \leftarrow |\underbrace{r + \gamma max_b Q(s',b) - Q_k(s,a)}_{\delta_t \text{ : erreur de prédiction}}|$
- (f) **si** $P > \theta$ **alors** insérer (s,a) dans $PQueue$ avec une priorité P
- (g) **pour** N *éléments* **faire**
  - **tant que** $PQueue$ *n'est pas vide* **faire**
    - 1. $(s,a) \leftarrow début(PQueue)$
    - 2. $s', r \leftarrow Modele(s,a)$
    - 3. $Q(s,a) \leftarrow Q(s,a) + \alpha[(r + \gamma max_b Q(s',b) - Q_k(s,a))]$
    - **pour** *tout* $(\overline{s},\overline{a})$ *prédit pour emmener dans l'état* s **faire**
      - i. $\overline{r} \leftarrow$ récompense prédite pour $(\overline{s},\overline{a},s)$
      - ii. $P \leftarrow |\overline{r} + \gamma max_b Q(s,b) - Q_k(\overline{s},\overline{a})|$
      - iii. **si** $P > \theta$ **alors** insérer $(\overline{s},\overline{a})$ dans $PQueue$ avec une priorité P
    - **fin**
  - **fin**
- **fin**

fin

Comme nous pouvons le voir, le fonctionnement de l'algorithme *Prioritized Sweeping* est très similaire à *Dyna-Q*. La différence principale réside dans le fait que les mises à jour ne sont plus choisies au hasard, mais par ordre de priorité. Chaque paire (s,a) dont l'erreur de prédiction δ changerait de manière non triviale si la valeur $Q(s,a)$ était mise-à-jour est ajoutée dans une fille d'attente, ordonnée selon un ordre de priorité correspondant à la valeur absolue de l'erreur de prédiction δ_t. Ensuite, dès lors que la valeur $Q(s,a)$ en haut de la file est mise-à-jour par l'algorithme, les erreurs de prédiction de chacun de ses prédécesseurs (les paires d'état-action emmenant dans l'état s) sont calculées, et si la valeur absolue de ces erreurs de prédiction sont à nouveau supérieures à un seuil, les paires (s,a) correspondantes sont ajoutées dans la file. De cette façon, les effets des changements sont efficacement propagés en amont. Avec cette amélioration, la durée d'apprentissage en est réduite, ce qui permet à l'agent d'être plus performant sur les tâches discrètes évaluées.

2.3.4 Calcul de la politique et compromis exploration/exploitation

Comme illustré précédemment, le calcul de la politique π, et donc le choix de l'action a que l'agent va effectuer dans l'état s, consiste souvent

à choisir l'action associée à la plus grande valeur d'état-action $Q(s,a)$. Lorsque le modèle de la tâche est entièrement connu, ce choix garantit de trouver la politique optimale. Cependant, dans un certain nombre de situations, le modèle de la tâche n'est pas connu par l'agent, et celui-ci doit l'explorer pour acquérir les données nécessaires à son apprentissage. Dans ce cas là, choisir l'action associée à la plus grand valeur d'état-action $Q(s,a)$ ne garantit pas que l'on suive la politique optimale, celle-ci n'étant peut-être simplement qu'un optimum local. Cela peut arriver dans un environnement où plusieurs chemins différents mènent à l'état récompensé : dès lors que l'agent ne connaît qu'un seul chemin, celui-ci sera nécessairement son chemin optimal, même s'il est plus long que les chemins qu'il ne connaît pas encore. De la même manière, l'agent peut connaître la position d'un état récompensé, sans savoir qu'un autre état apporte encore plus de récompense. N'oublions pas aussi que le modèle de la tâche peut évoluer au cours du temps dans le cas des problèmes non-stationnaires. Ici, nous comprenons donc en quoi choisir une action a priori sous-optimale peut s'avérer payant, et en quoi l'agent doit faire un compromis entre l'exploitation de ses connaissances et l'exploration du modèle de la tâche. Pour répondre à ce compromis, il suffit de choisir de temps à autre aléatoirement l'action que l'agent va effectuer. C'est ce que propose la règle de sélection *epsilon-greedy* :

$$a = \begin{cases} \text{action aléatoire} & \text{si } X \leq \epsilon \\ argmax_a Q(s,a) & \text{sinon} \end{cases} \tag{2.13}$$

où ϵ est une faible probabilité et X une variable aléatoire suivant une loi uniforme sur $[0,1]$. Malheureusement, cette méthode ne tient pas compte de la valeur des états, et de ce fait, une action menant vers une impasse (et donc associée à une faible valeur) aura autant de chances d'être sélectionnée qu'une action amenant malgré tout vers une récompense (et donc associée à une plus grande valeur). À l'inverse, la règle de sélection *soft-max*, qui transforme la distribution des valeurs d'état-action en une distribution de probabilités P, permet de prendre en compte ces valeurs dans le choix de l'action :

$$P(a|s) = \frac{\exp(Q(s,a)/\tau)}{\sum_{b \in \mathcal{A}} \exp(Q(s,b)/\tau)} \tag{2.14}$$

où τ est le paramètre de compromis exploration/exploitation, aussi appelé *température de sélection*. Plus sa valeur est faible, plus l'agent tend à sélectionner l'action la plus optimale selon lui, et plus sa valeur est grande, plus l'agent tend à sélectionner une action aléatoirement. Si dans notre travail, nous utiliserons une valeur de τ fixée par l'expérimentateur, certains travaux s'intéressent à l'adaptation en ligne de la valeur de τ, ainsi que des autres paramètres du système de manière plus générale (Khamassi et al. 2011, Aklil et al. 2014).

2.3.5 Reward shaping

Comme nous l'avons vu précédemment, la fonction de récompense R est indispensable à l'apprentissage de l'agent, qu'on parle d'apprentissage

par renforcement direct ou indirect, puisqu'elle indique dans quelle mesure il a résolu le problème auquel il est confronté. Cependant, l'information associée reste peu informative, du fait que seul un ou quelques états sont en réalité récompensés, et qu'elle ne spécifie pas directement les actions à entreprendre dans chaque état. Or pour dériver une politique, le robot doit savoir dans quelle mesure chaque action a contribué à sa progression vers l'objectif. Ce problème, connu comme le *problème de l'attribution de crédits temporels*, peut être résolu par les méthodes d'*apprentissage par différence temporelle* et de *programmation dynamique* que nous avons présentées précédemment, et qui permettent de propager l'information associée à la récompense dans l'espace d'état S. Cependant, ces méthodes sont soit lentes à converger, soit coûteuses en termes de ressources calculatoires.

La méthode du *reward shaping* est une solution permettant d'accélérer et de faciliter le processus d'apprentissage en ajoutant des récompenses intermédiaires (Gullapalli et Barto 1992). Cependant, si mal utilisée, elle peut causer des problèmes de convergence. Par exemple, l'agent peut rester coincé dans des minima locaux en répétant sans cesse des comportements sous-optimaux, car maximisant les récompenses intermédiaires plutôt que celles associées à la résolution de l'objectif (Randløv et Alstrøm 1998) : c'est le problème des *cycles positifs*.

Pour faire face à ce problème, des méthodes de *reward shaping* permettent d'ajouter des récompenses intermédiaires tout en garantissant en même temps le fait que l'agent apprendra un comportement optimal (Ng et al. 1999, Wiewiora et al. 2003). Ces méthodes dites *basées sur le potentiel* impliquent la construction d'une fonction de potentiel adaptée reposant sur l'extraction de connaissances depuis des tâches apprises précédemment, et sur leur transfert vers une tâche cible. Ces méthodes sont très liées à *l'apprentissage par transfert* (Pan et Yang 2009, Doncieux 2013) dont nous ne parlerons pas dans ce manuscrit.

Pour faire face à la nouvelle difficulté que représente le fait de devoir concevoir une fonction de potentiel adaptée au problème, certaines méthodes qualifiées d'*autonomes* permettent de l'apprendre de manière automatique en approximant la fonction de valeur associée au problème sur un espace d'état alternatif de complexité réduite. Dans Konidaris et Barto (2006), les auteurs dotent l'agent de capteurs supplémentaires lui permettant d'apprendre une fonction de valeur alternative durant une session pré-expérimentale, qui sera utilisée pour initialiser les valeurs d'état du problème. Dans Marthi (2007), les auteurs ne dotent pas l'agent de capteurs supplémentaires, mais approximent la fonction de valeur dans un espace d'état abstrait, afin de l'utiliser ensuite pour initialiser, comme précédemment, les valeurs d'état du problème. Ces méthodes sont basées sur le travail de Wiewiora (2003), qui montre qu'utiliser une fonction de potentiel pour faire du *reward shaping* revient à utiliser cette fonction de potentiel pour initialiser les valeurs d'état-action du problème.

Pour finir, même si les méthodes de *reward shaping* permettent d'accélérer l'apprentissage, elles reposent soit sur la nécessité de construire au préalable des fonctions de potentiel adaptées, ou soit sur la capacité de l'agent de le faire lui-même, moyennant un coût calculatoire supplémentaire. Une autre manière de faire du *reward shaping* consiste à laisser un humain instruit interagir avec l'agent au cours de la tâche afin de le

guider vers la résolution de l'objectif, moyennant une perte d'autonomie. Nous aborderons ce cas spécifique dans le chapitre 3 dédié à l'apprentissage interactif.

2.4 Apprentissage ensembliste

Les différents algorithmes que nous avons présentés jusqu'alors affichent des bénéfices et des inconvénients différents : alors que les algorithmes d'AR direct mettent beaucoup plus de temps à converger vers une politique optimale que les algorithmes d'AR indirect, qui font appel à la programmation dynamique, ils sont en revanche bien plus économiques en terme de coût calculatoire. Pouvoir utiliser les deux méthodes pourrait permettre à un agent de tirer avantage des bénéfices de chacune d'entre elle. Plus largement, lorsqu'un agent doit prendre une décision importante, il est souvent plus sage de recueillir plusieurs avis que de se fier à un seul. C'est là tout le principe des méthodes dites *d'ensemble* ou *méthodes ensemblistes*, qui consistent à mettre en parallèle plusieurs *experts* cherchant à répondre à un problème donné, et à prendre en compte la réponse de chacun pour décider du comportement final de l'agent. Est appelé *expert* un algorithme modélisant une méthode d'apprentissage spécifique. À nouveau, si la problématique de l'efficacité et du coût calculatoire de l'agent est moins contraignante dans le cas de tâches simulées où le temps est virtuellement accéléré, nous nous intéressons avant tout dans ce travail de recherche à des problèmes de robotique bien réels. Or, dans des problèmes de grande dimension ou de grande complexité, comme décrit dans Kober et al. (2013), l'AR devient un véritable défit. Les méthodes d'ensemble participent, à leur juste mesure, à le relever.

2.4.1 État de l'art

S'il existe une littérature conséquente des méthodes d'ensemble en apprentissage supervisé (Jacobs et al. 1991, Wolpert et Kawato 1998, Freund et al. 1999, Sewell 2008), nous nous intéressons ici avant tout à la littérature de l'apprentissage par renforcement (Jiang et Kamel 2006, Khamassi et al. 2006). Récemment, les méthodes d'ensemble appliquées à l'AR ont regagné un certains succès, pour les raisons évoquées plus haut concernant l'application de l'AR à la robotique réelle, mais aussi du fait que la structure des algorithmes d'AR s'adapte assez bien à la multiplication des processeurs dans les ordinateurs modernes (Wiering et Van Otterlo 2012). Les possibilités de combinaisons d'experts sont nombreuses : tandis que Doya et al. (2002) combinent plusieurs algorithmes d'AR direct de même nature, comme Faußer et Schwenker (2011; 2015a) et leurs réseaux de neurones apprenant par différences temporelles, Wiering et Van Hasselt (2007; 2008) combinent quant à eux des algorithmes d'AR direct différents, Hans et Udluft (2010) des réseaux de neurones à la topologie différente et Renaudo (2016) des algorithmes d'AR direct et indirect. Plus récemment, Chen et al. (2018) ont appliqué la méthode ensembliste à l'AR profond, et plus spécifiquement à l'algorithme *DQN*, en intégrant plusieurs réseaux cibles différents (*DQN*, *double DQN*, *Deep Sarsa Network*) à leur architecture et en combinant les différentes politiques retournées.

Les similitudes ou les différences entre les experts d'apprentissage sont un facteur important, étant donné que chaque algorithme apprend différemment, et a donc des biais différents vis à vis du problème. Par exemple, l'algorithme *SARSA* calcule son erreur de prédiction en fonction de l'action a' réalisée par l'agent dans l'état s' suivant la politique π, tandis que le *Q-learning* calcule son erreur de prédiction en fonction de l'action a^* prédite comme étant optimale dans l'état s'. L'un sera donc biaisé vers sa politique actuel, tandis que l'autre sera biaisé vers la politique estimée comme étant optimale. Qui dit apprentissage ensembliste dit aussi importance de la gestion du résultat retourné par chacun des experts. Cela est d'autant plus vrai lorsque les experts sont portés par des algorithmes assez différents, comme dans Caluwaerts et al. (2012b), qui combine un algorithme de parcours de graphes et un algorithme qui apprend par AR direct à associer des stimulus visuels avec des actions. Face à ce type de combinaison, la solution la plus évidente revient à faire un choix entre le retour des différents experts, et donc de définir la politique selon ce retour. C'est ce que font Caluwaerts et al. (2012b). À l'inverse, lorsque les retours sont de nature identique, il devient possible de les fusionner en une solution unique, donnant ainsi naissance à une politique hybride.

2.4.2 Méthodes de fusion

Ici, nous nous concentrerons sur les méthodes dites de *fusion*, qui nous permettent de mettre en avant des manières de combiner des politiques. Les méthodes dites de *sélection*, illustrées précédemment par les travaux de Caluwaerts et al. (2012b) et qui reposent plutôt sur des critères de choix, seront évoquées dans le chapitre 4. Wiering et Van Hasselt (2008) définissent quatre méthodes différentes de *fusion* permettant de combiner les politiques de N experts qui calculent un score de préférence p pour chaque action $a[i]$:

— La technique *Majority Voting*, où le score de préférence est défini par :

$$p_t(s_t, a[i]) = \sum_{e=1}^{N} I(a[i], a_t^e) \tag{2.15}$$

avec a_t^e l'action la plus probable de l'expert e dans l'état s_t et $I(x, y)$ la fonction indicatrice qui vaut 1 lorsque $a[i] = a_t^e$ et 0 sinon. L'action la plus probable est donc celle considérée comme meilleure par la majorité des experts.

— La technique *Rank Voting*, où le score de préférence est défini par :

$$p_t(s_t, a[i]) = \sum_{e=1}^{N} w_t^e(a[i]) \tag{2.16}$$

avec $w_t^e(a[i])$ un poids affecté aux actions en fonction de leur probabilité de sélection pour l'expert e. Plus la probabilité de sélection est grande, plus le poids sera important.

— La technique *Boltzmann Multiplication*, où le score de préférence est défini par :

$$p_t(s_t, a[i]) = \prod_{e=1}^{N} \pi_t^e(s_t, a[i]) \tag{2.17}$$

c'est-à-dire par le produit des probabilités de sélection par chaque expert de l'action préférée. Cette technique implique que dès lors que la probabilité de sélection d'une action d'un des experts est nulle, sa préférence sera nulle peu importe sa probabilité de sélection pour les autres experts.

— La technique *Boltzmann Addition*, qui est une variante du *Rank Volting* préservant les écarts relatifs entre les valeurs d'actions, où le score de préférence est défini par :

$$p_t(s_t, a[i]) = \sum_{e=1}^{N} \pi_t^e(s_t, a[i]) \tag{2.18}$$

Wiering et Van Hasselt (2008), à l'origine de ces différentes méthodes de *fusion*, les évaluent sur des problèmes de labyrinthes discrets, et montrent que *Majority Voting* et *Boltzmann Multiplicaton* exhibent les meilleures performances face aux deux autres méthodes, mais aussi face aux algorithmes pris individuellement. Hans et Udluft (2010) et Faußer et Schwenker (2011; 2015a) confirment l'efficacité de *Majority Voting* sur des réseaux de neurones, respectivement dans le problème du *Cart Pole* (Sutton et Barto 1998) et dans les jeux *tic-tac-toc* et *tétris*. Quand à Renaudo et al. (2015c;b), ils montrent que dans une tâche de poussée de cubes sur un tapis roulant simulé, la technique *Rank Voting* affiche de meilleures performances que les autres méthodes de *fusion* et que celles de *sélection* lorsque l'environnement est non-stationnaire, et que lorsque celui-ci est stationnaire, *Rank Voting*, *Boltzmann Multiplication* et *Boltzmann Addition* affichent des performances identiques. Ils montrent aussi que les méthodes de *fusion* ou de *sélection* affichent aussi souvent de meilleures performances que celles des algorithmes seuls. Chen et al. (2018) n'utilisent pas l'une des quatre méthodes de *fusion* et somment simplement les retours de chaque réseau cible pour calculer la Q-valeur cible de leur fonction de perte. Ils montrent aussi que leur réseau ensembliste affiche des meilleures performances que les réseaux traditionnels d'AR profond sur les problèmes du *Cart Pole* et du *Mountain Car* (Sutton et Barto 1998). Pour finir, Faußer et Schwenker (2015b) montrent dans leurs derniers travaux l'avantage de fusionner les politiques d'un sous-ensemble d'experts sélectionnés selon certains critères saillants, plutôt que de fusionner les politiques de tous les experts, ce qui ouvre la porte à des améliorations futures pour l'apprentissage ensembliste.

2.5 Conclusion

Dans ce chapitre, nous avons expliqué comment modéliser des problèmes de décision séquentielle avec les Processus de Décision Markoviens, et comment résoudre ces problèmes à l'aide de la programmation dynamique et de l'apprentissage par renforcement. Généralement, résoudre le problème consiste pour l'agent à accumuler de la récompense au cours du temps.

Si la programmation dynamique est très performante lorsque le robot a accès à un modèle du problème, elle est aussi coûteuse calculatoirement parlant, et perd en intérêt lorsque le modèle du problème n'est pas disponible. L'apprentissage par renforcement est pour sa part capable d'apprendre à résoudre le problème sans modèle et à moindre coût en interagissant avec son environnement, même si cela implique un apprentissage long et fastidieux. Pour tirer bénéfices des avantages de chacune des méthodes, nous avons présenté les méthodes ensemblistes, qui proposent de faire fonctionner plusieurs méthodes en parallèle en fusionnant leurs retours. Nous verrons dans le chapitre 4 qu'une autre solution consiste à développer des critères permettant de choisir entre les différents retours de l'agent. C'est cette solution que nous avons utilisée pour développer l'architecture de contrôle robotique au cœur de ce travail de recherche, et que nous avons évaluée dans les chapitres 6, 7 et 8.

Dans ce chapitre, nous nous sommes avant tout intéressés à l'apprentissage par renforcement tabulaire et discret comme d'un outil permettant de résoudre des problèmes de prise de décision de haut niveau. Comme nous aurons l'occasion de le voir plus tard dans le manuscrit, nous considérons que discrétiser un problème continu est une solution efficace, bien que nécessairement très dépendant de la méthode de discrétisation. Nous arguons aussi que l'apprentissage par renforcement profond, bien que très populaire de nos jours, n'est pas forcément l'outil le plus adapté dès lors que les ressources calculatoires mises à disposition de l'agent sont limitées.

Apprentissage interactif

3

Sommaire

Dans ce chapitre, nous nous intéressons à l'apprentissage interactif comme d'une méthode permettant d'accélérer et de faciliter le processus d'apprentissage d'agents artificiels. Si différentes manières d'apprendre de l'humain pour un robot existent, nous mettons ici l'emphase sur l'*apprentissage par les rétroactions évaluatives* (Knox et Stone 2012b), qui s'intègre très bien au cadre de l'apprentissage par renforcement (Sutton 1992) que nous avons présenté dans le chapitre précédent, et qui est au cœur de notre travail de recherche. Dans un premier temps, nous commençons par faire une revue des différents types d'interactions humain-robot et des différentes manières d'apprendre de l'humain décrites dans la littérature. Ensuite, nous expliquons comment les rétroactions évaluatives d'un humain peuvent être interprétées par un agent effectuant de l'apprentissage par renforcement en présentant différentes méthodes de *shaping*. Pour finir, nous nous intéressons aux bénéfices et inconvénients de l'*apprentissage par les rétroactions évaluatives* face à l'*apprentissage par la démonstration*, une autre manière interactive d'apprendre pour un robot reconnue comme étant plus performante. Comme dans le chapitre précédent, nous nous intéressons dans ce chapitre avant tout à un problème symbolique de haut niveau, qui est ici la manière dont les interactions humaines peuvent influencer (négativement ou positivement) le système de prise de décision du robot. De nombreuses autres études s'intéressent à d'autres problématiques de l'apprentissage interactif humain-robot, de

tout niveau, telles que la perception par le robot des signaux des humains, la manière dont l'humain communique avec le robot, le développement d'interfaces d'interaction ergonomiques ou encore la nécessité de pouvoir transférer l'apprentissage de tâche en tâche (Goodrich et Schultz 2008). Dans ce travail de recherche, l'enjeu est avant tout d'améliorer l'autonomie décisionnelle d'agents artificiels incarnés. Nous montrons dans ce chapitre en quoi l'apprentissage interactif humain-robot peut améliorer la performance de ces agents, au prix d'une perte certaine d'autonomie.

3.1 Définitions

3.1.1 Types d'interactions enseignant-élève

L'apprentissage interactif est un processus collaboratif qui implique la participation d'au moins deux agents désignés comme étant l'enseignant et l'élève (Thomaz et Breazeal 2008, Vollmer et al. 2014). D'un côté, l'enseignant doit organiser sa stratégie d'enseignement afin de communiquer efficacement la tâche à l'élève, et de l'autre, l'élève doit participer activement au processus d'apprentissage et aider l'enseignant à s'améliorer en lui fournissant des informations pertinentes. Quatre grandes idées communément rencontrées dans la littérature sur l'apprentissage interactif structurent les types d'interactions que peuvent réaliser l'enseignant et l'élève : la *transparence*, l'*apprentissage actif*, l'*étayage* et le *façonnage*.

La transparence

La *transparence* décrit le mécanisme par lequel l'élève tient au courant l'enseignant de sa progression dans la résolution de la tâche. Pour l'enseignant, cela représente une source d'informations importante lui permettant de progresser lui-même dans son enseignement, et ainsi de guider de manière plus fine l'élève dans son apprentissage (Thomaz et Breazeal 2006).

L'apprentissage actif

L'*apprentissage actif* décrit le processus par lequel l'élève va interroger l'enseignant sur les aspects de la tâche qu'il maîtrise mal ou dont la compréhension lui est difficile (Settles 2009). C'est ce caractère *actif* qui le différencie de l'idée de *transparence*. À l'inverse de la *transparence*, l'*apprentissage actif* va permettre de cibler des aspects importants de la tâche qui pourraient être manqués par l'enseignant, et que seul l'élève a connaissance du fait qu'il est celui qui réalise la tâche.

L'étayage

L'*étayage*, qui est issu de la littérature sur le développement de l'enfant, décrit le processus qui permet à un enfant ou à un novice de résoudre un problème, de mener à bien une tâche ou d'atteindre un but qui aurait été, sans cette assistance, au-delà de ses capacités (Wood et al. 1976). Bruner (2015) dénombre six fonctions à l'*étayage* : l'enrôlement, qui consiste

à faire adhérer l'élève à la résolution de la tâche ; la réduction des degrés de liberté, qui consiste à simplifier la tâche de l'élève ; le maintien de l'orientation, qui consiste à motiver l'élève à rester concentrer sur la tâche ; la signalisation des caractéristiques déterminantes, qui consiste à faire comprendre à l'élève ce qu'il aurait pu faire pour mieux réussir la tâche ; le contrôle de la frustration, qui permet de rassurer l'élève lorsqu'il commet des erreurs ; et la démonstration, qui consiste à réaliser la tâche devant l'élève. Pour l'enseignant, l'*étayage* consiste donc essentiellement à prendre en main les éléments du problème qui excèdent pour le moment les capacités de l'élève, lui permettant ainsi de concentrer ses efforts sur les éléments qu'il maîtrise.

Le façonnage

L'idée du *façonnage* est issu de la littérature en psychologie et fût proposé pour la première fois comme une méthode de dressage des animaux dans le cadre du conditionnement opérant (Skinner 2019). Nous aurons l'occasion de parler plus longuement du conditionnement opérant dans le chapitre 4. Concernant l'intelligence artificielle, les premiers travaux s'intéressant au *façonnage* datent des années 90, avec Gullapalli et Barto (1992), Singh (1992) et Dorigo et Colombetti (1994). Parmi les plus récents, nous pouvons citer Knox et Stone (2009), Judah et al. (2010) et Cederborg et al. (2015). Concrètement, le *façonnage* consiste à faire converger le comportement de l'élève vers un comportement désiré par l'enseignant. Cette idée est principalement utilisée dans le cadre de l'apprentissage par renforcement comme un mécanisme permettant d'accélérer le processus d'apprentissage en fournissant à l'agent des récompenses intermédiaires. Ce mécanisme que l'on nomme *reward shaping*, et qui a été présenté dans le chapitre 2, sera explicité dans un cadre interactif plus loin dans le chapitre.

Globalement, l'idée générale derrière le *façonnage* est très proche de celle de l'*étayage*, qui implique en plus des mécanismes complexes de structuration de l'environnement (Saunders et al. 2006, Thomaz et Breazeal 2008, Argall et al. 2011). Comme nous le verrons dans le chapitre 7, la tâche d'interaction humain-robot que nous avons définie peut totalement être réalisée sans l'enseignant. Or, l'*étayage* considère que l'élève peut, grâce à l'enseignant, atteindre un objectif qu'il n'aurait pas pu réaliser sans lui. Dans ce manuscrit, nous considérons donc que notre tâche expérimentale s'inscrit davantage dans l'idée du *façonnage* que dans celle de l'*étayage*.

3.1.2 Différentes manières d'apprendre de l'humain

Les interactions humain-robot (*HRI* pour *Human-Robot Interactions* en anglais) font l'objet d'un champ de recherche interdisciplinaire à la frontière entre la psychologie, la robotique et l'ergonomie, où l'humain devient l'enseignant et où le robot devient l'élève. Dans la littérature sur l'apprentissage interactif en robotique (Knox et Stone 2009, Judah et al. 2010, Griffith et al. 2013), trois grandes manières d'apprendre d'un être humain pour un robot sont communément définies : l'*apprentissage par les*

conseils, l'*apprentissage par la démonstration* et l'*apprentissage par les rétroactions évaluatives*.

L'apprentissage par les conseils

L'*apprentissage par les conseils* est des trois manières d'apprendre celle qui est la moins bien définie. En effet, la définition de ce qu'est un *conseil* est très vague. Pradyot (2012) définissent les conseils comme toutes entrées externes au robot qui pourraient être utilisées par celui-ci pour prendre des décisions, et modifier la progression de son exploration ou renforcer sa croyance en une politique. Cela représente donc à la fois des préférences d'action ou d'état qui pourraient être transmises au robot, mais aussi plus directement des instructions, des démonstrations, des retours sur ses décisions, etc.

Deux types de conseils sont néanmoins catégorisés :

— Les *conseils généraux*, qui regroupent les informations générales sur la tâche que l'humain peut transmettre au robot, telles que les limites de ses capacités, des informations sur des caractéristiques de l'environnement, etc. Ces informations ont de commun le fait qu'elles soient auto-suffisantes, qu'elles ne dépendent pas de l'état dans lequel se trouve le robot, et qu'elles peuvent donc lui être transmises n'importe quand, c'est-à-dire avant même qu'il commence à réaliser la tâche. Ces informations prennent généralement la forme d'instructions conditionnelles (*Si ... Alors ...*) (Maclin et Shavlik 1996).

— Les *conseils contextuels*, qui comme leur nom l'indique, dépendent cette fois-ci du contexte, et donc de l'état dans lequel se trouve le robot. Ils regroupent entre autre les conseils d'orientation (Thomaz et al. 2006) et les retours sur les décisions du robot (Whitehead 1991, Judah et al. 2010).

Nous ne rentrerons pas plus dans le détail de l'*apprentissage par les conseils* dans ce manuscrit. Notons simplement que récemment, Najar et Chetouani (2020) ont présenté une revue des méthodes existantes permettant d'intégrer les conseils humains à l'apprentissage par renforcement. Étant donnée la variété de ce qui peut être considéré comme un *conseil*, les auteurs commencent d'abord par proposer une taxonomie des différents types de conseils qui peuvent être transmis à un agent. Ensuite, ils décrivent les différentes méthodes qui peuvent être utilisées pour interpréter et intégrer ces conseils à un processus d'apprentissage. Certaine de ces méthodes, dites de *shaping*, sont utilisées dans le cadre de l'*apprentissage par les rétroactions évaluatives*, et seront présentées plus en détails dans la partie associée de ce chapitre.

L'apprentissage par la démonstration

L'*apprentissage par la démonstration* (abrégé *ApD* par la suite), aussi connu sous l'appellation *apprentissage par imitation*, est apparu dans les années 80 comme une méthode de programmation de robots industriels (Lozano-Perez 1983). Depuis, elle a donné lieu à de nombreux travaux en robotique visant à développer des méthodes d'enseignement intuitives

pour des utilisateurs non experts (Billard et al. 2008, Chernova et Thomaz 2014). L'idée au centre de l'*ApD* est celle d'un humain enseignant à un robot en lui donnant des exemples de parties de la tâche à accomplir, soit en lui montrant quoi faire, soit en prenant contrôle de son comportement. Généralement, ces exemples sont définis par des séquences de couple état-action $(s_0, a_0), ..., (s_t, a_t)$. L'objectif de l'agent est alors de généraliser depuis cette démonstration et d'apprendre une politique $\pi : S \rightarrow A$, où π couvre tous les états, qui lui permettra d'imiter la démonstration (Knox et al. 2011).

L'apprentissage par les rétroactions évaluatives

L'*apprentissage par les rétroactions évaluatives* (abrégé *ApE* par la suite) consiste à faire des retours au robot sur la qualité de ses actions en prenant en considération les objectifs qu'il doit atteindre. Dans la littérature, il existe plusieurs manière de représenter les rétroactions évaluatives : par une valeur scalaire $f \in [-1, 1]$ (Knox et Stone 2009), par une valeur binaire $f \in -1, 1$ (Thomaz et al. 2006, Najar et al. 2020), par un renforcement positif $f \in -"Bravo!", "Correct!"$ (Kaplan et al. 2002) ou encore par une information catégorielle $f \in Correct, Faux$ (Griffith et al. 2013, Loftin et al. 2016). Ces retours vont être ensuite traduits par le robot et pris en compte dans le calcul de ses futures décisions. D'un point de vue pratique, l'*ApE* est largement considéré comme une technique de *shaping*, et s'intègre donc totalement au cadre des interactions de type *façonnage* présentées précédemment.

Bien que cette taxonomie soit couramment utilisée dans la littérature, nous pouvons voir que les différentes catégories peuvent se chevaucher facilement, et notamment du fait de l'*apprentissage par les conseils*, qui regroupe un ensemble très hétérogène d'informations pouvant être transmises au robot. Par exemple, dans certains travaux (Lin 1991, Whitehead 1991), les *démonstrations* sont considérées comme des *conseils* du fait qu'elles ne sont pas exécutées par l'humain mais simplement communiquées au robot. Étant des informations transmises au robot, les *rétroactions évaluatives* sont donc aussi parfois considérées comme des *conseils* (Whitehead 1991, Judah et al. 2010, Griffith et al. 2013).

Dans ce chapitre, nous nous intéressons avant tout à la manière dont l'*ApE* peut être utilisé par le robot pour biaiser sa prise de décision et à ses avantages et inconvénients face à l'*ApD*.

3.2 Méthodes de shaping pour l'apprentissage par les rétroactions évaluatives

Plusieurs manières de dériver une politique à partir de rétroactions évaluatives humaines sont référencées dans la littérature, souvent associées à une terminologie incluant la notion de *shaping*, qui définit le fait d'influencer le robot vers un comportement souhaité. Techniquement, pour le robot, cela consiste à intégrer des entrées humaines dans le calcul

de ses stratégies comportementales : la rétroaction évaluative de l'humain est combinée à une source d'information interne au robot, telle qu'une fonction de récompense (Thomaz et al. 2006), une fonction de valeur (Knox et Stone 2010) ou une politique (Griffith et al. 2013). Nous parlons alors de *reward shaping*, de *value shaping* ou de *policy shaping*. Empruntant à la terminologie présentée dans le chapitre 2, ces méthodes peuvent être *model-free* ou *model-based*. Dans les méthodes de *shaping model-free*, la rétroaction évaluative perçue par l'agent est directement intégrée au processus d'apprentissage (Zaraki et al. 2019), tandis que dans les méthodes de *shaping model-based*, un modèle de l'enseignant humain est construit et maintenu en parallèle du modèle de la tâche de l'agent (Knox et Stone 2012a). Dans certains cas, la rétroaction évaluative constitue l'unique source d'information et n'est combinée avec aucune autre. Malgré tout, ces méthodes peuvent aussi être considérées comme du *shaping*. En effet, combiner la rétroaction évaluative de l'humain avec aucune source d'information revient à la combiner avec des fonctions de récompense ou de valeur nulles, ou avec une politique uniforme. C'est ce que font Loftin et al. (2014) dans leur étude, qui ne décrivent pas leur méthode comme étant du *shaping*.

3.2.1 Reward shaping

Dans le cadre du *reward shaping*, après sa conversion en valeur numérique, la rétroaction évaluative est considérée comme une récompense supplémentaire de même nature que les récompenses du Processus de Décision Markoviens (MDP), qui sera donc prise en compte dans le calcul de la fonction de valeur. De ce fait, la rétroaction évaluative n'aura pas simplement un effet sur la dernière action a réalisée par le robot dans l'état s, mais se propagera grâce à la fonction de valeur qui diverge de la fonction de récompense. Techniquement, faire du *model-free reward shaping* consiste à définir une nouvelle fonction de récompense $R'(s,a)$ en sommant la récompense donnée par l'humain à la fonction de récompense du robot (Isbell et al. 2001, Thomaz et al. 2006, Mathewson et Pilarski 2016) :

$$R'(s,a) = R(s,a) + R^H \tag{3.1}$$

où $R(s,a)$ est la récompense du MDP obtenue par le robot lorsqu'il effectue l'action a dans l'état s et R^H la récompense instantanée transmise par l'humain.

Knox et Stone (2009; 2010) proposent quant à eux, avec leur algorithme TAMER, une méthode de *model-based reward shaping* où la nouvelle fonction de récompense $R'(s,a)$ est calculée comme ceci :

$$R'(s,a) = R(s,a) + \beta \times \hat{H}(s,a) \tag{3.2}$$

où $R(s,a)$ est la récompense du MDP obtenue par le robot lorsqu'il effectue l'action a dans l'état s, $\hat{H}$ un modèle de régression appelé *fonction de renforcement humain* calculé en convertissant les rétroactions évaluatives en récompense, et β un paramètre permettant de pondérer le poids de la récompense prédite par l'humain $\hat{H}(s,a)$ pour la paire d'état-action (s,a).

Si de nombreuses études ont montré l'efficacité du *reward shaping* dans le support de l'apprentissage d'agents (Gullapalli et Barto 1992, Isbell

et al. 2001, Thomaz et al. 2006, Mathewson et Pilarski 2016), Knox et Stone (2012a) ont pour leur part pointé du doigt le fait que cette efficacité était très dépendante de la valeur du facteur d'actualisation γ, qui permet de prendre en compte l'horizon temporel des récompenses futures (voir chapitre 2). Très souvent, lorsqu'un agent doit apprendre à résoudre des tâches épisodiques basées sur des objectifs (Sutton et Barto 1998), les meilleures performances sont atteintes avec une valeur de γ élevée, signe que prendre en compte les récompenses lointaines est prolifique pour l'agent. Knox et Stone (2012a) montrent pourtant qu'en faisant du *reward shaping* avec un agent effectuant de l'apprentissage par renforcement indirect, ils obtiennent en moyenne de meilleures performances avec de très petites valeurs de γ (où la performance est mesurée comme l'accumulation de la récompense du MDP). Pour les hautes valeurs de γ, ils observent régulièrement le problème de *cycles positifs* présenté dans le chapitre 2, c'est-à-dire le fait que l'agent reste coincé dans des minima locaux en répétant sans cesse des comportements sous-optimaux (Randløv et Alstrøm 1998). Ce problème découle du fait que l'agent n'est pas capable de faire la distinction entre les récompenses du MDP (qui sont les seules récompenses permettant de remplir l'objectif) et les récompenses intermédiaires transmises par l'humain. D'après eux, la positivité de la récompense humaine conduit inévitablement à des circuits comportementaux infinis qui peuvent potentiellement éviter les objectifs (les récompenses du MDP) : l'agent va apprendre à accumuler de la récompense plutôt qu'à remplir les objectifs. Et par conséquent, Knox et Stone (2012a) arguent le fait que la récompense humaine ne peut pas être apprise naïvement comme s'il s'agissait d'une récompense de MPD conventionnelle.

3.2.2 Value shaping

Pour éviter les problèmes de *cycles positifs*, la solution consiste donc à faire du *reward shaping* avec une valeur de γ de 0, ce qui revient à considérer les rétroactions évaluatives comme des récompenses instantanée. Ainsi, les rétroactions évaluatives deviennent des renforcements immédiats en réponse aux actions effectuées par l'agent, et les récompenses associées à ces rétroactions se muent en valeurs d'état-action (Ho et al. 2017). C'est ce que proposent de faire les méthodes de *value shaping*, qui consistent à considérer la rétroaction évaluative comme un indice de préférence d'une paire d'état-action (s, a), qui se matérialise sous la forme d'un bonus ou d'un malus (d'un biais positif ou négatif), plutôt que comme une récompense. Cette représentation numérique de la rétroaction évaluative permet de modifier directement la fonction de valeur d'état-action $Q(s, a)$. Techniquement, faire du *model-free value shaping* consiste à définir une nouvelle fonction de valeur d'état-action $Q'(s, a)$ en sommant la préférence transmise par l'humain à la fonction des états-actions du robot (Maclin et Shavlik 1996, Maclin et al. 2005) :

$$Q'(s, a) = Q(s, a) + Q^H \tag{3.3}$$

où $Q(s, a)$ est la Q-valeur associée à la réalisation de l'action a dans l'état s et Q^H la préférence transmise par l'humain.

Knox et Stone (2010), Knox et al. (2011), Knox et Stone (2012b) proposent toujours avec leur algorithme TAMER une méthode de *model-based value shaping* qu'ils nomment *Q-Augmentation*, où la nouvelle fonction de valeur d'état-action $Q'(s,a)$ est calculée comme ceci :

$$Q'(s,a) = Q(s,a) + \beta \times \hat{H}(s,a) \tag{3.4}$$

où $Q(s,a)$ est la Q-valeur associée à la réalisation de l'action a dans l'état s, $\hat{H}$ un modèle de régression calculé en convertissant les rétroactions évaluatives en préférences et β un paramètre permettant de pondérer le poids de la préférence prédite par l'humain $\hat{H}(s,a)$ pour la paire d'état-action (s,a).

Dans Knox et Stone (2012b), les auteurs montrent que plus les rétroactions évaluatives humaines affectent directement le processus de sélection des actions du robot, plus le robot est performant, et plus elles affectent la mise-à-jour des valeurs d'état-action pour chaque transition expérimentée, moins celui-ci sera performant. Pour les méthodes de *value shaping*, cela se traduit par des problèmes de convergence dus au fait que les informations combinées dans l'équation 3.4 ne sont pas tout à fait identiques. En effet, si le biais transmis par l'humain informe seulement de la préférence d'une paire d'état-action (s,a), la fonction de valeur d'état-action $Q(s,a)$ informe quant à elle aussi de la proximité de l'objectif (Ho et al. 2015).

3.2.3 Policy shaping

Pour se rapprocher au plus près du processus de sélection des actions du robot comme préconisé par Knox et Stone (2012b), la solution consiste à utiliser des méthodes de *policy shaping*. Ici, la rétroaction évaluative de l'humain est utilisée pour biaiser la politique de l'agent sans interférer avec les fonctions de récompense ou de valeur d'état-action. Dans (Najar et al. 2020), les auteurs proposent une méthode de *model-free policy shaping* où les rétroactions évaluatives de l'humain sont utilisées pour calculer la nouvelle valeur de préférence $p'(s,a)$ de l'acteur d'un algorithme acteur-critique :

$$p'(s,a) = p(s,a) + \beta \times f \tag{3.5}$$

où $p(s,a)$ est la valeur de préférence associée à la transition expérimentée (s,a), $f \in [-1,1]$ la rétroaction évaluative transmise par l'humain et β un paramètre permettant de pondérer le poids de f.

Knox et Stone (2010), Knox et al. (2011), Knox et Stone (2012b) proposent quand à eux une méthode de *model-based policy shaping* qui consiste toujours à biaiser la valeur d'état-action associée à l'action préférée par l'humain comme avec la méthode *Q-Augmentation* 3.4, mais cela uniquement durant le processus de décision, et sans modifier la fonction de valeur d'état-action : c'est la méthode *Action Biaising*. Concrètement, lorsque l'agent doit utiliser une règle de sélection pour définir l'action qu'il va exécuter, la Q-valeur associée à la préférence de l'humain sera biaisée par rapport aux Q-valeurs des autres actions. Par exemple, si la règle de sélection est une fonction *argmax*, alors :

$$a^* = argmax_a[Q(s,a) + \beta \times \hat{H}(s,a)] \tag{3.6}$$

où a^* est l'action sélectionnée par l'agent, $Q(s,a)$ la Q-valeur évaluée par l'agent et $\hat{H}(s,a)$ la préférence prédite par l'humain pour la paire d'état-action (s,a).

Les mêmes auteurs proposent une manière alternative de faire du *model-based policy shaping*, en permettant à l'agent d'arbitrer entre les décisions de l'humain et du robot sur la base d'un critère de probabilité. Cette méthode, qui emprunte à l'*ApD*, est appelée *Control Sharing* :

$$Prob(a = argmax_a[\hat{H}(s,a)]) = min(\beta, 1) \tag{3.7}$$

où $argmax_a[\hat{H}(s,a)]$ est la préférence prédite par l'humain et β est le seuil qui permet de déterminer la probabilité de sélection de la décision de l'humain plutôt que celle du robot.

Loftin et al. (2016) proposent une autre méthode de *model-based policy shaping* où la rétroaction évaluative de l'humain n'est pas convertie en une valeur numérique, mais en une information catégorielle de type $Correct, Faux$. La distribution de la rétroaction fournie est utilisée dans un cadre bayésien afin de dériver directement une politique sans passer par une fonction de valeur.

Quant à Griffith et al. (2013), ils proposent une méthode inspirée des méthodes ensemblistes présentées dans le chapitre 2, et plus précisément de la technique *Boltzmann Multiplication*, qui définit la préférence d'une action comme le produit de ses probabilités par chaque *expert*, où les experts sont ici l'humain et le robot. Griffith et al. (2013) montre que cette méthode est plus efficace que les méthodes de *reward shaping*, d'*Action Biaising* et de *Control Sharing* présentées précédemment.

Finalement, les études comparatives de Griffith et al. (2013) et Ho et al. (2015) confirment la conclusion de Knox et Stone (2012b) concernant le fait que plus une méthode modifie le modèle de la tâche (*reward shaping*, *value shaping*), et moins cette méthode sera efficace. On aura donc plutôt tendance à s'orienter vers le *policy shaping*, qui a en plus l'avantage de s'appliquer aux méthodes qui dérivent directement des politiques sans calculer de fonction de valeur ou sans même utiliser de récompense. C'est le choix que nous ferons dans le chapitre 7 de ce manuscrit.

La figure 3.1, adaptée d'une figure de Najar et Chetouani (2020), résume les différences et les similitudes des méthodes de *shaping* que nous avons présentées.

3.3 Apprentissage par les rétroactions évaluatives et apprentissage par la démonstration : forces et faiblesses

Dans Knox et al. (2011), les auteurs comparent sous différentes conditions les performances d'agents résolvant les problèmes du *Cart Pole* et du *Mountain Car* (Sutton et Barto 1998), et apprenant d'un humain durant une phase d'entraînement, soit par démonstration, soit par rétroactions évaluatives.

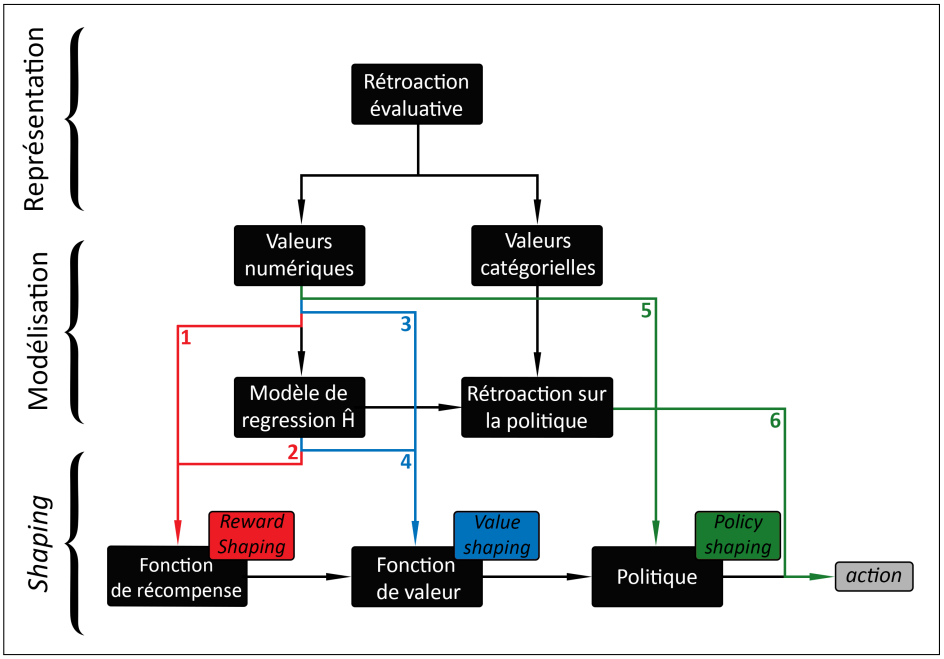

FIGURE 3.1 – *Méthodes de shaping pour les rétroactions évaluatives. 1 : model-free reward shaping. 2 : model-based reward shaping. 3 : model-free value shaping. 4 : model-based value shaping (Q-Augmentation). 5 : model-free policy shaping. 6 : model-based policy shaping (Action Biaising, Control Sharing). Inspirée d'une figure de Najar et Chetouani (2020).*

Knox et al. (2011) montrent que dans tous les cas de figures, les agents obtiennent de meilleures performances avec l'*ApD*. L'explication tient du fait que durant l'*ApD*, l'humain prend directement le contrôle du comportement du robot pour plusieurs actions consécutives, alors pour l'*ApE*, le robot garde le contrôle sur son comportement et l'humain ne peut que lui faire des retours indirects sur les actions qu'il choisit. De la même manière, l'*ApE* met plus longtemps à exhiber des performances intéressantes par rapport à l'*ApD*, qui affiche rapidement de bonnes performances. D'un point de vue plus pratique maintenant, l'*ApD* a l'avantage de pouvoir plus ou moins ressembler à un jeu-vidéo pour l'enseignant humain, un loisir de plus en plus pratiqué aujourd'hui. De ce fait, l'interface d'utilisation peut être particulièrement facile à prendre en main pour l'enseignant. À l'inverse, contrôler un interface d'*ApE* peut demander plus de pratique. Les auteurs suspectent aussi qu'en contrôlant directement le robot, l'*ApD* peut permettre à l'enseignant d'élaborer plus facilement une stratégie efficace par rapport à l'*ApE*, qui repose sur des interactions indirectes.

Bien sûr, cette intuition est très dépendante de la tâche : nous pourrions imaginer des tâches où contrôler le robot avec l'*ApD* serait très compliqué (par exemple un bras robotique à plusieurs dimensions), et où le caractère indirect de l'*ApE* permettrait justement de faciliter l'interaction. Et pour continuer sur les interfaces, l'interface de l'*ApE* peut aussi avoir pour avantage de pouvoir demeurer constante d'une tâche à l'autre, alors que l'interface de contrôle de l'*ApD* est de fait très dépendante de la nature de la tâche et du robot. Un autre avantage de l'*ApE* est le fait qu'il demande moins d'expertise sur la tâche que l'*ApD*, dû au fait que l'on peut intuitivement évaluer l'avantage global des conséquences d'une ac-

tion sans savoir qu'elle est l'action optimale. L'*ApD* demande pour sa part à l'enseignant de prendre le contrôle du robot pour réaliser la tâche à sa place, et donc de planifier la succession d'actions qu'il va réaliser. Concrètement, l'enseignant doit donc raisonner au niveau du *modèle de transition* de la tâche, et les auteurs suspectent que l'*ApE* demande une charge cognitive plus faible à l'enseignant. Pour finir, durant l'*ApE*, le robot montre à l'enseignant la politique qu'il apprend, ce qui pourra permettre à l'humain de prendre en compte les éventuelles forces et faiblesses de cette politique durant son interaction. À l'inverse, l'*ApD* ne permet généralement pas de montrer à l'enseignant la politique du robot, et force donc l'humain à enseigner de manière moins informée. D'ailleurs, Knox et al. (2011) montrent dans leurs expériences que la performance de la politique finalement apprise grâce aux rétroactions évaluatives est *au moins* aussi bonne que la performance atteinte durant l'entraînement avec l'humain en *ApE*, alors que la performance de la politique apprise grâce aux démonstrations est *au mieux* aussi bonne que la performance atteinte durant l'entraînement avec l'humain en *ApD*. Cela illustre l'écart de performance qu'il peut exister entre la politique du robot et celle de l'humain dans le cadre de l'*ApD*.

Finalement, nous comprenons que bien que l'*ApD* affiche des performances meilleures que l'*ApE*, les forces et les faiblesses de ces deux manières d'apprendre sont très dépendantes de la tâche que le robot doit réaliser et de la nature même du robot. Ces facteurs, associés à d'autres tels que la maîtrise de l'interface par l'enseignant, l'expertise nécessaire de la tâche et les ressources cognitives disponibles, seront déterminants dans le choix de l'apprentissage que l'on souhaite réaliser. Pour tirer avantage des bénéfices de chacune des manières d'apprendre, certaines études proposent donc un apprentissage interactif mixte entre *ApD* et *ApE* (Argall et al. 2007, Li et al. 2018, Ezzeddine et al. 2018, Mourad et al. 2020).

3.4 Conclusion

Dans ce chapitre, nous nous sommes intéressés à comment un agent artificiel effectuant de l'apprentissage par renforcement pouvait interpréter des interactions humaines afin de biaiser son système de prise de décision. Parmi les quelques manières d'apprendre d'un enseignant humain référencées dans la littérature, nous nous sommes concentrés avant tout sur l'*apprentissage par les rétroactions évaluatives* et les différentes méthodes de *shaping* associées, qui s'intègrent très bien au cadre de l'apprentissage par renforcement. Nous expliquons pourquoi, d'après la littérature, les méthodes de *policy shaping* affichent de meilleures performances que les méthodes de *reward shaping* et de *value shaping*. Nous nous sommes aussi intéressés à des études comparant l'*apprentissage par les rétroactions évaluatives* avec l'*apprentissage par la démonstration*, une autre manière d'apprendre d'un enseignant humain décrite comme étant plus performante, afin de montrer que le choix de l'apprentissage interactif à adopter dépendait avant tout beaucoup du robot utilisé et de la tâche qu'il devait effectuer.

Ne pouvant pas aborder en seul chapitre toute l'immensité des études réalisées sur l'apprentissage interactif, nous avons fait le choix de ne pas

mentionner un grand nombre de sujets, tels que ceux s'intéressant à la manière dont le robot peut capter et interpréter le regard de l'humain (Cambuzat et al. 2018, Khamassi et al. 2018a;b), ou encore les recherches s'intéressant aux substrats neuronaux de cette forme apprentissage. Nous noterons simplement que dans Burke et al. (2010), les auteurs proposent que l'*apprentissage par la démonstration* puisse être expliqué par deux formes d'erreurs de prédiction relatives à l'observation de l'humain, et jusqu'alors non caractérisées : les erreurs de prédiction d'actions observationnelles (le choix réel moins le choix prédit par le partenaire) et les erreurs de prédiction de résultats observationnels (le résultat réel moins la prédiction du résultat reçu par le partenaire). Dans une expérience d'IRM fonctionnelle, ils montrent que l'activité cérébrale dans le *cortex préfrontal dorsolatéral* et le *cortex préfrontal ventromédial* correspondrait respectivement à ces deux signaux d'apprentissage observationnel distincts. Nous mentionnons cette étude du fait que nous reviendrons dans le chapitre 4 sur ces substrats neuronaux, étant tout deux centraux dans l'apprentissage animal, qu'il soit interactif ou non.

Nous rappelons que dans ce chapitre, les différentes méthodes d'apprentissage interactif que nous avons présentées sont comparées selon leur capacité à permettre à un agent *artificiel* de résoudre efficacement des tâches expérimentales. Si, concernant l'*apprentissage par les rétroactions évaluatives*, les méthodes de *policy shaping* sont évaluées comme étant les plus optimales, cela ne signifie pas qu'elles décrivent correctement le même mécanisme d'apprentissage interactif effectué par un humain bien *réel*, et potentiellement non optimal. Par exemple, Najar et al. (2019) montrent avec des données expérimentales que les méthodes de *value shaping* modélisent mieux l'*apprentissage par la démonstration* de sujets humains que les méthodes de *policy shaping*. Leur conclusion pourrait expliquer en partie la force et l'omniprésence des phénomènes liés à l'influence sociale, et computationnellement, cela s'expliquerait du fait que les méthodes de *value shaping*, à l'inverse des méthodes de *policy shaping*, modifient directement le modèle de valeur d'état-action des sujets, et ont donc des conséquences plus immuables sur ses croyances et son comportement.

Finalement, nous nous sommes donc ici avant tout intéressés à la manière dont l'apprentissage interactif pouvait agir sur les processus d'apprentissage par renforcement d'un agent artificiel. En effet, l'apprentissage par renforcement et l'amélioration de l'autonomie décisionnelle de robots restent les sujets d'étude principaux de notre travail de recherche. Dans le chapitre 7 de ce manuscrit, nous aurons l'occasion d'évaluer l'architecture cognitive de contrôle que nous avons développée avec une tâche d'interaction humain-robot, et de montrer en quoi celle-ci est capable de tirer judicieusement avantage des bénéfices de l'apprentissage de l'enseignant, sans perdre en autonomie décisionnelle.

Études et modèles computationnels du comportement animal

4

Sommaire

Dans ce chapitre, nous nous intéressons à l'étude du comportement des mammifères en psychologie et en neurosciences, et aux modélisations proposées pour les comprendre. Dans un premier temps, nous retraçons succinctement l'histoire du béhaviorisme, du conditionnement pavlovien à la mise en évidence computationnelle et neuro-anatomique de la dualité entre comportements dirigés vers un but et *habitudes comportementales*. Dans un second temps, nous montrons les liens forts qui existent entre l'apprentissage par renforcement, présenté dans le chapitre 2, et les comportements instrumentaux. Suite à ça, nous faisons une revue des études s'étant intéressées à la modélisation des comportements instrumentaux en adressant la question des mécanismes de leur coordination, et examinons des modèles issus d'études du comportement instrumental, de navigation ou de robotique. Finalement, nous discutons de la notion d'*habitudes comportementales* et de la manière dont celles-ci sont modélisées. En effet, certains auteurs pointent le fait que les algorithmes d'apprentissage par renforcement reproduisent imparfaitement les données associées aux *habitudes comportementales* observées chez des animaux.

4.1 Étude du comportemental animal

4.1.1 Naissance du Béhaviorisme

Historiquement, le béhaviorisme est apparu vers la fin du 19^{e} siècle et le début du 20^{e} siècle en réaction aux approches dites « mentalistes », qui visaient jusqu'alors à comprendre le fonctionnement de l'esprit humain (et plus particulièrement celui de la conscience) en utilisant largement l'introspection, et qui peinaient à produire des énoncés scientifiques empiriquement testables. À une époque où le concept de conscience est de plus en plus remis en cause aux États-Unis, le psychologue américain John Broadus Watson, pour qui le seul objet d'étude est le comportement, et non la conscience (les états mentaux), établit en 1913 les principes de base du béhaviorisme (Watson 1913). Pour Watson, l'objectif de la science du comportement est d'étudier les relations entre les stimuli (S) de l'environnement et les comportements réponses (R) qu'ils provoquent, qui donnent lieu au modèle du comportement *S-R*. Sans nier l'existence d'un individu recevant les stimuli et générant les comportements réponses, les béhavioristes ne s'en préoccupent pas, considérant l'individu comme une « boîte noire » et écartant toutes les questions relatives à ses états mentaux. Cette position de principe correspond à ce qui fût par la suite appelé le *béhaviorisme méthodologique*, permettant ainsi de le différentier des autres courants auxquels il donnera naissance.

À la même époque en 1917, le psychologue américain Edward Thorndike découvre la notion d'essai-erreur avec sa fameuse *Boîte de Thorndike* (Thorndike 1970) : un chat affamé est placé dans une boîte grillagée lui permettant d'observer de la nourriture à l'extérieur, et dont la porte peut être ouverte en tirant sur une chaîne puis en pressant un levier. Celui-ci doit découvrir ce mécanisme afin de pouvoir sortir de la cage et accéder à la nourriture. Thorndike observe que le chat exhibe dans un premier temps des comportements sans stratégies apparentes (tourner en rond, griffer la cage, miauler, etc.), jusqu'à découvrir par hasard comment sortir de la cage. Ensuite, à chaque fois que le chat a faim, il est remis dans la cage, et devient alors capable de sortir de plus en plus rapidement de celle-ci. À l'époque, les conceptions dominantes concernant l'intelligence des chiens et des chats suggéraient que leur capacité de résolution de problème était largement due au hasard. Ces résultats ont donc été vivement critiqués, en pointant du doigt le fait que l'expérience empêchait le chat de procéder autrement que par le hasard pour résoudre la tâche, et que le fait de réussir à sortir de la cage n'était qu'un artefact de ce comportement hasardeux. L'objection la plus sérieuse provient de l'allemand Wolfgang Köhler, qui met en avant à la même époque l'*apprentissage soudain* (*insight* en anglais) chez le chimpanzé (Köhler 1917), c'est-à-dire la découverte soudaine de la solution à un problème, sans passer par une série d'essais-erreurs progressifs. Suite à ça, Thorndike développa de nombreuses expériences similaires sur l'être humain, qui lui permirent de consolider ses résultats et de formuler plusieurs grandes lois de l'apprentissage telles que la *Loi de l'Effet*, qui établit que le comportement est fonction de ses conséquences, et qui fut rejetée par un certain nombre de béhavioristes méthodologiques tel que Watson.

Vingt-cinq ans plus tôt en Russie, Ivan Pavlov, qui travaillait à l'époque sur la fonction gastrique des chiens et la chimie de la salive, observe le fait que ces animaux avaient tendance à saliver avant d'entrer réellement en contact avec les aliments. S'intéressant à ce phénomène, il réalise de nombreux travaux entre les années 1890 et 1900, qui durent attendre 1927 avant d'être entièrement traduits (Pavlov 2010) 4.1. Son expérience la plus connue consiste dans un premier temps à présenter de la nourriture à un chien (on parle de stimulus inconditionnel SI) afin d'observer sa salivation (on parle de réponse inconditionnelle RI). Dans un second temps, un stimulus neutre (SN) tel le son d'une cloche, est associé au stimulus inconditionnel, et donc au fait de donner de la viande au chien. L'association de ces deux stimuli est appelée stimulus conditionnel (SC). Pavlov répète le stimulus conditionnel un certain nombre de fois jusqu'à observer qu'en enlevant le stimulus inconditionnel (c'est-à-dire en enlevant la viande), et en présentant uniquement le stimulus neutre au chien (c'est-à-dire le son de la cloche), celui-ci se met malgré tout à saliver comme si la viande était présente : la réponse inconditionnelle est devenue une réponse conditionnelle (RC), ou encore un *réflexe conditionnel*. Cette manière d'influencer le comportement de l'animal, utilisant le paradigme *S-R* de Watson, est appelée *conditionnement classique* ou *conditionnement répondant*. Rescorla (1972) proposeront bien des années plus tard un modèle computationnel du *conditionnement classique*, qui sera la source d'inspiration de nombreux autres algorithmes d'apprentissage par renforcement.

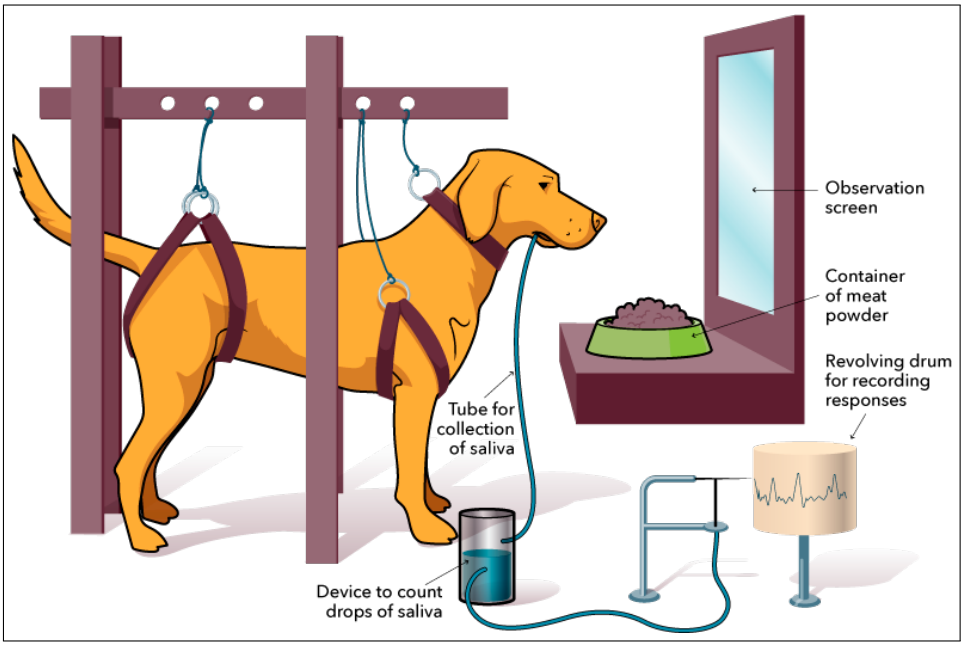

FIGURE 4.1 – *Illustration de l'expérience de Pavlov (Pavlov 2010).*

Comme beaucoup de psychologues de l'époque, Burrhus Frederic Skinner, un chercheur américain, considère que ni le *conditionnement répondant* de Pavlov, ni le modèle classique *S-R* de Watson ne permettent d'expliquer la majorité des comportements, et en particulier les comportements pour lesquels il n'y a pas de causes antérieures apparentes dans l'environnement. S'appuyant sur la *Loi de l'Effet* de Thorndike, Skinner développa les notions de renforcement, de façonnage et d'apprentissage

programmé. Ces principes marquent une divergence profonde avec le *béhaviorisme méthodologique* de Watson en acceptant l'idée que des variables internes à l'individu puissent finalement intervenir dans l'analyse du comportement. En effet, pour Skinner, l'approche pavlovienne pèche du fait qu'elle ne prenait pas en compte l'action de l'environnement sur l'individu après qu'une réponse aie été produite. Pour reprendre l'exemple du chien de Pavlov, Skinner considérait que si un chien pouvait effectivement saliver à la suite d'un son de cloche, cela pouvait être dû au fait de récompenser ou non l'acte de saliver. Avec cette notion de *récompense*, Skinner fait apparaître la notion de *contingence de renforcement* pour définir l'environnement qui va produire le comportement dit *opérant* du chien, c'est-à-dire un comportement qui va produire des conséquences renforçantes. Concrètement, si le *conditionnement répondant* de Pavlov définit une réponse comportementale *réflexe* d'un animal à un sitmulus, Skinner s'intéresse aux réponses comportementales *volontaires* de l'animal, prises en considération de l'environnement : c'est le *conditionnement instrumental* ou *conditionnement opérant* (Skinner 2019). La réponse de l'animal (R) est désormais liée aux stimuli (I) mais aussi à son utilité et à ses conséquences (C) dans le contexte actuel, de la même manière qu'un instrument est lié à la réalisation de la tâche pour laquelle il a été pensé, donnant ainsi lieu au modèle opérant *S-R-C*.

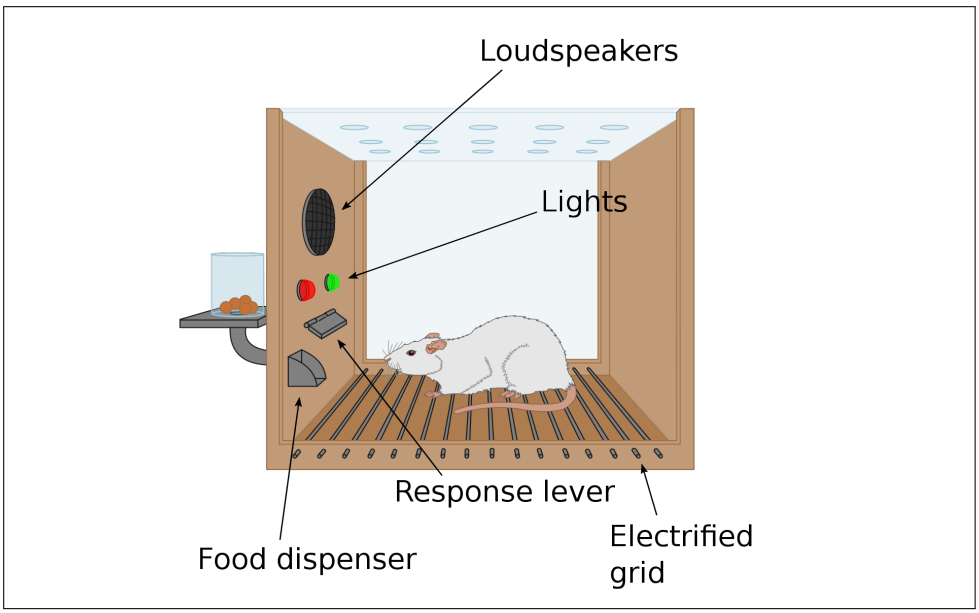

FIGURE 4.2 – *Illustration de la boite de Skinner (Skinner 2019).*

La *Boîte de Skinner* 4.2 permet de tester les capacités de rongeurs à subir plusieurs types de *conditionnements instrumentaux* :

— Le *renforcement positif*, où un rat dans une cage peut appuyer sur un levier pour recevoir de la nourriture. On observe ici une augmentation de la probabilité de voir le rat appuyer sur le levier.
— Le *renforcement négatif*, où un rat dans une cage reçoit des chocs électriques depuis le plancher, et peut appuyer sur un levier pour les faire cesser. On observe ici à nouveau une augmentation de la probabilité de voir le rat appuyer sur le levier.
— La *punition positive*, où un rat dans une cage peut appuyer sur un

levier pour recevoir un choc électrique. On observe ici une diminution de la probabilité de voir le rat appuyer sur le levier.
— La *punition négative*, où un rat est dans une cage avec de la nourriture, qui disparaît dès lors qu'il appuie sur le levier. On observe ici à nouveau une diminution de la probabilité de voir le rat appuyer sur le levier.

Dans les années 1950, Skinner pousse encore plus loin son idée béhavoriste en considérant les processus internes tels que les pensées ou les émotions comme des comportements, ce qui revient à dire que tout est comportement. Pour cette raison, le béhaviorisme de Skinner est appelé *béhaviorisme radical*. Au même moment, le professeur de psychologie Donald Hebb publie un livre (Hebb 1949) où il défend une conception biologique de la psychologie, et affirme qu'elle n'est ni plus ni moins que l'étude du système nerveux. Il y expose ses idées clés sur l'apprentissage et l'association entre les neurones : deux neurones en activité au même moment créent ou renforcent leur connexion de sorte que l'activation de l'un par l'autre sera plus facile à l'avenir. Cette idée est appelée *apprentissage hebbien* ou encore *règle de Hebb*, et est à l'origine des premiers systèmes artificiels capables d'apprendre par expérience, tel que le Perceptron (Rosenblatt 1958). Aujourd'hui, elle est toujours la pierre angulaire des systèmes de réseaux de neurones les plus sophistiqués, que nous avons rapidement abordés dans le chapitre 2.

Dans ce chapitre, nous nous concentrerons sur le *conditionnement instrumental* tel que définit par Skinner et sur les modèles qui visent à le décrire et à l'expliquer.

4.1.2 Études et substrats neuronaux du comportement instrumental

On distingue deux types de comportements instrumentaux :
— *Les comportements dirigés vers un but*, qui définissent les comportements informés de leurs conséquences et tournés vers l'obtention d'un effet désirable. Ils sont caractérisés par leur grande flexibilité et leur capacité à s'adapter aux changements environnementaux. Cela implique une délibération active de l'agent au prix d'un coût calculatoire mental élevé nécessaire à l'évaluation les conséquences de ses actions (Dayan 2009). Le mécanisme de contrôle associé est appelé *réponse-conséquence*.
— Les *habitudes comportementales*, qui définissent les comportements non informés de leurs conséquences futures, non tournés vers l'obtention d'un effet, mais simplement causés par un stimulus perceptuel. Elles sont caractérisées par leur automatisme, leur robustesse aux changements environnementaux, mais également par leur faible coût calculatoire mental. Le mécanisme de contrôle associé est appelé *stimulus-réponse*.

Comme nous pouvons le deviner, pouvoir coordonner à la volée ces deux types de stratégies comportementales offre un avantage évolutif certain à l'animal. Cela lui permet d'un côté de s'adapter efficacement aux changements environnementaux, et ainsi d'accomplir au mieux ses objectifs, mais aussi d'économiser des ressources calculatoires, et donc de la précieuse énergie (sous forme de lipides ou de glucides).

Les premières hypothèses sur la co-existence de ces deux types de comportements chez les mammifères viennent des théories des cartes cognitives, un type de représentation mentale permettant à un individu d'acquérir, de coder, de stocker et de décoder des informations relatives à son environnement (Tolman 1948). En effet, en observant des souris aller et venir entre plusieurs chemins dans un labyrinthe, le psychologue américain Tolman interprète ces comportements comme l'indication que l'animal évalue les conséquences attendues de plusieurs options. Cette hypothèse est confirmée par Adams et Dickinson (1981) qui montrent qu'un rat ayant appris à obtenir une récompense sous forme alimentaire en pressant un levier, arrête de le faire dès lors que l'aliment le rend malade, signe que celui-ci est capable de prendre instantanément en considération le changement environnemental (*comportement dirigé vers un but*). En revanche, Adams (1982) montre par la suite que le rat est aussi capable de devenir totalement insensible à la nocivité de la nourriture lorsqu'il est surentraîné, comme s'il appuyait sur le levier uniquement par habitude (*habitude comportementale*). À la suite de ces travaux, Dickinson (1985) met en lumière l'existence d'un transfert de contrôle entre ces deux types de comportements.

Pour identifier le comportement d'un animal dans un cadre expérimental, les chercheurs réalisent souvent des études de lésions. Ces études reposent sur des méthodes anciennes consistant à étudier les conséquences de lésions cérébrales en fonction de leur localisation, afin d'en déduire la fonction des aires lésées. Dans le cas des études visant à mettre en évidence les comportements instrumentaux, deux méthodes sont généralement utilisées et comparées chez les animaux lésés et intacts :

— la dévaluation de la récompense (*outcome devaluation* en anglais), qui consiste à rendre neutre ou répulsif l'élément qui était précédemment considéré comme une récompense (Adams et Dickinson 1981). Après une phase d'apprentissage sur une tâche récompensée, on applique la dévaluation durant une phase dite d'*extinction*, c'est-à-dire durant une phase où la récompense précédente n'est plus distribuée. Si l'animal arrête immédiatement de chercher à obtenir la nourriture, c'est qu'il a un *comportement dirigé vers un but*, puisqu'il est capable d'associer la *réponse* à la *conséquence*. S'il persiste, c'est qu'il a une *habitude comportementale*, puisqu'il n'est plus capable d'associer la *réponse* à la *conséquence*, mais seulement le *stimulus* à la *réponse*.

— la dégradation de la contingence (*contingency degradation* en anglais), qui consiste à modifier les conséquences d'une action après une phase d'apprentissage, par exemple en supprimant la livraison de récompense ou en donnant de la récompense sans lien avec l'action effectuée. Si l'animal adapte son comportement, c'est qu'il a un *comportement dirigé vers un but*, et s'il persiste à effectuer l'action qui lui apportait autrefois de la récompense, c'est qu'il a une *habitude comportementale*.

Nous reviendrons à la fin du chapitre sur un effet de bord délétère engendré par l'utilisation de ces deux méthodes.

Les ganglions de la base

À la fin des années 90, Balleine et Dickinson (1998) renforcent la découverte de Dickinson (1985) concernant l'existence d'un transfert de contrôle entre les deux types de comportements instrumentaux en effectuant des *études de lésions* préliminaires sur le rat. Dès lors, les années 2000 seront marquées par de grande avancées concernant l'étude neuro-anatomique et neuro-computationnelle des comportements instrumentaux.

Quelques années plus tard, Balleine et al. (2003) montrent que l'endommagement ou l'inactivation de l'amygdale, un ensemble de noyaux sous-corticaux distincts des ganglions de la base (abrégé BG par la suite), atténue l'effet de la punition et entraîne la persistance d'une action même lorsqu'elle donne un résultat manifestement dévalué, tandis que Yin et al. (2004) montrent que l'endommagement ou l'inactivation du striatum dorsolatéral, une partie du noyau principal d'entrée des BG, entraîne à l'inverse le blocage du contrôle habituel, de sorte que même des actions très entraînées restent dirigées vers un but. À la suite de ces travaux, Yin et al. (2005), Yin et Knowlton (2006), Balleine et al. (2007) réalisent d'autres études de lésions leur permettant de confirmer le rôle central des BG dans la régulation de ces comportements. Notamment, les auteurs montrent que deux sous-parties des BG, le striatum dorsomédian (abrégé DMS par la suite) et le striatum dorsolatéral (abrégé DLS par la suite) semblent respectivement supporter l'encodage des *comportements dirigés vers un but* et des *habitudes comportementales*. Quelques années plus tard, Yin et al. (2009), Thorn et al. (2010) montrent que l'activité neuronale du DMS et du DLS change dynamiquement au cours de l'apprentissage : en début d'entraînement, le DMS (et donc les *comportements dirigés vers un but*) est préférentiellement engagé, tandis qu'en fin d'expérience, le DLS (et donc les *habitudes comportementales*) devient le substrat neuronal le plus engagé des deux.

Finalement, les effets de la dévaluation de la récompense et celui de la dégradation de la contingence, jusqu'alors bien identifiés chez les rongeurs, sont aussi retrouvés chez l'être humain (Valentin et al. 2007, Tricomi et al. 2009).

Le cortex préfrontal

Si notre attention s'est pour le moment avant tout portée sur les BG, le cortex préfrontal (abrégé PFC par la suite) reste une aire cérébrale centrale dans la modulation des comportements instrumentaux.

En effet, le PFC est connu comme étant le siège des fonctions cognitives dites supérieures, telles que le langage (Gabrieli et al. 1998), la mémoire de travail (Lara et Wallis 2015) ou encore le raisonnement (Donoso et al. 2014). Plus globalement, le PFC est le siège du contrôle cognitif (Koechlin et al. 1999), c'est-à-dire l'ensemble des fonctions de haut niveau qui permettent de faire varier de manière adaptative le traitement et le comportement de l'information à chaque instant en fonction des objectifs de l'individu. Sachant cela, nous comprenons en quoi il est important de mentionner le rôle de cette aire cérébrale lorsque l'on s'intéresse à l'étude du comportement animal.

Computationnellement parlant, on considère que les rôles des BG et du PFC sont interdépendants : si le contrôle cognitif permet de manipuler les valeurs internes de comportements qui ont été appris par AR dans les BG, apprendre de nouveaux comportements grâce à l'AR nécessite de pouvoir structurer un problème sous forme de modèles, de pouvoir réguler l'équilibre entre l'exploration de l'environnement et l'exploitation des connaissances, de pouvoir changer de *task-set* etc. (Koechlin et Summerfield 2007, Gläscher et al. 2012, Jeon et Friederici 2013). Un *task-set* représente, étant donnée une tâche cognitive spécifique, une configuration de processus mentaux permettant de réaliser cette tâche de manière appropriée, et donc une stratégie de résolution de celle-ci. C'est la stratégie optimale connue qui sera maintenue tout au long de la tâche, et qui sera donc imposée par le contrôle cognitif (Monsell 2003, Sakai 2008). Tout au long de la tâche, le *task-set* s'adaptera alors à l'environnement en utilisant les modèles mis à jour par l'AR. Le PFC est une aire cérébrale centrale dans la création et la manipulation de ces modèles de la tâche (Daw et al. 2011, Gläscher et al. 2010, Wunderlich et al. 2012).

Le PFC est aussi le siège du contrôle motivationnel (Kouneiher et al. 2009). Si le contrôle cognitif permet de faire varier de manière adaptative le traitement et le comportement de l'information en fonction des objectifs, le contrôle motivationnel permet de les faire varier en fonction de l'état motivationnel de l'individu. Il régule le contrôle cognitif.

Très récemment, O'Doherty et al. (2021), avec le support d'études d'imagerie à résonance magnétique fonctionnelle et de neurostimulation, ont proposé que la partie antérieure du PFC était capable de déterminer à chaque instant quel système devait contrôler le comportement de l'animal. D'après eux, le cerveau utilise un cadre vaguement analogue à la technique du mélange d'experts en apprentissage automatique (*Mixture of experts* en anglais), où un gestionnaire situé dans le PFC lit la fiabilité des prédictions de chacun des systèmes d'apprentissage (ceux associés aux *habitudes comportementales* ou aux *comportements dirigés vers un but*, par exemple), et utilise ces prédictions pour attribuer le contrôle du comportement à ces systèmes d'une manière qui est proportionnelle à la précision ou aux incertitudes relatives de leurs prédictions. Ils suggèrent aussi que cette fiabilité soit médiée par les erreurs de prédiction (formalisées dans le chapitre 2), qui sont susceptibles d'être présentes dans chaque système d'apprentissage, et qui sont encodées dans les BG.

Finalement, il est intéressant de noter que le PFC, qui représente un tiers de la surface du cerveau humain, est évolutivement la partie du cerveau qui le différencie le plus des autres grands primates, à l'inverse des BG, par exemple, présents aussi chez les rongeurs. Si le PFC n'est donc pas indispensable à l'émergence des comportements instrumentaux, il permet en revanche de les réguler de manière très fine.

Il est important de garder à l'esprit qu'il est impossible de réduire une fonction à une aire cérébrale. Tout au plus, une aire aura un rôle prédominant dans la réalisation de cette fonction. Car c'est avant tout le résultat de l'interaction entre différentes aires cérébrales qui permet d'expliquer une fonction cognitive. Pour le lecteur souhaitant en apprendre d'avan-

tage sur les substrats neuronaux des comportements instrumentaux, nous le réorientons vers le travail de Balleine et O'Doherty (2010), qui propose une revue très complète du sujet. Nous pouvons aussi le réorienter vers le travail de Boraud et al. (2018), qui réévaluent la fonction du réseau BG / cortex et montrent que les théories actuelles pourraient tout à fait être englobées dans un cadre plus large de l'apprentissage des compétences et de la performance.

4.2 Modèles computationnels du comportement instrumental

4.2.1 L'apprentissage par renforcement au coeur des modèles du comportement instrumental

Après la lecture de ce début de chapitre, le lecteur attentif remarquera sans doute les analogies entre l'apprentissage par renforcement (AR) décrit dans le chapitre 2 et l'idée d'une évolution du comportement motivée et renforcée par l'obtention d'une récompense. C'est ce dont se sont rendus compte les chercheurs à l'origine des premiers algorithmes d'AR dans les années 80-90, notamment Sutton et Barto (1987), qui conceptualisent l'apprentissage par différence temporelle en s'inspirant du modèle du conditionnement classique de Rescorla (1972). Dans les années 90, des recherches en neurophysiologie montrent l'importance du rôle joué par la dopamine dans les tâches de conditionnement (Ljungberg et al. 1992), et quelques années plus tard, Schultz et al. (1997) montrent une corrélation forte entre l'activité phasique des neurones dopaminergiques (qui sécrètent la dopamine) et l'erreur de prédiction calculée par les algorithmes d'AR direct (*SARSA*, *Q-learning*, voir chapitre 2). En effet, nous pouvons voir sur la première ligne de la figure 4.3 qu'un pic d'activité des neurones dopaminergiques apparaît chez un animal recevant une récompense non prédite, ce qui correspond à un retour reçu supérieur à la valeur prédite des algorithmes d'AR direct. Lorsque sur la deuxième ligne, l'animal apprend le lien entre un stimulus et l'obtention de la récompense, le pic se décale au moment du stimulus, et n'est plus présent au moment de l'obtention de la récompense, ce qui correspond au fait que la prédiction des algorithmes est équivalente au retour reçu. Pour finir avec la dernière ligne, lorsque la prédiction de l'animal est fausse, l'activité des neurones est inhibée au moment où la récompense aurait dû être délivrée, ce qui correspond à un retour reçu inférieur à la valeur prédite par les algorithmes. Si l'hypothèse du lien entre AR et neurones dopaminergiques sera discutée à de nombreuses reprises durant les années qui suivront (Berridge 2007, Schultz 2013, Bellot et al. 2012, Kishida et al. 2016), celle-ci reste encore aujourd'hui très présente en modélisation et globalement bien admise en neurosciences.

Un an après la découverte de Schultz et al. (1997) concernant les liens entre l'activité phasique des neurones dopaminergiques et l'erreur de prédiction calculée par les algorithmes d'AR direct, Doya (1999) propose une association des différents types d'apprentissage automatique à des aires cérébrales : l'apprentissage non-supervisé est associé au cortex cérébral, tandis que l'apprentissage supervisé est associé au cervelet. Concer-

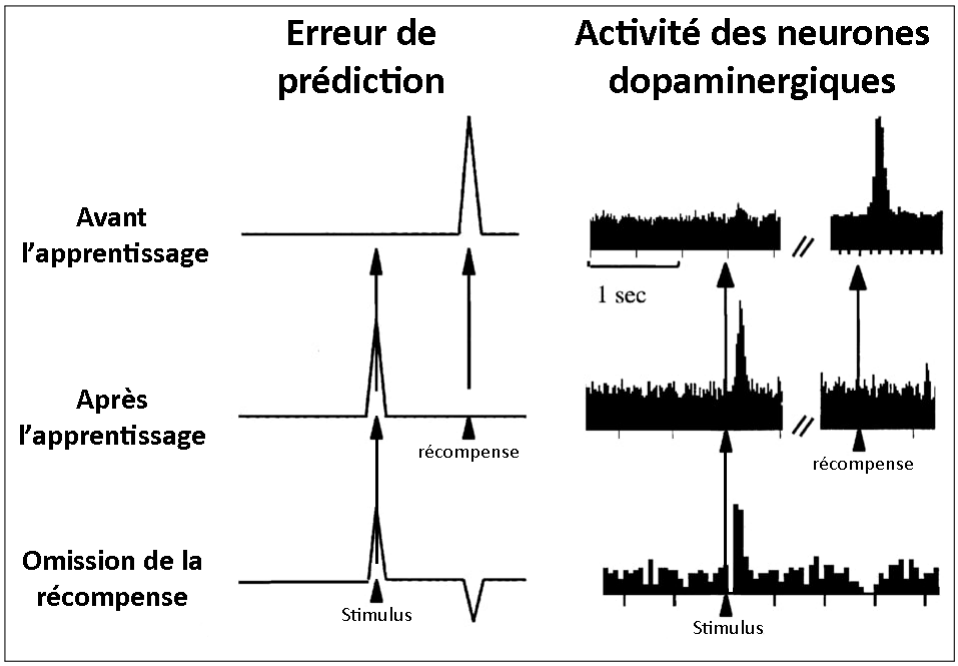

FIGURE 4.3 – ***Gauche***. *Dynamique des valeurs de l'erreur de prédiction dans les algorithmes d'apprentissage par renforcement.* ***Droite***. *L'activité des neurones dopaminergiques telle que relevée dans Schultz et al. (1997). La première ligne montre l'activité avant l'apprentissage, où l'erreur de prédiction et l'activité dopaminergique ont un pic au moment de l'obtention de la récompense. La ligne du milieu montre l'activité après l'apprentissage et le décalage du pic au moment de la présentation du stimulus. La dernière ligne montre l'activité après l'apprentissage et avec omission de la récompense au moment attendu. Cette figure est adaptée de Schultz et al. (1997).*

nant l'AR, il est associé aux BG, que l'on sait à l'époque être impliqués dans l'apprentissage de comportement séquentiel (Dominey et Arbib 1992, Graybiel 1995, Berns et Sejnowski 1998). Comme mentionné précédemment, Yin et Knowlton (2006) associeront quelques années plus tard le rôle du DMS aux *comportements dirigés vers un but*, et celui du DLS aux *habitudes comportementales*. Avec ce nouvel éclairage, une nouvelle association semble désormais évidente en neurosciences : celle entre AR et comportements instrumentaux.

4.2.2 Modélisation de la coordination de stratégies comportementales

Daw et al. (2005) sont les premiers à faire le rapprochement entre la dichotomie de l'apprentissage par renforcement (AR indirect et AR direct, voir chapitre 2) et celle des comportements instrumentaux (*comportements dirigés vers un but* et *habitudes comportementales*). Leur hypothèse est la suivante :

— L'AR indirect (ou AR *model-based*) est un bon mécanisme pour décrire les *comportements dirigés vers un but*. En effet, comme les *comportements dirigés vers un but*, il converge rapidement vers la solution grâce à la planification, s'adapte rapidement aux changements environnementaux, mais cela au prix d'un temps de réponse plus long (et donc d'un coût calculatoire plus élevé).

— L'AR direct (ou AR *model-free*) est un bon mécanisme pour décrire les *habitudes comportementales*. En effet, comme les *habitudes comportementales*, il converge lentement vers la solution et s'adapte mal aux changements environnementaux, dû au fait qu'il apprend sans planifier, mais avec des temps de réaction très courts (et donc avec un faible coût calculatoire).

Dès lors, la grande majorité des recherches s'intéressant à la modélisation du comportement instrumental animal s'appuyèrent sur ce paradigme : le comportement instrumental est le résultat d'un partage de contrôle entre ces deux types d'algorithmes d'AR. Parmi ces travaux, nous pouvons citer Lesaint et al. (2014) qui reproduit en simulation les différents types de comportements instrumentaux observés chez le rat en utilisant un modèle coordonnant des algorithmes d'AR direct et indirect, renforçant ainsi l'hypothèse de Daw et al. (2005). Dans Collins et Frank (2012) en revanche, les auteurs remettent en question ce paradigme en étudiant dans quelle mesure l'apprentissage par renforcement relève-t-il en réalité plutôt de la mémoire de travail. Pour ce faire, ils combinent un algorithme de mémoire de travail à un algorithme d'AR direct et réussissent à nouveau à reproduire les différents types de comportements instrumentaux observés chez le rat. Viejo et al. (2015) raffinent le modèle de Collins et Frank (2012) et considèrent que ceci est une autre façon de coordonner un système *model-based* (utilisant de la mémoire de travail) et un système *model-free* (d'AR direct), même si le système *model-based* est formalisé sous forme bayésienne plutôt que sous forme d'AR indirect. Dans Topalidou et al. (2018) les auteurs proposent quand à eux un modèle incluant des interactions entre le cortex, les BG et le thalamus, sur la base d'une double compétition entre un circuit moteur et un circuit cognitif, respectivement pilotés par AR direct et apprentissage hebbien (Hebb 1949). Au

final, même si la nature des mécanismes qui sous-tendent les comportements instrumentaux est encore aujourd'hui débattue, l'hypothèse de l'existence d'un mécanisme de coordination de systèmes d'apprentissage et de prise de décision est rarement remise en cause. Dans Khamassi et Humphries (2012), les auteurs montrent que la dichotomie entre AR indirect (AR *model-based*) et direct (AR *model-free*) permet d'expliquer la variété des comportements observés expérimentalement chez les rongeurs dans les tâches de navigation. Ils proposent à cette occasion une organisation des stratégies de navigation et du contrôle de l'apprentissage au sein des BG 4.4.

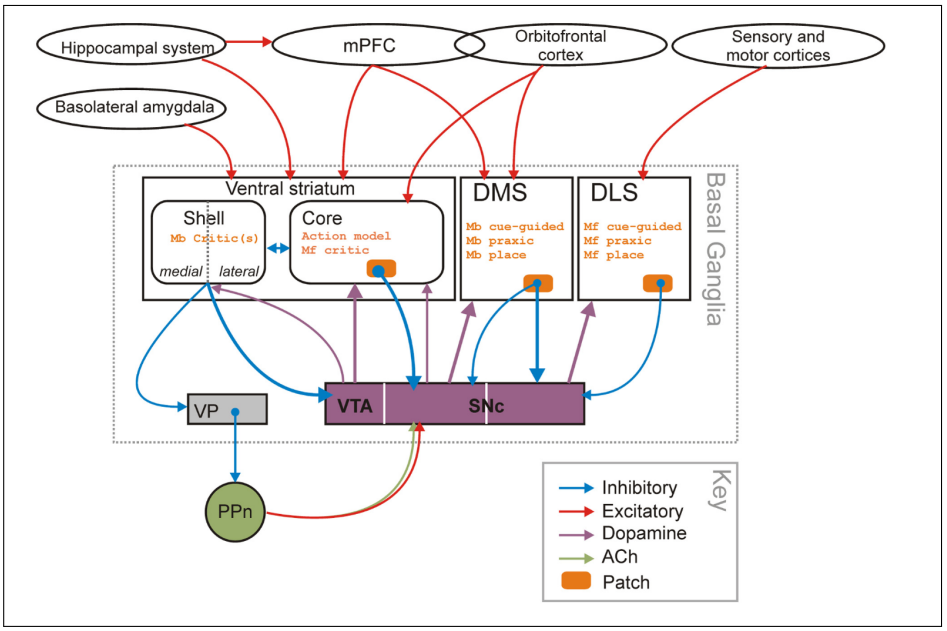

FIGURE 4.4 – *Substrats du domaine striatal des comportements dirigés vers un but et des habitudes comportementales. Abréviations : Mb, model-based ; Mf, model-free ; PPn, noyau pédonculopontin ; SNc, substantia nigra pars compacta ; VP, pallidum ventral ; VTA, aire tegmentaire ventrale. Cette figure est reprise de Khamassi et Humphries (2012).*

Comme nous allons le voir, de nombreuses autres études se sont intéressées à la problématique du partage de contrôle d'*experts décisionnels*. Notre recherche s'inscrit dans ce corpus. En effet, l'enjeu de notre travail est d'améliorer l'autonomie décisionnelle de robots en leur permettant d'adapter leur comportement dynamiquement en fonction de l'évolution de l'environnement. Nous proposons avec ce manuscrit une solution originale, que nous aurons l'occasion de présenter dans le chapitre 5. Dans la suite, nous appellerons *expert* un algorithme modélisant une stratégie comportementale spécifique (par exemple une stratégie de type *habitude comportementale* ou de type *comportement dirigé vers un but*).

Les études s'intéressant au transfert de contrôle entre *comportements dirigés vers un but* et *habitudes comportementales* s'organisent souvent selon deux angles :

— **L'organisation des experts**. Deux manières d'organiser les experts sont référencées dans la littérature. Dans les *modèles horizontaux*, les experts évaluent en parallèle la solution au problème tandis qu'un troisième système se charge de faire un choix entre les propositions de chaque expert. Dans les *modèles hiérarchiques*, un expert dirige le

comportement de l'agent par défaut, et sous certaines conditions, la solution de l'autre expert est mise en avant.

— **La méthode d'arbitrage**. Deux manières d'arbitrer entre les solutions des experts sont référencées dans la littérature. Les *méthodes de fusion*, présentées dans le chapitre 2, consistent à fusionner le retour de chacun des experts en un retour unique. Les *méthodes de sélection* consistent à définir un critère de sélection qui permettra à un système tiers (généralement appelé *méta-contrôleur*) de faire un choix entre les propositions de chaque expert.
Dans les sous-parties suivantes, nous ferons une revue de la littérature s'intéressant à la coordination d'experts décisionnels sous l'angle de leur organisation.

Modèles horizontaux

Dans leur travail, Daw et al. (2005) étudient la coordination entre un expert d'AR direct et un expert d'AR indirect, et plus précisément la coordination entre les versions bayésiennes d'un algorithme *Q-learning* et d'un algorithme *Value Iteration* (voir chapitre 2). Lorsqu'il utilise l'algorithme *Q-learning*, l'agent n'estime plus l'intérêt d'une action comme une valeur unique, mais comme une distribution de probabilité de la valeur d'état-action associée. Lorsqu'il utilise l'algorithme *Value iteration*, la distribution de probabilité est maintenue sur les probabilités de transitions et de récompenses, pour en déduire les distributions sur les probabilités des valeurs d'état-action. Ainsi, si pour un couple (s, a) donné, la distribution est proche d'une loi uniforme, cela signifie que l'agent est très incertain de la valeur d'état-action. À l'inverse, si la distribution est piquée sur une valeur particulière, c'est que l'agent a une bonne estimation de la valeur d'état-action. Finalement, la valeur de chaque action est estimée en prenant la valeur moyenne de la distribution de probabilité de l'expert le plus confiant, c'est-à-dire celui dont la variance est la plus faible. Ainsi ici, le critère de sélection dépend du niveau d'incertitude de chaque expert.

Une autre approche, telle que celle de Chavarriaga et al. (2005), consiste à apprendre à sélectionner à chaque étape de décision l'un des experts. Ici, les auteurs font évoluer un agent virtuel dans un *labyrinthe de Morris* (Morris 1984), un dispositif aquatique circulaire très utilisé en neurosciences comportementales afin d'évaluer la mémoire des rongeurs. Les auteurs proposent un modèle coordonnant une stratégie de navigation *locale*, basée sur la position de l'agent dans l'espace, et une stratégie de navigation *taxon*, basée sur des indices perceptifs (signaux visuels du labyrinthe). À l'inverse de Daw et al. (2005), les deux experts supportant ces stratégies sont ici modélisés par des algorithmes d'AR direct, et l'association entre état et stratégie est apprise par un troisième algorithme d'AR direct, appelé *gating network*. La sélection d'une stratégie s'effectue en fonction de la valeur de l'action associée à la stratégie, mais aussi en fonction de la force de l'association apprise entre état et stratégie. Ainsi, une stratégie qui aurait été peu associée à l'état mais qui prédirait une forte récompense a quand même une chance d'être sélectionnée. Chacun de ces experts apprend en fonction de son erreur de prédiction, mais également de sa probabilité d'avoir été sélectionné.

Dollé et al. (2010) proposent un modèle qui, comme celui de Chavarriaga et al. (2005), coordonne différentes stratégies de navigation grâce à un méta-contrôleur (un *gating network*). En revanche trois stratégies sont désormais possibles : guidage visuel (approche *taxon*), navigation (cette fois-ci modélisée par un algorithme de recherche de chemin dans un graph), et exploration. Une autre différence par rapport à Chavarriaga et al. (2005) est que le *gating network* apprend par renforcement uniquement à associer un état à une stratégie, et donc que la méthode s'appuie désormais seulement sur la capacité des experts à faire obtenir de la récompense à l'agent.

Deux ans plus tard, Caluwaerts et al. (2012b) étendent ce modèle à la robotique en l'implémentant sur un *robot-rat*. Désormais, le *gating network* manipule des états qui ne sont composés plus que de l'estimation de position, c'est-à-dire uniquement de l'information de la stratégie de navigation. Les auteurs montrent que leur architecture parvient à apprendre à utiliser le meilleur expert dans les différents points de l'espace en fonction de l'information perceptive disponible, ainsi qu'à ignorer l'expert explorateur lorsque celui-ci est couplé à un expert ayant appris suffisamment pour être meilleur.

Du côté des architectures de contrôle utilisant des réseaux de neurones profonds, Hafez et al. (2019) proposent un modèle axé sur la curiosité qui coordonne un DDPG (*deep deterministic policy gradientl* en anglais), un réseau de neurones profond faisant de l'AR direct, et un réseau de neurones profond dynamique faisant de l'AR indirect avec correction anticipatrice. L'objectif du robot simulé est d'attraper un objet sur une table. Ici, le méta-contrôleur tient compte d'un critère la fiabilité du modèle de la tâche pour choisir entre l'action du réseau d'AR direct ou indirect. Dans Sheikhnezhad Fard et Trappenberg (2019), les auteurs associent cette fois-ci un DDPG à un algorithme d'apprentissage supervisé qui utilise les expériences passées pour généraliser un modèle inverse et un modèle avancé du bras. Ici, le méta-contrôleur se charge de comparer la valeur absolue de l'erreur de prédiction de l'expert des *habitudes comportementales* à un seuil fixé par les expérimentateurs. Si cette valeur est en dessous du seuil, alors l'action de l'expert des *habitudes comportementales* est sélectionnée, sinon, celle de l'expert des *comportements dirigés vers un but* l'est.

Si pour le moment, nous avons présentés uniquement des méthodes de sélection, de nombreux modèles horizontaux proposent de fusionner les valeurs estimées par chaque expert en les sommant. C'est l'approche proposée par Guazzelli et al. (1998), qui combinent d'une manière assez semblable à Chavarriaga et al. (2005) deux stratégies de navigation modélisées par des algorithmes d'AR direct, et exploitant des informations de nature différente : un modèle *Taxon-Affordances*, qui s'appuie sur les stimuli perceptibles, et un modèle *World-Graph*, qui associe ces stimuli à un lieu dans l'environnement. Un méta-contrôleur somme directement la valeur estimée pour chaque action par chacun des experts avant d'effectuer la sélection de l'action à exécuter.

Dans Hanoune (2015), un expert utilisant une carte cognitive et un expert utilisant des associations sensorimotrices évaluent chacun les actions disponibles à partir d'une information commune formée par les transi-

tions entre les lieux. Comme dans Guazzelli et al. (1998), ces estimations sont sommées avant compétition entre les valeurs résultantes.

Les deux dernières études présentées proposent une manière simple de mettre en commun les informations issues de différents experts, mais donnent en conséquence la même importance aux propositions de chacun. Dans Girard et al. (2005), un robot confronté à une tâche de survie a le choix entre une stratégie de navigation de type *taxon* (à nouveau basée sur des stimuli perceptibles) et une stratégie de navigation basée sur une carte topologique de l'environnement. Ici, les auteurs proposent de sommer de manière pondérée les *saliences* de la stratégie *taxon* (où la pondération est un paramètre fixe du modèle), ce qui permet de favoriser l'un des experts lorsque les deux sont en désaccord sur l'action à suivre. Ils montrent que la performance de la combinaison des experts est meilleure que celle des experts pris individuellement.

Prenant inspiration de Collins et Frank (2012), l'expert des *comportements dirigés vers un but* de Viejo et al. (2015) est modélisé 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 probabilité des experts sont hautes, la probabilité de décider reste faible. En revanche, dès lors que ces valeurs d'entropie diminuent ou que le nombre de fois où l'agent choisit de raffiner les valeurs d'état-action augmente, la probabilité de décider s'élève.

Nous pouvons aussi citer Hasson (2012), qui introduit la notion de frustration. Ici, la frustration correspond à une mesure de l'évolution de la distance au but qui augmente si la distance augmente ou stagne. Lorsque le niveau de frustration dépasse un certain seuil, alors l'agent change de stratégie. Si le mécanisme se veut générique, il nécessite cependant que le but soit connu de l'agent, ce qui n'est pas directement applicable à d'autres contextes que la navigation. Afin de contrer cette faiblesse, Jauffret et al. (2013) étendent ce modèle en basant la notion de frustration sur l'erreur de prédiction sensorimotrice, le rendant ainsi plus générique.

Du côté des modèles à assemblées de neurones, nous pouvons citer le modèle de Topalidou et al. (2018), qui fonctionnent selon deux régimes pilotés par AR direct et par apprentissage hebbien (Hebb 1949), et où la décision finale est prise en fonction d'une combinaison linéaire de ces deux mécanismes. Les auteurs montrent un transfert progressif du contrôle de l'AR direct vers l'apprentissage hebbien et montrent que ce modèle permet de reproduire des données comportementales chez le singe.

Enfin, le modèle de coordination que nous proposons dans ce manuscrit est une évolution de la méthode Renaudo (2016). Étant plus liée à notre notre travail que les autres méthodes présentées ici, nous détaillerons son fonctionnement dans le chapitre 5. Nous mentionnerons simplement le fait que le modèle de Renaudo (2016) s'inspire grandement de celui de Daw et al. (2005), et coordonne un algorithme d'AR direct et indirect manipulant des états abstraits composés d'informations dont la nature dépend des tâches expérimentées. Dans Renaudo et al. (2015c;b), les auteurs évaluent une multitude de critères de coordination, allant des méthodes de sélection aux méthodes de fusion.

Modèles hiérarchiques

Les modèles hiérarchiques sont plus récents que les modèles horizontaux. À la différence de ces derniers, ils donnent un rôle prépondérant à un expert. Si leur vision a été formalisée par Dezfouli (2015), on retrouve les prémisses dans des modèles antérieurs.

Dans Keramati et al. (2011), les auteurs étendent le modèle proposé par Daw et al. (2005) sous une forme hiérarchique. Si Daw et al. (2005) ne se préoccupaient que de l'incertitude de chaque agent, les auteurs proposent ici un arbitrage par sélection qui optimise le compromis entre la précision de l'information et la rapidité de la décision. Dans ce modèle, l'expert ayant des *habitudes comportementales* maintient un suivi constant de son incertitude sur les valeurs d'état-action à l'aide d'un filtre de Kalman, comme proposé par Geist et al. (2009). Cette incertitude lui permet de calculer la *Valeur de Parfaite Information*, qui représente l'intérêt de l'agent à raffiner son estimation de la valeur d'une action. Cette *Valeur de Parfaite Information* est ensuite comparée au coût estimé à raffiner cette estimation, calculé en multipliant la récompense moyenne reçue jusqu'ici par le temps passé à délibérer. Concrètement, lorsque l'agent se trouve dans un état s, il effectue cette comparaison pour chaque action a qu'il peut réaliser. Si la *Valeur de Parfaite Information* est inférieure au coût, la valeur d'état-action $Q(s,a)$ de l'expert des *habitudes comportementales* est maintenue. En revanche, si la *Valeur de Parfaite Information* est supérieure au coût, l'expert des *comportements dirigés vers un but* va raffiner la valeur $Q(s,a)$ grâce à son processus de planification. Finalement, l'agent utilise la règle de sélection pour choisir l'action qu'il va effectuer. Comparativement aux modèles horizontaux de sélection, il n'y a donc ici pas de compétition entre les propositions des deux experts : l'expert des *comportements dirigés vers un but* est l'expert au rôle prépondérant, qui planifie dès lors que le bénéfice de la planification est supérieur au coût. Si cette approche offre l'avantage de limiter l'utilisation du processus de planification aux cas où il est réellement nécessaire, il repose néanmoins sur une hypothèse très forte : l'expert des *comportements dirigés vers un but* peut obtenir une information *parfaite* sur l'intérêt de chaque action en planifiant dans ses modèles internes. Quelques années plus tard, le modèle hiérarchique de Keramati et al. (2011) est comparé au modèle horizontal de Renaudo (2016), qui obtient de meilleures performances sur une tâche simulée de poussée de cube comptant un trop grand nombre d'états (>200) pour que la planification puisse couvrir tous les états dans un délai limité, confirmant ainsi que cette hypothèse est effectivement trop forte pour passer à l'échelle d'un grand espace d'états.

Une méthode similaire est utilisée dans Pezzulo et al. (2013), où aucune hypothèse concernant la perfection de l'expert des *comportements dirigés vers un but* n'est cette fois-ci faite. Ici, les auteurs calculent une valeur correspondant au ratio entre l'incertitude de l'action et la différence entre la valeur de l'action considérée et celle de son alternative. Cette valeur est à nouveau comparée à une valeur de coût, et permet de déclencher le processus de planification de l'expert des *comportements dirigés vers un but*, comme dans Keramati et al. (2011). Aucune hypothèse concernant la perfection de l'expert des *comportements dirigés vers un but* n'étant faite,

la valeur d'état-action de l'expert des *habitudes comportementales* n'est pas remplacée mais simplement mise-à-jour à l'aide de l'estimation de la valeur d'état-action de l'expert des *comportements dirigés vers un but*.

Dans Dezfouli et Balleine (2012), les auteurs utilisent aussi une méthode hiérarchique mais proposent une tout autre conception des *habitudes comportementales*, jusqu'alors modélisées par des algorithmes d'AR direct. Les auteurs pointent en effet le fait que les algorithmes d'AR direct reproduisent imparfaitement les résultats sur les animaux testés dans des situations de dégradation de la contingence : alors que les animaux qui ont été entraînés suffisamment pour former des *habitudes comportementales* sont insensibles aux situations où le taux de récompense associé à une action varie brutalement au cours de l'expérience, les algorithmes d'AR direct classiques présentent dans ce cas là une erreur de prédiction importante, et tendent à modifier la valeur d'état-action associée, puis le comportement de l'agent au fur et à mesure que la valeur se propage. Plutôt que de proposer une variante d'un algorithme d'AR direct permettant de palier à ce défaut de modélisation, les auteurs proposent de modéliser les habitudes comme des séquences d'actions. Ici, le comportement de l'agent est donc uniquement contrôlé par un expert des *comportements dirigés vers un but* modélisé avec un algorithme d'AR indirect. Pour manipuler ces séquences d'actions, les auteurs définissent une mesure du coût et du bénéfice d'exécuter une séquence depuis un certain état, et comparent ensuite ces mesures. Si le coût est inférieur au bénéfice, une séquence d'action est formée, et s'il est supérieur, la séquence est divisée en actions élémentaires. Une séquence d'action sera traitée par l'expert des *comportements dirigés vers un but* de la même manière qu'une action traditionnelle : l'expert choisit l'action selon son processus de sélection, et une fois celle-ci exécutée par l'agent, toutes les autres actions de la séquence sont exécutées automatiquement. L'état d'arrivée n'est évalué qu'après que toutes les actions de la séquence aient été exécutées, et le coût de celle-ci n'est pas réévalué. Cette approche originale est appuyée par des données expérimentales humaines (Dezfouli et Balleine 2013). Au delà du fait que cette méthode propose une tout autre conception des habitudes, Balleine et Dezfouli (2019) insistent aussi sur son aspect coopératif (entre les *habitudes comportementales* et les *comportements dirigés vers un but*) en comparaison de l'aspect souvent plus compétitif des modèles horizontaux.

Cushman et Morris (2015) proposent un modèle hiérarchique "inversé", où l'expert des *habitudes comportementales* a cette fois-ci le rôle prépondérant. À l'inverse des études précédentes, un expert des *habitudes comportementales* fournit ici à l'agent des "objectifs" vers lesquels le second expert pourra planifier. Si cette idée est intéressante, elle nécessite un élargissement significatif de ce qui est traditionnellement considéré comme l'objet manipulé par l'expert des *habitudes comportementales*. Comme nous l'avons vu jusqu'alors, une habitude est généralement considérée dans la littérature comme un mouvement moteur spécifique manipulant des états du monde, et non pas des objectifs. Un animal s'efforçant de modifier le monde pour qu'il corresponde à ses désirs, comme ici, sera généralement plutôt considéré comme travaillant d'une manière dirigée vers un but.

4.3 Discussion sur la notion d'habitude

Comme nous venons de le voir avec les modèles hiérarchiques, la question de savoir comment modéliser l'*habitude comportementale* reste un sujet d'étude important du comportement instrumental. Aujourd'hui, il est communément admis que le conditionnement instrumental peut en partie être expliqué par les *comportements dirigés vers un but*, des comportements montrant une sensibilité aux changements de la relation entre action, valeur et conséquence. En cas de surentraînement, il a été montré que les *comportements dirigés vers un but* s'amenuisaient et que les actions pouvaient être effectuées indépendamment de leurs conséquences estimées : ce sont les *habitudes comportementales*.

En adoptant cette dichotomie, des progrès considérables ont été réalisés, non seulement en fournissant des preuves de l'existence des *comportements dirigés vers un but* chez diverses espèces (dont l'être humain), mais aussi en découvrant des bases neurales associées. Cette approche et ce succès ont cependant eu un effet délétère : les tests effectués à l'origine pour distinguer les *comportements dirigés vers un but* ont été utilisés pour discriminer sans distinction ce qui ne relevait pas de ces comportements : des comportements "*non dirigés vers un but*", rapidement qualifiés d'*habitudes comportementales*, presque par défaut. Or, les *comportements dirigés vers un but* peuvent aussi s'amenuiser en cas de stress, de neurodégénérescence, ou d'exposition à diverses drogues. Bien sûr, les *habitudes comportementales* restaient reconnaissables à leur régularité non cognitive et répétitive, à leur contrôle des stimuli, etc., mais toujours en comparaison des *comportements dirigés vers un but*, et sans savoir si d'autres formes de comportement pouvaient expliquer ces régularités.

Le premier indice du problème engendré par cette généralité excessive est la difficulté de différentier *l'habitude comportementale* du *réflexe conditionné de pavlov* sur la base de leur insensibilité aux changements dans la relation action/conséquence. Plus globalement, affirmer qu'une action est une habitude dès lors qu'elle ne satisfait pas aux tests des *comportements dirigés vers un but*, c'est considérer que des actions réalisées par un acteur confus, sujet à une perte de mémoire, ou ayant une croyance perturbatrice concernant les résultats de ces actions, sont des habitudes. Or, dans de telles situations, le comportement peut sembler "habituel" alors qu'il est en fait contrôlé par un expert dirigé vers un but altéré. Au final, malgré leur apparente simplicité, les comportements définis comme étant des "*habitudes comportementales*" sont plus compliqués qu'ils n'y paraissent.

Récemment, Balleine et Dezfouli (2019) ont essayé de prendre en compte les différentes définitions alternatives de l'habitude afin de dériver des critères permettant de catégoriser les réponses associées aux *habitudes comportementales*. D'après eux, les habitudes sont des actions s'accordant avec les quatre observations suivantes : (1) elles sont relativement rapides à déployer et à exécuter, (2) elles sont relativement invariables dans la cinématique de leur mouvement moteur, (3) elles sont incorporées dans des séquences d'actions groupées, et (4) elles sont insensibles aux changements dans leur relation avec leurs conséquences individuelles et avec la valeur de ces conséquences. Les observations (3) et (4), appuyées par des données expérimentales humaines (Dezfouli et Balleine 2013), sont notam-

ment celles ayant motivé la conceptualisation récente des habitudes en tant que "séquences d'actions" (Dezfouli et Balleine 2012, Dezfouli 2015).

Comme nous l'avons montré dans ce chapitre, il existe de nombreuses preuves comportementales, neurales et computationnelles de la coordination et de la concurrence entre *comportements dirigés vers un but* et *habitudes comportementales*. Pourtant, Balleine et Dezfouli (2019) considèrent qu'une grande parties d'entre elles peuvent être réinterprétées. Par exemple, les facteurs qui influencent l'arbitrage entre les *comportements dirigés vers un but* et les *habitudes comportementales* dans le cadre des modèles horizontaux (tels que le bénéfice, le coût, l'incertitude, etc.), peuvent tout aussi bien influencer le processus qui choisit dans un modèle hiérarchique entre action élémentaire ou séquence d'actions. Les raisons de ces influences seront aussi globalement les mêmes : sélectionner une séquence d'actions implique un coût réduit (coût réduit des *habitudes comportementales*), tandis que sélectionner une action élémentaire implique potentiellement une hausse du gain de récompense en raison de leur ajustement plus immédiat aux contraintes environnementales (meilleure performance et adaptabilité des *comportements dirigés vers un but*). D'un point de vue comportemental, dans la mesure où la charge cognitive et la complexité accrue de la planification favorisent les habitudes (Otto et al. 2013), on s'attend à ce qu'en début d'expérience, l'expert des *comportements dirigés vers un but* d'un modèle hiérarchique sélectionne davantage les actions élémentaires (les séquences d'actions n'étant pas encore formées), tandis qu'en fin d'expérience, il sélectionnera davantage les séquences d'actions. En effet, l'évaluation des séquences d'actions est moins exigeante sur le plan cognitif qu'un ensemble d'actions élémentaires, les premières ne reposant pas sur le calcul des valeurs d'état-action intermédiaires. Finalement, on obtient le même type de schéma temporel "planifier jusqu'à l'habitude" mis en évidence avec des modèles horizontaux de sélection (Renaudo 2016, Dromnelle et al. 2020b;b) et correspondant aux observations faites par Keramati et al. (2016) chez des sujets humains réalisant des tâches de bandit manchot. À noter que comme avec les modèles horizontaux, la complexité de la planification dans un modèle hiérarchique peut aussi changer en fonction de la tâche : dans les environnements avec peu d'états, la planification peut être gérée avec des actions élémentaires, qui ont eu une plus grande précision, et lorsque l'environnement se complexifie, le recours aux séquences d'actions devient plus important car le coût de l'évaluation des actions élémentaires augmente de manière exponentielle avec le nombre d'états.

Enfin, une autre hypothèse expliquant la formation des *habitudes comportementales* a été proposée par Topalidou et al. (2015). Selon l'hypothèse que les BG guident initialement l'apprentissage du cortex, les auteurs proposent de modéliser les *habitudes comportementales* non plus comme un algorithme d'AR direct, qui apprend en fonction de la récompense, mais comme un algorithme d'apprentissage hebbien (Hebb 1949) qui apprend des associations stimulus-action. 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. À la suite

de ces travaux, Topalidou et al. (2016) montrent que ce modèle permet de reproduire des données comportementales chez le singe.

4.4 Conclusion

Dans ce chapitre, nous nous sommes intéressés d'une part à l'étude du comportement instrumental chez les mammifères, et d'autre part aux modèles computationnels qui cherchent soit à expliquer ces résultats, soit à contrôler des robots.

D'un point de vue évolutif, être capable de coordonner à la volée plusieurs types de stratégies comportementales est un avantage certain. Si cette capacité permet de s'adapter efficacement aux changements environnementaux, et ainsi d'accomplir au mieux ses objectifs, elle permet aussi d'économiser des ressources calculatoires, et donc de l'énergie. En effet, la problématique de l'économie de ressources énergétiques est centrale pour un animal. Certes, pouvoir trouver efficacement de la nourriture afin de remplir ses réserves glucidiques est très utile. Mais si cela se fait au prix d'une consommation excessive de glucide ou de glucose, l'opération n'est pas nécessairement gagnante. De la même manière, être capable d'économiser ses réserves, c'est éviter de prendre des risques en devant partir à la recherche de nourriture. Chez un être humain, le poids de son cerveau représente environ 2% du poids total de son corps. Or, son cerveau consomme en permanence environ 20% de l'énergie nutritive absorbée. C'est dire l'importance de l'activité du cerveau et de sa nécessité à utiliser avec parcimonie ses processus cognitifs les plus complexes.

Les modèles du comportement instrumental que nous avons présentés dans ce chapitre adoptent une approche qui consiste à doter un agent de plusieurs stratégies comportementales, et à les combiner selon diverses méthodes. Dans le cas où ils servent à contrôler des agents artificiels, l'objectif de ces modèles est de tirer bénéfices des avantages de chacune des stratégies et de minimiser les inconvénients. Dans la majorité de ces modèles, les comportements sont organisés de manière horizontale, c'est-à-dire avec une méthode d'arbitrage qui choisit entre les différentes stratégies ou les fusionne. Dans ces modèles, aucun comportement n'est actif par défaut. Il existe à l'inverse certains modèles, qualifiés de hiérarchiques, où l'un des comportements est prépondérant. Si l'apprentissage par renforcement est souvent au cœur des processus d'apprentissage de ces modèles, nous mettons en lumière quelques exceptions qui proposent d'autres approches. À cette occasion, nous discutons des *habitudes comportementales*, et du problème que pose leur modélisation.

En effet, il semble rarement y avoir de consensus définitif concernant ces questions de modélisation. Ces modèles s'inspirant des neurosciences, et les connaissances biologiques évoluant régulièrement, les hypothèses sont souvent amenées à être révisées. Si l'importance des BG et du PFC dans l'encodage et la coordination des *comportements dirigés vers un but* et des *habitudes comportementales* est globalement admise en neurosciences, il reste encore beaucoup à découvrir concernant leurs nombreuses subtilités. La contribution de la robotique à ce débat apporte un point de vue plus fonctionnel sur ces mécanismes, même si sa motivation première est d'améliorer les capacités des robots, des entités relativement différentes

des animaux. Dans le chapitre 5, nous présentons l'architecture robotique de coordination de comportements que nous avons développée, et qui sera évaluée expérimentalement dans les chapitres 6, 7 et 8.

5 Architecture robotique neuro-inspirée pour la coordination de stratégies d'apprentissage

Sommaire

Dans ce chapitre, nous nous intéressons aux architectures de contrôle pour la robotique et à la manière dont un agent artificiel incarné peut être doté de capacités de délibération lui permettant de prendre les bonnes décisions face aux situations qu'il expérimente. Dans un premier temps, nous commençons par présenter les travaux de recherche passés ayant permis le développement d'outils de planification automatique et leur incarnation dans des agents robotiques à l'origine des architectures de contrôle modernes. Dans un second temps, et à la lumière de ce bagage scientifique, nous présentons une nouvelle architecture de contrôle robotique neuro-inspirée, générique et simplifiée, en nous concentrant sur la couche décisionnelle. Celle-ci présente des capacités de planification permettant au robot de résoudre rapidement le problème auquel il fait face, mais coûteuses en termes de ressources calculatoires, ainsi que des capacités d'apprentissage par renforcement direct, demandant au robot un plus long temps d'assimilation, mais aussi bien moins énergivores. Pour

permettre au robot de tirer bénéfice des avantages de chacune de ces capacités, c'est-à-dire d'atteindre à moindre coût la meilleure performance face à un problème donné, la couche décisionnelle est aussi pourvue de capacités de méta-contrôle, coordonnant à la volée de manière coopérative les statégies comportementales du robot.

5.1 Les architectures de contrôles

5.1.1 Architectures robotiques

L'architecture de contrôle que nous proposons est construite sur la base d'une architecture robotique dite hybride, permettant de faire l'interface entre un niveau symbolique, maintenant des représentations à long terme, et un niveau sous-symbolique, s'appuyant directement sur les données reçues par les capteurs du robot. Ce type d'architecture, démocratisée au cours des années 90, vise à réconcilier l'approche délibérative et l'approche réactive :

— L'approche délibérative : cette approche consiste pour le robot à former un modèle du monde grâce à ses capteurs embarqués et à utiliser ce modèle afin de planifier une solution au problème auquel il est confronté. Dans le cadre robotique, la planification consiste à trouver l'enchaînement des actions symboliques qui permettent au robot d'atteindre ses buts à long terme (Nilsson 1969, Fikes et Nilsson 1971). Malheureusement, l'approche délibérative se heurte à la difficulté que représente le fait de générer, à moindre coût, un modèle satisfaisant d'un monde complexe et évolutif.

— L'approche réactive : cette approche consiste à fournir au robot une hiérarchie de comportements évalués en parallèle, à des échelles de temps différents, et pouvant potentiellement s'inhiber les uns les autres et remplacer leur sortie motrice respective (Brooks 1986). L'argument étant qu'il est ainsi possible d'obtenir un robot réactif pouvant adopter des comportements complexes, et ce, sans se confronter aux limites de la planification. Pour autant, l'approche réactive n'est pas sans défaut, et est rapidement confrontée à ses propres limites, notamment au manque de modularité engendré par la définition a posteriori du niveau de priorité des comportements, et par sa difficulté intrinsèque à prendre en compte les buts à long terme.

Une architecture hybride tend à bénéficier des avantages des deux approches. En 1989, l'architecture robotique AuRA permettait de naviguer grâce à des *motor schemas*, des comportements primitifs encodés sous la forme de champs de potentiel, et dont l'interaction générait des comportements plus complexes (Arkin 1989). Quelques années plus tard, conscient des limites de l'approche réactive, une approche hybride est adoptée par Arkin, et les *motor schemas* sont couplés avec des planificateurs et un séquenceur de plans (Arkin et al. 1987, Arkin 1990, Arkin et Balch 1997). À l'époque, Bonasso (1991) proposait déjà l'utilisation d'un planificateur inspiré de l'approche délibérative mais dont les opérateurs étaient des comportements hiérarchisés selon l'approche réactive. Suite à ces propositions, une multitude d'architectures hybrides vont être développées, dont

les architectures à trois couches. De manière générale, ces trois couches sont :

— La couche fonctionnelle : ce niveau contient les compétences de base du robot, définies par des boucles sensorimotrices réactives qui contrôlent les actionneurs lors de l'interaction avec l'environnement. Elle fonctionne à une échelle de temps rapide, sans états internes, et gère les fonctions bas-niveau spécialisées (e.g. déplacement et évitement d'obstacle dans le cas d'une tâche de navigation, saisie d'un objet dans le cas d'une tâche de rangement). Cette couche correspond à l'approche réactive.

— La couche décisionnelle : ce niveau contient les capacités de planification du robot. Elle fonctionne à une échelle de temps plus longue, avec des états internes, et calcule à un niveau symbolique le plan que le robot doit suivre pour atteindre ses objectifs. Cette couche correspond à l'approche délibérative.

— La couche exécutive : ce niveau représente l'interface entre les plans symboliques de la couche décisionnelle et les contrôleurs numériques de la couche fonctionnelle. Son rôle est d'exécuter le plan calculé par la première couche, en recrutant depuis la seconde les compétences du robot nécessaires à son exécution, ainsi que de superviser l'effet des actions réalisées.

Parmi les architectures robotiques à trois couches, nous pouvons mentionner l'architecture 3T (Bonasso et al. 1997) ; l'architecture ATLANTIS (Gat 1998), donnant à la couche exécutive la possibilité d'appeler la couche décisionnelle lorsque nécessaire ; l'architecture du LAAS (Alami et al. 1998), transférant le rôle de supervision de la couche exécutive vers la couche décisionnelle ; ou encore l'architecture proposée par l'équipe AMAC de l'ISIR (Renaudo 2016), dont est issue l'architecture robotique que nous proposons et présenterons dans la sous-partie suivante. Nous pouvons aussi mentionner les architectures à deux couches, telle que CLARAty (Volpe et al. 2001), dans laquelle la couche fonctionnelle est organisée selon le niveau d'abstraction du robot.

5.1.2 Architectures cognitives

Si notre travail s'inscrit avant tout dans l'étude des architectures robotiques, des architectures cognitives ont aussi été développées en parallèle, proposant de modéliser et d'organiser les propriétés internes du système cognitif d'un être vivant. Nous pouvons notamment citer l'architecture ACT-R, développée pour la première fois en 1993 (Anderson 1993), qui vise à définir les opérations cognitives et perceptuelles fondamentales et irréductibles qui permettent à l'esprit humain de fonctionner. Dans sa version la plus récente (Anderson 2009), elle est constituée de huit modules : le module d'objectif, le module d'imagination, celui de la perception visuelle, celui du contrôle manuel, celui de la perception orale, celui du contrôle vocal, celui de la mémoire déclarative et pour finir le module de la mémoire procédurale, au point de jonction de tous les autres. Les architectures robotiques étant vouées à être fonctionnelles, il est intéressant de chercher des points communs entre les problématiques de la robotique et celles considérées dans les sciences cognitives (Khamassi et al. 2016), ainsi

que d'adresser la question de l'architecture cognitive des robots (Vernon et al. 2007a;b, Kurup et Lebiere 2012). D'ailleurs, plusieurs architectures cognitives robotiques inspirées du modèle ACT-R ont dors-et-déjà été développées avec succès (Trafton et al. 2013, Bono et al. 2020).

5.2 Architecture proposée

5.2.1 Vue d'ensemble

L'architecture que nous proposons est une évolution de l'architecture de Renaudo (2016). Elle est une architecture à trois couches telle que présentée dans la sous-partie précédente. Dans ce travail, nous nous concentrons sur les capacités délibératives de la couche décisionnelle, les autres couches étant volontairement simplifiées.

La couche décisionnelle de l'architecture proposée (Fig. 5.1) est composée de deux experts organisés selon un modèle horizontal (voir chapitre 4), qui apprennent à leur manière en fonction de la récompense reçue, et qui génèrent en conséquence leur propre proposition d'action : l'expert model-based et l'expert model-free (abrégés MB et MF par la suite). Ces deux experts tirent leur nom des algorithmes d'AR éponymes, dont ils utilisent les méthodes. Tous deux exécutent trois processus à la suite :

- Le processus d'apprentissage, qui désigne pour l'expert MF la mise-à-jour de la valeur de l'association entre l'état courant et l'action effectuée, et pour l'expert MB la mise-à-jour de ses modèles du monde.
- Le processus d'inférence, qui désigne pour les deux experts l'évaluation des valeurs d'action de l'état courant, impliquant pour l'expert MB un travail de planification utilisant ses modèles du monde.
- Le processus de décision, qui désigne pour les deux experts la conversion des valeurs d'action de l'état courant en une distribution de probabilités d'actions, et à un tirage de leur proposition d'action depuis cette distribution.

Cette couche est également dotée d'un méta-contrôleur (abrégé MC par la suite) chargé d'arbitrer entre les propositions des deux experts, et donc de coordonner les deux types de comportements sous-jacents. À chaque itération, chacun des deux experts envoie au MC des données traduisant la qualité de leur apprentissage et le coût de leur inférence (t1). Ces données, dont nous détaillerons la nature dans la sous-partie traitant du MC, sont relatives à l'état courant dans lequel le robot se situe au moment de l'envoi, et sont calculées sur la base de la mise-à-jour des connaissances du robot effectuée la dernière fois que celui-ci est passé dans ce même état. Le MC utilise ces données issues du passé afin de calculer des valeurs de compromis selon un critère d'arbitrage, qui va lui permettre de designer l'expert dont les processus d'inférence et de décision seront inhibés, et celui dont ces même processus seront exécutés, qui dirigera donc le comportement du robot à cette itération. Nous décrirons précisément le critère d'arbitrage dans la partie traitant du MC. Ce signal d'exécution est ensuite envoyé aux experts (t2). Après avoir pris une décision, l'expert gagnant envoie sa proposition d'action au MC (t3), qui transmet lui même l'action à effectuer à la couche exécutive (t4). La couche exécutive s'assure

de son accomplissement en recrutant les compétences du robot depuis la couche fonctionnelle. Une fois l'action exécutée, le robot atteint (ou non) un nouvel état et obtient (ou non) une récompense. Il est important d'insister sur le fait que peu importe l'identité de l'expert désigné par le MC pour contrôler le comportement du robot à cette itération, les deux experts vont mettre à jour leurs connaissances de l'environnement, c'est à dire intégrer la conséquence de l'action qui a été exécutée (potentiel nouvel état atteint, potentielle récompense obtenue). En effet, à l'inverse des processus d'inférence et de décision, le processus d'apprentissage n'est jamais inhibé. Ainsi, les deux experts sont capables d'apprendre de la décision de l'autre, même lorsqu'ils ne dirigent pas le comportement du robot. Cette spécificité est un point central de notre architecture, car dès lors, les experts ne sont plus vraiment en compétition, mais coopèrent en réalité pleinement. La prochaine fois que le robot passera à nouveau dans l'état courant, c'est donc sur la base de ces connaissances mises à jour que les deux experts transmettront au MC les informations de *qualité d'apprentissage* et de *coût d'inférence*.

Cette architecture a été construite afin de pouvoir contrôler un robot réalisant tout type de tâches, et n'est donc pas astreinte à une catégorie de problèmes. Pour ce faire, la couche décisionnelle reçoit et produit des données symboliques discrètes, interprétées et traduites numériquement par la couche exécutive, et originaires ou à destination de la couche fonctionnelle. Dans toutes les tâches, l'objectif symbolique est le même : atteindre l'état récompensé et accumuler de la récompense au cours du temps. Adopter un meilleur comportement consiste donc à adopter un comportement maximisant ce gain de récompense.

À noter que dans l'architecture de Renaudo (2016), dont nous tirons notre inspiration principale, les experts et le MC communiquaient de manière relativement différente. En effet, dans Renaudo (2016), les deux experts commençaient d'abord par prendre une décision, alors que dans notre cas, seul l'expert ayant été choisi au préalable par le MC lui envoie sa décision (t3). Ensuite, dans Renaudo (2016), les deux experts envoyaient cette décision au MC accompagnée des données nécessaires à l'arbitrage, alors que dans notre cas, les experts envoient dans un premier temps seulement les données nécessaire à l'arbitrage (t1). Selon le critère utilisé, le MC de Renaudo (2016) prenait finalement une décision et choisissait l'action d'un des experts à envoyer à la couche exécutive, alors que dans notre cas, le MC envoie dans un premier temps un signal d'exécution aux experts (t2), et dans un second temps l'action de l'expert choisit à la couche exécutive (t4). Dans Renaudo (2016), à l'itération suivante, l'expert dont l'action n'avait pas été sélectionné précédemment voyait son processus d'inférence être inhibé. Dans notre cas, l'expert qui n'est pas choisit par le MC voit son processus d'inférence être inhibé à cette itération-ci (entre t2 et t3).

5.2.2 Expert model-based

Suivant la proposition faite par Daw et al. (2005) et les connaissances mises en avant dans le 4, nous nous appuyons sur l'analogie entre les comportements dirigés vers un but et l'AR model-based (aussi appelé AR

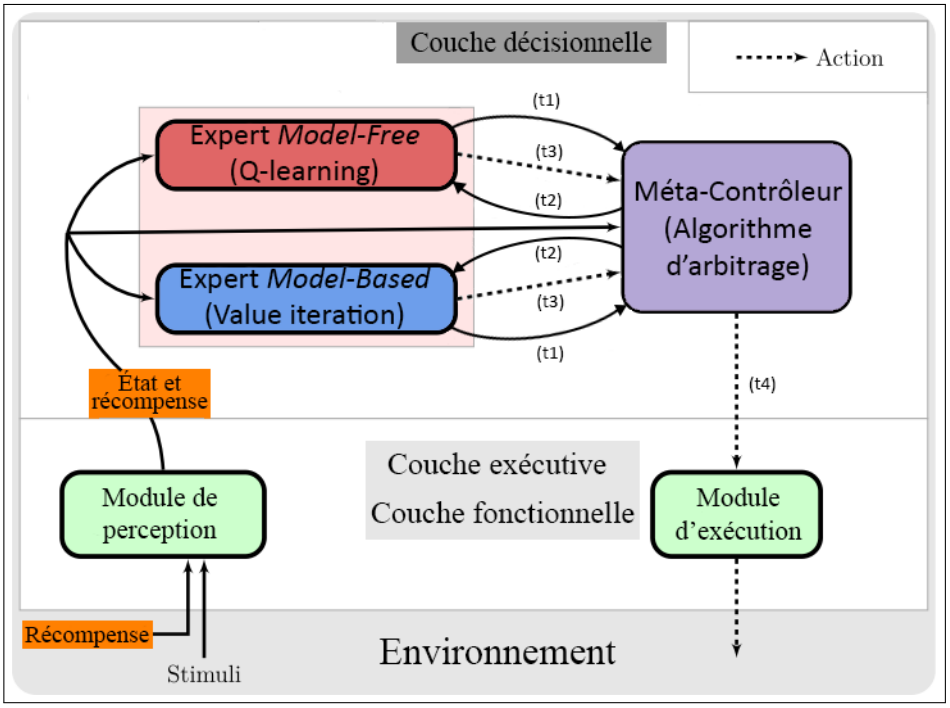

FIGURE 5.1 – *Version générique de l'architecture. Deux experts ayant des propriétés différentes calculent la prochaine action à faire dans l'état actuel s. Ils envoient chacun au méta-contrôleur (MC) des données relatives à leur d'apprentissage et à leur inférence (t1). Le MC désigne l'expert gagnant selon un critère qui utilise ces données et l'autorise à exécuter ses processus d'inférence et de décision (t2). Après avoir pris une décision, l'expert gagnant envoie sa proposition au MC (t3), qui envoie l'action à la couche exécutive (t4). L'effet de l'action exécutée génère une nouvelle perception, transformée en un état markovien abstrait s, et associée à une récompense r (nulle ou non). Ces deux données sont envoyées aux experts. Chaque expert apprend en fonction de l'action choisie par le MC, du nouvel état atteint et de la récompense.*

indirect, voir le chapitre 2) pour définir l'expert MB. Sommairement, son fonctionnement consiste en l'apprentissage d'un modèle de transition T et d'un modèle de récompense R du problème, suivi d'une planification dans l'espace d'état des modèles, c'est-à-dire la propagation des récompenses connues dans le graphique de transition, permettant l'évaluation des valeurs d'état-action de ces états, et donc une prise de décision.

Ces modèles du monde permettent de simuler sur plusieurs étapes les conséquences de la poursuite d'un comportement donné, et de rechercher les états souhaitables à atteindre. Par conséquent, lorsque le robot réalise que l'environnement a changé, il peut utiliser cette connaissance du monde pour trouver instantanément le nouveau comportement pertinent. Cependant, ce processus de recherche est coûteux en temps de calcul car il doit simuler plusieurs itérations de valeurs dans chaque état pour trouver la solution correcte (Sutton et Barto 1998).

Processus d'apprentissage

Le processus d'apprentissage de l'expert MB consiste à mettre à jour les modèles de récompense et de transition en interagissant avec le monde. Ici, l'agent ne connaît pas ces modèles a priori : il les construit au fur et à mesure de ses interactions avec l'environnement. Comme défini dans

2.7, le modèle de transition T est appris en comptant les occurrences des transitions (s, a, s') et en divisant ensuite cette valeur par la somme de tous les états atteints depuis s en effectuant l'action a.

Le modèle de récompense R stocke la valeur de récompense reçue la plus récente r_t, après avoir effectué une action a dans l'état s et avoir atteint l'état actuel s', multipliée par la probabilité de la transition (s, a, s'). La multiplication de la récompense reçue par la probabilité de transition vers l'état récompensé est un point critique. En effet, dans le cas où le modèle de transitions est probabiliste, c'est-à-dire qu'une même action réalisée dans un même état peut emmener dans plusieurs états, sans cette multiplication, réaliser l'action connue pour mener dans l'état récompensé sans y parvenir entraînerait pour l'agent l'oubli de la récompense. Cet ajout est l'une des modifications que nous avons effectuée par rapport à l'architecture de Renaudo (2016).

Processus d'inférence

La réalisation du processus d'inférence consiste à planifier en utilisant un algorithme tabulaire de *Value Iteration* 2.2 (Sutton et Barto 1998).

Dans l'algorithme original, à chaque étape de décision, les valeurs d'état-action sont réinitialisées et le modèle mis à jour est utilisé pour les estimer à nouveau. Dans notre version, nous étalons le calcul sur plusieurs étapes de décision : au lieu de réinitialiser les valeurs, nous commençons par des valeurs initiales égales au résultat de l'estimation précédente. Cette version réduit le coût global de l'inférence et accélère la convergence vers une fonction de valeur de qualité (Renaudo et al. 2015b).

Processus de décision

Le processus de décision consiste tout simplement à convertir l'estimation des valeurs d'état-action en une distribution de probabilités de sélection d'actions à l'aide d'une fonction *softmax* (2.14), et à tirer la proposition d'action de cette distribution.

5.2.3 Expert model-free

Ici, nous nous appuyons sur l'analogie entre l'habitude comportementale et l'AR model-free (aussi appelé AR direct, voir le chapitre 2) pour définir l'expert MF (Daw et al. 2005). À l'inverse de l'expert MB, et comme son nom l'indique, l'expert MF n'utilise pas de modèles du monde pour décider de l'action à faire dans chaque état, mais apprend directement les associations état-action correspondantes en mettant en cache les récompenses sous forme de valeurs d'état-action. Comme la mise-à-jour des valeurs d'état-action est locale à l'état courant, le processus est lent et le robot ne peut pas apprendre les relations topologiques entre les états. Par conséquent, lorsque l'environnement change, le robot effectue de nombreuses actions afin d'adopter le nouveau comportement pertinent. En revanche, cette méthode est bien moins coûteuse en termes de durée du processus d'inférence que celle employée par l'expert MB.

Processus d'apprentissage

Le processus d'apprentissage de l'expert MF consiste à estimer la valeur de l'action effectuée dans l'état précédent $Q(s, a)$ en utilisant un algorithme de *Q-learning* 2.11 (Sutton et Barto 1998).

Dans la version précédente de l'architecture (Renaudo 2016), le *Q-learning* de l'expert MF était implémenté sous la forme d'un réseau de neurones sans couche cachée, dont chaque neurone d'entrée encodait l'état renvoyé par le module de perception, chaque neurone de sortie encodait une des actions réalisables, et où l'activité du neurone représentait la valeur d'état-action $Q(s, a)$ associée. Nous sommes passés à une version tabulaire du *Q-learning*, jugeant dans ce cadre précis les avantages d'un réseau de neurones trop peu nombreux face à ses inconvénients structurels : approximation de la fonction de valeur, absence de méthode systématique permettant de définir la meilleure topologie du réseau et le nombre de neurones à placer dans les éventuelles couches cachées, choix des valeurs initiales des poids du réseau, réglage du pas d'apprentissage, inintelligibilité des valeurs de poids du réseau pour l'utilisateur, etc. Cela nous permet aussi une certaine symétrie entre les deux experts, l'expert MB utilisant lui aussi un algorithme d'apprentissage tabulaire.

Processus d'inférence

Comme l'expert MF n'utilise pas de méthode de planification, à l'inverse de l'expert MB, son processus d'inférence consiste uniquement à évaluer depuis le tableau qui contient toutes les valeurs d'état-action celles associées à l'état s. De fait, le processus d'inférence de l'expert MF est extrêmement peu coûteux, mais ne l'aide pas à aboutir à un comportement efficient et à s'adapter aux changements environnementaux.

Processus de décision

Le processus de décision est le même que celui de l'expert MB (2.14).

5.2.4 Méta-contrôleur

À chaque itération, le MC est chargé d'évaluer chaque expert afin de choisir qui d'entre eux deux générera le comportement du robot. Pour cela, il utilise un nouveau critère d'arbitrage que nous avons défini sur la base des théories psychologiques s'accordant à dire que l'humain effectue une analyse coûts-avantages lorsqu'il alloue un "effort mental" à une tâche (Kurzban et al. 2013, Shenhav et al. 2017), sur la base des paradigmes expérimentaux affinant ces théories (Boureau et al. 2015, Kool et al. 2017; 2018) et sur le travail exploratoire entreprit dans Renaudo et al. (2015c) avec l'architecture dont la nôtre est l'évolution. Ce nouveau critère d'arbitrage est donc un compromis entre la qualité de l'apprentissage (à maximiser) et le coût de l'inférence (à minimiser).

L'autre nouveauté fondamentale par rapport à Renaudo (2016) est le fait que les valeurs de compromis sont dorénavant calculées localement comme dans Caluwaerts et al. (2012a). Auparavant, les valeurs de compromis étaient calculées seulement de manière globale : à un temps donné, il

existait uniquement deux valeurs de compromis (une par expert), et cela peu importe l'état de l'environnement dans lequel le robot se situait. Maintenant, chaque état possède une valeur de compromis par expert. Grâce à cette évolution, à un temps donné, le robot peut décider de favoriser tel ou tel expert en fonction de la qualité de l'apprentissage et du coût de l'inférence de chacun d'entre eux, mais cela aussi dépendamment de l'état de l'environnement dans lequel il se situe. Pour résumer, à chaque itération, le MC évalue qui des deux experts avait exhibé la meilleure qualité d'apprentissage la dernière fois que le robot s'était retrouvé dans l'état dans lequel il se trouve actuellement, tout en évaluant à quel point son processus d'inférence avait été onéreux, et choisit en conséquence l'expert qui dirigera le comportement du robot dans cet état.

Qualité de l'apprentissage

En nous appuyant sur des études ultérieures (Viejo et al. 2015), nous définissons la qualité de l'apprentissage comme l'entropie de la distribution des probabilités de sélection d'actions, lissée au cours du temps par un filtre passe-bas permettant d'être moins sensible aux perturbations. Concrètement, nous considérons que plus une action est choisie par un expert, et plus cela signifie qu'elle lui permet d'accumuler de la récompense au cours du temps, et donc qu'elle est l'action à favoriser par le système. Si c'est le cas, la probabilité de sélectionner cette action sera plus haute que les autres, ce qui entraînera l'apparition d'un "pic" au niveau de la distribution des probabilités de sélection. De ce fait, plus la distribution sera piquée, et plus l'entropie sera basse. À l'inverse, plus la distribution sera lissée, et plus l'entropie sera haute. Maximiser la qualité de l'apprentissage, comme nous souhaitons le faire, revient donc à minimiser cette valeur d'entropie.

À chaque étape t, si l'expert E est sélectionné pour diriger la décision, ses probabilités de sélection d'actions (3) sont filtrées à l'aide d'un filtre passe-bas et stockées par le système :

$$f(P(a|s,E,t)) = (1,0-\alpha) * f(P(a|s,E,t-1)) + \alpha * P(a|s,E,t) \tag{5.1}$$

Sinon, aucun filtre passe-bas n'est appliqué :

$$f(P(a|s,E,t)) = f(P(a|s,E,t-1)) \tag{5.2}$$

En utilisant la distribution de probabilité de l'action filtrée $f(P(a|s,E,t))$, le MC peut calculer l'entropie $H(s,E,t)$ de chaque expert :

$$H(s,E,t) = -\sum_{a=0}^{|\mathcal{A}|} f(P(a|s,E,t)) \cdot log_2(f(P(a|s,E,t)) \tag{5.3}$$

Coût de l'inférence

À chaque étape de temps t et pour chaque état s, la durée $T(s,E,t)$ du processus d'inférence de chaque expert est tout simplement enregistrée, et filtrée de la même manière que les probabilités de sélection d'actions (5.1).

Critère d'arbitrage

En utilisant la qualité de l'apprentissage et le coût de l'inférence, le MC calcule une valeur de compromis $Q(s, E, t)$ pour chaque expert :

$$Q(s, E, t) = -(H(s, E, t) + e^{-\kappa H(s,MF,t)} T(s, E, t)) \tag{5.4}$$

Ici, $e^{-\kappa H(s,MF,t)}$ permet de pondérer l'impact du coût de l'inférence $T(s, E, t)$ dans la valeur de compromis. En effet, nous considérons que lorsque les valeurs d'entropie sont hautes, et donc que la qualité d'apprentissage des experts est médiocre, l'important est avant tout de faire en sorte que la qualité globale de l'apprentissage augmente. Pour cela, et comme les experts sont capables d'apprendre l'un de l'autre, même lorsqu'ils ne dirigent pas le comportement du robot, il est plus intéressant de donner le contrôle du comportement à l'expert ayant la meilleure qualité d'apprentissage, et donc la valeur d'entropie la plus faible. Dans cette situation, se préoccuper du coût du processus d'inférence des experts passe au second plan. La valeur de $e^{-\kappa H(s,MF,t)}$ doit donc être faible, pour donner moins d'importance au membre $T(s, E, t)$ de l'équation 5.4. À l'inverse, plus la qualité d'apprentissage des experts est haute, et donc plus les valeurs d'entropie sont faibles, plus il devient intéressant de pouvoir discriminer les deux experts afin de donner le contrôle du comportement du robot à celui qui saura économiser le plus de ressources calculatoires. Se préoccuper du coût du processus d'inférence devient alors nécessaire afin de distinguer les deux experts exhibant une bonne qualité d'apprentissage. Dans ce cas, la valeur de $e^{-\kappa H(s,MF,t)}$ doit être haute, pour donner plus d'importance au membre $T(s, E, t))$ de l'équation 5.4. La figure 5.2 illustre cette réflexion en présentant la manière dont $e^{-\kappa H(s,MF,t)}$ évolue en fonction de la valeur d'entropie $H(s, MF, t)$ et du paramètre κ.

κ est un paramètre constant pondérant le terme d'entropie et fixé selon une analyse de front de Pareto (Powell et Sammut-Bonnici 2015). Nous présenterons en détail cette analyse dans le chapitre 6. Dans tous les cas, nous recherchions une valeur de κ qui maximisait la capacité de l'agent à accumuler de la récompense au fil du temps, tout en minimisant le coût de son inférence.

Le choix d'utiliser la valeur d'entropie de l'expert MF dans le calcul de $e^{-\kappa H(s,MF,t)}$ plutôt que celle de l'expert MB, ou une valeur d'entropie moyenne, s'explique par le fait que, de par sa méthode d'apprentissage par renforcement, l'expert MF est quasi systématiquement l'expert qui prend le plus de temps à converger vers un comportement de qualité. De ce fait, en utilisant la valeur d'entropie de l'expert MF, souvent supérieure à celle de l'expert MB, la valeur de $e^{-\kappa H(s,MF,t)}$ va plus longtemps rester basse, et donc, la qualité de l'apprentissage restera plus longtemps le levier majeur du calcul du compromis. Ainsi, on s'assure de donner plus longtemps le contrôle du comportement du robot à l'expert ayant la meilleure qualité d'apprentissage, et donc de stimuler l'apprentissage de l'expert en retard (très souvent l'expert MF). Ce choix marque à nouveau le caractère avant tout coopératif de notre modèle de coordination.

Finalement, une fois ces valeurs de compromis $Q(s, E, t)$ calculées, le MC les convertit en une distribution de probabilités d'experts à l'aide d'une fonction *softmax* (2.14), et tire l'expert gagnant de cette distribution.

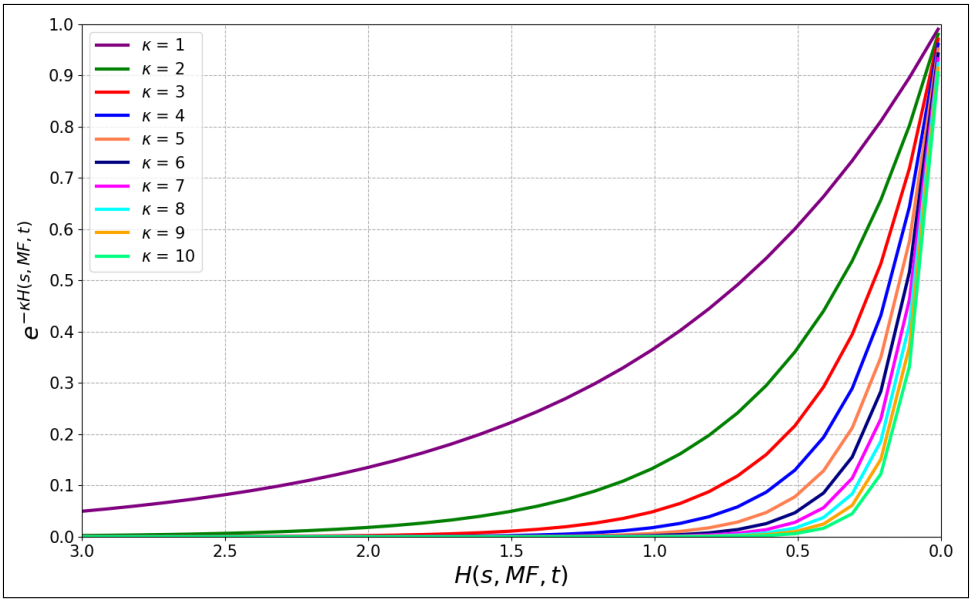

FIGURE 5.2 – *Évolution de la valeur de $e^{-\kappa H(s,MF,t)}$ en fonction de la valeur d'entropie $H(s, MF, t)$ et du paramètre κ.*

Le processus d'inférence de l'expert non choisi est inhibé, ce qui permet au système d'économiser du temps de calcul.

5.2.5 Module de perception

Lorsqu'un robot évolue dans un environnement réel, et donc hautement complexe, il devient rapidement impossible d'énumérer les différents états de l'environnement dans lequel il se trouve. Dans ces conditions, il devient plus judicieux de générer automatiquement les états que le robot traverse à partir des informations perceptuelles qu'il reçoit. Ces informations sont sélectionnées selon leur niveau de pertinence pour la tâche, ancrant ainsi les informations utilisées au niveau décisionnel. Dans notre architecture, c'est le module de perception qui se charge de cela, en transformant dynamiquement les informations perçues et éventuellement stockées en un état abstrait et discret. Cette génération automatique d'états à partir de données numériques n'est pas tâche-indépendante, et les critères pertinents définissant ce que représente un état dans telle tâche doivent être fixés à l'avance. À noter que dans ce travail, nous ne cherchons pas à proposer une méthode de génération automatique d'états tâche-indépendante. Passé ce niveau de spécificité, la couche décisionnelle ayant été construite de la manière la plus générique qui soit, elle devient en revanche capable de manipuler des états représentant des informations de nature totalement différentes. Pour autant, et même si le robot est capable d'abstraire et de généraliser ses perceptions, un environnement réel reste bien souvent très volatile et sujet au changement. Dans ces conditions, il n'est pas nécessairement judicieux de fournir au robot un modèle explicite du problème qu'il doit résoudre, aussi abstrait soit-il. Une fois les critères définissant les états fixés, on préférera alors laisser le robot se faire sa propre représentation (abstraite) du monde en expérimentant. C'est pour cela qu'aucun modèle du monde n'est donné au robot a priori,

et que celui-ci doit les construire au fur et à mesure de ses interactions avec l'environnement. Pour terminer, le module de perception reçoit aussi un signal de récompense abstrait, permettant aux experts d'apprendre. Nous considérons que cette récompense est directement donnée par l'environnement (récompense localisée dans l'environnement, récompense donnée par un humain, etc.) ou par un système branché à notre architecture de contrôle (système d'auto-valorisation du robot, système d'interprétation des signaux sociaux d'un humain, etc.).

5.2.6 Module d'exécution

Tout comme les informations perçues par le robot sont interprétées en état abstraits dont les critères ont été définis par le concepteur, dans cette architecture, les propositions des experts sont générées sous la forme d'actions symboliques et discrètes, dont le sens réel échappe au robot. Au niveau de sa couche décisionnelle, ces actions n'ont de sens qu'en termes de valeurs et de conséquences directes sur l'environnement. En utilisant une symbolique décorrélée du sens réel des actions et partagée entre les deux experts, cela facilite grandement le transfert du contrôle du robot vers l'un ou l'autre expert par le MC, et permet à l'architecture générique que nous proposons de manipuler des actions de natures totalement différentes. De plus, et contrairement à certaines architectures robotiques, jamais aucune notion d'échec d'une action n'est formalisée dans cette architecture. Une action n'échoue jamais : elle emmène ou non dans un nouvel état, et permet ou non d'obtenir une récompense. Ne pas changer d'état ou ne pas obtenir de récompense n'est pas considéré en tant que tel comme un échec. Cela est simplement la conséquence d'une action. C'est uniquement à la lumière de la définition de la tâche à réaliser pour le robot, et du comportement appris par celui-ci, qu'une action pourra être considérée comme ayant échoué ou non. Cela implique aussi la capacité du système à traiter des actions probabilistes, c'est-à-dire des actions pouvant emmener dans plusieurs états différents depuis le même état, ce que notre architecture est capable de faire, comme mentionnée précédemment. Au final, l'ensemble des actions symboliques que les experts peuvent proposer représentent l'ensemble des compétences du robot, que les experts peuvent ou non recruter pour atteindre un but particulier. Une fois la proposition d'action symbolique finale retenue par le MC, il devient nécessaire de la traduire en ordres moteurs continus effectifs pour le robot. Là est le rôle du module d'exécution : il va recruter depuis la couche fonctionnelle les compétences du robot nécessaires à son exécution, et superviser l'effet de l'action réalisée sur l'environnement réel.

5.3 Conclusion

Dans ce chapitre, nous avons présenté une architecture robotique générique à trois couches intégrant des mécanismes de coordination d'experts d'apprentissage. Par rapport à la version précédente (Renaudo 2016), la nouveauté principale réside dans la mesure explicite en ligne des performances et du coût de chaque expert, de manière à donner le contrôle à l'expert exhibant à chaque instant le meilleur compromis entre les deux.

Chacun de ces deux experts coopère avec l'autre et possède des avantages et des inconvénients :

— l'expert MB est coûteux en termes de ressources calculatoires, mais rapide à mettre en place un comportement pertinent et à s'adapter aux changements environnementaux.
— l'expert MF est peu coûteux en termes de ressources calculatoires, mais lent à mettre en place un comportement pertinent et à s'adapter aux changements environnementaux.

Avec un tel système, nous supposons observer la dynamique de coordination suivante, peu importe la tâche que le robot doit réaliser : en début d'expérience, le MC donne à l'expert MB le contrôle du comportement du robot, étant l'expert permettant d'accumuler le plus rapidement de la récompense au cours du temps, et donc exhibant la meilleure performance à court terme. Les deux experts bénéficiant de l'apprentissage de chacun, même lorsque ceux-ci ne dirigent pas le comportement, l'expert MF va au fur et à mesure du temps réussir à atteindre la performance de l'expert MB. Étant moins coûteux que l'expert MB en termes de ressources calculatoires, arrivera donc le moment où l'expert MF sera plus intéressant à utiliser, et où le MC lui donnera le contrôle du comportement du robot. Si au cours de l'expérience l'environnement est amené à évoluer, nous supposons que le MC rendra le contrôle du comportement à l'expert MB, plus apte à s'adapter aux changements environnementaux, et donc plus apte à continuer d'accumuler de la récompense. Lorsque l'expert MF, plus lent, aura lui aussi réussi à s'adapter à ces changements, et à adopter un comportement pertinent, alors le MC pourra à nouveau lui donner le contrôle du robot. Cette dynamique que nous pouvons nommer "planifier jusqu'à l'habitude" est celle observée par Keramati et al. (2016) chez des sujets humains réalisant des tâches de bandit manchot. Plus largement, nous souhaitons aussi illustrer le caractère générique de notre architecture robotique et sa capacité à contrôler des agents réalisant différentes tâches d'apprentissage. Pour ce faire, nous évaluons dans le chapitre 6 notre architecture robotique dans une tâche de navigation réelle et simulée. Dans les chapitres 7 et 8, nous l'évaluerons dans des tâches d'interaction et de coopération humain-robot. À l'heure de la crise climatique, et même si là n'était pas l'objectif premier de cette thèse, notre architecture robotique s'inscrit aussi dans le développement d'IA plus économiques, et donc plus écologiques. Une problématique qui participe à argumenter en faveur de notre méthode.

6 Évaluation du système d'arbitrage dans une tâche de navigation

Sommaire

Dans ce chapitre, nous évaluons l'architecture robotique et le système de coordination dont nous avons présenté le fonctionnement dans le chapitre 5 lors d'expériences de navigation autonome réelles. Si la navigation autonome est aujourd'hui un sujet phare de la recherche scientifique internationale, en témoigne les nombreux progrès réalisés dans la conception de véhicules autonomes ces quinze dernières années (Barbier et al. 2018, Van Brummelen et al. 2018, Saraydaryan et al. 2018, Litman 2020), nous la considérons ici avant tout comme un cadre expérimental adapté à notre travail. Notre objectif était donc moins de faire avancer la recherche en navigation autonome que d'évaluer notre architecture robotique. Dans ce chapitre, nous commençons par présenter le protocole expérimental défini ainsi que le matériel utilisé. Dans un second temps, nous expliquons en quoi le développement d'un environnement simulé facile à mettre en place nous a permis d'évaluer et d'optimiser rapidement notre système robotique. Pour finir, nous présentons les résultats obtenus en simulation

puis dans l'environnement réel, et montrons que notre système de coordination permet au robot de maintenir un haut niveau de performance tout en diminuant plus de deux fois son coût calculatoire, et cela même lorsque l'environnement est sujet à des changements de but ou à l'ajout d'obstacles. Ce chapitre correspond à une version étendue de la publication Dromnelle et al. (2020b).

6.1 Matériel et protocole expérimental

6.1.1 Arène et robot

La première tâche expérimentale avec laquelle nous avons évalué notre architecture robotique de coordination est une tâche de navigation. Pour la réaliser, nous avons construit une arène de 2,6 m x 9,5 m contenant deux chemins possibles permettant de se déplacer entre les parties droites et gauches de l'arène ainsi qu'une impasse (Fig. 6.1). Le robot que nous avons utilisé est un *TurtleBot-1*. L'ordinateur servant de calculateur au robot utilise *ROS* (*Robot Operating System*, Quigley et al. (2009)) pour traiter les signaux de ses capteurs, contrôler la base mobile et s'interfacer avec notre architecture. Le robot possède deux types de capteurs : un capteur visuel Kinect-1, qui renvoie une estimation de la distance aux obstacles dans son champ de vision, et des capteurs de contact, à l'avant et sur les côtés de la base mobile.

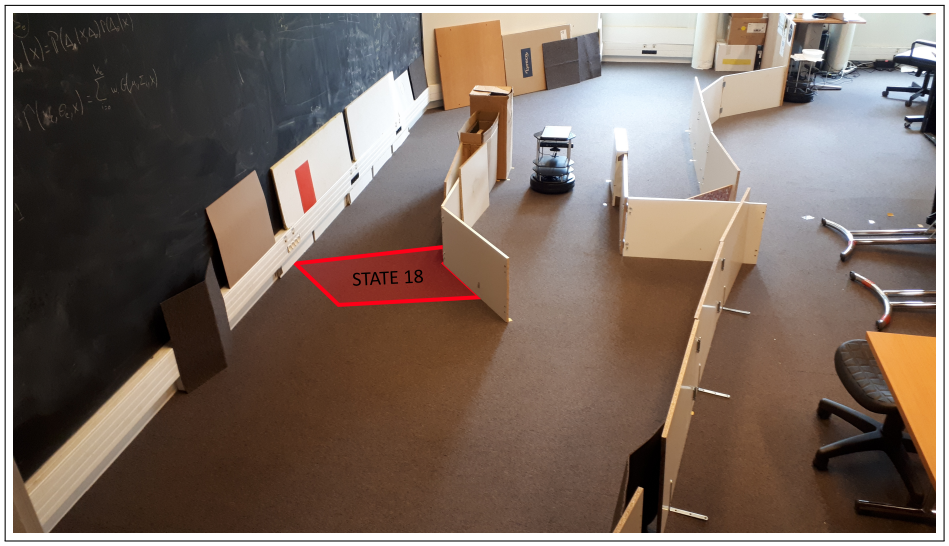

Figure 6.1 – ***A****. Photo de l'arène et du TurtleBot-1 empruntant le couloir central. L'état initialement récompensé (le numéro* 18*) est représenté en rouge.*

6.1.2 Perception du robot

Pour se localiser dans l'environnement, le robot utilise un algorithme de cartographie et de localisation simultanées (abrégé SLAM par la suite, Grisetti et al. (2007)) par filtrage particulaire appelé GMapping. De manière générale, lorsqu'un robot est en mouvement, il peut utiliser la navigation à l'estime pour se situer dans l'espace, une méthode de navigation consistant à déduire la position d'un véhicule de sa route et de la distance

parcourue depuis sa dernière position connue, grâce à des instruments embarqués mesurant par exemple son cap, sa vitesse, le temps passé ou encore l'influence de l'environnement sur son déplacement. Malheureusement, du fait qu'elle repose sur la mesure d'informations internes au robot, cette méthode est incertaine, car dépendante de la précision de ces mesures, mise à mal par les potentiels frottements, glissements, jeux, etc. L'autre façon dont dispose le robot pour se situer dans l'espace consiste alors à utiliser des informations non plus internes, mais collectées depuis sont environnement, via des capteurs extéroceptifs. La méthode du SLAM propose d'allier ces deux méthodes, en permettant au robot de se localiser dans l'espace tout en construisant en même temps une carte de son environnement. Dans notre cas, l'algorithme maintient un nombre N de particules, chacune étant une estimation de la position du robot, associée à la probabilité que le robot y soit effectivement. Ces estimations sont mises à jour en fonction de l'odométrie du robot (source d'information interne) et d'une observation réalisée par sa Kinect (source d'information environnementale), correspondant à la mesure de profondeur limitée à une hauteur fixe. À noter qu'en neurosciences, l'hippocampe semble être impliqué dans des calculs de type SLAM (Howard et al. 2005, Fox et Prescott 2010), impliquant entre autre le fonctionnement des *cellules de lieu*, signe que cette méthode de navigation semble proche de celle utilisée chez l'animal. Il existe d'ailleurs des systèmes SLAM bio-inspirés, tels que RatSLAM (Milford et al. 2004). À partir de ces informations, le robot est donc capable de construire progressivement une carte topologique de l'environnement. Si on lui offre en plus de cela la possibilité de placer de manière autonome des centres régulièrement espacés sur cette carte, il sera alors capable de discrétiser la carte en cellules (Fig. 6.2 gauche). Nous considérons que la cellule dans laquelle se trouve le robot est celle associée au centre le plus proche de sa position lorsqu'il a terminé de réaliser son déplacement. Les cellules ont toutes un rayon de 35 centimètres, fixé par l'expérimentateur par rapport à la taille des couloirs, afin qu'aucun mur ne soit à cheval entre plusieurs états. Dans cette tâche, nous considérons que la carte discrétisée de 38 états de la figure 6.2 représente l'espace d'état du robot. Ici, les capteurs du robot ne lui servent donc qu'à savoir dans quel état il se trouve à chaque instant.

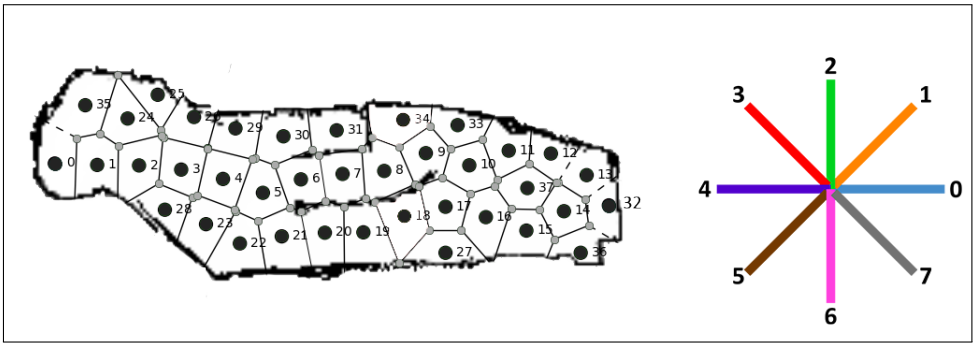

FIGURE 6.2 – ***Gauche***. *Carte de l'arène créée de manière autonome par le robot et utilisée dans les expériences. Elle est composée de 38 états.* ***Droite***. *L'étoile de couleur à huit branches indique chacune des directions dans laquelle le robot peut se déplacer dans l'arène, et donc sur la carte.*

6.1.3 Actions du robot

Le robot peut se déplacer dans 8 directions allocentriques équitablement réparties, représentées dans la figure 6.2. Son espace d'action est donc un espace d'action discret. Lorsqu'une direction est sélectionnée, le robot s'oriente, avance jusqu'à atteindre un nouvel état puis parcourt une distance fixe de 0,20 mètres, l'empêchant ainsi de s'arrêter en bordure de deux états. Une fois cette distance parcourue, l'action est considérée comme finie et une nouvelle décision peut être prise dans l'état courant. Le robot est aussi capable de gérer de manière autonome l'évitement d'obstacle : lorsqu'il se rapproche progressivement d'un mur et que son déplacement n'est pas terminé, il s'oriente progressivement vers le côté de plus grande ouverture perçue (à une vitesse angulaire de 0 :2 rad/s). S'il est trop proche du mur, le robot s'arrête et pivote vers ce même côté. Enfin, si le robot est trop proche d'un obstacle et que la Kinect ne reçoit plus aucune donnée, il risque de percuter le mur, activant ainsi ses capteurs de contacts. Si cela arrive, il recule alors de 0,15 mètres et met fin à l'action qu'il était en train de réaliser.

6.1.4 Tâche de navigation

La tâche de navigation que nous avons conçue est divisée en deux phases distinctes : la phase préliminaire et la phase expérimentale.

Au cours de la phase préliminaire, le robot construit par exploration une carte topologique de l'environnement en suivant la méthode que nous avons présentée précédemment. Théoriquement, ce processus de création de carte pourrait être réalisé par le robot en même temps qu'il effectue sa tâche d'apprentissage. Réaliser cette phase à part nous assure cependant de générer une carte cohérente, quitte à recommencer le processus de construction si nous jugeons que la carte n'est pas satisfaisante. En effet, dû à certains facteurs environnementaux (qualité de l'observation de la Kinect, reflets, usure des roues, ressemblance entre plusieurs portions de l'arène), il a fallu nous y reprendre à plusieurs reprises afin d'obtenir une carte de la qualité de celle présentée dans la figure 6.2. De plus, en construisant cette carte à l'avance et en la réutilisant systématiquement pour chacune des expériences, nous réduisons les sources de variabilité de celles-ci en nous concentrant sur l'essentiel : la variabilité due à l'apprentissage du robot et la transferabilité des résultats de simulation lors du test ultérieur sur robot réel.

Une fois la carte est créée, l'expérience peut réellement commencer. Dès lors, un état est choisi comme étant l'état récompensé. L'objectif du robot est d'atteindre cet état. Pour ce faire, le robot utilise le système de méta-contrôle que nous avons présenté dans le chapitre 5, où un expert MF et un expert MB apprennent la tâche à réaliser, mais où seulement un seul des deux infère et décide à chaque pas de temps (chaque action). Lorsque le robot réussit à atteindre cet état, il reçoit une récompense unitaire et est renvoyé au hasard à l'une de ses deux positions initiales (les états 0 et 32), situées aux extrémités de l'arène. L'expérience débute systématiquement par une période stable, dans laquelle l'état 18 est l'état récompensé et où l'environnement ne change pas. S'en suit une seconde période marquée par un changement environnemental, dans laquelle l'état

34 devient l'état récompensé, ou dans laquelle des obstacles sont ajoutés à l'intérieur de l'arène. Les différents changements environnementaux sont illustrés dans la figure 6.3. Parmi les obstacles ajoutés, certains sont systématiquement placés afin d'empêcher le déplacement entre les états 16 et 37 et 16 et 15. D'autres sont aussi placés, soit entre les états 6 et 7, soit entre les états 20 et 21. Le choix de ces derniers placements est effectué en fonction de l'identité du couloir que le robot empruntait le plus souvent pour atteindre l'état 18, lorsqu'il se situait dans la partie gauche de l'arène. Si le robot empruntait davantage le couloir du bas, alors un obstacle était placé entre les états 20 et 21 afin de le forcer à emprunter le couloir central. Réciproquement, s'il empruntait majoritairement le couloir central, alors l'obstacle était placé entre les états 6 et 7.

Dans tous les cas, le changement environnemental a lieu à partir du 1600ème pas de temps. Cette durée a été choisie à partir de l'observation empirique suivante : le robot doit réaliser environ 800 actions minimum dans cette expérience afin que l'expert MF commence à exhiber un comportement pertinent. Du fait que nous nous intéressons à la coordination de systèmes d'apprentissage, et que notre méthode d'arbitrage repose en partie sur la qualité d'apprentissage des différents experts, il est indispensable que ces experts puissent avoir le temps de correctement apprendre à accumuler de la récompense au cours du temps. Comme mentionné dans la partie précédent, l'expert MF est celui des deux mettant le plus de temps à accumuler de la récompense. C'est donc par rapport à lui que le choix de la durée de la première période a été défini : nous avons choisi deux fois le minimum d'actions nécessaires à l'exhibition d'un comportement pertinent, soit 1600 actions. La seconde période dure 800 pas de temps, ce qui emmène chaque expérience à une durée totale de 2400 pas de temps.

À noter que dans cette expérience, en fonction des coordonnées exactes auxquelles se situe le robot à l'intérieur d'un même état, l'état d'arrivée ne sera pas nécessairement identique pour la même action effectuée. La représentation de l'environnement du robot étant probabiliste, cela implique donc l'utilisation par l'expert MB de modèles de transition T et de récompense R probabilistes, comme mentionné dans le chapitre 5.

6.2 Optimisation et extension de l'architecture

6.2.1 Développement d'un environnement simulé

Comme mentionné précédemment, si nous voulons observer une coordination entre les deux experts d'apprentissage, il est nécessaire que l'expert MF puisse apprendre à accumuler efficacement de la récompense au cours du temps. Lui donner assez de temps est une première solution. Une seconde consisterait à trouver le jeu de paramètres maximisant sa performance. Malheureusement, en moyenne, le robot met environ cinq heures pour exécuter 1600 actions (2 heures d'expérience, 1 heure de recharge, 2 heures d'expérience). Les expériences étant longues, il devenait difficile de tester une multitude de jeux de paramètres.

Pour nous permettre de contrer cette limite temporelle, nous avons développé une version simulée de la tâche de navigation. Certes dans notre expérience, le robot évolue dans un environnement continu réel difficile-

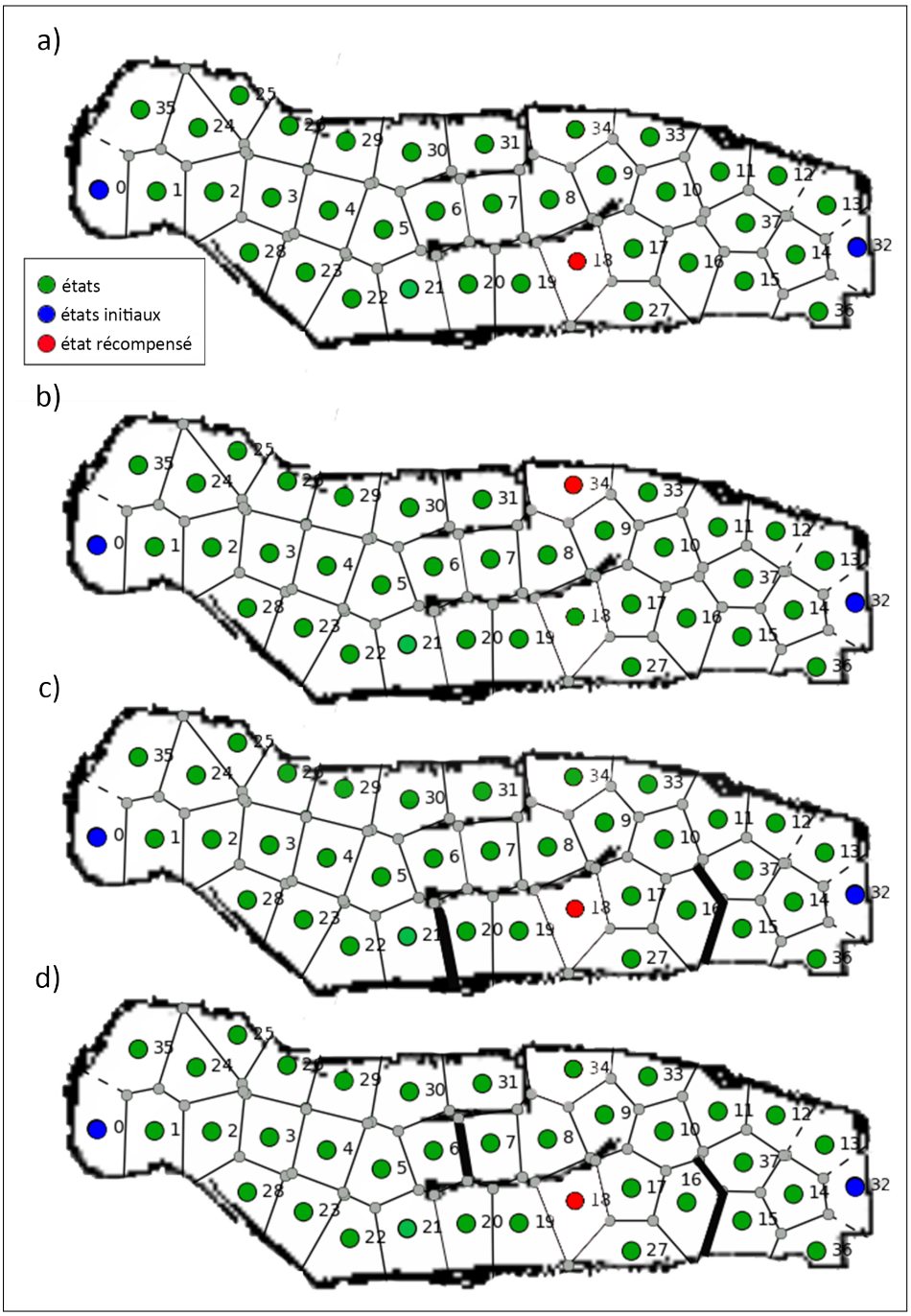

FIGURE 6.3 – ***a***. *Carte de l'arène durant la première période de l'expérience.* ***b*** *Carte de l'arène durant la seconde période de l'expérience, où le changement environnemental consiste en une modification de l'état récompensé.* ***c*** *Carte de l'arène durant la seconde période de l'expérience, où le changement environnemental consiste en un ajout d'obstacles, avec obstruction du couloir du bas.* ***d*** *Carte de l'arène durant la seconde période de l'expérience, où le changement environnemental consiste en un ajout d'obstacles, avec obstruction du couloir central.*

ment simulable. Cependant, durant la phase préliminaire, le robot a lui même réduit l'environnement en un ensemble d'états markoviens, formant une carte du monde discrétisée sur laquelle il se déplace, et qui est utilisée par ses experts d'apprentissage. À nouveau, à chaque instant, les capteurs du robot ne lui servent qu'à savoir dans quel état de la carte il se trouve. L'environnement étant réduit à un ensemble d'états markoviens, il peut donc être décrit par un ensemble de probabilités de transition entre états. Idéalement, si nous laissions le robot explorer l'environnement (non récompensé) durant un temps infini, nous obtiendrions un modèle de transition décrivant à lui seul toute la complexité de l'environnement réel. C'est sur cette base que nous avons développé notre environnement simulé : nous avons laissé le robot explorer l'arène (et donc la carte de la figure 6.2, mais sans état récompensé), et avons arrêté l'exploration lorsque nous nous étions assurés que chaque état avait été parcouru au moins 30 fois par le robot. Cette valeur arbitraire a été définie par l'expérimentateur, en considération de son suivi visuel de l'exploration du robot, et du temps que cela a pris (environ 13 heures, sans inclure l'heure de recharge ayant lieu toutes les 2 heures). Le modèle de transition généré à la suite de cette phase exploratoire, véritable "image des possibilités du robot", a ensuite été récupéré. Il représente l'environnement de simulation.

Concrètement, le premier état dans lequel se trouve le robot virtuel (état initial 0, par exemple) est défini par l'expérimentateur. Ensuite, si le robot effectue par exemple l'action 0, qui consiste à se déplacer vers la droite, l'identité de l'état d'arrivée ne sera pas défini selon l'odométrie et les observations d'un capteur visuel, le robot virtuel n'en possédant pas, mais selon les informations contenues dans le modèle de transition généré précédemment dans la réalité. D'après le modèle de transition, lorsque le robot effectue l'action 0 dans l'état 0, il arrive dans l'état 1 dans $94,4\%$ des cas, et dans l'état 35 situé juste au dessus dans 6.6% des cas. L'état d'arrivée est choisi selon ces probabilités. À noter que le modèle de transition servant d'environnement à la simulation ne remplace pas le modèle de transitions que l'expert MB du robot virtuel construira lui-même au cours de son apprentissage.

Cet environnement simulé nous a permis de tester rapidement de multiples jeux de paramètres pour l'expert MF, mais aussi tout simplement la fonctionnalité de notre architecture et divers critères de coordination, qui ont ensuite profité au robot réel. Comme vous le verrez dans la suite de ce chapitre, nous comparerons systématiquement nos résultats obtenus en simulation à ceux obtenus dans l'environnement réel.

La figure 6.4 résume en un schéma l'ensemble des différentes phases de notre protocole expérimental : phase préliminaire de construction de la carte ; exploration longue de la carte discrétisée afin de générer un modèle de transition décrivant l'environnement ; utilisation de ce modèle de transition afin de développer une version simulée de la tâche permettant d'optimiser plus facilement le système de coordination et ses experts ; réalisation de la tâche d'apprentissage dans l'environnement réel grâce au système de coordination optimisé en simulation.

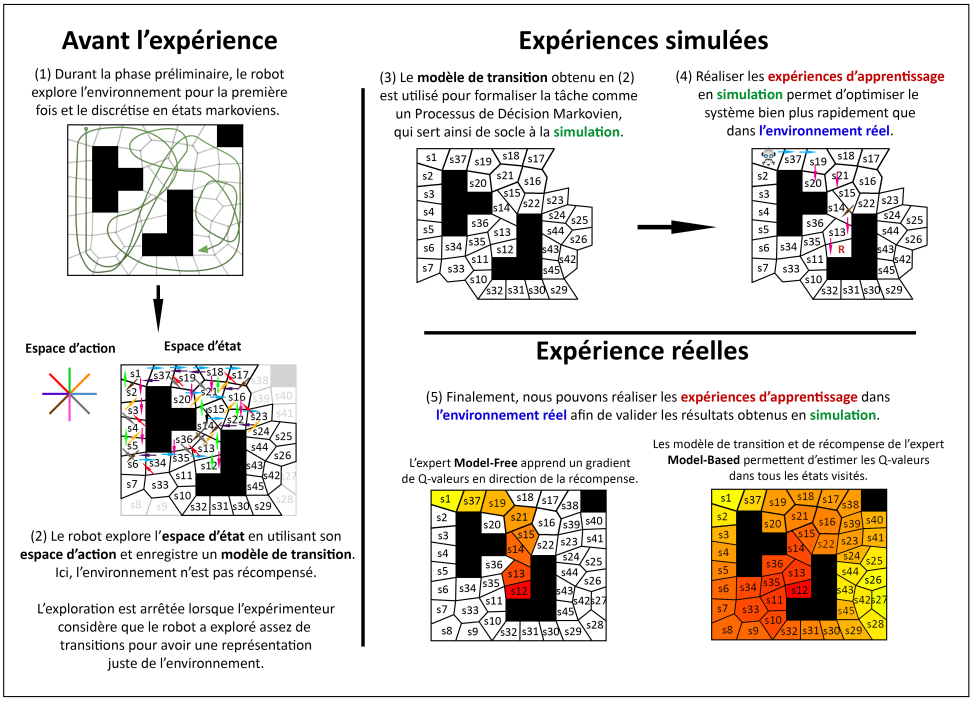

FIGURE 6.4 – *Schéma résumant l'ensemble des différentes phases du protocole expérimental dans un cadre générique. L'exemple prit est celui d'une expérience de navigation, car facile à conceptualiser, mais la méthode est applicable à toutes expériences dont l'espace d'état et l'espace d'action peuvent être discrétisés.*

6.2.2 Paramétrage des experts

Premièrement, pour les deux experts, nous avons initialisé toutes les valeurs d'état-action à 1 afin de faciliter l'exploration des actions non sélectionnées précédemment. Nous nous sommes pour cela inspiré de l'algorithme *Rmax* (Sutton et Barto 1998), proposant d'initialiser le modèle de l'environnement de manière optimiste, c'est-à-dire d'initialiser toutes les valeurs d'état-action au retour maximal (1 dans notre cas). Grâce à cela, le fait de sélectionner une action n'emmenant pas dans l'état récompensé entraîne une diminution de sa valeur d'état-action, et donc une diminution de sa probabilité de sélection la prochaine fois que le robot atteindra cet état, favorisant ainsi la sélection des actions moins sélectionnées précédemment, et donc l'exploration de l'environnement.

Dans les travaux précédents ayant inspiré ce travail (Renaudo 2016), les résultats préliminaires ne permettaient pas de montrer que l'expert MF était capable d'atteindre la performance de l'expert MB. Or, comme mentionné précédemment, la capacité de l'expert MF à adopter un comportement pertinent, et donc la qualité de son apprentissage, est un élément central du critère qui coordonne les experts. Pour maximiser la capacité de l'expert MF à accumuler de la récompense au cours du temps, nous avons donc effectué une recherche par grille. Avec cette analyse, nous cherchions à trouver le meilleur ensemble de paramètres, c'est-à-dire les paramètres maximisant la récompense totale accumulée par un robot utilisant uniquement un expert MF, sur une durée fixe de 1600 pas de temps. Les paramètres de l'expert MB et du MC sont les mêmes que ceux de l'expert

MF, afin de limiter le nombre de paramètres différents. Les valeurs des paramètres sont listées dans le tableau 6.1.

TABLE 6.1 – *Valeurs choisies des paramètres des experts dans l'expérience de navigation.*

Param	MB	MF	MC
α	n.a.	0,6	n.a.
τ	0,02	0,02	0,02
γ	0,9	0,9	n.a.

Concernant la valeur du paramètre κ, le paramètre qui pondère le terme d'entropie dans le calcul des valeurs de compromis $Q(s, E, t)$, nous la fixons selon une analyse de front de Pareto 6.5. Ici, nous recherchions une valeur de κ qui maximisait la capacité du robot à accumuler de la récompense au fil du temps, tout en minimisant le coût de son inférence. Nous avons choisi $\kappa = 7$, c'est-à-dire la valeur de κ pour laquelle les performances ne sont réduites que d'environ 1% par rapport à la performance du robot MB-seul, aussi bien pour les expériences de changement de but que d'ajouts obstacles.

6.2.3 Évaluation d'un expert Model-Free profond

À des fins de comparaison avec notre modèle, nous avons développé spécifiquement pour cette expérience une version alternative de notre expert MF, dont nous avons comparé l'efficacité à celle de la version originelle. Dans la version originelle, présentée dans le chapitre 5, le processus d'apprentissage de l'expert MF est réalisé par un algorithme de *Q-learning*, un algorithme d'apprentissage par renforcement tabulaire. Dans cette seconde version, le processus d'apprentissage est réalisé par un algorithme *DQN* 2.1, un algorithme d'apprentissage par renforcement de référence dans la littérature (Mnih et al. 2015b), utilisant un réseau de neurones profond pour approximer la fonction d'état-action $Q(s, a)$.

Nous avons évalué itérativement plusieurs réseaux avec un nombre et une taille de couches différents, et nous avons sélectionné l'ensemble des paramètres qui offrait les meilleures performances. Le réseau de neurones choisi est composé de deux couches cachées de 76 neurones, prenant en entrée un vecteur de taille 38 (correspondant à l'activité des 38 états de l'arène de navigation, avec 1 si l'état est actif, et 0 sinon), renvoyant un vecteur de taille 8 (correspondant aux 8 valeurs d'état-action l'état courant), utilise un réseau jumeau pour calculer la valeur cible et fait de l'*experience replay*. Ses paramètres sont $\alpha = 0,1$ $\gamma = 0,95$ et $\tau = 0,05$.

6.3 Résultats

6.3.1 Résultats dans l'environnement simulé

Pour évaluer les performances du robot virtuel, nous avons étudié cinq combinaisons d'experts : (1) un robot virtuel MF-seul, en rouge, utilisant uniquement la version originelle de l'expert MF pour décider, (2) un robot virtuel MB-seul, en bleu, utilisant uniquement l'expert MB pour décider, (3) un robot virtuel DQN-seul, en jaune, utilisant uniquement la seconde

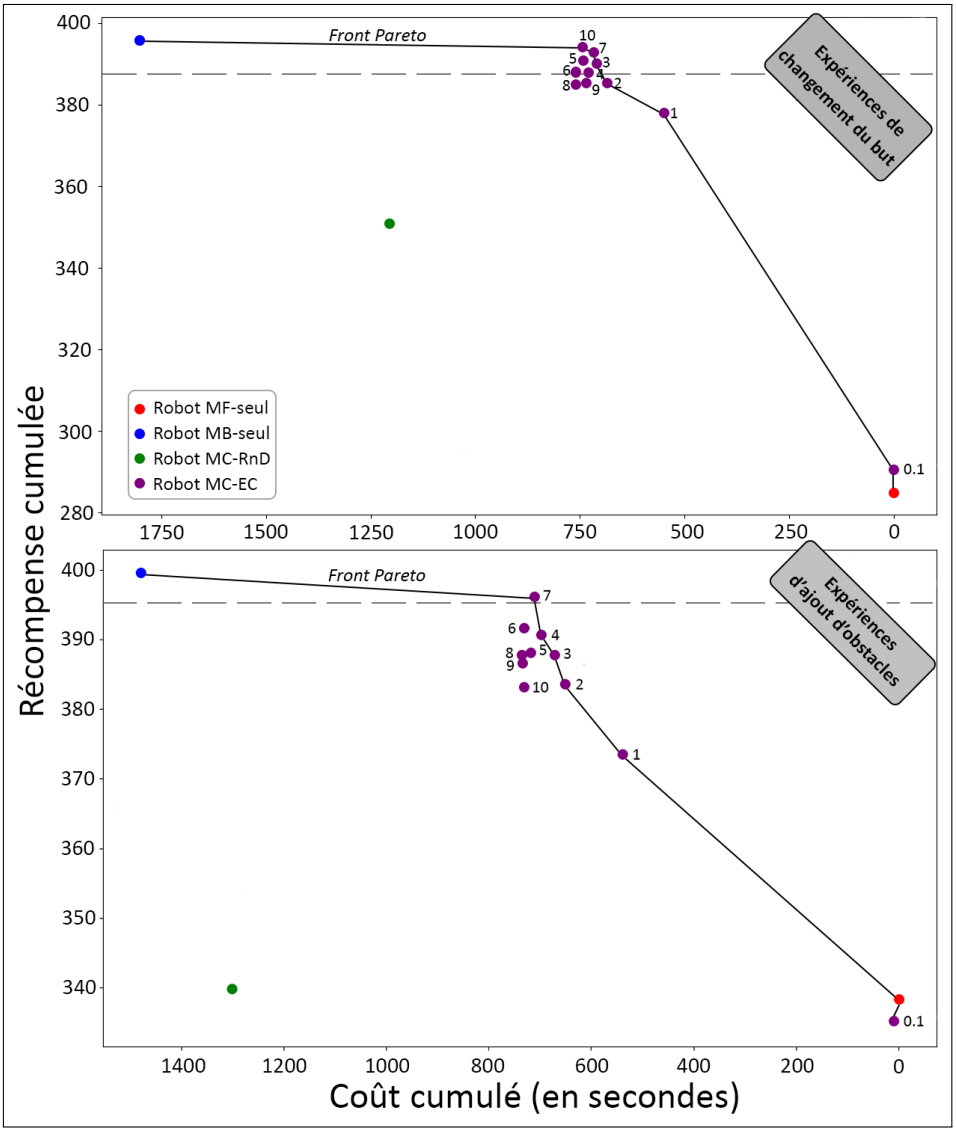

FIGURE 6.5 – *Front de pareto. Chaque point correspond au résultat moyen obtenu pour 100 robots simulés. La couleur rouge correspond aux résultats obtenus par le robot virtuel utilisant uniquement la version originelle de l'expert MF pour décider, la couleur bleu à ceux obtenus par un robot virtuel utilisant uniquement l'expert MB, la couleur verte à ceux obtenus par un robot virtuel coordonnant les deux experts de manière aléatoire et la couleur violette à un robot virtuel coordonnant les deux experts en utilisant le critère de coordination que nous avons défini. Pour les points violets, le nombre associé est la valeur du paramètre κ. La ligne noire reliant les points représente le front de Pareto. La ligne pointillée délimite 99% de la performance de l'expert MB. La figure du haut présente les résultats pour les expériences de changements de but, et la figure du bas ceux pour les expériences d'ajout d'obstacles.*

version de l'expert MF présenté précédemment, (4) un robot virtuel de coordination aléatoire (MC-RnD), en vert, qui coordonne les deux experts originaux de manière aléatoire et (5) un robot virtuel "Entropie et Coût" (MC-EC), en violet, qui coordonne les deux experts originaux en utilisant le critère d'arbitrage que nous avons proposé. Ce code couleur sera systématiquement repris dans chacun des prochains chapitres.

Expériences de changement de but

Nous pouvons observer dans la figure 6.6.A l'évolution de la performance moyenne des différents robots simulés lors des expériences de changement de but. Ici, la performance est définie comme la capacité du robot à accumuler de la récompense sur la durée de l'expérience. Comme attendu, des quatre robots principaux, le robot MF-seul est celui affichant la moins bonne performance. Le robot MC-RnD, quant à lui, affiche une performance intermédiaire, alors que les robots MB-seul et MC-EC affichent une performance équivalente et supérieure à celles des deux autres robots. Nous pouvons aussi voir que les robots MB-seul et MC-EC sont ceux des quatre les moins perturbés par le changement de l'état récompensé, et que le robot MF-seul, à l'inverse, met du temps à réadopter un comportement performant. À ce stade là, nous pouvons donc dire que notre critère d'arbitrage permet au robot de maintenir une performance maximale, malgré la volatilité de l'environnement, si nous considérons que le robot MB-seul, de par sa conception, est le robot qu'on attendait être le plus performant des quatre. Nous pouvons aussi voir qu'arbitrer entre les deux experts en utilisant le critère que nous avons défini est plus efficace qu'arbitrer de manière aléatoire.

Dans la figure 6.6.B, nous pouvons observer l'évolution du coût cumulé moyen des différents robots simulés lors des expériences de changement de but. Ici, le coût de calcul est défini comme le temps cumulé pris pour exécuter le processus d'inférence du robot sur la durée de l'expérience, en secondes. Comme attendu, le robot MF-seul est le moins coûteux des quatre robots principaux, son processus d'inférence étant réduit simplement à évaluer depuis le tableau qui contient toutes les valeurs d'état-action celles associées à l'état dans lequel il se trouve actuellement, et le robot MB-seul est celui le plus coûteux, son processus d'inférence consistant à faire de la planification, et n'étant jamais inhibé par le MC. Les robots MC-RnD et MC-EC affichent logiquement un coût intermédiaire, puisqu'ils inhibent de temps à autre les processus d'inférence de leurs experts, et le robot MC-EC est celui des deux étant le moins coûteux. Il est à noter que les computations effectuées par le méta-contrôleur ont dans tous les cas un coût très faible, similaire à celui de l'expert en MF, de $10^{-}5$ secondes par itération en moyenne. Dans ce système, seul l'expert MB est onéreux, avec un coût moyen de $10^{-}2$ secondes par itération. Le coût de l'utilisation d'un méta-contrôleur est donc négligeable par rapport à ce qu'il apporte en termes d'économies globales.

Enfin, la figure 6.6.A nous montre aussi que le robot DQN-seul apprend et s'adapte moins bien que tous les autres robots. Comme l'algorithme *DQN* est un algorithme model-free, il n'est pas surprenant que les robots utilisant l'expert MB soient plus efficaces et plus adaptatifs. Le

fait que l'algorithme *DQN* soit également moins performant que l'algorithme de *Q-learning* tabulaire du robot MF-seul s'explique par le nombre de valeurs mémorisées (c'est-à-dire les poids du réseau) qu'il doit adapter avant de pouvoir fournir des sorties correctes. En effet, l'apprentissage des réseaux neuronaux profonds nécessite souvent plusieurs centaines de milliers d'itérations. Un tel nombre d'itérations est beaucoup trop important lorsque l'on vise à développer des applications robotiques réelles, où un apprentissage à la volée est nécessaire. Si des mécanismes de *replay* 2.1 peuvent être utilisés pour accélérer l'apprentissage du *DQN*, ces calculs supplémentaires augmentent néanmoins le coût de calcul du système résultant (Fig. 6.6.B).

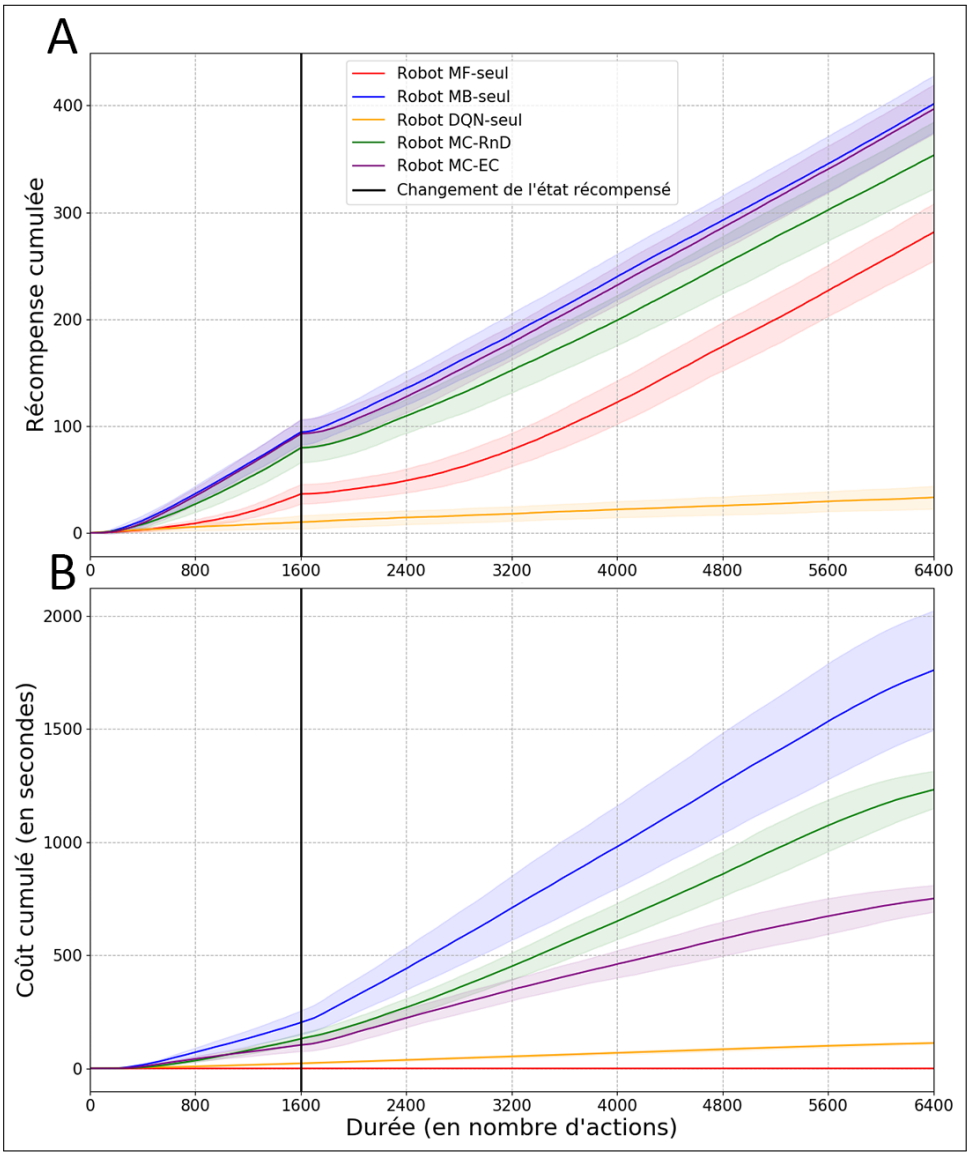

FIGURE 6.6 – *A. Performance moyenne pour 100 expériences simulées. La performance est définie comme la capacité du robot à accumuler de la récompense sur la durée de l'expérience. La durée est représentée par le nombre d'actions effectuées par le robot. Nous utilisons l'écart-type comme indicateur de dispersion. À la 1600ème action, la récompense passe de l'état 18 à l'état 34. **B**. Coût moyen de calcul pour 100 expériences simulées. Le coût de calcul est défini comme le temps cumulé pris pour exécuter le processus d'inférence du robot sur la durée de l'expérience, en secondes.*

Nous pouvons observer dans la figure 6.7.A l'évolution de la dynamique de sélection des experts par le robot MC-EC, exprimée en termes de probabilité moyenne de sélection. Trois phases différentes semblent se distinguer, et se répéter au cours du temps :

— **La phase exploratoire de l'expert MF (1 sur la figure 6.7.A).** Avant la découverte de l'état récompensé, le robot MC-EC utilise principalement l'expert MF. Ceci est dû à la différence de méthode de mise-à-jour des valeurs d'état-action entre les deux experts. Avec les mêmes valeurs initiales et l'ensemble des paramètres que nous avons définis, les valeurs d'état-action de l'expert MF diminuent en effet légèrement plus que celles de l'expert MB, ce qui entraîne une diminution plus prononcée de l'entropie de la distribution des probabilités de sélection d'actions, et donc une augmentation de la probabilité de sélection de la décision de l'expert MF . En outre, comme nous ne disposons pas d'un expert spécialisé dans l'exploration, il est juste de faire appel à l'expert le moins cher jusqu'à ce que la position de la récompense soit découverte. Concernant l'exploration, d'autres études proposent justement d'ajouter un troisième expert spécialisé dans l'exploration de l'environnement (Caluwaerts et al. 2012b).

— **La phase de guidage de l'expert MB (2 sur la figure 6.7.A).** Après avoir trouvé la récompense pour la première fois, l'expert MB prend progressivement la main sur le comportement du robot, son processus d'inférence ayant besoin de faire face à la récompense une seule fois pour propager correctement les valeurs d'état-action dans son modèle de transition. Ainsi, les valeurs d'entropie de la distribution des probabilités de sélection de ses actions vont diminuer plus rapidement que celles de l'expert MF, et sa décision sera de fait plus souvent sélectionnée par le méta-contrôleur.

— **La phase de guidage de l'expert MF (3 sur la figure 6.7.A).** L'expert MF ne possédant pas le processus d'inférence de l'expert MB, il doit propager les valeurs d'état-action progressivement d'un état à l'autre, ce qui rend son apprentissage beaucoup plus lent. À noter que le processus d'apprentissage des experts n'étant jamais inhiber, il apprend malgré tout par démonstration de l'expert MB, ce qui accélère son apprentissage. Finalement, après un certain nombre d'itérations (environ 800) l'expert MF atteint la performance de l'expert MB. À ce moment là, puisque l'expert MF est systématiquement moins coûteux que l'expert MB en termes de ressources calculatoires, à performance équivalente, le méta-contrôleur du robot MC-EC sélectionnera automatiquement plus souvent la décision de l'expert MF.

La phase exploratoire de l'expert MF recommence à la 1600ème itération, lorsque la récompense passe de l'état 18 à l'état 34. Ensuite, les phases de guidage de l'expert MB puis celle de l'expert MF se répètent automatiquement.

Après le changement de l'état récompensé, lorsque le robot atteint l'état 18 sans obtenir de récompense, le processus d'inférence de l'expert MB par *Value Iteration* (2) est tel qu'il va rapidement propagé la valeur nulle de récompense à l'ensemble du modèle, et donc ainsi réinitialiser

toutes les valeurs d'état-action du robot, ce qui donnera lieu à une augmentation de ses valeurs d'entropie de la distribution des probabilités de sélection d'actions. À l'inverse, le robot devra passer plusieurs fois sur les états menant à l'état 18 pour diminuer les valeurs d'état-action de l'expert MF, et donc augmenter ses valeurs d'entropie de distribution des probabilités de sélection d'actions. Cette différence explique le "pic" de la probabilité de sélection de l'expert MF qui suit le changement de l'état récompensé, ainsi que le très léger retard dans l'accumulation de la récompense que le robot MC-EC prend sur le robot MB-seul, ayant lieu au même moment sur la figure 6.6.A (bien que cette différence de performance ne soit pas significative à la dernière itération : test statistique de Mann-Whitney, $p = 0.116 > 0.0125$). Concrètement, lorsque la récompense change d'état, l'expert MF va donc être durant un moment favorisé par le système d'arbitrage, du fait de son incapacité à intégrer directement le changement et à réinitialiser ses valeurs d'état-action. Si cette particularité ne pose pas de problème dans cette tâche de navigation, nous y serons à nouveau confrontés au cours du chapitre 8, où nous proposerons une solution. Dès lors que le robot découvre la nouvelle récompense dans l'état 34 (barre orange sur la figure 6.7.A), l'expert MB propage les valeurs d'état-action, fait baisser ses valeurs d'entropie de la distribution des probabilités de sélection d'actions, et prend donc la main sur la décision, pour un temps.

Enfin, la taille de l'écart-type s'explique par le fait que pour chaque expérience prise de manière individuelle, la stratégie et le comportement du robot peuvent varier énormément, notamment en raison du nombre d'états et d'actions possibles, de la multiplicité des chemins que le robot peut emprunter, mais aussi de la nature probabiliste de l'environnement. En conséquence, la durée de passage d'une des trois phases à la suivante varie beaucoup d'une expérience à l'autre. Néanmoins, le profil de sélection des experts est cohérent avec le comportement moyen présenté dans la figure 6.7.B : nous retrouvons bien les trois phases.

Nous pouvons observer dans la figure 6.8 des cartes de l'environnement, à différentes périodes de l'expérience, indiquant l'identité des experts ayant pris la dernière décision dans chacun des états (couleur bleu pour l'expert MB, couleur rouge pour l'expert MF). Ces cartes correspondent aux données de l'expérience 9, que nous avons choisie pour sa représentativité. Celles-ci nous permettent d'observer sous un autre angle la formation d'un schéma temporel de coordination des experts : durant la phase de guidage de l'expert MB, les cartes sont majoritairement colorées en bleues, tandis que durant la phase de guidage de l'expert MF, les cartes sont majoritairement colorées en rouge. Plus intéressant encore, nous pouvons observer la formation d'un profil de coordination spatial, se dessinant lui aussi au cours du temps : lors de la phase de guidage de l'expert MF, des "chemins d'états rouge" (bien que parfois entrecoupés d'états bleus) se dessinent, reliant les états initiaux aux états récompensés. Lorsque la récompense change d'état, les chemins se déconstruisent progressivement, avant de finalement se reformer correctement en direction du nouvel état récompensé. Concrètement, nous pouvons donc identifier deux types d'états :

— Des états situés sur le chemin optimal menant à la récompense, dans lequel l'expert MF est bien entraîné, et dans lequel le robot

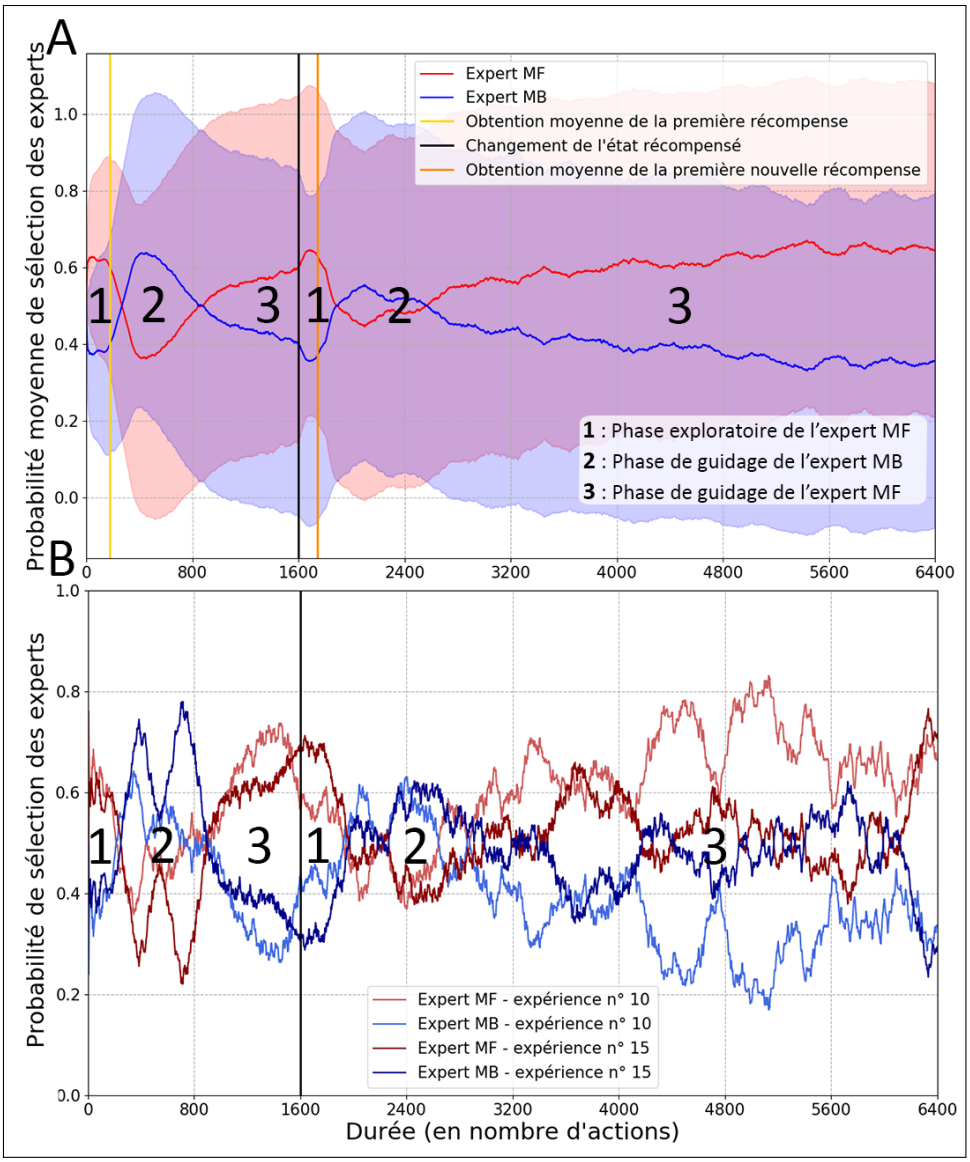

FIGURE 6.7 – *A. Probabilité moyenne de sélection des experts par le méta-contrôleur du robot MC-EC pour 100 expériences simulées. Nous utilisons l'écart-type comme indicateur de dispersion. À la 1600ème action, la récompense passe de l'état 18 à l'état 34.* ***B****. Probabilité de sélection des experts par le méta-contrôleur du robot MC-EC pour deux expériences simulées choisies.*

passe régulièrement. Par "chemin optimal" nous entendons "plus court chemin reliant les états initiaux à l'état récompensé".

— Des états situés en bordure du chemin optimal, dans lequel l'expert MF est peu entraîné, où l'expert MB a donc la main sur la décision, et dans lequel le robot passe moins régulièrement. Si jamais le robot s'aventure hors du chemin, son expert MB le renverra dessus.

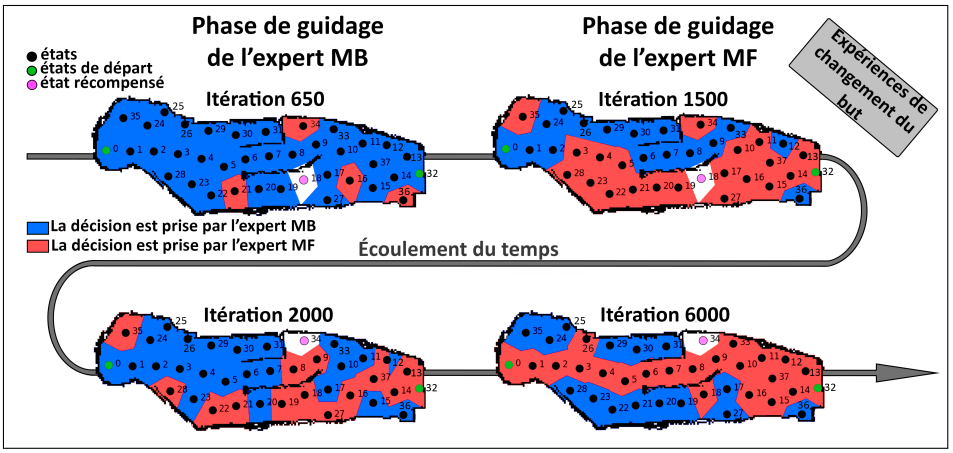

FIGURE 6.8 – *Carte de sélection des experts par le méta-contrôleur du robot MC-EC pour l'expérience simulée numéro 9. À la 1600ème action, la récompense passe de l'état 18 à l'état 34. Lorsqu'un état est coloré en rouge, cela signifie que la dernière décision enregistrée a été prise par l'expert MF, et lorsqu'un état est coloré en bleu, cela signifie que la dernière décision enregistrée a été prise par l'expert MB. Les phases de guidage de l'expert MF et de l'expert MB correspondent aux phases identifiées dans la figure 6.7.*

Pour résumer, le robot MC-EC, grâce à son méta-contrôleur et son critère d'arbitrage, est capable de maintenir une performance maximale tout en exhibant un coût calculatoire plus que deux fois plus petit que celui du robot MB-seul, et ce malgré le changement de but en cours d'expérience. Cette haute performance, associée à son bas coût calculatoire, s'explique du fait de la formation d'un schéma temporel de coordination alternant entre trois phases distinctes tirant bénéfices des avantages de chacun des experts : (1) une phase durant laquelle l'expert le moins coûteux (MF) guide l'exploration, (2) une phase durant laquelle l'expert le plus performant mais aussi le plus coûteux (MB) guide le comportement du robot tandis que l'expert le moins performant (MF) apprend par démonstration et (3) une phase durant laquelle l'expert le moins coûteux (MF) réussi a atteindre la performance de l'expert le plus performant (MB), et prend finalement la main sur le guidage du comportement. Ces différentes phases sont associées à un profil de coordination spatial, illustré par la formation d'un chemin d'états reliant les états initiaux à l'état récompensé, sur lequel l'expert MF est celui qui contrôle le comportement du robot. Ces chemins se réforment dynamiquement lorsque la récompense change d'état.

Expériences d'ajouts d'obstacles

Dans cette sous-partie, nous présentons les résultats des expériences avec ajouts d'obstacles. De manière générale, ils montrent des tendances très semblables à ceux observés dans les expériences avec changement de but, ce qui montre la capacité de notre système à s'adapter à différents

type de changements environnementaux, le changement de but ayant un impact sur le modèle de récompense du robot, tandis que l'ajout d'obstacles impacte son modèle de transition.

Dans la figure 6.9.A, nous observons à nouveau la faible performance du robot MF-seul comparée à celle des robot MB-seul et MC-EC. La performance du robot MC-RnD, toujours intermédiaire, chute dorénavant progressivement au cours du temps. Cela semble être aussi le cas pour la performance de l'expert MC-EC, même si sa différence avec celle de l'expert MB-seul reste non significative en fin d'expérience (test statistique Mann-Whitney, $p = 0.171 > 0.0125$). Cette fois-ci, nous n'observons plus le très léger retard dans l'accumulation de la récompense que le robot MC-EC prenait sur le robot MB-seul juste après le changement du but. En effet, comme expliqué précédemment, cette particularité s'explique par l'incapacité de l'expert MF à réinitialiser ses valeurs d'états-action lorsque le robot découvre que la récompense a changé d'état, à l'inverse de l'expert MB. Ici, l'état récompensé restant le même, le robot n'est pas sujet à ce problème. À nouveau, l'évolution du coût calculatoire cumulé moyen des différents robots simulés adopte une dynamique identique à celle observée précédemment 6.9.B : le robot MF-seul ne coûte presque rien, tandis que le robot MB-seul est celui le plus coûteux de tous. Les robots MC-RnD et MC-EC affichent des coûts intermédiaires, et le robot MC-EC a un coût toujours moins de deux fois plus petit que celui du robot MB-seul.

Comme pour les expériences de changement de but, nous observons toujours dans la figure 6.10.A le schéma temporel de coordination des experts : phase exploratoire de l'expert MF, puis phase de guidage de l'expert MB, puis phase de guidage de l'expert MF. De la même manière que nous n'observons pas le retard d'accumulation de la récompense du robot MC-EC après l'ajout des obstacles dans la figure 6.10.A, nous n'observons ici pas le "pic" de la probabilité de sélection de l'expert MF qui suit le changement environnemental. Ces trois phases réapparaissent couramment lorsque nous observons les expériences individuellement 6.10.B.

Pour finir, nous observons à nouveau la formation du profil de coordination spatial, et l'apparition / disparition de "chemins d'états rouge" reliant les états initiaux à l'état récompensé 6.11. À noter que dans cet exemple représentatif, un obstacle a été rajouté dans le couloir du bas à partir de la 1600ème action. De ce fait, les états 19 et 20 ne seront plus jamais visités par le robot, étant coincés entre l'obstacle et l'état récompensé. Comme pour les expériences de changement de but, les "chemins d'états rouge" sont rarement complets, et souvent entrecoupés d'états bleus.

6.3.2 Résultats dans l'environnement réel

Comme expliqué précédemment, le fait de développer un environnement simulé nous a permis d'évaluer et d'optimiser notre système bien plus rapidement et efficacement. Suite à cela, nous voulions savoir si les résultats obtenus durant les expériences réelles seraient concluants, afin de répliquer les résultats de simulation et de confirmer l'utilité de notre environnement simulé. Nous avons donc évalué aussi notre système de coordination avec un robot physique dans un environnement réel, avec changements du but et ajouts d'obstacles à la 1600ème action.

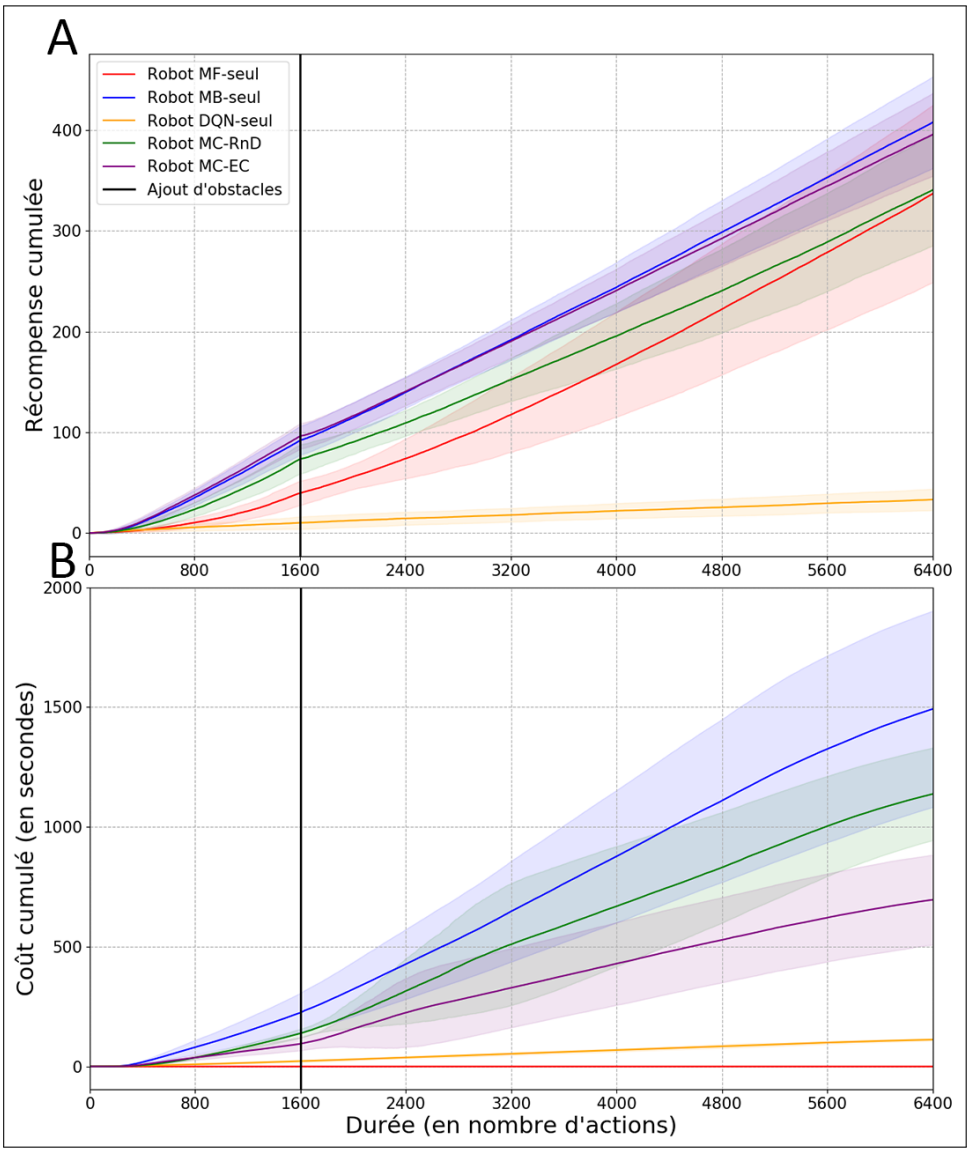

FIGURE 6.9 – *A. Performance moyenne pour 100 expériences simulées. La performance est définie comme la capacité du robot à accumuler de la récompense sur la durée de l'expérience. La durée est représentée par le nombre d'actions effectuées par le robot. Nous utilisons l'écart-type comme indicateur de dispersion. À la 1600ème action, des obstacles sont rajoutés dans l'environnement.* ***B****. Coût moyen de calcul pour 100 expériences simulées. Le coût de calcul est défini comme le temps cumulé pris pour exécuter le processus d'inférence du robot sur la durée de l'expérience, en secondes.*

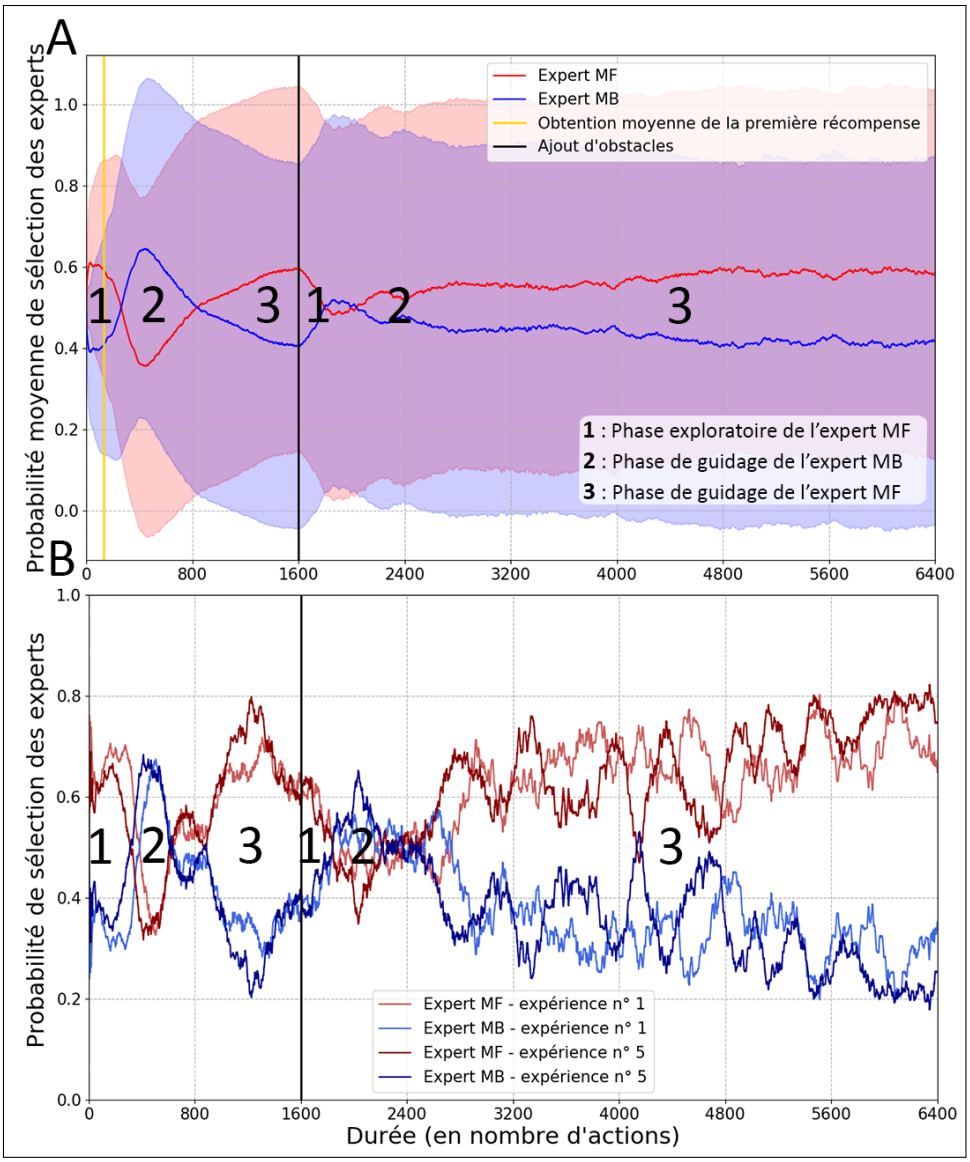

FIGURE 6.10 – *A. Probabilité moyenne de sélection des experts par le méta-contrôleur du robot MC-EC pour 100 expériences simulées. Nous utilisons l'écart-type comme indicateur de dispersion. À la 1600ème action, des obstacles sont rajoutés dans l'environnement.* ***B****. Probabilité de sélection des experts par le méta-contrôleur du robot MC-EC pour deux expériences simulées choisies.*

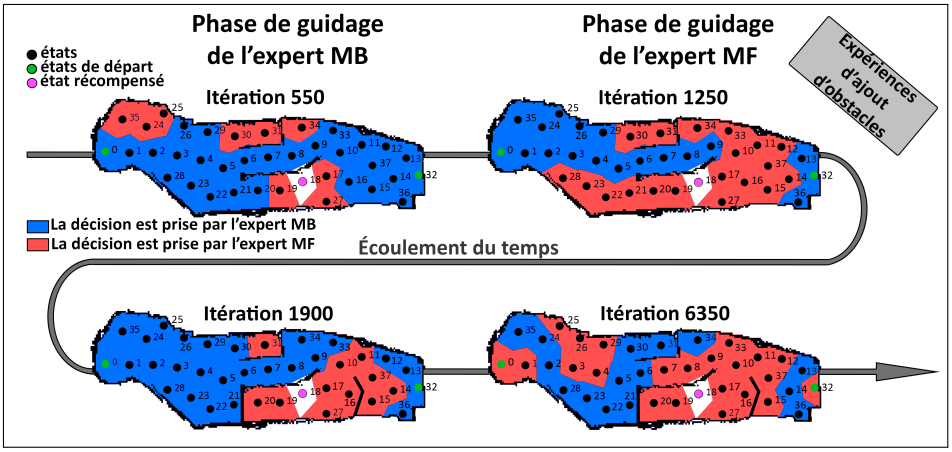

FIGURE 6.11 – *Carte de sélection des experts par le méta-contrôleur du robot MC-EC pour l'expérience simulée numéro 17. À la 1600ème action, des obstacles sont rajoutés dans l'environnement. Lorsqu'un état est coloré en rouge, cela signifie que la dernière décision enregistrée a été prise par l'expert MF, et lorsqu'un état est coloré en bleu, cela signifie que la dernière décision enregistrée a été prise par l'expert MB. Les phases de guidages de l'expert MF et de l'expert MB correspondent aux phases identifiées dans la figure 6.10.*

Nous pouvons observer sur la figure 6.12.A et B la comparaison de la performance et du coût calculatoire du robot MC-EC simulé et réel, pour les deux types de changements environnementaux. L'écart avec la réalité est visible, avec une baisse de performance d'environ 20% pour les expériences avec changement du but et d'environ 30% pour les expériences avec ajout d'obstacle. Nous pouvons aussi observer une augmentation du coût calculatoire du robot réel par rapport à la simulation. Toutefois, notre système de coordination permet au robot MC-EC réel d'apprendre malgré tout à accumuler de la récompense au cours du temps d'une manière comparable, et l'économie calculatoire reste avantageuse.

À nouveau, nous pouvons observer la formation du même schéma temporel de coordination des experts pour les expériences réelles, et cela pour les deux types de changements environnementaux (figure 6.13. A et B), et la succession et la répétition des trois différentes phases.

Pour finir, nous observons aussi toujours le même profil de coordination spatial dans les figures 6.14 et 6.15. Cependant, les "chemins d'états rouges" restent plus incomplets qu'en simulation, signe que l'environnement réel est, comme attendu, plus complexe que l'environnement simulé, ce qui impacte la capacité du robot à réaliser correctement la tâche.

6.4 Conclusion

Dans ce chapitre, nous avons évalué le comportement d'une architecture robotique à trois couches intégrant des mécanismes de coordination d'experts d'apprentissage. La particularité du système de coordination réside dans sa mesure explicite en ligne de la performance et du coût calculatoire de chaque expert, de manière à donner le contrôle à l'expert exhibant à chaque instant le meilleur compromis entre les deux. Nous avons évalué le robot dans des tâches de navigation réelles et simulées, où l'environnement est complexe et non stationnaire (environnement probabiliste,

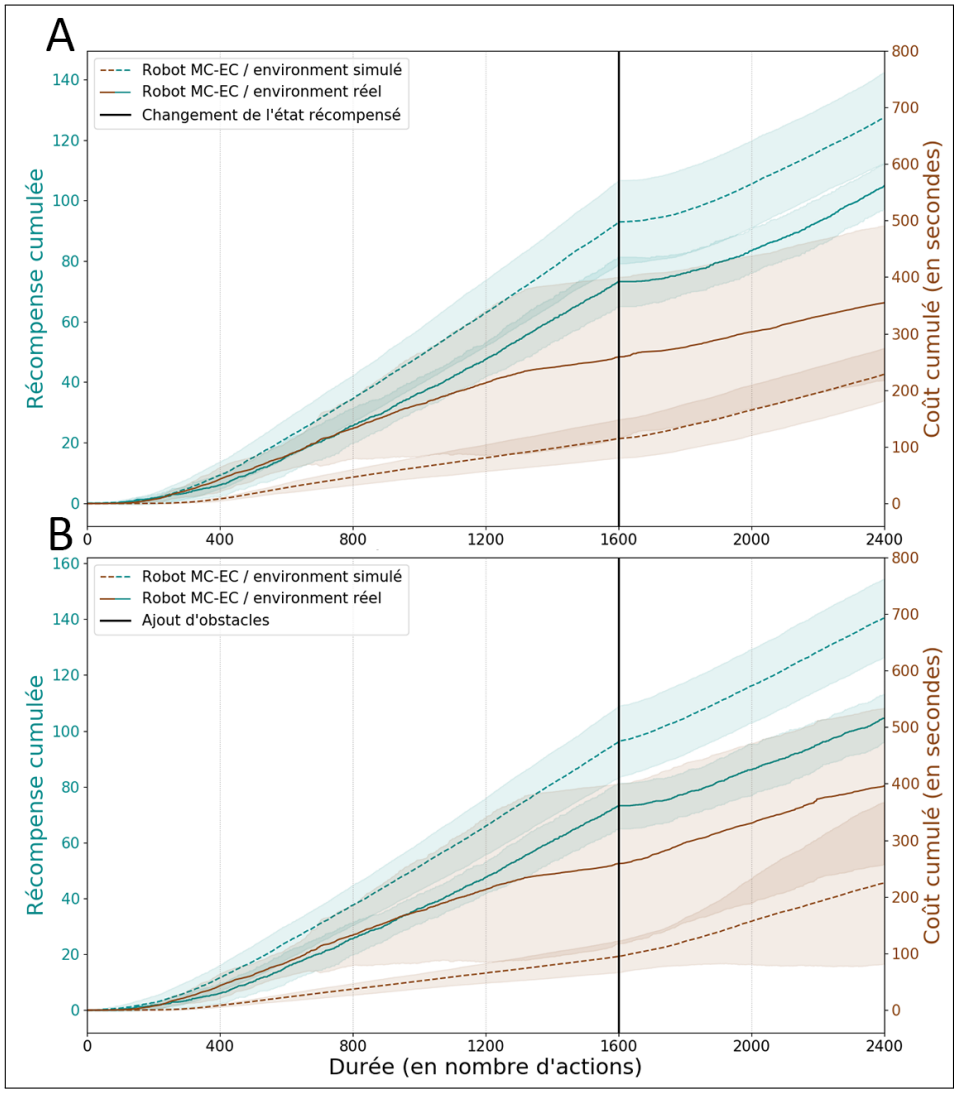

FIGURE 6.12 – *A. Performance (en cyan) et coût de calcul (en marron) moyen du robot MC-EC pour 10 expériences réelles. La performance est définie comme la capacité du robot à accumuler de la récompense sur la durée de l'expérience. Le coût de calcul est défini comme le temps cumulé pris pour exécuter le processus d'inférence du robot sur la durée de l'expérience, en secondes. À la 1600ème action, la récompense passe de l'état 18 à l'état 34.* ***B****. Performance (en cyan) et coût de calcul (en marron) moyen du robot MC-EC pour 10 expériences réelles. À la 1600ème action, des obstacles sont rajoutés dans l'environnement.*

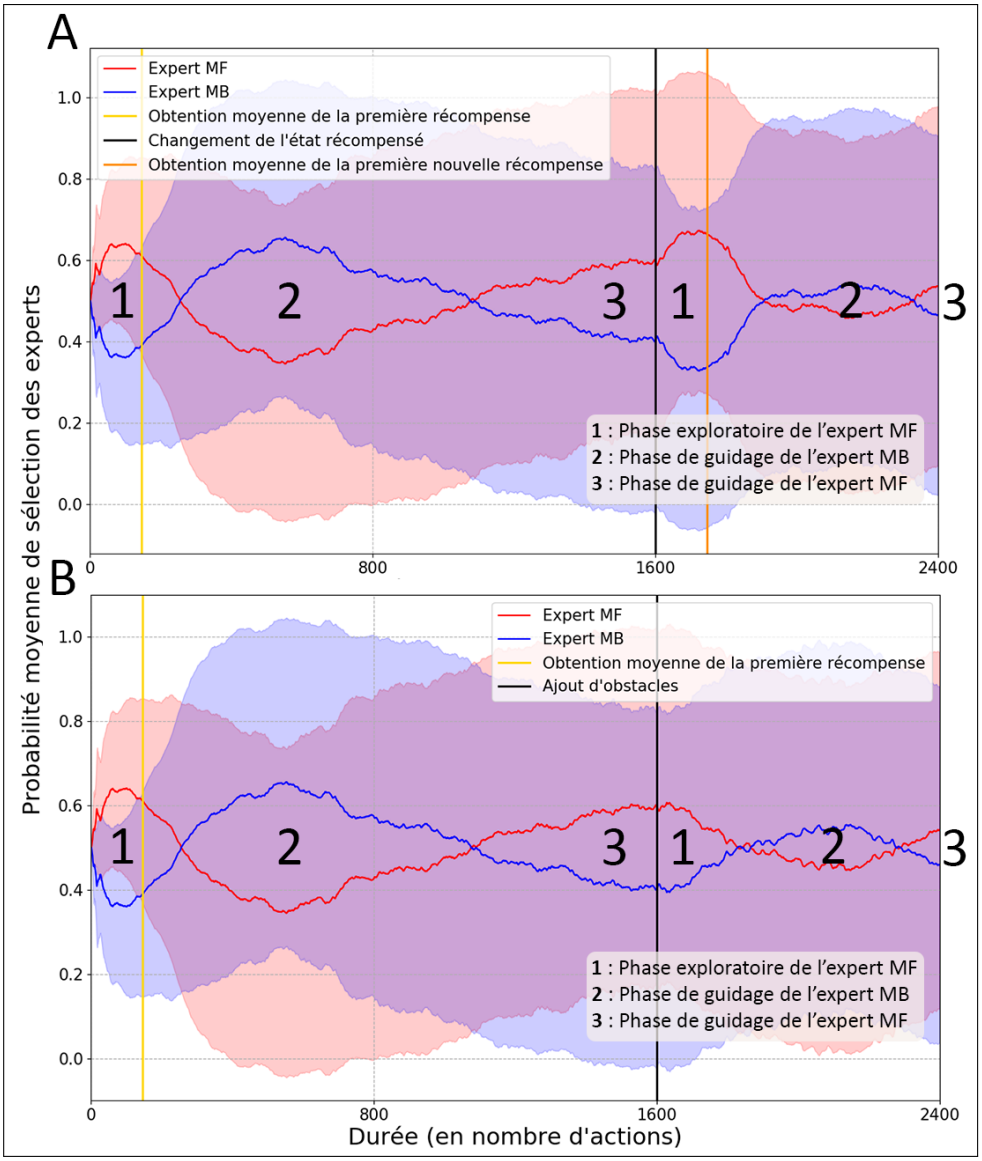

FIGURE 6.13 – *A. Probabilité moyenne de sélection des experts par le méta-contrôleur du robot MC-EC pour 10 expériences réelles. Nous utilisons l'écart-type comme indicateur de dispersion. À la 1600ème action, la récompense passe de l'état 18 à l'état 34.* ***B****. Probabilité moyenne de sélection des experts par le méta-contrôleur du robot MC-EC pour 10 expériences réelles. À la 1600ème action, des obstacles sont rajoutés dans l'environnement.*

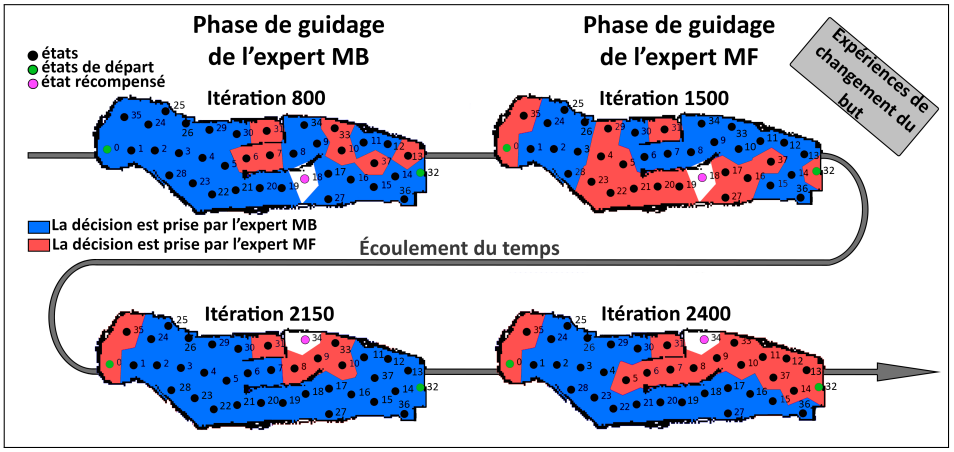

FIGURE 6.14 – *Carte de sélection des experts par le méta-contrôleur du robot MC-EC pour l'expérience réelle numéro 6. À la 1600ème action, la récompense passe de l'état 18 à l'état 34. Lorsqu'un état est coloré en rouge, cela signifie que la dernière décision enregistrée a été prise par l'expert MF, et lorsqu'un état est coloré en bleu, cela signifie que la dernière décision enregistrée a été prise par l'expert MB. Les phases de guidages de l'expert MF et de l'expert MB correspondent aux phases identifiées dans la figure 6.13.*

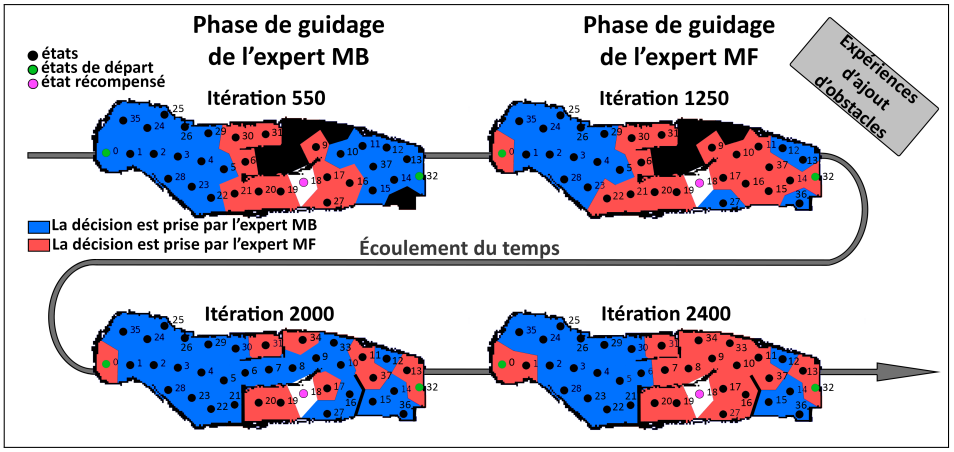

FIGURE 6.15 – *Carte de sélection des experts par le méta-contrôleur du robot MC-EC pour l'expérience réelle numéro 6. À la 1600ème action, des obstacles sont rajoutés dans l'environnement. Lorsqu'un état est coloré en rouge, cela signifie que la dernière décision enregistrée a été prise par l'expert MF, et lorsqu'un état est coloré en bleu, cela signifie que la dernière décision enregistrée a été prise par l'expert MB. Les états noirs sont des états non encore explorés par le robot. Les phases de guidage de l'expert MF et de l'expert MB correspondent aux phases identifiées dans la figure 6.13.*

ajouts de murs et changement de l'état récompensé en cours de tâche). Le critère d'arbitrage de notre système de coordination permet au robot de déterminer de manière autonome quand changer d'expert décisionnel durant l'apprentissage, générant ainsi un schéma temporel cohérent de prise de décision qui alterne entre plusieurs stratégies au cours de l'expérience. Ce schéma temporel de type "planifier jusqu'à l'habitude" correspond aux observations faites par Keramati et al. (2016) chez des sujets humains réalisant des tâches de bandit manchot. Grâce à cette capacité de coordination, le robot acquière une plus grande flexibilité face aux changements environnementaux qu'un robot contrôlé uniquement par un algorithme d'apprentissage model-free, et atteint le même niveau de performance que celui d'un robot contrôlé uniquement par un algorithme d'apprentissage model-based, tout en divisant son temps de calcul par plus de deux. La comparaison avec le robot contrôlé uniquement par un algorithme DQN a aussi montré que l'apprentissage par renforcement profond a un coût de calcul peu compatible avec les contraintes robotiques réelles, et ainsi que la construction et l'utilisation d'une représentation de données adaptée à la tâche à accomplir réduit la charge de travail du système d'apprentissage, permettant un apprentissage à la volée à faible coût. Dans le chapitre 7, nous évaluons notre modèle de coordination dans une tâche d'interaction humain-robot où le robot doit désormais s'adapter à la variabilité du comportement humain.

7 Évaluation du système d'arbitrage dans une tâche d'interaction humain-robot

Sommaire

Dans ce chapitre, nous évaluons notre architecture robotique et notre système de coordination dans une nouvelle tâche d'interaction humain-robot. Dans un premier temps, nous présentons une tâche de rangement de cubes dans des boîtes en simulation, les deux types d'humains simulés que nous avons définis pour interagir avec le robot, ainsi que les changements minimaux que nous avons dû opérer sur l'architecture robotique. Dans la seconde partie, nous présentons les résultats obtenus et montrons en quoi notre système de coordination permet au robot, dans une tâche plus fournie en états, et sans changement majeur de notre architecture, de maintenir à nouveau un haut niveau de performance tout en diminuant grandement son coût calculatoire, mais aussi de faire face à la volatilité du comportement humain. Ce chapitre correspond à une version étendue de la la publication Dromnelle et al. (2020a).

7.1 Matériel et protocole expérimental

7.1.1 Environnement et robot simulés

Contrairement aux travaux précédents, cette expérience a été réalisée uniquement en simulation. Ici, un robot possédant au moins un bras mobile, un capteur visuel et un capteur sonore fait face à une table. Sur la table, trois boîtes et trois cubes de couleurs différentes sont posés. Le robot est capable de distinguer les couleurs des cubes et des boîtes, et de manipuler chacun des cubes. De l'autre côté de la table, un humain peut interagir oralement avec le robot, mais aussi prendre le contrôle de son bras. Le robot est capable d'entendre et d'interpréter les messages de l'humain. La figure 7.1 illustre l'expérience.

Comme pour la tâche de navigation, nous n'utilisons pas de logiciel de simulation robotique, mais représentons l'environnement par un modèle de transitions. Ici, le modèle de transitions représentant l'environnement simulé n'est pas généré par un robot dans le monde réel, puisqu'il n'existe aucune expérience réelle, mais défini à la main par l'expérimentateur. N'étant pas issu d'une expérience réel, ce modèle n'est pas probabiliste : chaque action réalisée dans chaque état par le robot emmène dans un unique état terminal. Il serait sans nul doute bien plus complexe s'il avait été généré par un robot réalisant cette tâche dans le monde réel, comme pour l'expérience de navigation du chapitre 6. Initialement, nous avions prévu de réaliser la tâche avec des vrais sujets humains et un robot *Baxter*, mais la situation sanitaire en 2020 nous a fait abandonner ce projet et rester sur des simulations. Pourtant, ce modèle de la tâche est dans un sens déjà plus complexe que l'environnement de navigation, comme nous allons le voir dans les deux sous-parties suivantes.

Dans cette tâche de rangement, l'objectif du robot est d'apprendre à mettre chacun des cubes, initialement posés sur la table, dans la boîte de la couleur correspondante. Lorsque cela est fait, le robot obtient une unité de récompense virtuelle, et les cubes sont automatiquement remis sur la table.

7.1.2 Espace d'état et d'action du robot

Comme pour l'expérience de navigation, l'espace d'état du robot est un espace d'état discret. Ici, un état ne représente pas une coordonnée dans l'espace, mais la position des trois cubes de couleur. Chacun des cubes peut se situer : dans la boîte rouge, dans la boîte verte, dans la boîte bleu, sur la table, dans la main du robot, dans la main de l'humain. Si l'on supprime les états où le robot et l'humain tiennent plusieurs cubes à la fois. Cela représente donc un total de 112 états, c'est-à-dire trois fois plus d'états que dans l'expérience de navigation.

Concernant l'espace d'action, le robot peut réaliser 7 actions différentes : prendre le cube rouge, prendre le cube vert, prendre le cube bleu, poser le cube tenu en main dans la boîte rouge, dans la boîte verte, dans la boîte bleu et sur la table.

Alors que d'autres façons de modéliser la tâche, comme avec l'apprentissage par renforcement relationnel (Džeroski et al. 2001), auraient été possibles, nous avons choisi cette décomposition de l'espace d'état pour

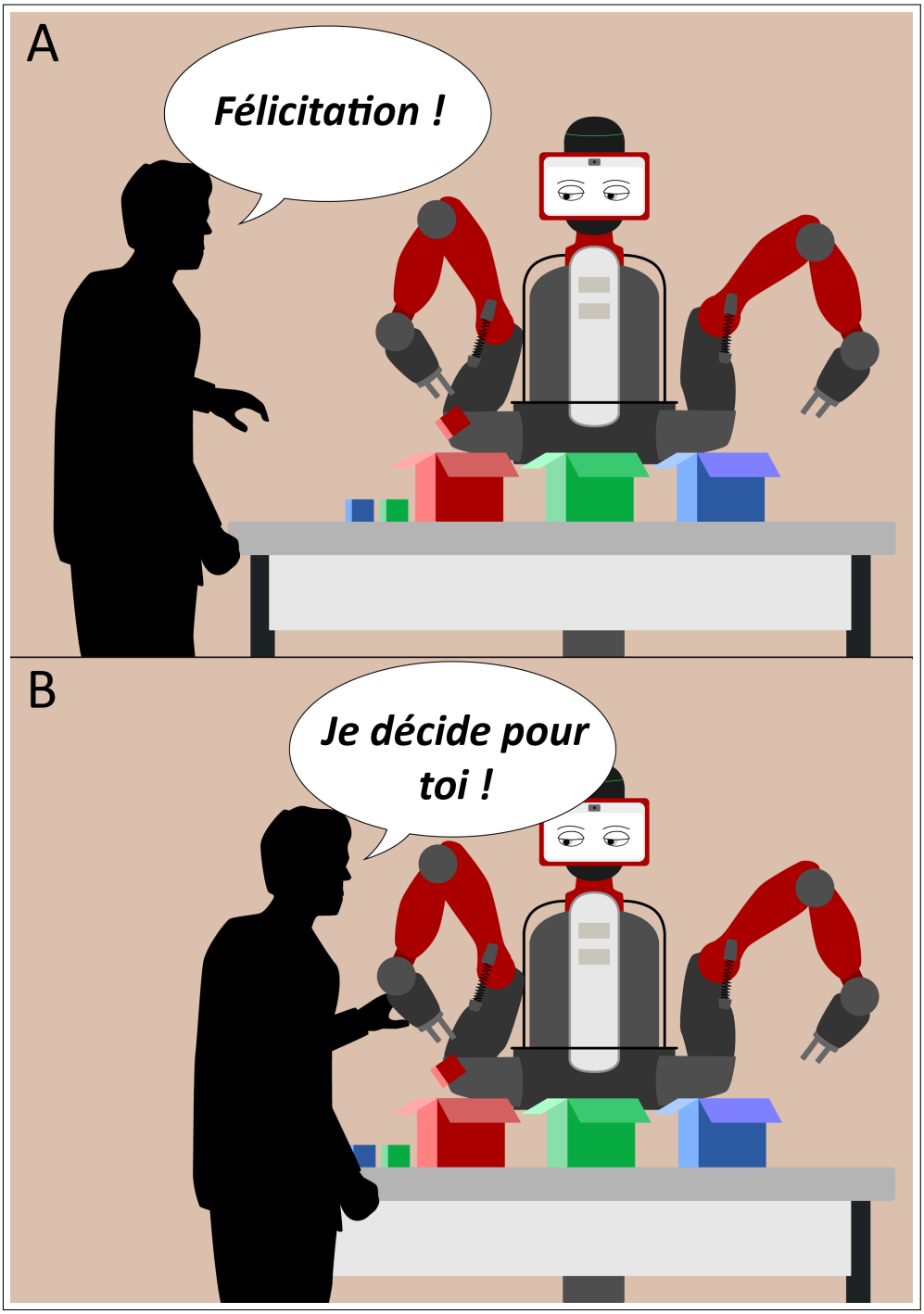

FIGURE 7.1 – *A. Interventions humaines de type félicitation. B. Interventions humaines de type prise de contrôle.*

plusieurs raisons : rester en adéquation avec la représentation utilisée dans l'expérience de navigation ; pour sa simplicité d'usage ; comme preuve de concept de l'intérêt de combiner des stratégies d'apprentissage model-free et model-based dans le domaine de l'interaction humain-robot.

7.1.3 Phase pré-expérimentale de babillage

Une période de babillage, où le robot peut manipuler les cubes, mais où aucune récompense ne lui est transmise, précède l'expérience. Nous avons défini cette phase pré-expérimentale du fait que dans cette tâche, le robot explore bien moins son environnement que dans la tâche de navigation (à taux d'exploration équivalent), et que cela entraîne des répercussions conséquences sur la performance de l'expert model-based. Les raisons à cette exploration de moins grande ampleur sont les suivantes :

— l'environnement de cette tâche de rangement est défini par environ trois fois plus d'états que dans la tâche de navigation (38 états contre 112),
— seulement 6 actions doivent être réalisées depuis l'état initial pour atteindre l'état final (soit environ 5% des états à traverser), contre 9 dans la tâche de navigation (soit environ 24% des états à traverser).
— l'environnement n'étant pas probabiliste, chaque action réalisée par le robot dans chaque état de cette tâche emmène dans un unique état terminal. Si l'environnemt probabiliste avait pour conséquence de rendre plus compliquée la traversée du robot dans la tâche de navigation, elle lui permettait aussi de découvrir par hasard des états non explorés.

Nous avons évalué les performances à plusieurs durées de babillage (Fig.7.2) pour le robot utilisant notre critère d'arbitrage (MC-EC) et sans interventions humaines, et avons finalement retenu la durée de 1200 itérations. Au-delà, cela ne permettait plus d'améliorer la performance du robot. Bien sûr, nous pourrions aussi faire le choix de donner au robot un modèle de transition plus ou moins complet avant le début de l'expérience. Nous considérons ici le cas où le robot n'a aucun a priori sur l'environnement et sait simplement quelles actions il peut ou non réaliser. De la même manière, nous pourrions très bien imaginer que le modèle de transition construit par le robot avant la première expérience pourrait être réutilisé pour toutes les prochaines expériences, dans le cas d'expériences réelles par exemple, où il est important de prendre en compte la durée des tâches expérimentales.

7.1.4 Humains simulés

Un humain simulé pouvant interagir avec le robot fait donc face à la table. Nous avons défini deux façons pour le robot d'apprendre des humains en nous inspirant des concepts d'*apprentissage par les rétroactions évaluatives* et d'*apprentissage par la démonstration* (voir chapitre 3). Nous nommons respectivement les deux types d'interventions sous-jacentes : intervention de type *félicitation* et intervention de type *prise de contrôle*. Plus précisément :

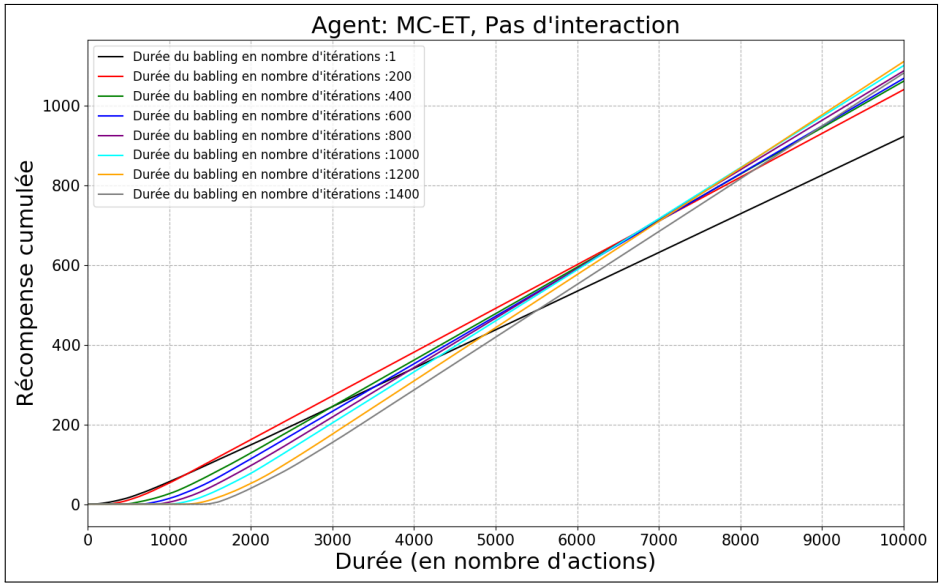

FIGURE 7.2 – *Performance moyenne du robot MC-EC pour différentes durées de babillage. Pour chaque durée, 50 expériences simulées ont été réalisées. La performance est définie comme la capacité du robot à accumuler de la récompense sur la durée de l'expérience. La durée est représentée par le nombre d'actions effectuées par le robot.*

— Dans le cas de l'intervention de type *félicitation*, l'humain peut féliciter le robot après qu'il ait mis un cube dans la boîte de sa couleur, par exemple le cube rouge dans la boîte rouge (Fig.7.1.A). L'effet de l'intervention sera effectif la prochaine fois que le robot se trouvera à nouveau dans la même situation (lorsqu'il tiendra à nouveau le cube rouge). Comme expliqué dans le chapitre 3, Knox et Stone (2012b) ont montré que plus les félicitations humaines affectent directement le processus de sélection des actions du robot, plus le robot est bon, et plus elles affectent la mise-à-jour des valeurs d'état-action pour chaque transition expérimentée, plus il est mauvais. Ainsi, dans notre travail, nous modélisons la félicitation de l'humain, et donc sa préférence, comme un biais positif (un bonus) d'une valeur d'état-action valable uniquement pendant le processus de décision, plutôt que comme une modification directe du modèle des valeurs d'état-action. Concrètement, nous nous inspirons de la méthode de *policy shaping* nommée *Action Biaising* et présentée dans le chapitre 3. Désormais, la fonction *softmax* (2.14) permettant de convertir l'estimation des valeurs d'état-action en une distribution de probabilités de sélection d'actions peut biaiser ces valeurs d'état-action :

$$P(a|s) = \frac{\exp((Q(s,a) + \alpha * \hat{H}(s,a))/\tau)}{\sum_{b \in \mathcal{A}} \exp((Q(s,b) + \alpha * \hat{H}(s,b))/\tau)} \tag{7.1}$$

où la préférence prédite par l'humain (le biais) $\hat{H}(s,a)$ est égale à 1 si l'humain a félicité le robot la dernière fois qu'il a exécuté l'action a dans l'état s, et 0 sinon. Le taux d'apprentissage de l'humain α est identique à celui du robot, donc égal à 0.6.

— Dans le cas de l'intervention de type *prise de contrôle*, l'humain peut passer outre le choix du robot, lorsqu'un cube est tenu par celui-

ci, en choisissant l'endroit où il sera déposé (Fig.7.1.B). Comme mentionné dans le chapitre 3, les démonstrations de l'enseignant humain correspondent généralement à des séquences de couples état-action $(s_0, a_0), ..., (s_t, a_t)$ que celui-ci réalise en observation du robot. Ici, comme pour la félicitation, la démonstration de l'humain est associée à un seul couple d'état-action (s_0, a_0). À noter que comparée à la félicitation, la démonstration a un effet instantané sur le robot. Et même s'il ne peut pas agir durant ces moments, le robot apprend quand même de l'observation des conséquences des actions choisies par l'humain.

Nous remarquons que dans les deux cas, aucun processus de mémorisation des interventions humaines n'a été modélisé. En interagissant avec le robot pour influencer ses décisions, l'humain biaise la mise-à-jour de ses valeurs d'état-action. De ce fait, la conséquence de l'intervention est donc incorporée dans le modèle de valeurs d'état-action du robot, qui illustre aussi bien les choix du robot que la préférence de l'humain, même s'il n'est pas possible de les dissocier.

7.1.5 Paramétrage des experts

Afin de montrer le caractère générique et tâche-indépendante de notre système d'apprentissage et de méta-contrôle, nous avons réutilisé le même jeu de paramètres que celui utilisé dans la tâche de navigation pour chacun des experts et pour le méta-contrôleur (tableau 7.1).

TABLE 7.1 – *Valeurs choisies des paramètres des experts et du méta-contrôleur dans la tâche de rangement de cubes.*

Param	MB	MF	MC
α	n.a.	0,6	n.a.
τ	0,02	0,02	0,02
γ	0,9	0,9	n.a.
κ	n.a.	n.a.	7.0

À l'inverse de la tâche de navigation, les valeurs d'état-action des experts ne sont pas initialisées à une valeur positive, et valent 0.0 en début d'expérience.

7.2 Résultats

Pour évaluer les performances des robots virtuels, nous réutilisons le code couleur de l'expérience de navigation : le rouge pour le robot MF-seul, le bleu pour le robot MB-seul, le vert pour le robot de coordination aléatoire (MC-RnD) et le violet pour le robot qui coordonne les deux experts en utilisant le critère d'arbitrage que nous avons proposé (MC-EC).

L'intérêt de cette expérience est d'évaluer l'apport du méta-contrôle dans une tâche où un robot peut interagir avec un humain. Nous commencerons par évaluer la performance des robots sans intervention humaines, puis avec les deux types d'interventions humaines définies précédemment.

7.2.1 Expériences sans intervention humaine

Nous pouvons observer dans la figure 7.3.A l'évolution de la performance moyenne des différents robots lorsque l'humain n'interagit pas avec eux. Comme dans les expériences de navigation, le robot MF-seul est celui affichant la moins bonne performance. En revanche, ici, la performance maximale est atteinte par les robots faisant du méta-contrôle, plutôt que par le robot MB-seul. La figure 7.3.B permet cependant de départager ces deux robots, le robot MC-EC affichant un coût calculatoire bien plus bas que celui du robot MC-RnD. Dans la figure 7.3.C, nous retrouvons la phase de guidage de l'expert MB, très courte, suivit de la phase de guidage de l'expert MF. Du fait que les valeurs d'état-action ont été initialisées à 0.0 en début d'expérience, et non pas à une valeur positive, nous n'observons pas la phase exploratoire de l'expert MF.

7.2.2 Interventions humaines de type *félicitation*

Dans la figure 7.4, nous pouvons visualiser la performance des différents robots à la dernière itération (la 10000ème) pour différentes durées d'interventions humaines de type *félicitation*. L'humain commence à intervenir directement après la fin de la période de babillage. Nous remarquons que seul le robot MF-seul semble être impacté significativement par l'intervention humaine. Pour vérifier si les différentes performances relatives aux différentes durées d'intervention des trois autres robots étaient effectivement toutes identiques, nous avons effectué trois tests de *Krustal-Wallis*. Le test de *Krustal-Wallis* est un test non paramétrique à utiliser lorsque nous sommes en présence de k échantillons indépendants, et lorsque nous voulons déterminer si les différents échantillons proviennent d'une même population (hypothèse nulle Ho), ou si au moins un échantillon provient d'une population différente des autres (hypothèse H1). Pour les robot MB-seul et MC-RnD, les tests ont montré qu'au moins une des performances était effectivement différente des autres (p-value MB-seul = $5.66 * 10^{-}5$ et p-value MC-RnD = 0.002 sont inférieures au seuil de significativité de 0.05). Afin d'identifier quelles performances étaient significativement différentes des autres, nous avons effectué des procédures de comparaisons multiples grâce à des tests de *Dunn* (Dunn 1964) avec *corrections de Bonferroni* (Fig.7.5). La *correction de Bonferroni* (Dunn 1961) est une méthode utilisée en statistique permettant de corriger le seuil de significativité lors de comparaisons multiples, comme par exemple durant un test de *Dunn*.

Si quatre tests de comparaisons de performance pour les robots MB-seul et MC-RnD ont en effet une p-value inférieure au seuil de significativité de 0.05, nous remarquons que l'effet semble avant tout dû à la variabilité des données. En témoigne la proximité de ces p-values du seuil de 0.05 comparativement à celles du robot MF-seul. Par exemple, pour le robot MB-seul, la performance relative à la durée de 10 interventions sort du lot, alors qu'aucune raison ne semble pouvoir l'expliquer. À l'inverse, l'effet de l'intervention de type *félicitation* sur la performance du robot MF-seul à un effet proportionnel à la durée de l'intervention, ce qui fait sens.

Un test de *Krustal-Wallis* entre les performances des quatre robots pour une durée d'intervention de 500 itérations nous confirme qu'au moins

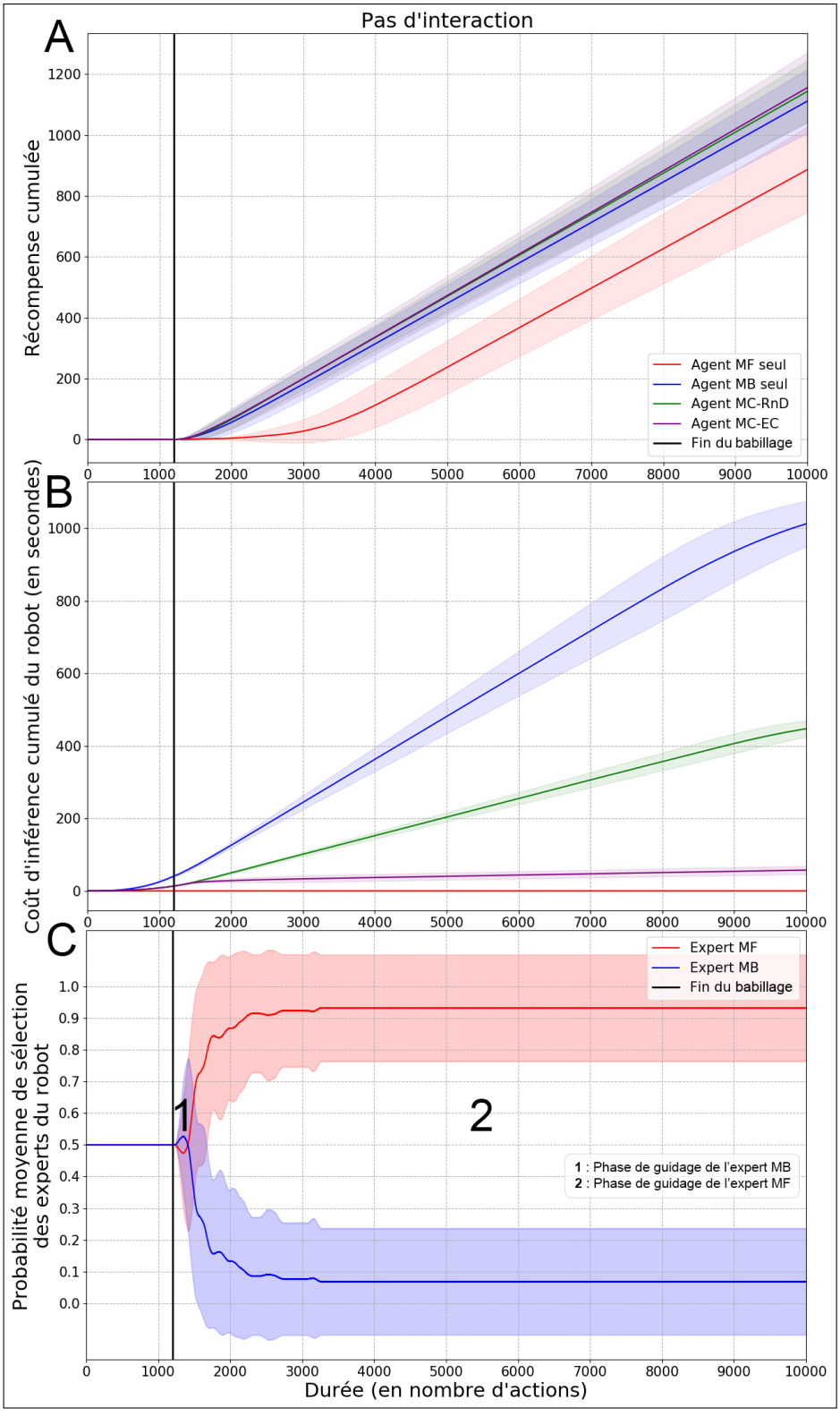

FIGURE 7.3 – ***A***. *Performance moyenne pour 50 expériences simulées. La performance est définie comme la capacité du robot à accumuler de la récompense sur la durée de l'expérience. La durée est représentée par le nombre d'actions effectuées par le robot.* ***B***. *Coût moyen de calcul pour 50 expériences simulées. Le coût de calcul est défini comme le temps cumulé pris pour exécuter le processus d'inférence du robot sur la durée de l'expérience, en secondes.* ***C***. *Probabilité moyenne de sélection des experts par le méta-contrôleur du robot MC-EC pour 50 expériences simulées. Nous utilisons l'écart-type comme indicateur de dispersion dans les trois figures.*

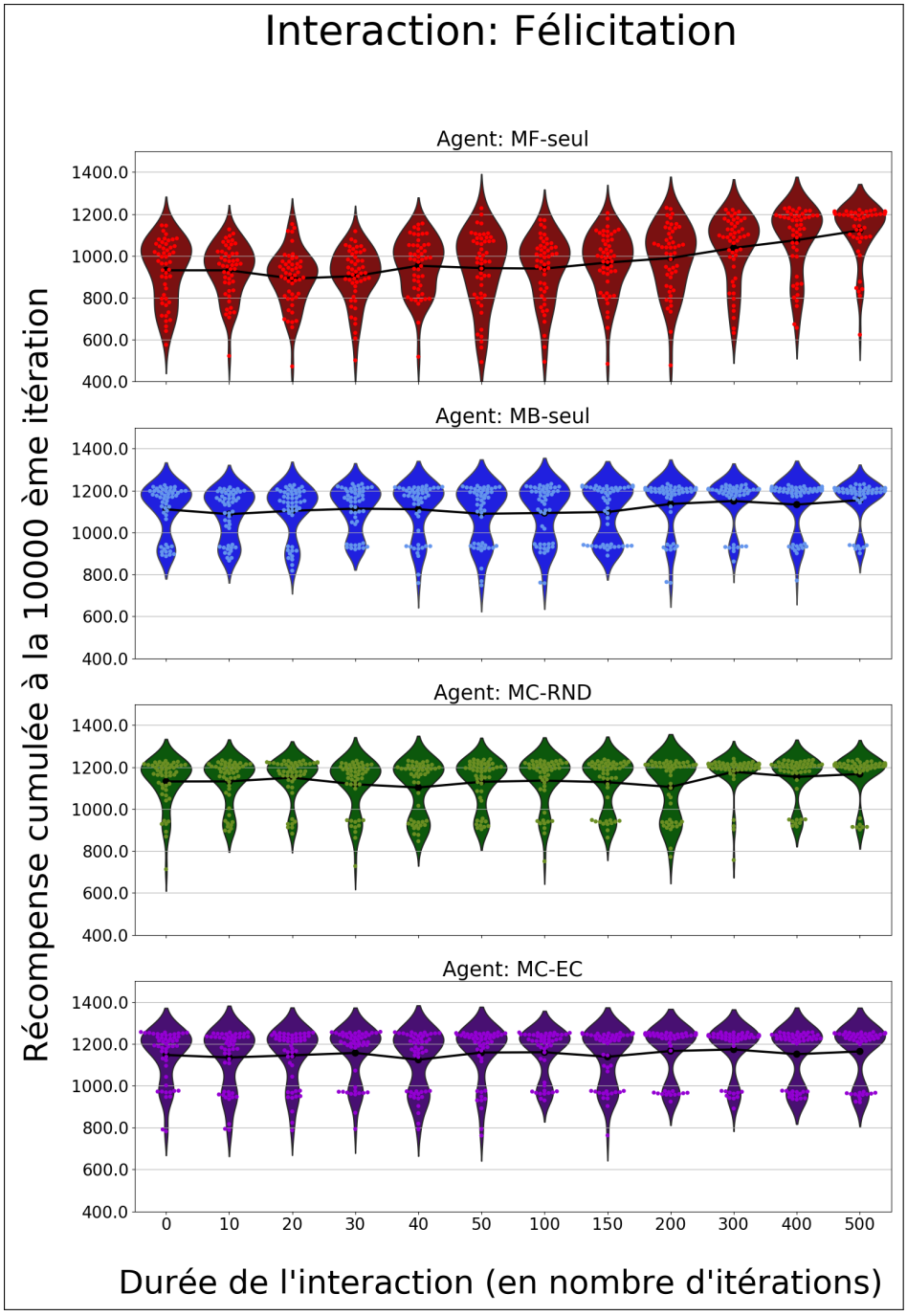

FIGURE 7.4 – ***A***. *Diagrammes en violon de la récompense totale cumulée à la 10000ème itération par le robot MF-seul pour différentes durées d'interventions humaines de type félicitation. Les points de couleur représentent les performances unitaires des différentes expériences et les points noirs les performances moyennes pour 50 expériences réalisées par durées d'intervention.* ***B***. *Identique à A pour le robot MF-seul.* ***C***. *Identique à A pour le robot MC-RnD.* ***D***. *Identique à A pour le robot MC-EC.*

MF / MB	0	10	20	30	40	50	100	150	200	300	400	500
0	0	1.0	1.0	1.0	1.0	1.0	1.0	0.408531	0.030455	0.000016	1.62689 1e-08	1.57263 4e-13
10	1.0	10	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.008622	3.91525 2e-05	3.02557 2e-09
20	1.0	1.0	20	1.0	1.0	1.0	1.0	0.563601	0.045360	0.000029	3.24847 2e-08	3.71462 0e-13
30	1.0	1.0	1.0	30	1.0	1.0	1.0	1.0	0.180889	0.000215	3.82907 2e-07	8.16058 5e-12
40	1.0	1.0	1.0	1.0	40	1.0	1.0	1.0	1.0	0.155930	1.69610 2e-03	4.40258 4e-07
50	1.0	1.0	1.0	1.0	1.0	50	1.0	1.0	1.0	0.203656	2.42317 8e-03	7.12742 3e-07
100	1.0	1.0	1.0	1.0	1.0	1.0	100	1.0	1.0	0.074458	6.37655 9e-04	1.18686 2e-07
150	1.0	1.0	1.0	1.0	1.0	1.0	1.0	150	1.0	1.0	2.16903 0e-02	1.44844 7e-05
200	0.568655	0.033144	0.406211	1.0	1.0	1.0	1.0	1.0	200	1.0	3.10143 3e-01	6.59296 9e-04
300	0.132280	0.005327	0.090044	1.0	1.0	0.478832	1.0	1.0	1.0	300	1.0	3.83120 0e-01
400	0.474306	0.026334	0.336678	1.0	1.0	1.0	1.0	1.0	1.0	1.0	400	1.0
500	0.172429	0.007403	0.118358	1.0	1.0	0.605852	1.0	1.0	1.0	1.0	1.0	500

RND / EC	0	10	20	30	40	50	100	150	200	300	400	500
0	0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.890272	1.0	1.0
10	1.0	10	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.455387	1.0	0.632360
20	1.0	1.0	20	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
30	1.0	1.0	1.0	30	1.0	1.0	1.0	1.0	1.0	0.029717	0.655518	0.044854
40	1.0	1.0	1.0	1.0	40	1.0	1.0	1.0	1.0	0.017303	0.431479	0.026507
50	1.0	1.0	1.0	1.0	1.0	50	1.0	1.0	1.0	0.819644	1.0	1.0
100	1.0	1.0	1.0	1.0	1.0	1.0	100	1.0	1.0	1.0	1.0	1.0
150	1.0	1.0	1.0	1.0	1.0	1.0	1.0	150	1.0	1.0	1.0	1.0
200	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	200	1.0	1.0	1.0
300	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	300	1.0	1.0
400	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	400	1.0
500	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	500

FIGURE 7.5 – *Résultats des tests de comparaisons multiples de Dunn pour les performances des quatre robots dans le cadre de l'intervention de type félicitation. Les p-values inférieures au seuil de significativité* 0.05 *sont colorées en rouge. Le seuil de significativé a été corrigé avec la correction de Bonferroni.*

une des performances est significativement différente des autres (p-value = $2.99 * 10^-7 < 0.05$). Finalement, un test de *Dunn* nous permet de voir que la performance du robot MC-EC à une durée d'intervention de 500 itérations est significativement différente de la performance des robots MC-RnD (p-value = $0.0408 < 0.05$), MB-seul (p-value = $9 * 10^-5 < 0.05$) et MF-seul (p-value = $3.94 * 10^-7 < 0.05$) à la même durée d'intervention. La performance du robot MC-RnD est elle aussi significativement différente de la performance du robot MF-seul (p-value = $0.042 < 0.05$) tandis que les robots MF-seul et MB-seul ont des performances identiques (p-value = $1.0 > 0.05$).

Pour le moment, nous avons donc montré que l'intervention humaine de type *félicitation* semble être utile uniquement au robot MF-seul embarquant un unique un expert model-free, sans pour autant lui permettre d'atteindre la performance maximale, uniquement atteinte par le robot MC-EC. Les robots MB-seul, MC-RnD et MC-EC embarquant un expert model-based, n'ont quant à eux pas besoin de l'intervention humaine pour améliorer leur performance.

La figure 7.6.A nous permet de comparer les coûts cumulés des processus d'inférences des différents robots à la fin de l'expérience dans le cas où l'humain n'interagit pas avec le robot, et dans le cas où l'humain félicite le robot. Globalement, nous pouvons dire que l'aide apportée par l'humain semble décharger légèrement le robot en coût de calcul. Cela est surtout observable pour le robot MB-seul (qui effectue de fait plus de calculs coûteux que les autres robots). Dans tous les cas, le coût affiché du robot MC-EC est à nouveau extrêmement bas comparativement à ceux des robots MB-seul et MC-RnD. Nous observons à nouveau les phases de guidage des deux expert dans la figure 7.6.B.

Finalement, nous pouvons conclure que le robot MC-EC est capable, à un coût minimal, de palier à l'absence de l'humain. Ici, l'expert MB se comporte comme un "expert de secours", qui permet au robot de ne pas être dépendant de l'humain. Lorsque l'humain est présent et interagit avec le robot, le coût de l'expert MB diminue, signe qu'il effectue moins de calcul coûteux. Lorsque la durée de l'intervention est longue, le robot MF-seul est totalement capable de réaliser la tâche efficacement à un coût calculatoire très bas. Pourtant, dès que la durée de l'intervention diminue, sa performance chute. Dans une situation où la présence de l'humain est incertaine, le robot MC-EC est donc l'expert idéal. La figure 7.7 nous permet de confirmer ce raisonnement. Ici, les différents robots interagissent avec des humains oubliant par moment de le féliciter. Si l'oubli a un effet visible à l'oeil nu sur la performance du robot MF-seul, la ramenant à la performance de non-intervention, les autres robots y font logiquement face sans souci, puisque l'intervention ne leur apporte rien.

Pour éprouver les capacités d'adaptation de ces différents robots face à des humains un peu plus réalistes, nous les avons fait interagir avec des humains pouvant commettre des erreurs (Fig.7.8). Dans le cadre de l'intervention de type *félicitation*, une erreur consiste à féliciter une mauvaise action du robot (par exemple mettre le cube rouge dans la boîte verte). Nous pouvons voir que si tous les robots subissent une dégradation de leur performance visible à l'oeil nu, le robot MF-seul est celui le plus touché par les erreurs humaines. Cela corrobore nos observations précédentes

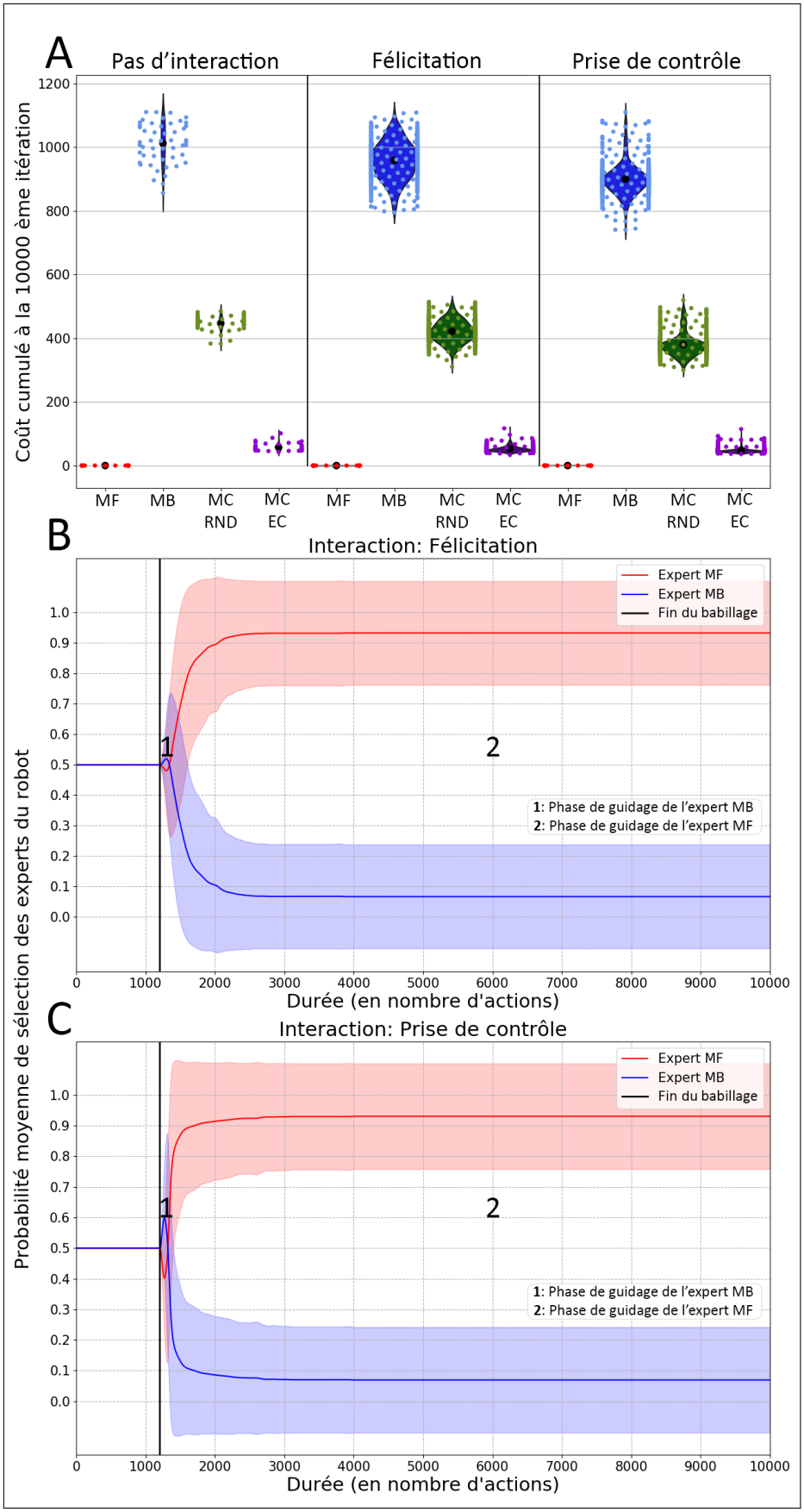

FIGURE 7.6 – ***A***. *Diagrammes en violon des coûts des processus d'inférence cumulés à la 10000ème itération par les différents robots et pour les différents types d'intervention. Les points de couleur représentent les performances unitaires des différentes expériences et les points noirs les performances moyennes pour toutes les expériences toutes durées d'interventions confondues, c'est-à-dire 600 expériences par type de robot.* ***B***. *Probabilité moyenne de sélection des experts par le méta-contrôleur du robot MC-EC pour 50 expériences simulées dans le cadre d'une intervention de type félicitation et d'une durée de 500 itérations.* ***C***. *Identique à B pour l'intervention de type prise de contrôle.*

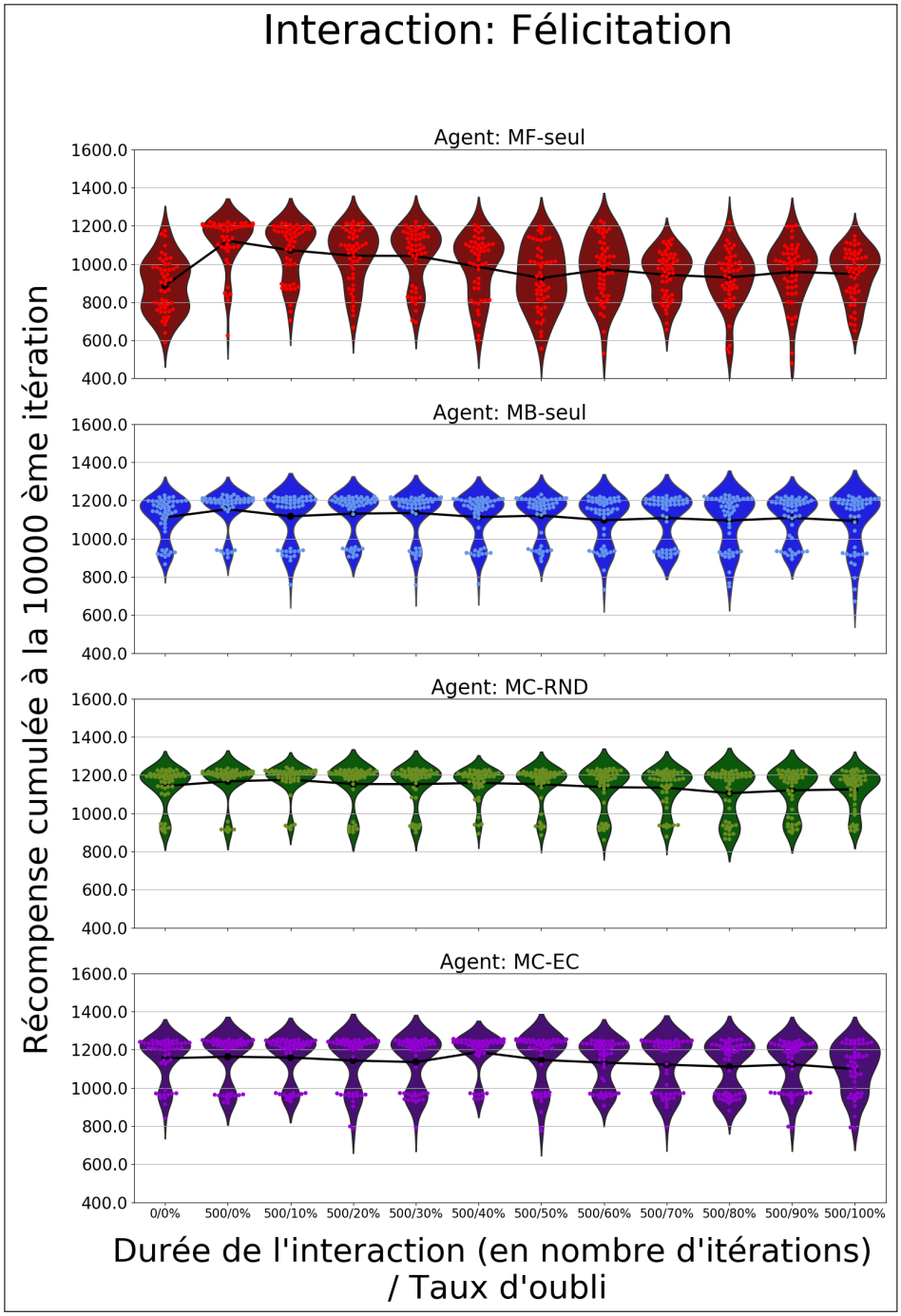

FIGURE 7.7 – ***A***. *Diagrammes en violon de la récompense totale cumulée à la 10000ème itération par le robot MF-seul dans le cas d'une intervention de type félicitation longue de 500 itérations pour différents taux d'oubli humains. Les points de couleur représentent les performances unitaires des différentes expériences et les points noirs les performances moyennes pour 50 expériences réalisées par durées d'intervention.* ***B***. *Identique à A pour le robot MF-seul.* ***C***. *Identique à A pour le robot MC-RnD.* ***D***. *Identique à A pour le robot MC-EC.*

concernant la dépendance du robot MF-seul, et donc celle l'expert MF, à l'intervention de l'humain. À nouveau, utiliser un expert MB est très profitable pour le robot. Dans les quatre cas, et même si la dégradation des performances des autres robots est minimale, nous observons qu'à très haut taux d'erreurs humaines, la quantité de récompenses cumulées à la fin de l'expérience reste inférieure à cette même quantité lorsque l'humain ne fait jamais d'erreurs ou n'interagit jamais avec le robot. Cela est dû au fait que pendant cette durée de 500 itérations d'interventions, les robots, quels qu'ils soient, peinent à accumuler de la récompense, ce qui crée donc un retard de performance par rapport aux robots n'interagissant pas avec l'humain ou avec un humain ne faisant pas d'erreurs.

Finalement, utiliser notre critère d'arbitrage permet au robot de ne pas être dépendant de l'humain pour atteindre l'objectif qui lui a été fixé, mais aussi d'encaisser plus efficacement ses potentielles erreurs.

7.2.3 Interventions humaines de type *prise de contrôle*

Dans cette sous-partie, nous nous intéressons à l'intervention humaine de type *prise de contrôle*. À la différence de l'intervention de type *félicitation*, nous pouvons voir dans la figure 7.9 que l'intervention de type *prise de contrôle* a un effet sur la performance de chaque robot, même si l'effet de la performance sur le robot MF-seul reste plus important. Pour les trois autres robots, nous pouvons voir à l'oeil nu qu'intervenir sur une durée supérieure à 100 actions n'augmente plus significativement la performance. Un test de *Krustal-Wallis* entre les performances des quatre robots pour une durée d'intervention de 500 itérations nous confirme qu'au moins une des performances est significativement différente des autres (p-value = $6.10 * 10^{-}35 < 0.05$). Un test de *Dunn* nous permet de voir que la performance du robot MC-EC à une durée d'intervention de 500 itérations est significativement différente de la performance des robot MC-RnD (p-value = $5.84 * 10^{-}16 < 0.05$), MB-seul (p-value = $1.73 * 10^{-}32 < 0.05$) et MF-seul (p-value = $2.50 * 10^{-}04 < 0.05$) à la même durée d'intervention. La performance du robot MC-RnD est elle aussi significativement différente de la performance des robots MF-seul (p-value = $1.53 * 10^{-}04 < 0.05$) et MB-seul (p-value = $1.25 * 10^{-}03 < 0.05$) qui ont tout deux aussi une performance significativement différente (p-value = $1.44 * 10^{-}04 > 0.05$). Ces performances dépassent en moyenne les 1200 récompenses accumulées, c'est-à-dire davantage que les performances maximales obtenues par les robots dans le cadre de l'intervention de type *félicitation* (Fig.7.4), inférieures à 1200 unités de récompense accumulées. Finalement, tous les robots ont des performances différentes, et à nouveau, le robot MC-EC est le meilleur d'entre tous. Nous pouvons expliquer la sur-performance de l'intervention de type *prise de contrôle* du fait que la décision de l'humain remplace celle du robot dans 100% des cas, alors que dans le cas de l'intervention de type *félicitation*, le processus décisionnel, bien que biaisé en faveur de l'humain, est toujours soumis à la fonction *softmax* (2.14), qui peut par moment sélectionner une action non optimale. De plus, l'intervention de type *prise de contrôle* agit sur le comportement du robot à l'itération à laquelle elle est effectuée, tandis que l'intervention de type *félicitation* aura de l'influence sur le comportement du robot uniquement la prochaine fois

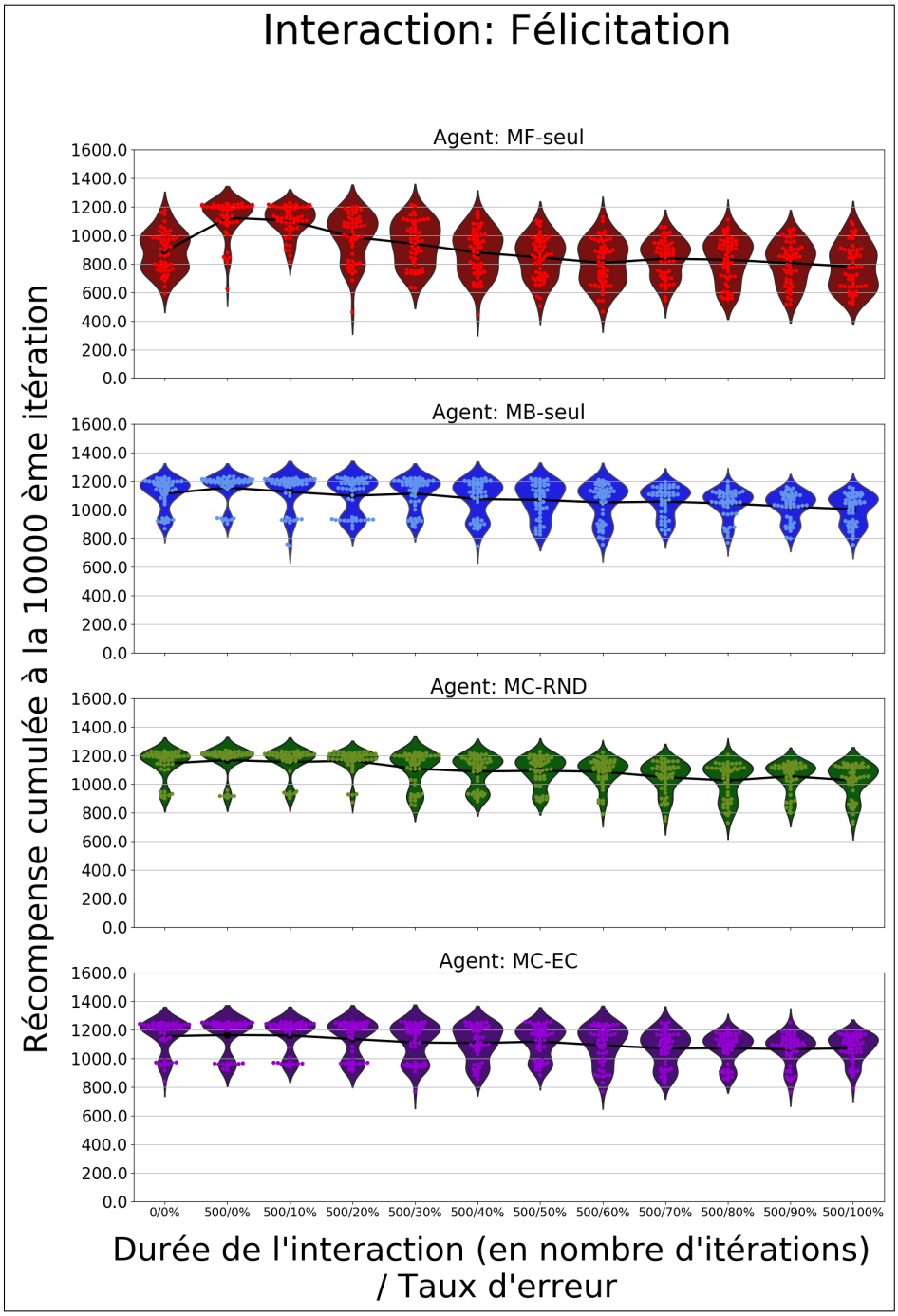

FIGURE 7.8 – ***A***. *Diagrammes en violon de la récompense totale cumulée à la 10000ème itération par le robot MF-seul dans le cas d'une intervention de type félicitation longue de 500 itérations pour différents taux d'erreurs humaines. Les points de couleur représentent les performances unitaires des différentes expériences et les points noirs les performances moyennes pour 50 expériences réalisées par durées d'interventions.* ***B***. *Identique à A pour le robot MF-seul.* ***C***. *Identique à A pour le robot MC-RnD.* ***D***. *Identique à A pour le robot MC-EC.*

que le robot effectuera la combinaison état-action que l'humain avait félicitée. Dans la figure 7.6.A, nous pouvons voir que les valeurs de coûts cumulés sont aussi faibles que dans le cadre de l'intervention de type *félicitation* : plus l'intervention humaine est efficace, moins l'expert MB a besoin de faire des calculs coûteux. Pour finir, nous observons à nouveau les phases de guidage des deux expert dans la figure 7.6.C.

Si nous observions précédemment que les robots MB-seul, MC-RnD et MC-EC n'étaient pas impactés par des humains oubliant d'intervenir, du fait que l'humain n'apportait aucune aide significative aux robots dotés d'un expert MB, les choses sont logiquement différentes ici, puisque l'intervention apporte une aide plus nette. En effet, nous pouvons voir dans la figure 7.10 qu'à hauts taux d'oubli, les performances de tous les robots se dégradent, même si à nouveau, la dégradation de la performance du robot MF-seul reste bien plus importante. Des trois autres robots, le robot MC-EC semble d'ailleurs être celui faisant le mieux face aux oublis de son partenaire humain.

Pour finir, nous mettons à nouveau les robots en face d'humains réalisant des erreurs (Fig.7.11). Dans le cadre de l'intervention de type *prise de contrôle*, cela signifie que l'humain prend le contrôle du bras du robot pour mettre le cube tenu dans la mauvaise boîte, ou pour retirer les cubes se trouvant dans les boîtes de la bonne couleur. Ici les résultats sont assez proches de ceux observés dans la figure 7.8 : nous observons une dégradation globale de la performance des robots, à nouveau bien plus intensive dans le cas du robot MF-seul. Comme précédemment, à très haut taux d'erreurs humaines, les quantités de récompenses cumulées à la fin de l'expérience sont inférieures à ces mêmes quantités lorsque l'humain n'interagit jamais avec les robots, du fait du retard de performance accumulé durant les 500 itérations d'interventions erronées.

Finalement, comme pour l'intervention de type *félicitation*, utiliser notre critère d'arbitrage permet au robot de ne pas être dépendant de l'humain effectuant une *prise de contrôle* pour atteindre l'objectif qui lui a été fixé, mais aussi d'encaisser plus efficacement ses erreurs. Pour autant, comme observé dans d'autres études (Knox et al. 2011), l'intervention de type *prise de contrôle* offre des meilleures performances aux robots que l'intervention de type *félicitation*. Elle est donc à privilégier.

7.3 Conclusion

Dans ce chapitre, nous avons évalué notre modèle de coordination d'experts d'apprentissage dans une tâche d'interaction humain-robot simulée. Dans cette nouvelle tâche, un humain virtuel pouvait interagir avec un robot de plusieurs façons (félicitation, prise de contrôle). Le nouvel environnement était aussi composé d'environ trois fois plus d'états que celui de la tâche de navigation (6). Malgré ces nombreuses différences, nous avons réutilisé les paramètres optimisés pour la tâche de navigation, afin de montrer le caractère générique et tâche-indépendant de notre système d'apprentissage et de méta-contrôle. Dans ce chapitre, nous montrons à nouveau que le critère d'arbitrage de notre système de coordination permet au robot d'atteindre le même niveau de performance

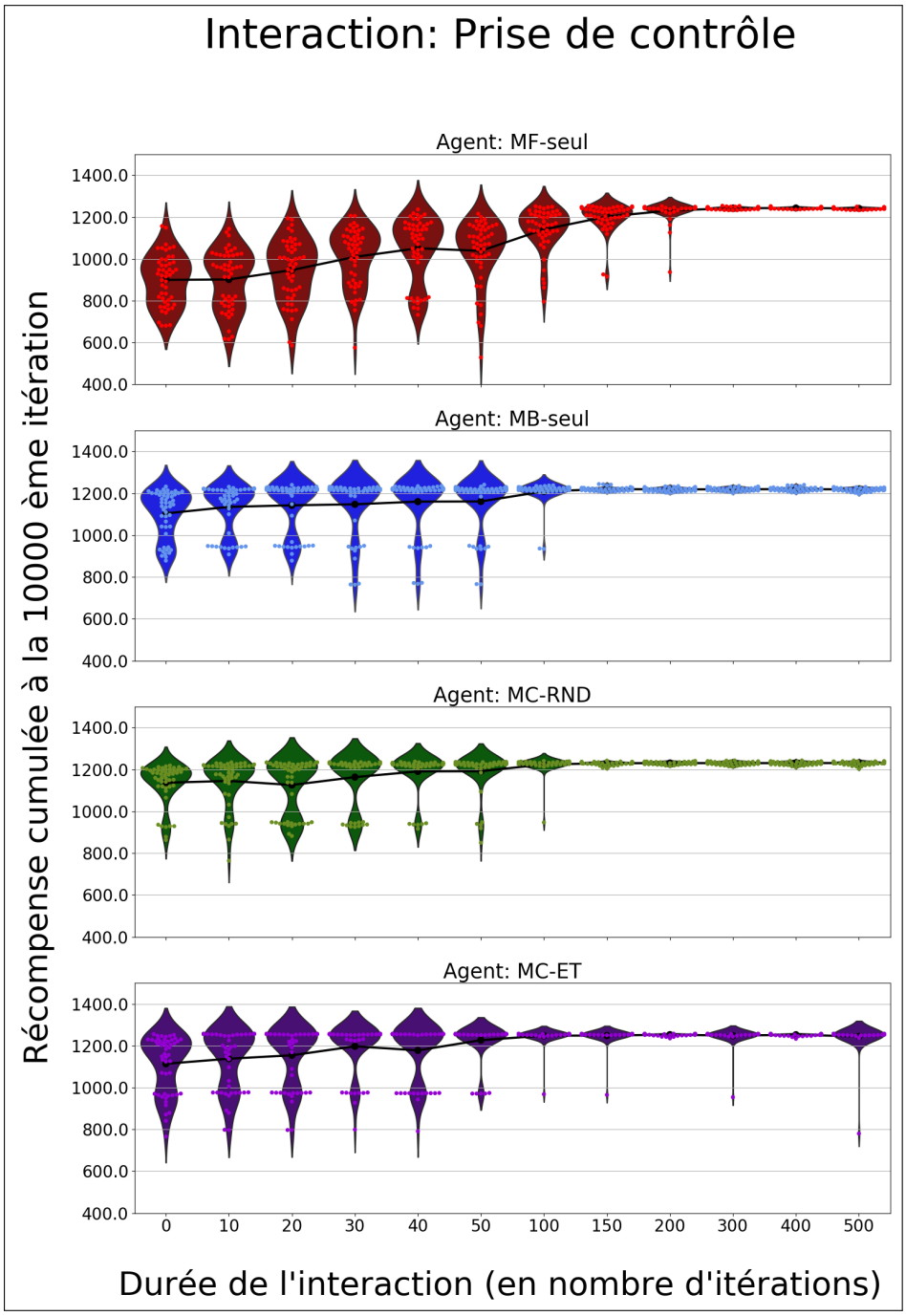

FIGURE 7.9 – ***A***. *Diagrammes en violon de la récompense totale cumulée à la 10000ème itération par le robot MF-seul pour différentes durées d'interventions humaines de type prise de contrôle. Les points de couleur représentent les performances unitaires des différentes expériences et les points noirs les performances moyennes pour 50 expériences réalisées par durées d'intervention.* ***B***. *Identique à A pour le robot MF-seul.* ***C***. *Identique à A pour le robot MC-RnD.* ***D***. *Identique à A pour le robot MC-EC.*

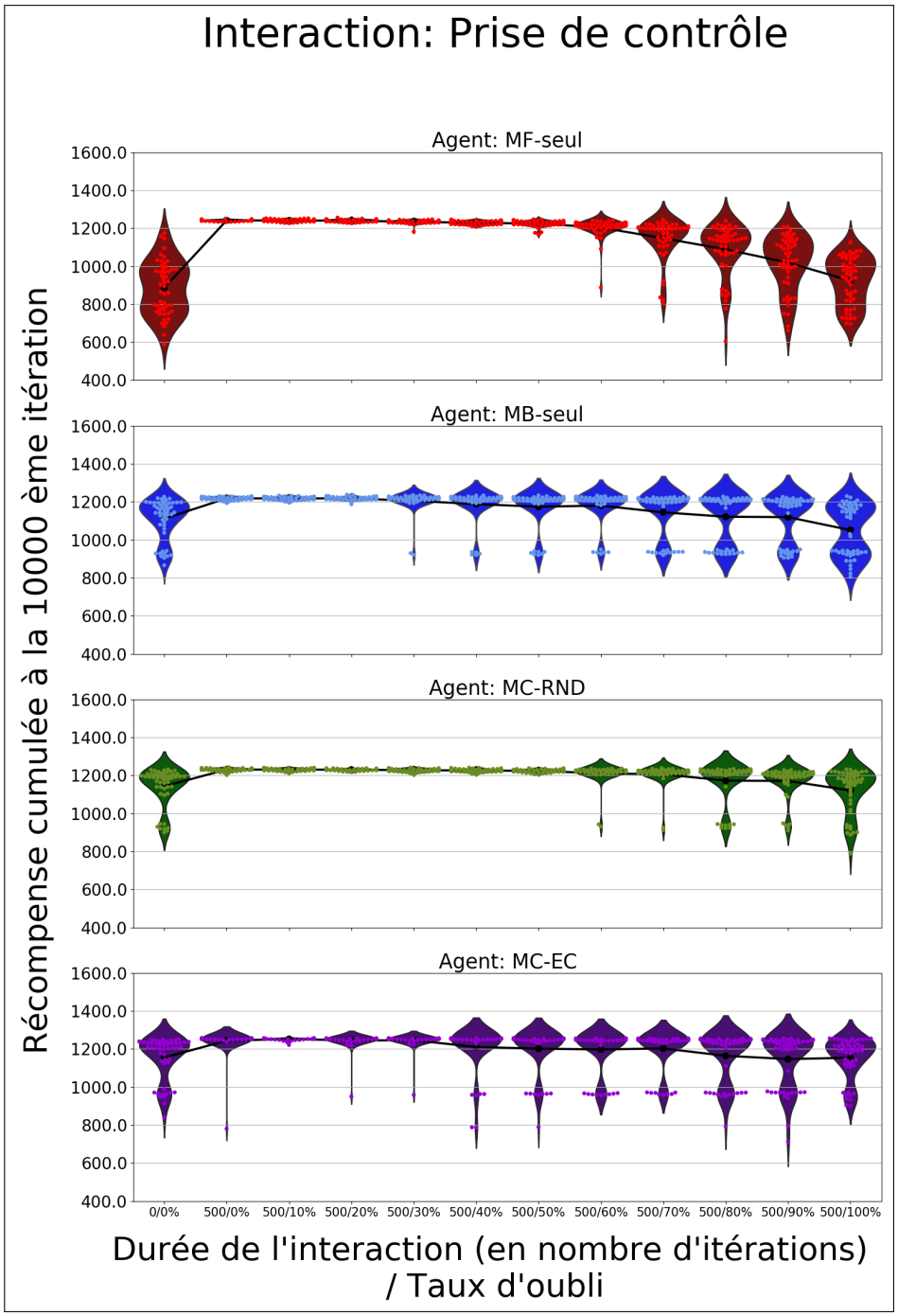

FIGURE 7.10 – ***A***. *Diagrammes en violon de la récompense totale cumulée à la 10000ème itération par le robot MF-seul dans le cas d'une intervention de type prise de contrôle longue de 500 itérations pour différents taux d'oubli humain. Les points de couleur représentent les performances unitaires des différentes expériences et les points noirs les performances moyennes pour 50 expériences réalisées par durées d'intervention.* ***B***. *Identique à A pour le robot MF-seul.* ***C***. *Identique à A pour le robot MC-RnD.* ***D***. *Identique à A pour le robot MC-EC.*

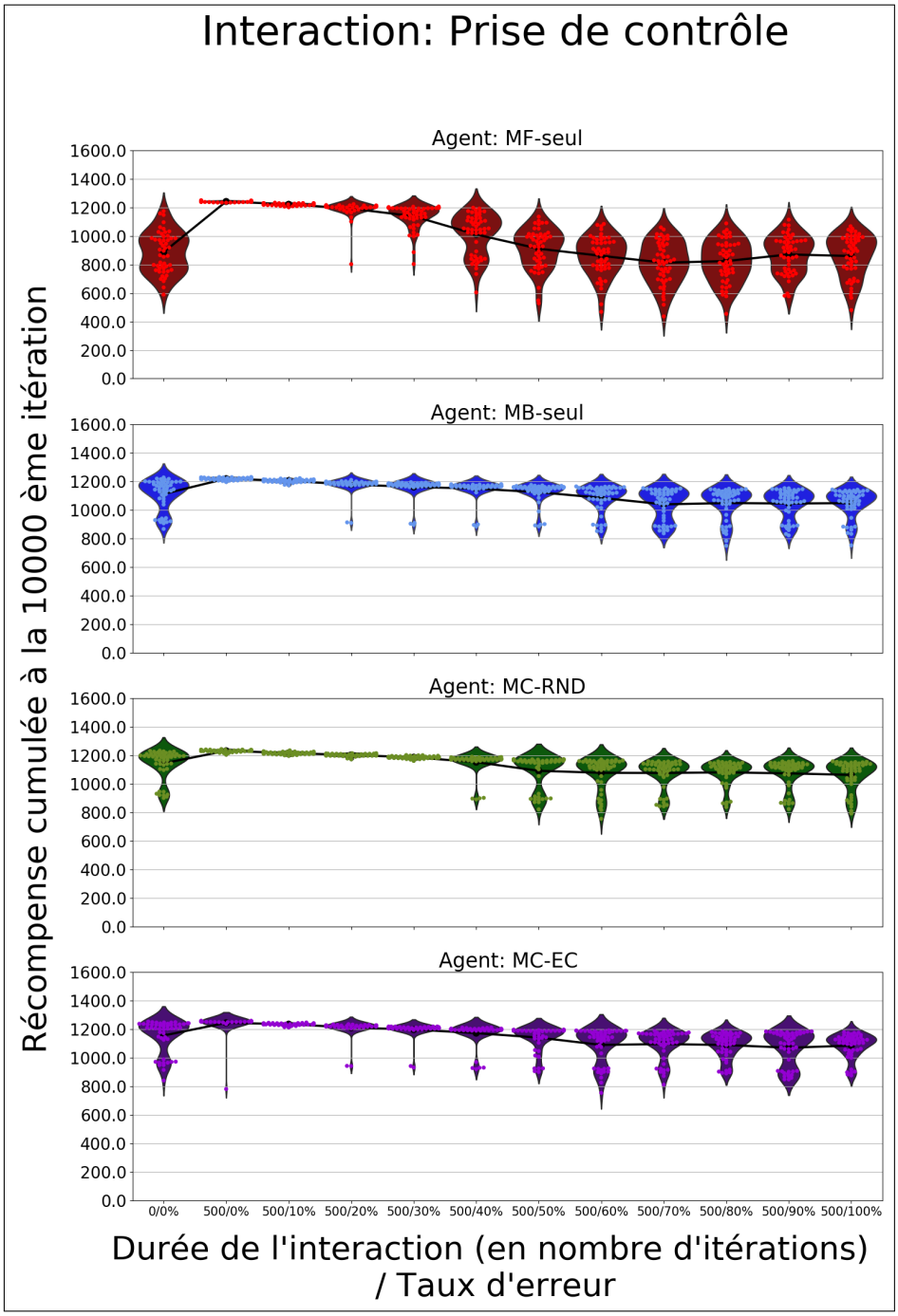

FIGURE 7.11 – ***A***. *Diagrammes en violon de la récompense totale cumulée à la 10000ème itération par le robot MF-seul dans le cas d'une intervention de type félicitation longue de 500 itérations pour différents taux d'erreurs humaines. Les points de couleur représentent les performances unitaires des différentes expériences et les points noirs les performances moyennes pour 50 expériences réalisées par durées d'intervention.* ***B***. *Identique à A pour le robot MF-seul.* ***C***. *Identique à A pour le robot MC-RnD.* ***D***. *Identique à A pour le robot MC-EC.*

que celui d'un robot contrôlé uniquement par un algorithme d'apprentissage model-based, tout en réduisant drastiquement son coût calculatoire, mais aussi de faire face efficacement à la volatilité du comportement humain (erreurs, oublis). Dans cette tâche, l'expert MB peut être considéré comme un expert de secours, utile au robot uniquement lorsque l'humain est absent ou que son comportement est délétère. L'intérêt de notre architecture est donc ici de permettre au robot d'être plus robuste à la présence/absence de l'humain et aux variations des interventions humaines. Nous montrons aussi que l'intervention de type *prise de contrôle* influence davantage positivement le comportement du robot, de part son impact plus direct sur celui-ci.

Finalement, nous illustrons avec cette nouvelle tâche d'interaction humain-robot le caractère tâche indépendant de notre modèle de coordination ainsi que sa capacité à faire face à la volatilité du comportement humain. Nous montrons aussi son caractère générique en réutilisant les mêmes valeurs de paramètres que pour la tâche de navigation (6). Pour améliorer le modèle, nous pourrions entre autres doter le robot d'un mécanisme lui permettant d'ignorer les interventions de l'humain lorsque celui-ci jugerait qu'elles ne sont pas appropriées. Dans le chapitre 8, nous évaluons notre modèle de coordination dans une tâche de coopération humain-robot où le robot a désormais obligatoirement besoin de l'humain pour effectuer la tâche qu'il doit accomplir.

8 Évaluation du système d'arbitrage dans une tâche de coopération humain-robot

Sommaire

Dans ce chapitre, nous évaluons notre système de coordination dans une tâche de coopération humain-robot différente de la précédente. Dans un premier temps, nous présentons la nouvelle tâche de rangement de cubes en simulation et la manière dont nous avons modélisé le partenaire humain avec qui le robot doit désormais obligatoirement coopérer pour mener à bien ses objectifs. Dans la seconde partie, nous présentons les résultats obtenus et montrons que dans une situation où le partenaire peut se muer en adversaire, notre système de coordination n'est plus capable de maintenir un haut niveau de performance. Pour contourner ce problème lié à une dissymétrie algorithmique naturelle entre les experts MF et MB, et non pas au partenaire humain, qui n'en est que le révélateur, nous proposons une solution peu coûteuse, sous la forme de l'ajout d'un mécanisme de détection de changements de contextes au robot. Avec ce mécanisme, le robot est à nouveau capable de maintenir un haut niveau de performance tout en diminuant toujours grandement son coût calculatoire. Ce chapitre correspond à un article en cours de rédaction.

8.1 Matériel et protocole expérimental

8.1.1 Environnement et robot simulés

Cette expérience est à nouveau réalisée uniquement en simulation. Le même robot que celui présenté dans le chapitre 7 fait face à une table. Cette fois-ci, la table est divisée en trois espaces distincts : un espace accessible uniquement par l'humain, un espace commun et un espace accessible uniquement par le robot. Chaque espace possède une boîte, désignée comme étant la boîte de l'humain et celle du robot. Trois cubes de couleur sont placés sur la table. La figure 8.1 illustre l'expérience. Cette tâche est inspirée de celles de Alami et al. (2011), Renaudo et al. (2015a).

Dans cette nouvelle tâche de rangement, le premier objectif du robot est d'apprendre à mettre chaque cube dans sa boîte. Lorsque cela est fait, le robot obtient une unité de récompense virtuelle, et les cubes sont automatiquement replacés dans la boîte de l'humain. Nous réintroduisons ici le changement d'objectif en cours d'expérience, comme pour la tâche de navigation. Ici, à la 5000ème itération, le robot doit désormais apprendre à mettre chaque cube dans la boîte de l'humain. Lorsque cela est fait, les cubes sont automatiquement replacés dans la boîte du robot.

Nous répliquerons les résultats de cette expérience avec une autre paire d'objectifs : mettre les cubes rouges et bleu dans la boîte du robot et le vert dans la boîte de l'humain (lorsque le robot obtient la récompense, les cubes rouges et bleu sont replacés dans la boîte de l'humain et le cube vert dans la boîte du robot), puis mettre les cubes rouges et bleu dans la boîte de l'humain et le vert dans la boîte du robot (lorsque le robot obtient la récompense, les cubes rouges et bleu sont replacés dans la boîte du robot et le cube vert dans la boîte de l'humain).

Nous pouvons voir qu'à l'inverse de la tâche du chapitre 7, où le robot pouvait réaliser l'expérience sans l'aide de l'humain, la participation de l'humain est ici indispensable, puisque le robot n'a pas accès au côté de la table de l'humain. Pour cette raison, nous parlons d'ici de *coopération avec l'humain*, et non plus seulement d'*intervention humaine*.

8.1.2 Espace d'état et d'action du robot

L'espace d'état est à nouveau un espace d'état discret. Un état représente toujours la position des trois cubes de couleur. Chacun des cubes peut se situer : dans la boîte de l'humain, dans l'espace commun, dans la boîte du robot, dans la main de l'humain et dans la main du robot. Si l'on supprime les états où le robot et l'humain tiennent plusieurs cubes à la fois, cela représente un total de 99 états, soit 13 de moins que la tâche du chapitre 7.

Concernant l'espace d'action, le robot peut réaliser 6 actions classiques : prendre le cube rouge, prendre le cube vert, prendre le cube bleu, poser le cube tenu en main dans sa boîte, poser le cube tenu en main dans l'espace commun, passer son tour. À cela s'ajoutent 2 actions interactives, permettant de donner directement le cube tenu en main à l'humain (si sa main est vide) ou à l'inverse de demander à l'humain de donner le cube qu'il tient en main (si la main du robot est vide), amenant à un total de 8 actions.

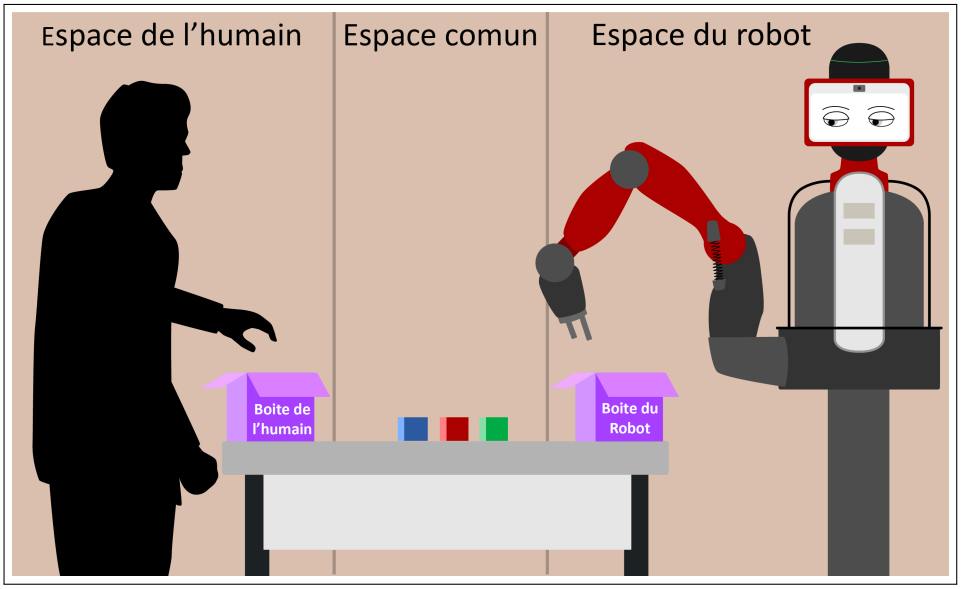

FIGURE 8.1 – *Illustration de l'expérience de coopération humain-robot.*

Comme nous allons le voir dans la sous-partie suivante, l'humain est considéré dans cette expérience comme un agent décisionnel, et possède donc son propre espace d'état équivalent à celui du robot.

8.1.3 L'humain simulé

Dans l'expérience du chapitre 7, l'humain pouvait de temps en temps interagir avec le robot. Ici, sa participation à la tâche est indispensable à la réussite du robot. Pour modéliser le comportement de l'humain, nous avons opté pour une version de notre agent MB-seul possédant un modèle de transition complet.

Comme expliqué dans le chapitre 4, ce travail de recherche s'inscrit entre autres dans le corpus scientifique s'intéressant aux liens entre l'apprentissage par renforcement et le comportement animal. Puisque nos experts model-based et model-free prennent respectivement inspiration des comportements dirigés par un but et des habitudes comportementales (Daw et al. 2005), deux types de comportement que l'on retrouve chez l'être humain (Dolan et Dayan 2013), nous considérons pouvoir modéliser le comportement de notre humain simulé de cette manière. Concrètement, nos différents robots simulés vont donc coopérer avec un humain qui n'a d'humain que le nom, et qui est en tout point de vue modélisé comme ces robots.

Finalement, la seule chose qui différencie ces deux types d'agents décisionnels est le fait que l'humain possède un modèle de transition complet alors que le robot doit construire le sien de manière itérative. Nous considérons en effet que si le robot doit dans un premier temps apprendre les conséquences de ses actions durant la phase de babillage, l'humain sait par exemple déjà que lorsqu'il prend le cube rouge depuis sa boîte, le cube se situe désormais dans sa main.

8.1.4 Phase pré-expérimentale de babillage

Une période de babillage, où le robot et l'humain peuvent manipuler les cubes, mais où aucune récompense ne peut être gagnée, précède l'expérience. Nous avons choisi d'ajouter cette phase pré-expérimentale pour les mêmes raisons que celles évoquées dans le chapitre 7. Cette fois-ci en revanche, plutôt que d'évaluer les performances du robot utilisant notre critère d'arbitrage (MC-EC) à différentes durées de babillage, nous les évaluons à différents pourcentages de transitions explorées (Fig.8.2). Nous choisissons un pourcentage d'exploration de 80% (courbe jaune) pour la première paire d'objectifs et un pourcentage d'exploration de 70% (orange) pour la seconde. Ces valeurs correspondent à celles au-dessus desquelles continuer à explorer ne permet plus d'accumuler plus rapidement de la récompense au cours du temps. À nouveau, nous pourrions choisir de donner au robot un modèle de transition plus ou moins complet avant le début de l'expérience ou de réutiliser le modèle de transition construit par le robot avant la première expérience pour toutes celles qui suivront, dans le cas d'expériences réelles où le temps n'est pas une ressource illimitée.

8.1.5 Paramétrage des experts

Nous réutilisons à nouveau le même jeu de paramètres que celui utilisé dans la tâche de navigation et celle d'interaction humain-robot pour chacun des experts et pour le méta-contrôleur (tableau 8.1), afin de montrer le caractère générique et tâche-indépendant de notre système d'apprentissage et de méta-contrôle. Les paramètres de l'humain simulé sont identiques à ceux du robots.

TABLE 8.1 – *Valeurs choisies des paramètres des experts et du méta-contrôleur dans la tâche de rangement de cubes en coopération avec un humain.*

Param	MB	MF	MC
α	n.a.	0,6	n.a.
τ	0,02	0,02	0,02
γ	0,9	0,9	n.a.
κ	n.a.	n.a.	7.0

Les valeurs d'état-action des experts et de l'humain valent à nouveau 0.0 en début d'expérience.

8.2 RÉSULTATS

Pour évaluer les performances des robots virtuels, nous réutilisons le code couleur des expériences précédentes : le rouge pour le robot MF-seul, le bleu pour le robot MB-seul, le vert pour le robot de coordination aléatoire (MC-RnD) et le violet pour le robot qui coordonne les deux experts en utilisant le critère d'arbitrage que nous avons proposé (MC-EC).

L'intérêt de cette expérience est d'évaluer l'apport du méta-contrôle dans une tâche où un robot doit obligatoirement coopérer avec un humain pour progresser, mais aussi de pousser notre architecture dans ses retranchements.

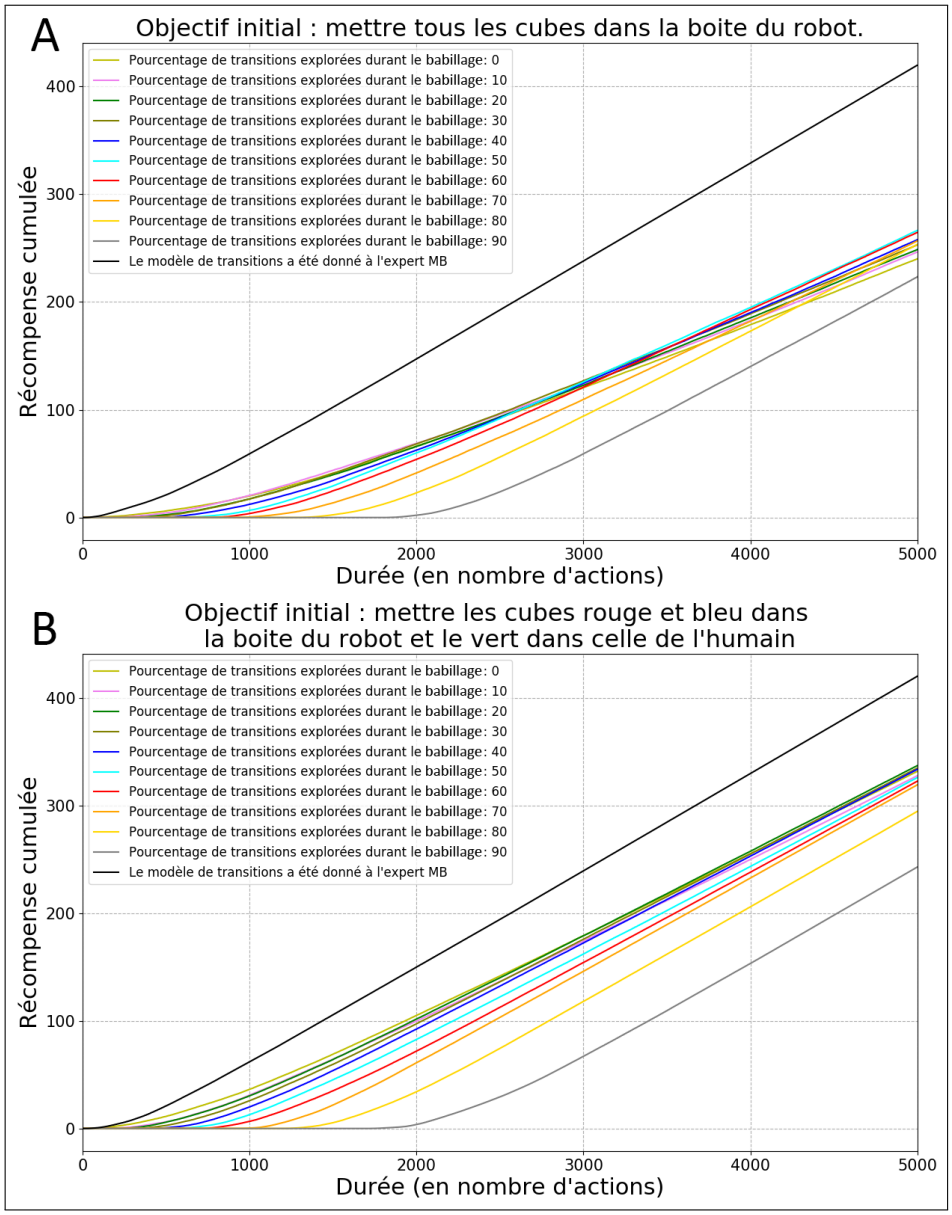

FIGURE 8.2 – ***A***. *Performance moyenne du robot MC-EC pour différents pourcentages de transitions explorées durant la phase de babillage et pour la première combinaison d'objectifs.* ***B***. *Performance moyenne du robot MC-EC pour différents pourcentages de transitions explorées durant la phase de babillage et pour la deuxième combinaison d'objectifs. Pour chaque valeur de pourcentage, 50 expériences simulées ont été réalisées. La performance est définie comme la capacité du robot à accumuler de la récompense sur la durée de l'expérience. La durée est représentée par le nombre d'actions effectuées par le robot.*

8.2.1 Quand le partenaire devient adversaire

Nous pouvons voir qu'avec la première paire d'objectifs, durant la première phase de l'expérience, la performance du robot MC-EC égalise à nouveau celle du robot MB-seul (Fig.8.3.A), pour un coût computationnel divisé par trois (Fig.8.3.B). Cela ne dure malheureusement pas. En effet, dès lors que l'objectif change, le robot MC-EC n'arrive plus à accumuler autant de récompense que le robot MC-seul, et se fait même rattraper par le robot MF-seul, jusqu'alors considéré comme le moins performant. Nous observons exactement la même chose avec la seconde paire d'objectifs (Fig.8.4.A et B). Dans les expériences précédentes, jamais nous n'avions fait face à une telle chute de performance du robot MC-EC. Pour l'expliquer, nous devons nous pencher sur ce qu'il se passe exactement à la 5000ème itérations.

Pour le robot et l'humain, la 5000ème itération n'est qu'une itération comme une autre : l'objectif change sans qu'ils ne soient mis au courant. Ne sachant pas que l'objectif a changé, les deux partenaires vont continuer à se passer les cubes comme si de rien n'était. Lorsqu'ils arrivent finalement par exemple à mettre tous les cubes dans la boîte du robot (dans le cas de la première paire d'objectifs), aucune récompense ne leur est délivrée et leurs modèles de récompense R sont donc modifiés en conséquence. Suite à cela, dès lors que les processus d'inférence des experts MB du robot MC-EC et de l'humain vont être activés, les valeurs d'état-action des experts MB vont être réinitialisées à 0.0 via l'action naturelle de l'algorithme de programmation dynamique *Value Iteration* 2.2.

Or, avant la 5000ème itération, nous pouvons voir que le comportement du robot MC-EC est majoritairement dirigé par l'expert MF (Fig.8.3.C et Fig.8.4.C), qui n'est quant à lui pas capable de réinitialiser d'une traite ses valeurs d'état-action. En effet, il lui faudra de nombreuses itérations et passages par les états menant à l'état récompensé pour que les valeurs d'état-action diminuent suite à l'absence de récompense. Le problème est donc le suivant : après s'être rendu compte que l'objectif a changé, l'humain va se remettre à explorer l'environnement afin de trouver le nouvel état récompensé, voir essayer de remplir le nouvel objectif s'il réussit à le découvrir, tandis que le robot MC-EC, dont le comportement est dirigé à ce moment de l'expérience majoritairement par son expert MF, va quant à lui continuer de tenter de réaliser le premier objectif. Ici, le partenaire devenu adversaire met en lumière une différence algorithmique dont nous avions déjà observé l'effet dans l'expérience de navigation du chapitre 6.

En effet, cette incapacité de l'expert MF à réinitialiser ses valeurs d'état-action de la même manière que l'expert MB était à l'origine d'un "pic" de la probabilité de sélection de l'expert MF (Fig.6.7.A) qui corrélait avec le très léger retard d'accumulation de récompenses que le robot MC-EC prenait sur le robot MB-seul (Fig.6.6.A). Pour rappel, notre critère d'arbitrage est un compromis entre le coût du processus d'inférence et la qualité de l'apprentissage définie comme l'entropie de la distribution des probabilités de sélection d'actions. Concrètement, plus les valeurs d'état-action d'un état sont proches les unes des autres, plus l'entropie sera grande, et plus la qualité d'apprentissage sera donc faible. Lorsque l'expert MB réinitialise ses valeurs d'état-action, il réinitialise aussi sa qualité d'apprentissage.

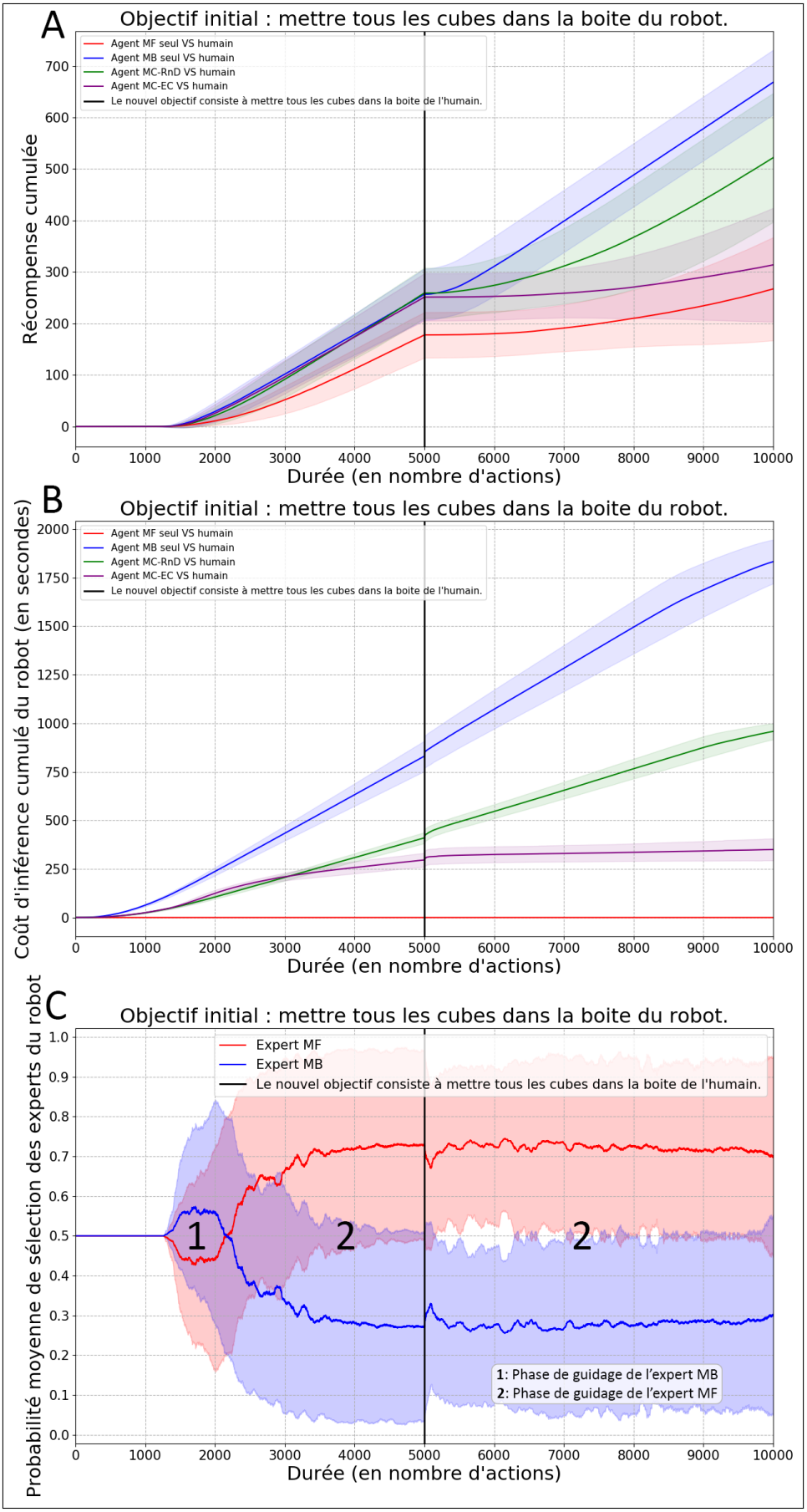

FIGURE 8.3 – ***A***. *Performance moyenne pour 50 expériences simulées.* ***B***. *Coût moyen de calcul pour 50 expériences simulées.* ***C***. *Probabilité moyenne de sélection des experts par le méta-contrôleur du robot MC-EC pour 50 expériences simulées. Nous utilisons l'écart-type comme indicateur de dispersion dans les trois figures.*

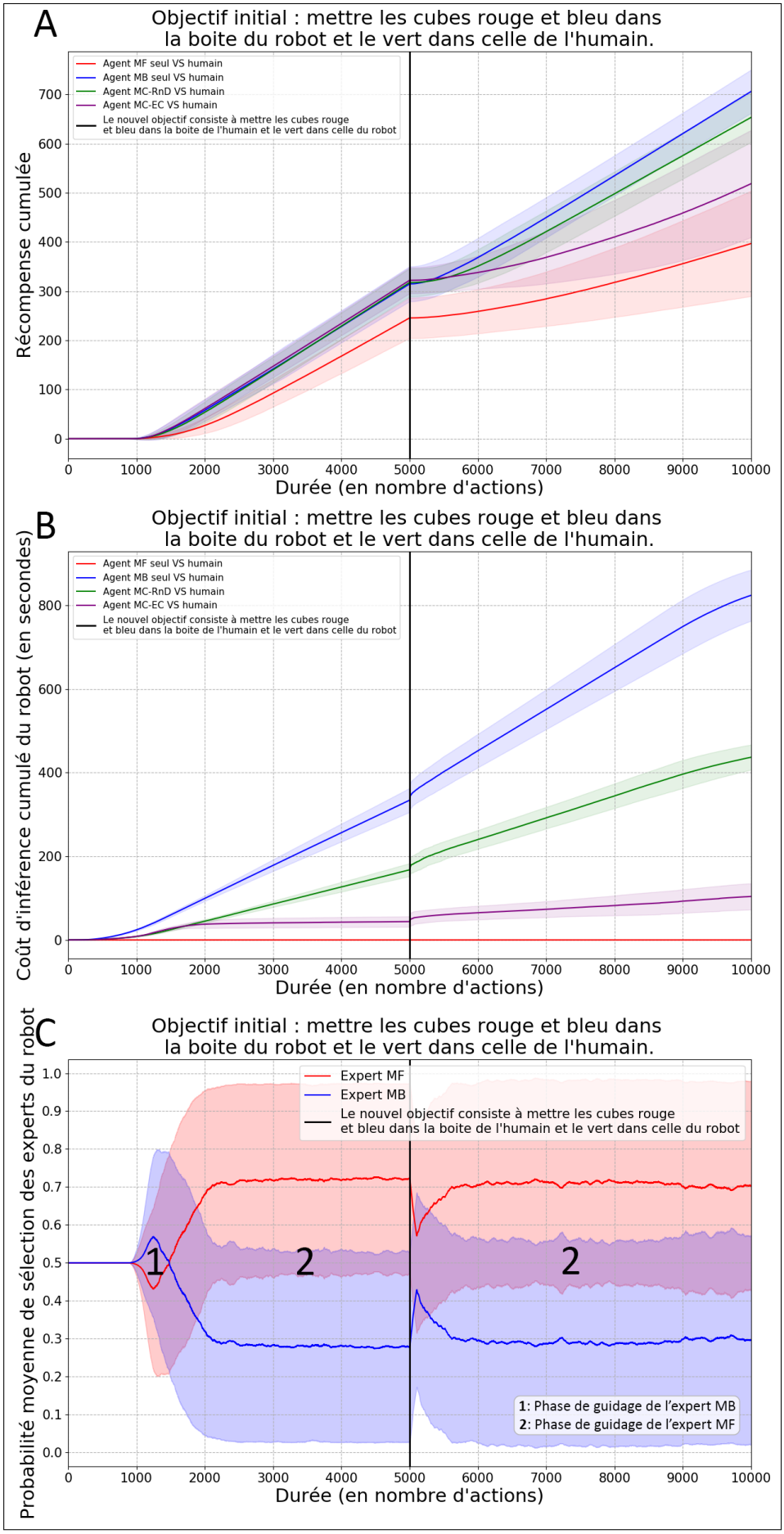

FIGURE 8.4 – ***A***. *Performance moyenne pour 50 expériences simulées.* ***B***. *Coût moyen de calcul pour 50 expériences simulées.* ***C***. *Probabilité moyenne de sélection des experts par le méta-contrôleur du robot MC-EC pour 50 expériences simulées. Nous utilisons l'écart-type comme indicateur de dispersion dans les trois figures.*

L'expert MF n'étant pas capable de le faire, il deviendra de facto l'expert ayant la meilleure qualité d'apprentissage, et donc l'expert contrôlant le comportement du robot, alors que le comportement judicieux serait justement de ne plus jouer le premier objectif.

Dans les deux expériences, l'observation est donc la même : si le changement environnemental implique une modification des modèles de récompense des experts MB, la dissymétrie algorithmique des experts MF et MB donne lieu à une période où l'expert MF dirige davantage le comportement du robot qu'il ne le devrait. Si cela n'empêchait pas le robot de conserver une bonne performance dans la tâche de navigation, l'expert MB n'est ici même plus capable de reprendre la main sur le comportement du robot (Fig.8.3.C et Fig.8.4.C) et reste donc coincé en phase 2 de guidage de l'expert MF.

A noter que, comparé à l'expérience de navigation du chapitre 6, nous n'observons pas ici la phase exploratoire de l'expert MF. Pour rappel, l'existence de cette phase était due à la différence de méthodes d'apprentissage des deux experts, à l'origine du fait que les valeurs d'état-action de l'expert MF diminuaient légèrement plus que celles de l'expert MB. Ici, les valeurs d'état-action des experts étant initialisées à 0.0 en début d'expérience, et non pas à 1.0 comme dans l'expérience de navigation, cet effet de la dissymétrie algorithmique n'est pas observé.

8.2.2 Détection de changements de contextes

Pour contrer ce problème, nous avons doté notre robot d'un mécanisme lui permettant de détecter automatiquement les changements d'objectifs en tenant compte uniquement de l'évolution de ses modèles de valeurs d'état-action. Pour ce faire, nous nous sommes appuyés sur la *similarité cosinus*, un outil mathématique permettant d'évaluer la similarité de deux vecteurs à n dimensions en déterminant le cosinus de leur angle. Généralement utilisé en tant que mesure de ressemblance entre deux documents, nous l'utilisons ici pour mesurer la ressemblance entre deux vecteurs de valeurs d'état-action :

$$cos(\theta) = \frac{A.B}{\|A\| \, \|B\|} \tag{8.1}$$

où A est le vecteur de valeurs d'état-action de l'état précédent avant sa mise-à-jour par l'expert MB et B est le vecteur de valeurs d'état-action de l'état précédent après sa mise-à-jour par l'expert MB. À noter que les valeurs des vecteurs ont toutes été multipliées par 100 et les valeurs nulles ont été remplacées par des valeurs très petites pour éviter la division par 0. Si les deux vecteurs sont identiques, θ vaut 1.

Cette idée nous est venue d'une étude précédemment réalisée (Caluwaerts et al. 2012b), où la mesure de la *similarité cosinus* était effectuée sur des vecteurs contenant des valeurs de diffusion de l'activité d'un graphique topologique représentant le modèle de transition de l'expert MB. Concrètement, ici, à chaque fois qu'il réalise son processus d'inférence, l'expert MB calcule la *similarité cosinus* θ de l'état précédent avant et après sa mise-à-jour, et la compare à un seuil. Si la valeur est inférieure à ce seuil, l'expert MB envoie alors un signal supplémentaire au méta-contrôleur

(flèche t1 de la figure 5.1), qui se chargera d'envoyer un signal à l'expert MF pour lui indiquer de remettre à jour son modèle d'état-action (flèche t2). Si nous souhaitions idéalement que la réinitialisation des valeurs d'état-action de l'expert MF ait lieu uniquement après la découverte de l'absence de l'ancienne récompense, nous pouvons en réalité dénombrer trois moments où la valeur de θ va forcément diminuer du fait de mises à jour des valeurs d'état-action :

— le moment où le robot trouve la récompense pour la première fois. Dans ce cas-ci, réinitialiser les valeurs d'état-action de l'expert MF ne pose pas de problème, puisque toutes sont déjà nulles.
— le moment où le robot se rend compte que l'état récompensé a changé, c'est-à-dire le moment qui nous intéresse.
— le moment où le robot trouve la nouvelle récompense pour la première fois. Dans ce cas-ci, réinitialiser les valeurs d'état-action de l'expert MF ne pose à nouveau pas de problèmes, puisqu'elles ont déjà toutes été réinitialisées précédemment.

Au final, plus qu'un mécanisme lui permettant de détecter automatiquement un changement d'objectif, la *similarité cosinus* permet aussi au robot de détecter l'apparition d'un nouvel objectif : c'est donc un mécanisme de détection de changements de contextes, comme le souligne Caluwaerts et al. (2012b). Dans notre algorithme, lorsque le robot découvre que l'état récompensé ne rapporte plus de récompense, les valeurs d'état-action de son expert MF sont réinitialisées. À la place, nous pourrions tout a fait lui permettre de les stocker en mémoire, afin de pouvoir potentiellement les réutiliser si l'état anciennement récompensé le redevient plus tard au cours de l'expérience, ce qui n'est pas le cas ici.

Bien sûr, la fonctionnalité du mécanisme dépend du seuil auquel la valeur de la *similarité cosinus* θ sera comparée. Pour le définir, nous avons regardé sur 200 simulations la valeur de la *similarité cosinus* à l'itération qui succède celle où le robot atteint l'état anciennement récompensé pour la première fois. θ était dans 100% des cas inférieure ou égale à 0.611 pour la première paire d'objectifs (100 simulations), et à 0.706 pour la seconde (100 simulations). Nous avons choisi un seuil commun de 0.7. Dans la figure 8.5, nous pouvons observer un histogramme des effectifs des différentes valeurs obtenues de θ pour une expérience de chacune des paires d'objectifs. Dans les deux cas, nous pouvons voir que la quasi totalité des valeurs de θ valent 1.0, signe que durant la quasi totalité de l'expérience, les valeurs d'état-action de l'expert MB n'évoluent pas. Dans la figure 8.5.A, nous pouvons compter un effectif de 4 valeurs de θ inférieures à 0.7 (3 de 0.558 et 1 de 0.611), et dans la figure 8.5.B, un effectif de 5 valeurs de θ inférieures à 0.7 (2 de 0.61 et 1 de 0.666). Dans les deux cas, cela correspond donc à plus de valeurs que les 3 moments que nous avons définis précédemment comme étant les moments où le contexte change effectivement (découverte de récompense, découverte de la disparition de la récompense, découverte de la nouvelle récompense). Cela signifie que par moments, les valeurs d'état-actions de l'expert MB évoluent sans que cela ne soit lié à un changement de contexte, mais par exemple plutôt à la découverte d'un nouvel état non encore exploré. En fonction du seuil défini, le robot peut donc prendre cette mise-à-jour "brutale" pour un change-

ment de contexte et réinitialiser les valeurs d'état-action de l'expert MF alors qu'il ne devrait pas.

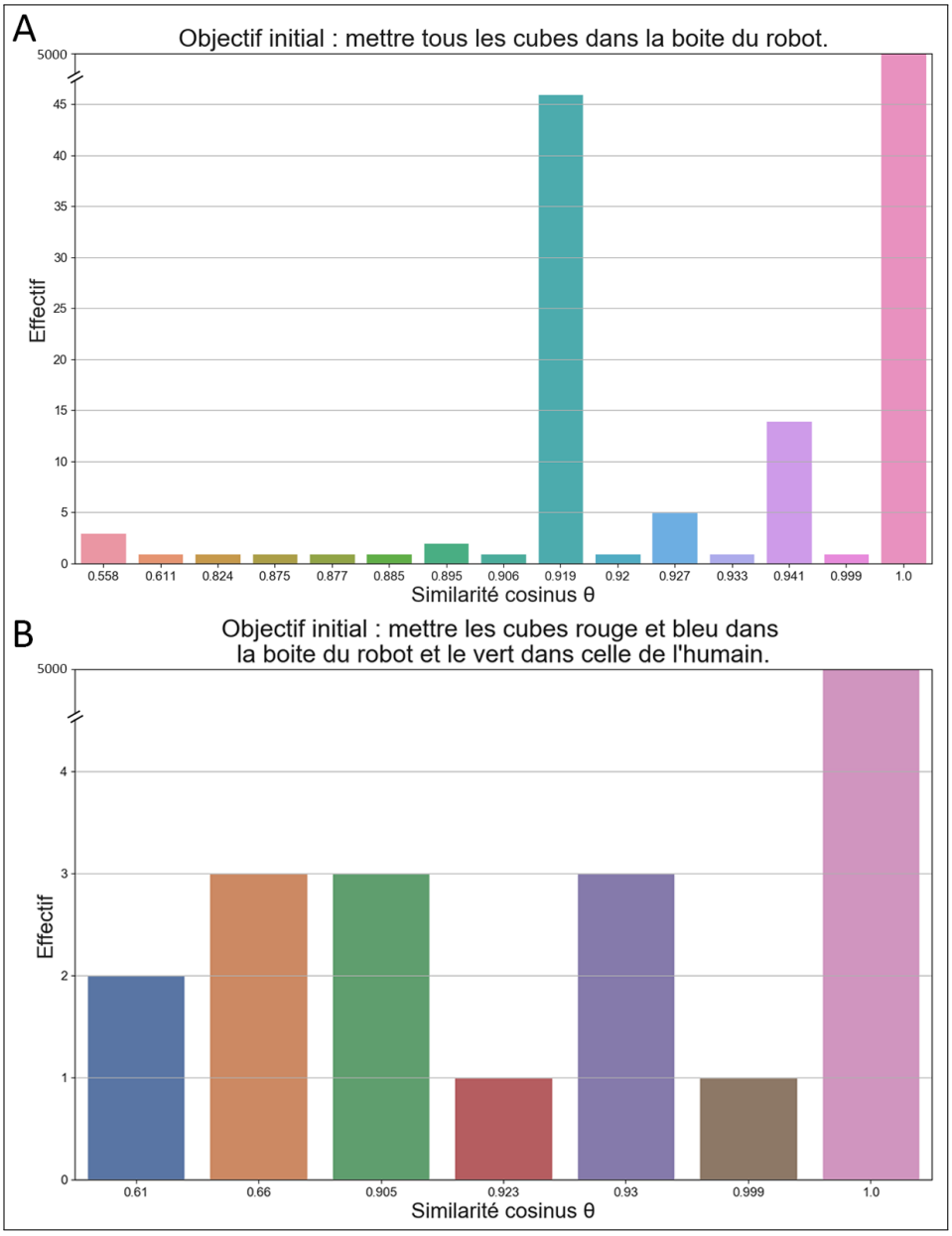

FIGURE 8.5 – ***A***. *Histogramme des effectifs des valeurs de similarité cosinus θ pour une expérience longue de 10 000 itérations avec la première paire d'objectifs.* ***B***. *Histogramme des effectifs des valeurs de similarité cosinus θ pour une expérience longue de 10 000 itérations avec la seconde paire d'objectifs. Le robot et l'humain jouant au tour par tour, cela fait donc un total de 5000 valeurs de θ par expérience.*

Pour autant, cela ne semble pas avoir un quelconque effet négatif sur les performances du robot, signe que cela arrive rarement. En effet, nous pouvons voir dans les figures 8.6.A et 8.7.A qu'avec le mécanisme de détection de changements de contextes et un seuil de 0.7, la performance du robot MC-EC est désormais identique à celle du robot MB-seul. Dans les figures 8.6.B et 8.7.B, nous pouvons voir que juste après le changement d'objectif, le coût computationnel du processus d'inférence augmente, signe que l'expert MB prend la main sur le comportement du robot

pour lui permettre de mieux faire face au changement environnemental. Les figures 8.6.C et 8.7.C confirment cela avec la réapparition des secondes phases de guidage de l'expert MB, absentes des expériences réalisées sans le mécanisme de détection de changements de contexte.

8.3 Conclusion

Dans ce chapitre, nous avons évalué notre modèle de coordination d'experts d'apprentissage dans une tâche de coopération humain-robot simulée. Dans cette nouvelle tâche, le robot doit coopérer activement avec l'humain pour mener à bien ses objectifs. L'humain n'est plus simplement présent pour aider le robot à améliorer sa performance, mais devient un véritable partenaire sans qui les objectifs ne peuvent être réalisés. À nouveau, nous avons réutilisé les paramètres optimisés pour la tâche de navigation, afin de montrer le caractère générique et tâche-indépendante de notre système d'apprentissage et de méta-contrôle. Dans ce chapitre, le robot est confronté à un problème déjà observé dans la tâche de navigation, mais qui ne l'empêchait jusqu'alors pas de progresser : l'incapacité de l'expert MF à réinitialiser ses valeurs d'état-action après le changement d'objectif par rapport à l'expert MB. Ici, du fait de la présence d'un humain n'étant pas impacté par ce problème, les deux partenaires peuvent un temps devenir adversaires, ce qui entraîne une chute drastique de la performance du robot. Ici, l'humain n'est pas le problème, mais simplement le révélateur. Pour contrer cela, nous avons donc développé un mécanisme de détection de changements de contexte, permettant au robot de réinitialiser automatiquement les valeurs d'état-action de son expert MF lorsque nécessaire. Avec ce mécanisme, le robot utilisant notre critère d'arbitrage obtient à nouveau le même niveau de performance que celui d'un robot contrôlé uniquement par un algorithme d'apprentissage model-based, tout en réduisant drastiquement son coût calculatoire.

Finalement, nous illustrons à nouveau avec cette nouvelle tâche de coopération humain-robot le caractère générique et tâche-indépendant de notre modèle de coordination et proposons une solution efficace et peu coûteuse lui permettant de contourner un problème pouvant survenir lorsque l'objectif change en cours de tâche, le rendant ainsi encore plus robuste.

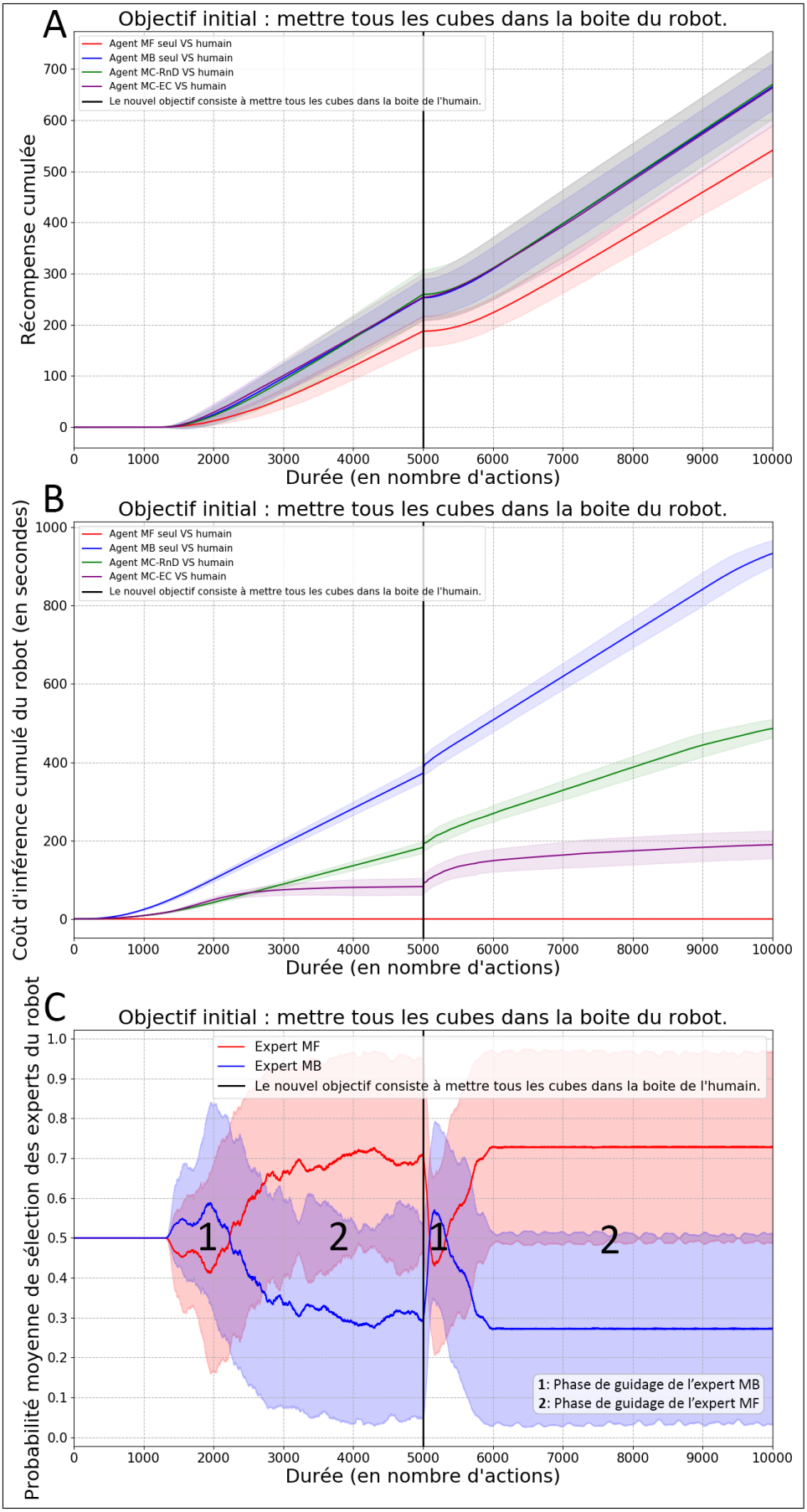

FIGURE 8.6 – ***A***. *Performance moyenne pour 50 expériences simulées.* ***B***. *Coût moyen de calcul pour 50 expérience simulées.* ***C***. *Probabilité moyenne de sélection des experts par le méta-contrôleur du robot MC-EC pour 50 expériences simulées. Nous utilisons l'écart-type comme indicateur de dispersion dans les trois figures. Dans ces expériences, les robots sont capables de détecter les changements de contextes.*

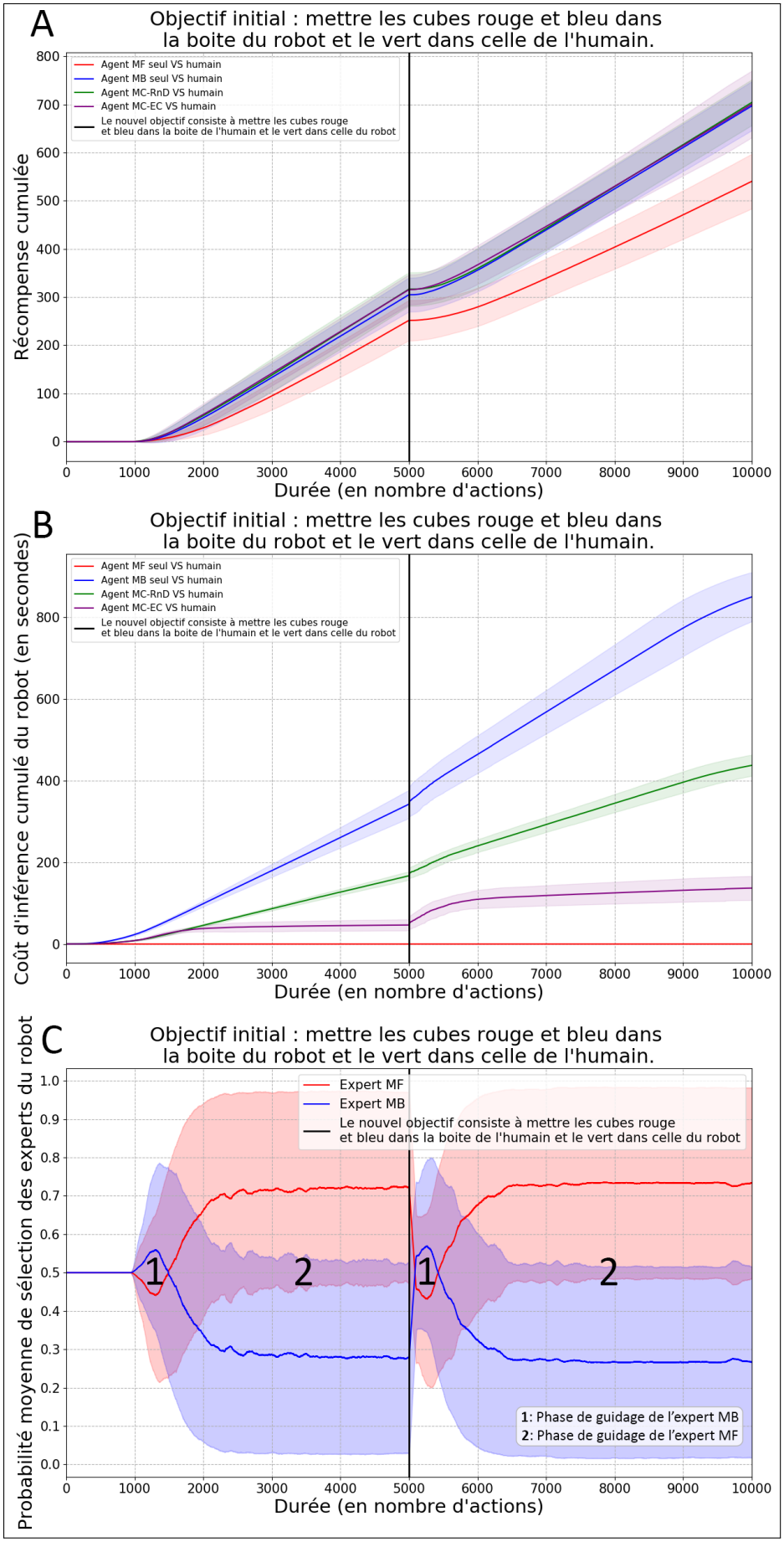

FIGURE 8.7 – ***A***. *Performance moyenne pour 50 expériences simulées.* ***B***. *Coût moyen de calcul pour 50 expérience simulées.* ***C***. *Probabilité moyenne de sélection des experts par le méta-contrôleur du robot MC-EC pour 50 expériences simulées. Nous utilisons l'écart-type comme indicateur de dispersion dans les trois figures. Dans ces expériences, les robots sont capables de détecter les changements de contexte.*

Discussion et perspectives

9

Sommaire

9.1 Résumé des contributions

Dans ce manuscrit, nous nous sommes intéressés à la manière dont nous pouvions doter un robot de la capacité à exprimer des habitudes comportementales en complémentarité de sa capacité à planifier pour agir. Ces habitudes comportementales avaient pour objectif de remplacer la planification coûteuse en ressources calculatoires dans le cas où le robot connaissait suffisamment son environnement pour s'en passer. Pour combiner ces deux manières d'agir, nous avons développé un modèle de coordination permettant au robot de changer de comportement dynamiquement en fonction de sa propre performance et du contexte environnemental. Pour modéliser ces deux manières d'agir, nous nous sommes inspirés de la dichotomie entre les *habitudes comportementales* et les *comportements dirigés vers un but*, mise en lumière par de nombreuses études physiologiques et computationnelles du comportement des mammifères. Ce modèle est intégré à une architecture de contrôle robotique dont la couche décisionnelle combine des algorithmes d'apprentissage *model-free* et *model-based*, modélisant respectivement les *habitudes comportementales* et les *comportements dirigés vers un but*.

Pour évaluer la capacité de cette architecture robotique à permettre au robot d'accumuler efficacement de la récompense au cours du temps en modulant dynamiquement son comportement, nous avons effectué trois tâches expérimentales différentes :

— une tâche de navigation en environnement réel, où le robot doit apprendre à localiser la position de la récompense dans une arène (Dromnelle et al. 2020b). L'environnement est non stationnaire,

dans le sens où la position de la récompense peut changer au cours de l'expérience et où des murs peuvent être rajoutés ;
— une première tâche simulée de rangement, où le robot doit apprendre à placer des cubes dans des boîtes (Dromnelle et al. 2020a). Ici, le robot peut interagir avec un humain virtuel de plusieurs manières différentes. Si l'humain peut aider le robot, il peut aussi oublier d'interagir ou faire des erreurs ;
— une seconde tâche simulée de rangement, où le robot doit apprendre à placer des cubes dans des boîtes (Dromnelle et al. 2020a). Ici, le robot doit obligatoirement coopérer avec l'humain pour mener à bien ses objectifs, n'ayant pas accès à certaines parties de l'environnement. Comme pour la tâche de navigation, l'objectif peut changer au cours de la tâche.

Grâce à ces différentes expériences, nous avons montré que notre méthode de coordination de comportements permettait au robot de maintenir une performance optimale en termes de performance tout en diminuant grandement le coût de calcul. Notre travail s'inscrivant dans la recherche et le développement de robots autonomes, le souci de l'économie calculatoire, et donc de l'autonomie, argumente en faveur de notre méthode. Cette performance à faible coût s'explique par la formation répétée d'un schéma temporel de coordination alternant entre trois phases distinctes tirant bénéfices des avantages de chacun des comportements. Nous avons aussi montré que cette méthode permet au robot de faire face à des changements brusques de l'environnement, des changements d'objectifs ou de comportement de partenaires humains dans le cas des tâches d'interaction humain-robot. Finalement, les robots de chacune de ces expériences, qu'ils soient réels ou virtuels, ont tous utilisé le même jeu de paramètres, montrant ainsi la généralité de notre méthode. Pour autant, nous n'affirmons pas avoir trouvé un jeu de paramètre universel qui donnerait de bonnes performances dans n'importe quelle tâche expérimentale. Ce caractère générique est sans nul doute conditionné par la manière dont nous représentons et abstrayons le monde en états, qui est identique dans les trois expériences.

Pour finir, il est important de rappeler qu'à l'heure de la crise climatique, il est de notre responsabilité de réfléchir au développement d'IA et de robots plus économiques, et donc plus écologiques. Si là n'était pas l'objectif premier de cette thèse, ce travail s'inscrit malgré tout dans cette dynamique.

9.2 Discussion et perspectives

9.2.1 Nécessité et difficultés de l'abstraction

Donner le contrôle d'un algorithme de décision à un agent évoluant dans un environnement réaliste pose de nombreux problèmes, bien définis en robotique, mais peu considérés dans les modèles du comportement des mammifères. Lorsqu'un modélisateur souhaite formaliser la tâche expérimentée par un animal sous la forme d'un MDP, il doit avant tout observer son comportement, puis définir un modèle de l'animal selon ces observations et à la lumière de ce MDP. Ici, le modèle est une abstrac-

tion de la réalité. À l'inverse, le roboticien fait face à des problèmes bien réels, où un robot évolue dans un environnement continu avec pour seules informations celles reçues par ses capteurs. Ces informations permettent généralement au robot de définir de façon simplifiée l'*état* dans lequel il se trouve.

Du point de vue de la délibération abstraite, cet *état* est une structure symbolique dénuée de sens et manipulée par le robot. Ce dernier ne reconnaît que leur forme, tout sens étant attribué par l'extérieur (le chercheur, la théorie, l'utilisateur). Dans Searle (1980), l'auteur argumente qu'un système incapable de remplir les symboles de sens peut malgré tout engendrer un comportement apparemment intelligent. C'est à cette occasion que le "problème de l'ancrage des symboles" est défini : peut-on réduire la cognition à la manipulation de symboles fondée sur leur forme ? Pour Harnad (1990), cela est le cas dès lors que l'on dote le robot de capacités à engendrer des discriminations et des identifications d'*input*. La capacité de discrimination permet de juger si deux *inputs* sont différents et à quel degré, tandis que la capacité d'identification permet d'attribuer un nom à un *input* et ainsi de dire de quelle catégorie il fait partie. Grâce à cela, le robot est capable de construire lui-même son ancrage et ainsi de manipuler des symboles abstraits dénués de sens.

Dans l'expérience de navigation du chapitre 6, nous avons doté un robot de la possibilité de placer de manière autonome des centres régulièrement espacés sur une carte topologique de l'environnement, construite à la volée grâce à un algorithme de SLAM (6.2). Nous considérions que l'état dans lequel se trouvait le robot était celui associé au centre le plus proche de sa position lorsqu'il avait terminé de réaliser son déplacement. L'environnement ainsi discrétisé en états, le robots pouvait utiliser ses algorithmes d'apprentissage par renforcement. Pourtant, l'application de ces algorithmes supposait que la *Propriété de Markov* définie dans le chapitre 2 tienne : toute information pour prendre une décision est contenue dans l'état précédent. Or, cette propriété n'était pas forcément toujours vérifiée. En effet, si effectuer l'action 0 dans l'état 19 emmenait généralement le robot dans l'état 18 récompensé, il arrivait par moment que celui-ci érafle le mur du bas et enclenche ses capteurs de contact, stoppant ainsi son déplacement : obtenir la récompense n'était donc jamais totalement assurée. Pour coller au mieux à la propriété, nous pourrions peut-être définir autrement ce qu'est un état, en nous appuyant par exemple sur les transitions, même s'il est probable que cette définition soit elle-même insuffisante à certains autres endroits de l'espace.

Toujours à propos de l'expérience de navigation, nous rappelons qu'il a fallu nous y reprendre à plusieurs reprises afin d'obtenir une carte de la qualité de celle présentée dans la figure 6.2. Or, la qualité de cette carte est très importante, puisque la performance du robot dépend directement de sa capacité à interpréter les états dans lesquels il se trouve. Une carte ne ressemblant pas assez à l'environnement réel, ou dont les murs se situent à cheval entre plusieurs états, sera nécessairement à l'origine de difficultés pour le robot. Notamment, comme dans Caluwaerts et al. (2012a), cela aurait pour conséquence d'amoindrir les performances de l'expert MB, et donc de changer potentiellement le schéma de coordination temporelle que nous avons observé. N'oublions pas non plus que si le robot est ca-

pable de générer lui-même de nouveaux états au gré de son exploration, il ne choisit pas leur taille, qui est définie par l'expérimentateur. Une perspective intéressante pourrait être d'améliorer notre méthode automatique de discrétisation de telle sorte que le robot puisse définir lui-même la taille des cellules en fonction de l'environnement (petites cellules dans les zones tortueuses, cellules plus grandes dans les zones ouvertes).

Mieux encore, nous pourrions passer à une représentation à plusieurs niveaux d'échelle, comme proposée par Llofriu et al. (2015). Dans cette étude, les auteurs s'intéressent à la nature multi-échelles des *cellules de lieu* et montrent qu'elles contribuent à un apprentissage plus rapide pendant la navigation dirigée vers un but d'un robot, par rapport à un système de navigation spatiale composé de cellules à échelle unique, comme le nôtre. Les *cellules de lieu* sont des neurones de l'hippocampe dont le champ récepteur est défini par une zone spatiale donnée sur une carte cognitive, qui est une carte mentale de l'environnement (O'keefe et Nadel 1978). Chaque *cellule de lieu* s'active donc quand l'animal se trouve dans le champ récepteur qui lui est associé, de telle sorte qu'elles forment collectivement une représentation cognitive de tout l'espace d'états de la tâche.

Les progrès récents de l'apprentissage profond offrent également une piste intéressante (Mnih et al. 2015a, Silver et al. 2016, Senior et al. 2020). En effet, 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, qui nécessite généralement un grand nombre d'exemples, ce qui est souvent difficile à obtenir dans une tâche robotique réelle, comme nous l'avons montré dans le chapitre 6.

Discuter de la manière dont nous pourrions améliorer notre système de représentation d'états, en permettant au robot de moduler dynamiquement la discrétisation de son environnement, nous amène à nous rappeler de travaux que nous avons présentés dans le chapitre 4, (Dezfouli et Balleine 2012, Dezfouli 2015, Balleine et O'Doherty 2010), où l'intérêt est porté sur la manière dont les actions peuvent être représentées. Dans ces études, les auteurs définissent la formation d'habitudes comportementales comme la capacité de créer des séquences d'actions : lorsque, depuis un état initial, le robot effectue la première action d'une séquence, toutes les actions suivantes sont ensuite réalisées automatiquement, lui permettant de traverser plusieurs états jusqu'à ce que la séquence arrive à son terme. Ici, ce n'est non pas la taille de l'état qui est redéfinie, mais la direction et la portée de ce qui devient une *macro-action*. En revanche, en fonction de la complexité locale de l'environnement, fusionner les états traversés en un *macro-état* pourrait potentiellement avoir le même effet sur le robot que de rassembler les actions élémentaires en une *macro-action*.

Plus généralement, discuter de la notion d'action nous amène à réfléchir à la manière de définir des actions pertinentes pour le robot, qui lui permettent d'agir dans l'ensemble des situations auxquelles il est confronté. Plutôt que de les définir a priori, comme nous l'avons fait jusqu'alors, nous pourrions doter le robot de la capacité de former ses propres actions, de la même manière qu'il génère ses propres états dans l'expérience de navigation. Dans leurs travaux, Konidaris et Barto (2009) et Konidaris et al. (2010) proposent pour cela une approche d'apprentissage

par renforcement hiérarchique qui peut apprendre par démonstration ou à partir de la politique suivie par le robot. Ils évaluent suite à ça leur approche sur un 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, telles que les actions de manipulation (qui sont indépendantes de la pièce dans laquelle le robot apprend). Ce travail est d'ailleurs étendu à la génération d'actions en langage PDDL (*Planning Domain Description Language*), un langage permettant de standardiser les données d'entrée d'un planificateur (Konidaris et al. 2014). Récemment, les mêmes auteurs ont développé un système sur un robot physique qui apprend de manière autonome et directement à partir de données sensorimotrices (nuages de points, emplacements sur la carte et angles d'articulation) sa propre représentation symbolique d'une tâche de manipulation, dont il se sert ensuite pour planifier (Konidaris et al. 2018).

9.2.2 Amélioration de la couche décisionnelle

Actuellement, la couche décisionnelle de notre architecture de contrôle robotique est constituée de trois modules : un expert des habitudes comportementale (expert MF), un expert des comportements dirigés vers un but (expert MB) et un méta-contrôleur (MC), chargé d'arbitrer entre les propositions des deux experts, et donc de coordonner les deux types de comportements sous-jacents.

Une première piste d'amélioration de la couche décisionnelle pourrait être de la doter de la capacité de créer des séquences d'actions. D'ailleurs, cette nouvelle capacité ne remplacerait pas nécessairement l'algorithme d'AR direct de l'expert MF, comme le font Dezfouli et Balleine (2012), Dezfouli (2015) et Balleine et O'Doherty (2010). Nous pourrions plutôt imaginer un troisième expert capable de créer ou diviser des séquences d'actions. Les *macro-actions* formées seraient alors considérées de la même manière que les actions élémentaires par les experts MF et MB : effectuer une *macro-action* dans un état initial permet d'arriver dans un état terminal (sans prendre en compte les états intermédiaires traversés). Les experts MF et MB, suivant leur propre méthode d'apprentissage, pourraient ainsi attribuer une valeur d'état-action à ces *macro-actions*.

Ensuite, comme mentionné dans le chapitre 6, notre système ne dispose pas d'un expert spécialisé dans l'exploration. Actuellement, lorsque les valeurs d'état-action sont initialisées à une valeur supérieure à 0.0, et que le robot n'a encore jamais découvert de récompenses, le MC donne majoritairement le contrôle du comportement du robot à l'expert MF. Ceci est dû à la différence entre les méthodes d'apprentissage des deux experts, à l'origine du fait que les valeurs d'état-action de l'expert MF diminuent légèrement plus que celles de l'expert MB, ce qui entraîne une diminution plus prononcée de l'entropie de la distribution des probabilités de sélection d'actions, et donc une augmentation de la probabilité de sélection de la décision de l'expert MF. Si cela ne pose pas de problème en début d'expérience, puisque les deux experts affichent la même performance nulle et que l'expert MF est le moins coûteux des deux, l'impact est bien plus notable lorsque, après apprentissage, l'état récompensé change. En effet,

comme observé dans le chapitre 6, mais surtout dans le chapitre 8, dès lors que le changement environnemental implique une modification du modèle de récompense de l'expert MB (dû au fait que l'état récompensé a changé), la dissymétrie algorithmique des experts MF et MB donne lieu à une période où l'expert MF dirige davantage le comportement du robot qu'il ne le devrait. Si cela a peu d'impact sur la performance du robot dans la tâche de navigation du chapitre 6, l'effet est bien plus handicapant dans la tâche de coopération humain-robot du chapitre 8. Or, si notre couche décisionnelle était dotée d'un troisième expert spécialisé dans l'exploration, celui-ci prendrait justement le contrôle du comportement du robot dans ce type de circonstances, où le contexte vient de changer et où explorer l'environnement à la recherche du prochain objectif est le comportement le plus judicieux. Pour ce faire, nous pourrions nous inspirer de Caluwaerts et al. (2012b) et Dollé et al. (2018), qui combinent un algorithme de parcours de graphes et un algorithme qui apprend par AR direct à un expert spécialisé dans l'exploration aléatoire (e.g., choix au hasard des directions de déplacement du robot). D'après les auteurs, les avantages sont doubles : d'une part cela permet de ne pas cumuler l'exploration aléatoire des experts MB et MF, mais de les rendre déterministes et d'avoir un seul système d'exploration ; d'autre part, cela permet au méta-contrôleur de sélectionner explicitement l'exploration, plutôt que d'en faire quelque chose de sélectionné en interne à un expert, et ainsi même d'apprendre dans quelles situations l'exploration permet de sortir d'une impasse ou de compenser les défaillances des experts MB et MF.

Plutôt que de rajouter de nouveaux experts à notre couche décisionnelle, nous pourrions aussi améliorer les experts MF et MB déjà présents. Dans le chapitre 2, nous présentions les algorithmes *Dyna-Q* et *Prioritized Sweeping*, des méthodes à mi-chemin entre l'apprentissage direct et l'apprentissage indirect. Des méthodes permettant au robot de mettre à jour ses valeurs d'état-action en fonction de l'expérience passée, comme pourrait le faire un algorithme de *Q-learning*, tout en mettant à jour en parallèle son modèle de la tâche pour les couples d'état-action connus, comme pourrait le faire un algorithme de *Value Iteration*. Dans Aubin et al. (2018), les auteurs examinent si un algorithme de *Prioritized Sweeping* est capable d'expliquer le mécanisme de *replay* des *cellules de lieu* qui ont été activées lors d'expériences antérieures, que l'on observe dans l'hippocampe des rongeurs durant le sommeil. Ils montrent que l'architecture qu'ils proposent est capable d'améliorer l'apprentissage d'agents simulés confrontés à une tâche de navigation et prédisent que, chez les animaux, l'apprentissage des modèles de transition et de récompense devrait se produire pendant les périodes de repos. Plus récemment, nous avons commencé à étudier des versions alternatives de nos experts MF et MB inspirées de ces algorithmes dans le cadre des stages de Guillaume Pourcel puis de Jeanne Barthelemy. Si les résultats du premier stage ont montré que la version *Prioritized Sweeping* de l'expert MB dégradaient la performance du robot de navigation simulé, par rapport à la version originale, les résultats du second stage sont plus nuancés et prometteurs. Ici, l'ajout d'un mécanisme de *replay* à l'expert MF améliore sa performance pour un coup réduit, ce qui diminue de fait la probabilité de sélection de l'expert MB et change la dynamique de coordination (sans pour autant changer le profil de co-

ordination à trois phases), tandis que l'ajout d'un mécanisme de *replay* à l'expert MB améliore aussi sa performance, mais cette fois-ci au prix d'une nouvelle augmentation du coût de calcul. D'autres idées intéressantes ont été abordées au cours de ce stage, telle que celle de mettre à jour entre les phases expérimentales le modèle d'état-action de l'expert MF grâce aux connaissances de l'expert MB (*replay offline*, Momennejad et al. (2018)), ou la possibilités d'effectuer des *replay* partagés entre les experts MF et MB, qui auraient donc accès au même modèle d'états-actions, afin de les rendre encore plus coopérateurs.

Pour finir, nous pourrions tout simplement imaginer un nouveau critère de coordination permettant de nous passer du mécanisme de détection de changements de contextes défini dans le chapitre 8. Si de nombreux critères ont déjà été présentés dans le chapitre 4, nous n'avons pas encore évoqué le travail de Lee et al. (2014), qui met en évidence l'existence d'un mécanisme d'arbitrage qui répartit le degré de contrôle du comportement entre un expert MB et MF en fonction de la fiabilité de leurs prédictions respectives. Cette année, Panayi et al. (2021) ont ré-implémenté ce critère de coordination pour illustrer comment certains résultats expérimentaux d'inactivation du cortex orbitofrontal chez les rongeurs pouvaient être reproduits par un mécanisme d'arbitrage déficient entre MB, MF et exploration. Ce critère de coordination basé sur la fiabilité des prédictions de l'agent, plutôt que sur la qualité de son apprentissage, pourrait permettre au système de donner le contrôle du comportement à l'expert MB après un changement de contexte, du fait de la probable chute rapide de la fiabilité de l'expert MF.

9.2.3 Extension de la tâche de rangement de cubes

Initialement, nous avions prévu de répliquer la tâche d'interaction humain-robot du chapitre 7 avec de vrais sujets humains et un robot *Baxter*. Malheureusement, la situation sanitaire en 2020 nous a fait abandonné ce projet, au profit de la tâche de coopération humain-robot simulée du chapitre 8. Il aurait notamment été très intéressant d'observer le comportement du robot face à des humains aux comportements très variés (humains attentifs, distraits, qui récompensent, qui punissent, qui se comportent comme des antagonistes, etc). À cet égard, nous aurions pu imaginer doter la couche décisionnelle de la capacité d'apprendre un comportement moyen de l'humain, afin de permettre au robot d'adapter encore mieux son comportement, et d'être encore plus robuste à la variété comportementale (et donc environnementale).

9.2.4 Conclusion

Dans ce travail de recherche, nous n'avons fait qu'effleurer la multitude de questions qui s'ouvrent dès lors que l'on teste une architecture robotique capable de coordonner dynamiquement plusieurs stratégies comportementales. Pour autant, nous avons réussi à montrer l'intérêt d'utiliser une méthode de coordination, notamment dans le cadre de la recherche et développement en robotique réelle, où il n'y a pas de petites économies, et où la préservation de l'autonomie est une problématique centrale. Nous

avons aussi mis en avant un certain nombre de perspectives, qui, nous l'espérons, pourront permettre d'alimenter les futures études qui s'inscriront dans ce travail de recherche.

Bibliographie

Christopher D Adams. Variations in the sensitivity of instrumental responding to reinforcer devaluation. *The Quarterly Journal of Experimental Psychology Section B*, 34(2b) :77–98, 1982. (Cité page 44.)

Christopher D Adams et Anthony Dickinson. Instrumental responding following reinforcer devaluation. *The Quarterly Journal of Experimental Psychology Section B*, 33(2b) :109–121, 1981. (Cité page 44.)

Nassim Aklil, Alain Marchand, Virginie Fresno, E Coutureau, Ludovic Denoyer, Benoît Girard, et Mehdi Khamassi. Modelling rat learning behavior under uncertainty in a nonstationary multi-armed bandit task. Dans *Fourth Symposium on Biology of Decision Making (SBDM 2014)*, 2014. (Cité page 21.)

R. Alami, R. Chatila, S. Fleury, M. Ghallab, et F. Ingrand. An architecture for autonomy. *IJRR Journal*, 17 :315–337, 1998. (Cité page 63.)

Rachid Alami, Mathieu Warnier, Julien Guitton, Severin Lemaignan, et Emrah Akin Sisbot. When the robot considers the human. Dans *The 15th International Symposium on Robotics Research (ISRR)*, 2011. (Cité page 120.)

John R Anderson. *How can the human mind occur in the physical universe ?*, volume 3. Oxford University Press, 2009. (Cité page 63.)

JR Anderson. Rules of the mind. *New Jersey, Laurence Erlbaum Associates,, 319p*, 1993. (Cité page 63.)

Brenna Argall, Brett Browning, et Manuela Veloso. Learning by demonstration with critique from a human teacher. Dans *2007 2nd ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 57–64. IEEE, 2007. (Cité page 37.)

Brenna D Argall, Brett Browning, et Manuela M Veloso. Teacher feedback to scaffold and refine demonstrated motion primitives on a mobile robot. *Robotics and Autonomous Systems*, 59(3-4) :243–255, 2011. (Cité page 29.)

Ronald C Arkin. Motor schema—based mobile robot navigation. *The International journal of robotics research*, 8(4) :92–112, 1989. (Cité page 62.)

Ronald C Arkin. Integrating behavioral, perceptual, and world knowledge in reactive navigation. *Robotics and autonomous systems*, 6(1-2) :105–122, 1990. (Cité page 62.)

Ronald C Arkin et Tucker Balch. Aura : Principles and practice in review. *Journal of Experimental & Theoretical Artificial Intelligence*, 9(2-3) :175–189, 1997. (Cité page 62.)

Ronald C Arkin, Edward M Riseman, et Allen R Hanson. Aura : An architecture for vision-based robot navigation. Dans *proceedings of the DARPA Image Understanding Workshop*, pages 417–431. Morgan Kaufmann Inc. Los Altos, California, 1987. (Cité page 62.)

Lise Aubin, Mehdi Khamassi, et Benoît Girard. Prioritized sweeping neural dynaq with multiple predecessors, and hippocampal replays. Dans *Conference on Biomimetic and Biohybrid Systems*, pages 16–27. Springer, 2018. (Cité pages 19 et 138.)

B. W. Balleine et J. P. O'Doherty. Human and rodent homologies in action control : corticostriatal determinants of goal-directed and habitual action. *Neuropsychopharmacology*, 35 :48–69, 2010. (Cité pages 47, 136 et 137.)

Bernard W Balleine, Mauricio R Delgado, et Okihide Hikosaka. The role of the dorsal striatum in reward and decision-making. *Journal of Neuroscience*, 27(31) :8161–8165, 2007. (Cité page 45.)

Bernard W Balleine et Amir Dezfouli. Hierarchical action control : Adaptive collaboration between actions and habits. *Frontiers in Psychology*, 10 :2735, 2019. (Cité pages 55, 56 et 57.)

Bernard W Balleine et Anthony Dickinson. Goal-directed instrumental action : contingency and incentive learning and their cortical substrates. *Neuropharmacology*, 37(4-5) :407–419, 1998. (Cité page 45.)

Bernard W Balleine, A Simon Killcross, et Anthony Dickinson. The effect of lesions of the basolateral amygdala on instrumental conditioning. *Journal of Neuroscience*, 23(2) :666–675, 2003. (Cité page 45.)

Mathieu Barbier, Christian Laugier, Olivier Simonin, et Javier Ibañez-Guzmán. Probabilistic decision-making at road intersections : Formulation and quantitative evaluation. Dans *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 795–802. IEEE, 2018. (Cité page 75.)

Andrew G Barto, Richard S Sutton, et Charles W Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics*, (5) :834–846, 1983. (Cité page 18.)

Edward Beeching, Jilles Debangoye, Oliver Simonin, et Christian Wolf. Deep reinforcement learning on a budget : 3d control and reasoning without a supercomputer. Dans *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 158–165. IEEE, 2021. (Cité page 18.)

Richard Bellman. Dynamic programming. *Science*, 153(3731) :34–37, 1966. (Cité page 10.)

Richard E Bellman. *Adaptive control processes : a guided tour*. Princeton university press, 2015. (Cité page 16.)

Jean Bellot, Olivier Sigaud, et Mehdi Khamassi. Which temporal difference learning algorithm best reproduces dopamine activity in a multi-choice task ? Dans *International Conference on Simulation of Adaptive Behavior*, pages 289–298. Springer, 2012. (Cité page 47.)

Gregory S Berns et Terrence J Sejnowski. A computational model of how the basal ganglia produce sequences. *Journal of cognitive neuroscience*, 10 (1) :108–121, 1998. (Cité page 49.)

Kent C Berridge. The debate over dopamine's role in reward : the case for incentive salience. *Psychopharmacology*, 191(3) :391–431, 2007. (Cité page 47.)

Aude Billard, Sylvain Calinon, Ruediger Dillmann, et Stefan Schaal. Survey : Robot programming by demonstration. Rapport technique, Springrer, 2008. (Cité page 31.)

R. Peter Bonasso. Integrating reaction plans and layered competences through synchronous control. Dans *IJCAI*, pages 1225–1233, 1991. (Cité page 62.)

R. Peter Bonasso, R James Firby, Erann Gat, David Kortenkamp, David P Miller, et Mark G Slack. Experiences with an architecture for intelligent, reactive agents. *Journal of Experimental & Theoretical Artificial Intelligence*, 9(2-3) :237–256, 1997. (Cité page 63.)

Adriana Bono, Agnese Augello, Giovanni Pilato, Filippo Vella, et Salvatore Gaglio. An act-r based humanoid social robot to manage storytelling activities. *Robotics*, 9(2) :25, 2020. (Cité page 64.)

Thomas Boraud, Arthur Leblois, et Nicolas P Rougier. A natural history of skills. *Progress in neurobiology*, 171 :114–124, 2018. (Cité page 47.)

Y-Lan Boureau, Peter Sokol-Hessner, et Nathaniel D Daw. Deciding how to decide : Self-control and meta-decision making. *Trends in cognitive sciences*, 19(11) :700–710, 2015. (Cité page 68.)

Olivier Bousquet et André Elisseeff. Stability and generalization. *The Journal of Machine Learning Research*, 2 :499–526, 2002. (Cité page 16.)

R. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2 :14–23, 1986. (Cité page 62.)

Jérôme Bruner. *Le développement de l'enfant : savoir faire, savoir dire*. Presses universitaires de France, 2015. (Cité page 28.)

Christopher J Burke, Philippe N Tobler, Michelle Baddeley, et Wolfram Schultz. Neural mechanisms of observational learning. *Proceedings of the National Academy of Sciences*, 107(32) :14431–14436, 2010. (Cité page 38.)

Liesle Caballero, Álvaro Perafan, Martha Rinaldy, et Winston Percybrooks. Predicting the energy consumption of a robot in an exploration task using optimized neural networks. *Electronics*, 10(8) :920, 2021. (Cité page 18.)

K. Caluwaerts, A. Favre-Félix, M. Staffa, S. N'Guyen, C. Grand, B. Girard, et M. Khamassi. Neuro-inspired navigation strategies shifting for robots : Integration of a multiple landmark taxon strategy. Dans T.J. et al. Prescott, éditeur, *Living Machines 2012, LNAI*, volume 7375/2012, pages 62–73. 2012a. (Cité pages 68 et 135.)

K. Caluwaerts, M. Staffa, S. N'Guyen, C. Grand, L. Dollé, A. Favre-Félix, B. Girard, et M. Khamassi. A biologically inspired meta-control navigation system for the psikharpax rat robot. *Bioinspiration & Biomimetics*, 7 : 025009, 2012b. (Cité pages 24, 52, 87, 127, 128 et 138.)

Remi Cambuzat, Frédéric Elisei, Gérard Bailly, Olivier Simonin, et Anne Spalanzani. Immersive teleoperation of the eye gaze of social robots-assessing gaze-contingent control of vergence, yaw and pitch of robotic eyes. Dans *ISR 2018 ; 50th International Symposium on Robotics*, pages 1–8. VDE, 2018. (Cité page 38.)

Thomas Cederborg, Ishaan Grover, Charles L Isbell Jr, et Andrea Lockerd Thomaz. Policy shaping with human teachers. Dans *IJCAI*, pages 3366–3372, 2015. (Cité page 29.)

Ricardo Chavarriaga, Thomas Strösslin, Denis Sheynikhovich, et Wulfram Gerstner. A computational model of parallel navigation systems in rodents. *Neuroinformatics*, 3(3) :223–241, 2005. (Cité pages 51 et 52.)

Xi-liang Chen, Lei Cao, Chen-xi Li, Zhi-xiong Xu, et Jun Lai. Ensemble network architecture for deep reinforcement learning. *Mathematical Problems in Engineering*, 2018, 2018. (Cité pages 23 et 25.)

Sonia Chernova et Andrea L Thomaz. Robot learning from human teachers. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 8 (3) :1–121, 2014. (Cité page 31.)

Anne GE Collins et Michael J Frank. How much of reinforcement learning is working memory, not reinforcement learning ? a behavioral, computational, and neurogenetic analysis. *European Journal of Neuroscience*, 35 (7) :1024–1035, 2012. (Cité pages 49 et 53.)

Fiery Cushman et Adam Morris. Habitual control of goal selection in humans. *Proceedings of the National Academy of Sciences*, 112(45) :13817–13822, 2015. (Cité page 55.)

Nathaniel D Daw, Samuel J Gershman, Ben Seymour, Peter Dayan, et Raymond J Dolan. Model-based influences on humans' choices and striatal prediction errors. *Neuron*, 69(6) :1204–1215, 2011. (Cité page 46.)

N.D. Daw, Y. Niv, et P. Dayan. Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. *Nat. Neurosci.*, 8(12) :1704–1711, 2005. (Cité pages v, 3, 49, 51, 53, 54, 65, 67 et 121.)

Peter Dayan. Goal-directed control and its antipodes. *Neural Networks*, 22 (3) :213–219, 2009. (Cité page 43.)

Amir Dezfouli. Hierarchical models of goal-directed and automatic actions. 2015. (Cité pages 54, 57, 136 et 137.)

Amir Dezfouli et Bernard W Balleine. Habits, action sequences and reinforcement learning. *European Journal of Neuroscience*, 35(7) :1036–1051, 2012. (Cité pages 55, 57, 136 et 137.)

Amir Dezfouli et Bernard W Balleine. Actions, action sequences and habits : evidence that goal-directed and habitual action control are hierarchically organized. *PLoS Comput Biol*, 9(12) :e1003364, 2013. (Cité pages 55 et 56.)

Anthony Dickinson. Actions and habits : the development of behavioural autonomy. *Philosophical Transactions of the Royal Society of London. B, Biological Sciences*, 308(1135) :67–78, 1985. (Cité pages 3, 44 et 45.)

Ray J. Dolan et Peter Dayan. Goals and habits in the brain. *Neuron*, 80(2) : 312–325, oct 2013. (Cité page 121.)

Laurent Dollé, Ricardo Chavarriaga, Agnès Guillot, et Mehdi Khamassi. Interactions of spatial strategies producing generalization gradient and blocking : A computational approach. *PLoS computational biology*, 14(4) : e1006092, 2018. (Cité page 138.)

Laurent Dollé, Denis Sheynikhovich, Benoît Girard, Ricardo Chavarriaga, et Agnès Guillot. Path planning versus cue responding : a bio-inspired model of switching between navigation strategies. *Biological cybernetics*, 103(4) :299–317, 2010. (Cité page 51.)

Peter F Dominey et Michael A Arbib. A cortico-subcortical model for generation of spatially accurate sequential saccades. *Cerebral cortex*, 2 (2) :153–175, 1992. (Cité page 49.)

Stephane Doncieux. Transfer learning for direct policy search : A reward shaping approach. Dans *Proceedings of ICDL-EpiRob conference*, pages 1–6, 2013. (Cité page 22.)

Stephane Doncieux, Nicolas Bredeche, Jean-Baptiste Mouret, et Agoston E Gusz Eiben. Evolutionary robotics : what, why, and where to. *Frontiers in Robotics and AI*, 2 :4, 2015. (Cité page 5.)

Maël Donoso, Anne GE Collins, et Etienne Koechlin. Foundations of human reasoning in the prefrontal cortex. *Science*, 344(6191) :1481–1486, 2014. (Cité page 45.)

Marco Dorigo et Marco Colombetti. Robot shaping : Developing autonomous agents through learning. *Artificial intelligence*, 71(2) :321–370, 1994. (Cité page 29.)

Kenji Doya. What are the computations of the cerebellum, the basal ganglia and the cerebral cortex ? *Neural networks*, 12(7-8) :961–974, 1999. (Cité page 47.)

Kenji Doya, Kazuyuki Samejima, Ken ichi Katagiri, et Mitsuo Kawato. Multiple model-based reinforcement learning. *Neural Computation*, 14 : 1347–1369, 2002. (Cité page 23.)

Rémi Dromnelle, Benoît Girard, Erwan Renaudo, Raja Chatila, et Mehdi Khamassi. Coping with the variability in humans reward during simulated human-robot interactions through the coordination of multiple learning strategies. Dans *29th IEEE International Conference on Robot and Human Interactive Communication, RO-MAN 2020, Naples, Italy, August 31 - September 4, 2020*, pages 612–617. IEEE, 2020a. URL `https://doi.org/10.1109/RO-MAN47096.2020.9223451`. (Cité pages v, 99 et 134.)

Rémi Dromnelle, Erwan Renaudo, Guillaume Pourcel, Raja Chatila, Benoît Girard, et Mehdi Khamassi. How to reduce computation time while sparing performance during robot navigation ? a neuro-inspired architecture for autonomous shifting between model-based and model-free learning. Dans *Living Machines 2020 : The 9th International Conference on Biomimetic and Biohybrid systems.*, 2020b. (Cité pages v, 57, 76 et 133.)

OJ Dunn. Multiple comparisons using rank sums technometrics 6 : 241–252. *Find this article online*, 1964. (Cité page 105.)

Olive Jean Dunn. Multiple comparisons among means. *Journal of the American statistical association*, 56(293) :52–64, 1961. (Cité page 105.)

Sašo Džeroski, Luc De Raedt, et Kurt Driessens. Relational reinforcement learning. *Machine learning*, 43(1-2) :7–52, 2001. (Cité page 100.)

Ali Ezzeddine, Nafee Mourad, Babak Nadjar Araabi, et Majid Nili Ahmadabadi. Combination of learning from non-optimal demonstrations and feedbacks using inverse reinforcement learning and bayesian policy improvement. *Expert Systems with Applications*, 112 :331–341, 2018. (Cité page 37.)

Stefan Faußer et Friedhelm Schwenker. Ensemble methods for reinforcement learning with function approximation. Dans *International Workshop on Multiple Classifier Systems*, pages 56–65. Springer, 2011. (Cité pages 23 et 25.)

Stefan Faußer et Friedhelm Schwenker. Neural network ensembles in reinforcement learning. *Neural Processing Letters*, 41(1) :55–69, 2015a. (Cité pages 23 et 25.)

Stefan Faußer et Friedhelm Schwenker. Selective neural network ensembles in reinforcement learning : taking the advantage of many agents. *Neurocomputing*, 169 :350–357, 2015b. (Cité page 25.)

Richard E Fikes et Nils J Nilsson. Strips : A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4) : 189–208, 1971. (Cité page 62.)

Charles Fox et Tony Prescott. Hippocampus as unitary coherent particle filter. Dans *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2010. (Cité page 77.)

Yoav Freund, Robert Schapire, et Naoki Abe. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780) : 1612, 1999. (Cité page 23.)

John DE Gabrieli, Russell A Poldrack, et John E Desmond. The role of left prefrontal cortex in language and memory. *Proceedings of the national Academy of Sciences*, 95(3) :906–913, 1998. (Cité page 45.)

E. Gat. On three-layer architectures. Dans *Artificial Intelligence and Mobile Robots*. MIT Press, 1998. (Cité page 63.)

Matthieu Geist, Olivier Pietquin, et Gabriel Fricout. Kalman temporal differences : the deterministic case. Dans *2009 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, pages 185–192. IEEE, 2009. (Cité page 54.)

Benoît Girard, David Filliat, Jean-Arcady Meyer, Alain Berthoz, et Agnès Guillot. Integration of navigation and action selection functionalities in a computational model of cortico-basal-ganglia–thalamo-cortical loops. *Adaptive Behavior*, 13(2) :115–130, 2005. (Cité page 53.)

Jan Gläscher, Ralph Adolphs, Hanna Damasio, Antoine Bechara, David Rudrauf, Matthew Calamia, Lynn K Paul, et Daniel Tranel. Lesion mapping of cognitive control and value-based decision making in the prefrontal cortex. *Proceedings of the National Academy of Sciences*, 109(36) : 14681–14686, 2012. (Cité page 46.)

Jan Gläscher, Nathaniel Daw, Peter Dayan, et John P O'Doherty. States versus rewards : dissociable neural prediction error signals underlying model-based and model-free reinforcement learning. *Neuron*, 66(4) : 585–595, 2010. (Cité page 46.)

Michael A Goodrich et Alan C Schultz. *Human-robot interaction : a survey*. Now Publishers Inc, 2008. (Cité page 28.)

Ann M Graybiel. Building action repertoires : memory and learning functions of the basal ganglia. *Current opinion in neurobiology*, 5(6) :733–741, 1995. (Cité page 49.)

Shane Griffith, Kaushik Subramanian, Jonathan Scholz, Charles L Isbell, et Andrea L Thomaz. Policy shaping : Integrating human feedback with reinforcement learning. Dans *Advances in neural information processing systems*, pages 2625–2633, 2013. (Cité pages 29, 31, 32 et 35.)

G. Grisetti, C. Stachniss, et W. Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *Trans. Rob.*, 23(1) :34–46, Février 2007. ISSN 1552-3098. URL `http://dx.doi.org/10.1109/TRO.2006.889486`. (Cité page 76.)

Alex Guazzelli, Mihail Bota, Fernando J Corbacho, et Michael A Arbib. Affordances. motivations, and the world graph theory. *Adaptive Behavior*, 6(3-4) :435–471, 1998. (Cité pages 52 et 53.)

Agnès Guillot et Jean-Arcady Meyer. *La bionique : Quand la science imite la nature*. Dunod, 2008. (Cité page 1.)

Vijaykumar Gullapalli et Andrew G Barto. Shaping as a method for accelerating reinforcement learning. Dans *Proceedings of the 1992 IEEE international symposium on intelligent control*, pages 554–559. IEEE, 1992. (Cité pages 22, 29 et 32.)

Muhammad Burhan Hafez, Cornelius Weber, Matthias Kerzel, et Stefan Wermter. Curious meta-controller : Adaptive alternation between model-based and model-free control in deep reinforcement learning. Dans *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019. (Cité page 52.)

S. Hanoune. *Apprentissage Multimodalité Neuroscience Robotique mobile Réseaux de neurones Selection de l'action.* PhD thesis, Université de Cercy Pontoise, Paris, FR, 2015. (Cité page 52.)

Alexander Hans et Steffen Udluft. Ensembles of neural networks for robust reinforcement learning. Dans *2010 Ninth International Conference on Machine Learning and Applications*, pages 401–406. IEEE, 2010. (Cité pages 23 et 25.)

Stevan Harnad. The symbol grounding problem. *Physica D : Nonlinear Phenomena*, 42(1-3) :335–346, 1990. (Cité page 135.)

P. Gaussier et S. Boucenna Hasson, C. Emotions as a dynamical system : the interplay between the meta-control and communication function of emotions. *Paladyn, Journal of Behavioral Robotics.*, 2(3) :111–125, 2012. (Cité page 53.)

Donald Olding Hebb. The organization of behavior ; a neuropsycholocigal theory. *A Wiley Book in Clinical Psychology*, 62 :78, 1949. (Cité pages 43, 49, 53 et 57.)

Donald Olding Hebb. *The organization of behavior : A neuropsychological theory*. Psychology Press, 2005. (Cité page 18.)

Mark K Ho, Michael L Littman, Fiery Cushman, et Joseph L Austerweil. Teaching with rewards and punishments : Reinforcement or communication ? Dans *CogSci*, 2015. (Cité pages 34 et 35.)

Mark K Ho, James MacGlashan, Michael L Littman, et Fiery Cushman. Social is special : A normative framework for teaching with and learning from evaluative feedback. *Cognition*, 167 :91–106, 2017. (Cité page 33.)

Marc W Howard, Mrigankka S Fotedar, Aditya V Datey, et Michael E Hasselmo. The temporal context model in spatial navigation and relational learning : toward a common explanation of medial temporal lobe function across domains. *Psychological review*, 112(1) :75, 2005. (Cité page 77.)

Ronald A Howard. Dynamic programming and markov processes. 1960. (Cité pages 7 et 10.)

Charles Isbell, Christian R Shelton, Michael Kearns, Satinder Singh, et Peter Stone. A social reinforcement learning agent. Dans *Proceedings of the fifth international conference on Autonomous agents*, pages 377–384, 2001. (Cité page 32.)

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, et Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1) :79–87, 1991. (Cité page 23.)

Adrien Jauffret, Marwen Belkaid, Nicolas Cuperlier, Philippe Gaussier, et Philippe Tarroux. 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)*, pages 1–7. IEEE, 2013. (Cité page 53.)

Hyeon-Ae Jeon et Angela D Friederici. Two principles of organization in the prefrontal cortex are cognitive hierarchy and degree of automaticity. *Nature Communications*, 4(1) :1–8, 2013. (Cité page 46.)

Ju Jiang et Mohamed S Kamel. Aggregation of reinforcement learning algorithms. Dans *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pages 68–72. IEEE, 2006. (Cité page 23.)

Kshitij Judah, Saikat Roy, Alan Fern, et Thomas Dietterich. Reinforcement learning via practice and critique advice. Dans *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, 2010. (Cité pages 29, 30 et 31.)

Daniel Kahneman. *Système 1/Système 2 : Les deux vitesses de la pensée*. Flammarion, 2012. (Cité page 3.)

Frederic Kaplan, Pierre-Yves Oudeyer, Enikö Kubinyi, et Adám Miklósi. Robotic clicker training. *Robotics and Autonomous Systems*, 38(3-4) :197–206, 2002. (Cité page 31.)

Mehdi Keramati, Amir Dezfouli, et Payam Piray. Speed/accuracy trade-off between the habitual and the goal-directed processes. *PLoS Comput Biol*, 7(5) :e1002055, 2011. (Cité page 54.)

Mehdi Keramati, Peter Smittenaar, Raymond J Dolan, et Peter Dayan. Adaptive integration of habits into depth-limited planning defines a habitual-goal–directed spectrum. *Proceedings of the National Academy of Sciences*, 113(45) :12868–12873, 2016. (Cité pages 57, 73 et 98.)

M. Khamassi, B. Girard, A. Clodic, S. Devin, E. Renaudo, E. Pacherie, R. Alami, et R. Chatila. Integration of action, joint action and learning in robot cognitive architectures. *Intellectica*, 2016. (Cité page 63.)

Mehdi Khamassi, Georgia Chalvatzaki, Theodore Tsitsimis, George Velentzas, et Costas Tzafestas. A framework for robot learning during child-robot interaction with human engagement as reward signal. Dans *2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 461–464. IEEE, 2018a. (Cité page 38.)

Mehdi Khamassi et Mark D Humphries. Integrating cortico-limbic-basal ganglia architectures for learning model-based and model-free navigation strategies. *Frontiers in behavioral neuroscience*, 6 :79, 2012. (Cité pages xii et 50.)

Mehdi Khamassi, Stéphane Lallée, Pierre Enel, Emmanuel Procyk, et Peter F Dominey. Robot cognitive control with a neurophysiologically inspired reinforcement learning model. *Frontiers in neurorobotics*, 5 :1, 2011. (Cité page 21.)

Mehdi Khamassi, Louis-Emmanuel Martinet, et Agnès Guillot. Combining self-organizing maps with mixtures of experts : application to an actor-critic model of reinforcement learning in the basal ganglia. Dans *International Conference on Simulation of Adaptive Behavior*, pages 394–405. Springer, 2006. (Cité page 23.)

Mehdi Khamassi, George Velentzas, Theodore Tsitsimis, et Costas Tzafestas. Robot fast adaptation to changes in human engagement during simulated dynamic social interaction with active exploration in parameterized reinforcement learning. *IEEE Transactions on Cognitive and Developmental Systems*, 10(4) :881–893, 2018b. (Cité page 38.)

Kenneth T Kishida, Ignacio Saez, Terry Lohrenz, Mark R Witcher, Adrian W Laxton, Stephen B Tatter, Jason P White, Thomas L Ellis, Paul EM Phillips, et P Read Montague. Subsecond dopamine fluctuations in human striatum encode superposed error signals about actual and counterfactual reward. *Proceedings of the National Academy of Sciences*, 113(1) :200–205, 2016. (Cité page 47.)

W Bradley Knox et Peter Stone. Interactively shaping agents via human reinforcement : The tamer framework. Dans *Proceedings of the fifth international conference on Knowledge capture*, pages 9–16, 2009. (Cité pages 29, 31 et 32.)

W Bradley Knox et Peter Stone. Combining manual feedback with subsequent mdp reward signals for reinforcement learning. Dans *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems : volume 1-Volume 1*, pages 5–12. Citeseer, 2010. (Cité pages 32, 33 et 34.)

W Bradley Knox et Peter Stone. Reinforcement learning from human reward : Discounting in episodic tasks. Dans *2012 IEEE RO-MAN : The 21st IEEE International Symposium on Robot and Human Interactive Communication*, pages 878–885. IEEE, 2012a. (Cité pages 32 et 33.)

W Bradley Knox et Peter Stone. Reinforcement learning from simultaneous human and mdp reward. Dans *AAMAS*, pages 475–482, 2012b. (Cité pages 27, 34, 35 et 103.)

W Bradley Knox, Matthew E Taylor, et Peter Stone. Understanding human teaching modalities in reinforcement learning environments : A preliminary report. Dans *IJCAI 2011 Workshop on Agents Learning Interactively from Human Teachers (ALIHT)*, 2011. (Cité pages 31, 34, 35, 37 et 114.)

Jan Kober, Andrew J. Bagnell, et Jan Peters. Reinforcement learning in robotics : A survey. *IJRR Journal*, 32(11) :1238–1274, 2013. (Cité pages 5, 6 et 23.)

Etienne Koechlin, Gianpaolo Basso, Pietro Pietrini, Seth Panzer, et Jordan Grafman. The role of the anterior prefrontal cortex in human cognition. *Nature*, 399(6732) :148–151, 1999. (Cité page 45.)

Etienne Koechlin et Christopher Summerfield. An information theoretical approach to prefrontal executive function. *Trends in cognitive sciences*, 11 (6) :229–235, 2007. (Cité page 46.)

Wolfgang Köhler. *Intelligenzprüfungen an anthropoiden. 1.-*. Numéro 1. Königl. akademie der wissenschaften, 1917. (Cité page 40.)

Vijay R Konda et John N Tsitsiklis. Actor-critic algorithms. Dans *Advances in neural information processing systems*, pages 1008–1014. Citeseer, 2000. (Cité page 18.)

George Konidaris et Andrew Barto. Autonomous shaping : Knowledge transfer in reinforcement learning. Dans *Proceedings of the 23rd international conference on Machine learning*, pages 489–496, 2006. (Cité page 22.)

George Konidaris et Andrew Barto. Skill discovery in continuous reinforcement learning domains using skill chaining. *Advances in neural information processing systems*, 22 :1015–1023, 2009. (Cité page 136.)

George Konidaris, Leslie Kaelbling, et Tomas Lozano-Perez. Constructing symbolic representations for high-level planning. Dans *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28, 2014. (Cité page 137.)

George Konidaris, Leslie Pack Kaelbling, et Tomas Lozano-Perez. From skills to symbols : Learning symbolic representations for abstract high-level planning. *Journal of Artificial Intelligence Research*, 61 :215–289, 2018. (Cité page 137.)

George Konidaris, Scott Kuindersma, Roderic Grupen, et Andrew Barto. Autonomous skill acquisition on a mobile manipulator. Dans *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 25, 2011. (Cité page 137.)

George Dimitri Konidaris, Scott Kuindersma, Andrew G Barto, et Roderic A Grupen. Constructing skill trees for reinforcement learning agents from demonstration trajectories. Dans *NIPS*, volume 23, pages 1162–1170, 2010. (Cité page 136.)

Wouter Kool, Samuel J Gershman, et Fiery A Cushman. Cost-benefit arbitration between multiple reinforcement-learning systems. *Psychological science*, 28(9) :1321–1333, 2017. (Cité page 68.)

Wouter Kool, Samuel J Gershman, et Fiery A Cushman. Planning complexity registers as a cost in metacontrol. *Journal of cognitive neuroscience*, 30(10) :1391–1404, 2018. (Cité page 68.)

Frédérique Kouneiher, Sylvain Charron, et Etienne Koechlin. Motivation and cognitive control in the human prefrontal cortex. *Nature neuroscience*, 12(7) :939–945, 2009. (Cité page 46.)

Unmesh Kurup et Christian Lebiere. What can cognitive architectures do for robotics ? *Biologically Inspired Cognitive Architectures*, 2 :88–99, 2012. (Cité page 64.)

Robert Kurzban, Angela Duckworth, Joseph W Kable, et Justus Myers. An opportunity cost model of subjective effort and task performance. *The Behavioral and brain sciences*, 36(6), 2013. (Cité page 68.)

Antonio H Lara et Jonathan D Wallis. The role of prefrontal cortex in working memory : a mini review. *Frontiers in systems neuroscience*, 9 : 173, 2015. (Cité page 45.)

Sang Wan Lee, Shinsuke Shimojo, et John P O'Doherty. Neural computations underlying arbitration between model-based and model-free learning. *Neuron*, 81(3) :687–699, 2014. (Cité page 139.)

Claude Lemaréchal. Cauchy and the gradient method. *Doc Math Extra*, 251(254) :10, 2012. (Cité page 16.)

Florian Lesaint, Olivier Sigaud, Shelly B Flagel, Terry E Robinson, et Mehdi Khamassi. Modelling individual differences in the form of pavlovian conditioned approach responses : a dual learning systems approach with factored representations. *PLoS Comput Biol*, 10(2) :e1003466, 2014. (Cité page 49.)

Guangliang Li, Bo He, Randy Gomez, et Keisuke Nakamura. Interactive reinforcement learning from demonstration and human evaluative feedback. Dans *2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 1156–1162. IEEE, 2018. (Cité page 37.)

Long Ji Lin. Programming robots using reinforcement learning and teaching. Dans *AAAI*, pages 781–786, 1991. (Cité page 31.)

Todd Litman. Autonomous vehicle implementation predictions : Implications for transport planning. 2020. (Cité page 75.)

Tomas Ljungberg, Paul Apicella, et Wolfram Schultz. Responses of monkey dopamine neurons during learning of behavioral reactions. *Journal of neurophysiology*, 67(1) :145–163, 1992. (Cité page 47.)

Martin Llofriu, Gonzalo Tejera, M Contreras, Tatiana Pelc, Jean-Marc Fellous, et Alfredo Weitzenfeld. Goal-oriented robot navigation learning using a multi-scale space representation. *Neural Networks*, 72 :62–74, 2015. (Cité page 136.)

Robert Loftin, James MacGlashan, Bei Peng, Matthew Taylor, Michael Littman, Jeff Huang, et David Roberts. A strategy-aware technique for learning behaviors from discrete human feedback. Dans *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28, 2014. (Cité page 32.)

Robert Loftin, Bei Peng, James MacGlashan, Michael L Littman, Matthew E Taylor, Jeff Huang, et David L Roberts. Learning behaviors via

human-delivered discrete feedback : modeling implicit feedback strategies to speed up learning. *Autonomous agents and multi-agent systems*, 30 (1) :30–59, 2016. (Cité pages 31 et 35.)

Tomas Lozano-Perez. Robot programming. *Proceedings of the IEEE*, 71(7) : 821–841, 1983. (Cité page 30.)

Richard Maclin, Jude Shavlik, Lisa Torrey, Trevor Walker, et Edward Wild. Giving advice about preferred actions to reinforcement learners via knowledge-based kernel regression. Dans *AAAI*, pages 819–824, 2005. (Cité page 33.)

Richard Maclin et Jude W Shavlik. Creating advice-taking reinforcement learners. *Machine Learning*, 22(1) :251–281, 1996. (Cité pages 30 et 33.)

Bhaskara Marthi. Automatic shaping and decomposition of reward functions. Dans *Proceedings of the 24th International Conference on Machine learning*, pages 601–608, 2007. (Cité page 22.)

Kory W Mathewson et Patrick M Pilarski. Simultaneous control and human feedback in the training of a robotic agent with actor-critic reinforcement learning. *arXiv preprint arXiv :1606.06979*, 2016. (Cité pages 32 et 33.)

Michael J Milford, Gordon F Wyeth, et David Prasser. Ratslam : a hippocampal model for simultaneous localization and mapping. Dans *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, volume 1, pages 403–408. IEEE, 2004. (Cité page 77.)

Marvin Minsky et Seymour A Papert. *Perceptrons : An introduction to computational geometry*. MIT press, 2017. (Cité page 18.)

Benjamin F Missner. The wirelessly directed torpedo. *Scientific American*, 107(3) :53–54, 1912. (Cité page 1.)

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540) :529–533, 2015a. (Cité pages xii, 16, 17 et 136.)

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, et Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540) :529–533, Février 2015b. ISSN 00280836. URL `http://dx.doi.org/10.1038/nature14236`. (Cité page 83.)

Ida Momennejad, A Ross Otto, Nathaniel D Daw, et Kenneth A Norman. Offline replay supports planning in human reinforcement learning. *Elife*, 7 :e32548, 2018. (Cité page 139.)

Stephen Monsell. Task switching. *Trends in cognitive sciences*, 7(3) :134–140, 2003. (Cité page 46.)

Andrew W Moore et Christopher G Atkeson. Prioritized sweeping : Reinforcement learning with less data and less time. *Machine learning*, 13(1) : 103–130, 1993. (Cité page 19.)

Richard Morris. Developments of a water-maze procedure for studying spatial learning in the rat. *Journal of neuroscience methods*, 11(1) :47–60, 1984. (Cité page 51.)

Nafee Mourad, Ali Ezzeddine, Babak Nadjar Araabi, et Majid Nili Ahmadabadi. Learning from demonstrations and human evaluative feedbacks : Handling sparsity and imperfection using inverse reinforcement learning approach. *Journal of Robotics*, 2020, 2020. (Cité page 37.)

Anis Najar, Emmanuelle Bonnet, Bahador Bahrami, et Stefano Palminteri. Imitation as a model-free process in human reinforcement learning. 2019. (Cité page 38.)

Anis Najar et Mohamed Chetouani. Reinforcement learning with human advice. a survey. *arXiv preprint arXiv :2005.11016*, 2020. (Cité pages xii, 30, 35 et 36.)

Anis Najar, Olivier Sigaud, et Mohamed Chetouani. Interactively shaping robot behaviour with unlabeled human instructions. *Autonomous Agents and Multi-Agent Systems*, 34 :1–35, 2020. (Cité pages 31 et 34.)

Andrew Y Ng, Daishi Harada, et Stuart Russell. Policy invariance under reward transformations : Theory and application to reward shaping. Dans *Icml*, volume 99, pages 278–287, 1999. (Cité page 22.)

Nils J Nilsson. A mobile automaton : An application of artificial intelligence techniques. Rapport technique, SRI INTERNATIONAL MENLO PARK CA ARTIFICIAL INTELLIGENCE CENTER, 1969. (Cité page 62.)

John O'keefe et Lynn Nadel. *The hippocampus as a cognitive map*. Oxford : Clarendon Press, 1978. (Cité page 136.)

A Ross Otto, Samuel J Gershman, Arthur B Markman, et Nathaniel D Daw. The curse of planning : dissecting multiple reinforcement-learning systems by taxing the central executive. *Psychological science*, 24(5) :751–761, 2013. (Cité page 57.)

John P O'Doherty, Sangwan Lee, Reza Tadayonnejad, Jeff Cockburn, Kyo Iigaya, et Caroline J Charpentier. Why and how the brain weights contributions from a mixture of experts. *Neuroscience & Biobehavioral Reviews*, 2021. (Cité page 46.)

Sinno Jialin Pan et Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10) :1345–1359, 2009. (Cité page 22.)

Marios Panayi, Simon Killcross, et Mehdi Khamassi. The rodent lateral orbitofrontal cortex as an arbitrator selecting between model-based and model-free learning systems. *Behavioral Neuroscience*, 2021. (Cité page 139.)

P Ivan Pavlov. Conditioned reflexes : an investigation of the physiological activity of the cerebral cortex. *Annals of neurosciences*, 17(3) :136, 2010. (Cité pages xii et 41.)

Jing Peng et Ronald J Williams. Efficient learning and planning within the dyna framework. *Adaptive behavior*, 1(4) :437–454, 1993. (Cité page 20.)

Giovanni Pezzulo, Francesco Rigoli, et Fabian Chersi. The mixed instrumental controller : using value of information to combine habitual choice and mental simulation. *Frontiers in psychology*, 4 :92, 2013. (Cité page 54.)

Rolf Pfeifer, Max Lungarella, et Fumiya Iida. Self-organization, embodiment, and biologically inspired robotics. *science*, 318(5853) :1088–1093, 2007. (Cité page 1.)

Taman Powell et Tanya Sammut-Bonnici. Pareto analysis. 01 2015. ISBN 9781118785317. (Cité page 70.)

KVN Pradyot. *Beyond rewards : Learning from richer supervision*. PhD thesis, Citeseer, 2012. (Cité page 30.)

M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, et A. Y. Ng. Ros : an open-source robot operating system. Dans *ICRA Workshop on Open Source Software*, 2009. (Cité page 76.)

J Ross Quinlan. *C4. 5 : programs for machine learning*. Elsevier, 2014. (Cité page 18.)

Marc Raibert, Kevin Blankespoor, Gabriel Nelson, et Rob Playter. Bigdog, the rough-terrain quadruped robot. *IFAC Proceedings Volumes*, 41(2) : 10822–10825, 2008. (Cité page 1.)

Jette Randløv et Preben Alstrøm. Learning to drive a bicycle using reinforcement learning and shaping. Dans *ICML*, volume 98, pages 463–471. Citeseer, 1998. (Cité pages 22 et 33.)

Erwan Renaudo. *Des comportements flexibles aux comportements habituels : meta-apprentissage neuro-inspiré' pour la robotique autonome*. PhD thesis, Université Pierre et Marie Curie, Paris, France, 2016. (Cité pages v, 3, 23, 53, 54, 57, 63, 64, 65, 67, 68, 72 et 82.)

Erwan Renaudo, Sandra Devin, Benoît Girard, Raja Chatila, Rachid Alami, Mehdi Khamassi, et Aurélie Clodic. Learning to interact with humans using goal-directed and habitual behaviors, 2015a. (Cité page 120.)

Erwan Renaudo, Benoît Girard, Raja Chatila, et Mehdi Khamassi. 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*, pages 178–184, Lyon, France, 2015b. (Cité pages 25, 53 et 67.)

Erwan Renaudo, Benoît Girard, Raja Chatila, et Mehdi Khamassi. 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)*, pages 254–260, Providence, RI, USA, 2015c. (Cité pages 25, 53 et 68.)

Robert A Rescorla. A theory of pavlovian conditioning : Variations in the effectiveness of reinforcement and nonreinforcement. *Current research and theory*, pages 64–99, 1972. (Cité pages 14, 41 et 47.)

Frank Rosenblatt. The perceptron : a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6) :386, 1958. (Cité page 43.)

Gavin A Rummery et Mahesan Niranjan. *On-line Q-learning using connectionist systems*, volume 37. University of Cambridge, Department of Engineering Cambridge, UK, 1994. (Cité page 15.)

Katsuyuki Sakai. Task set and prefrontal cortex. *Annu. Rev. Neurosci.*, 31 : 219–245, 2008. (Cité page 46.)

Jacques Saraydaryan, Fabrice Jumel, et Olivier Simonin. Navigation in human flows : Planning with adaptive motion grid. Dans *IROS Workshop CrowdNav*, 2018. (Cité page 75.)

Joe Saunders, Chrystopher L Nehaniv, et Kerstin Dautenhahn. Teaching robots by moulding behavior and scaffolding the environment. Dans *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, pages 118–125, 2006. (Cité page 29.)

Wolfram Schultz. Updating dopamine reward signals. *Current opinion in neurobiology*, 23(2) :229–238, 2013. (Cité page 47.)

Wolfram Schultz, Peter Dayan, et P Read Montague. A neural substrate of prediction and reward. *Science*, 275(5306) :1593–1599, 1997. (Cité pages xii, 47 et 48.)

John R Searle. Minds, brains, and programs. *The Turing Test : Verbal Behaviour as the Hallmark of Intelligence*, pages 201–224, 1980. (Cité page 135.)

Andrew W Senior, Richard Evans, John Jumper, James Kirkpatrick, Laurent Sifre, Tim Green, Chongli Qin, Augustin Žídek, Alexander WR Nelson, Alex Bridgland, et al. Improved protein structure prediction using potentials from deep learning. *Nature*, 577(7792) :706–710, 2020. (Cité pages 18 et 136.)

Burr Settles. Active learning literature survey. 2009. (Cité page 28.)

Martin Sewell. Ensemble learning. *RN*, 11(02) :1–34, 2008. (Cité page 23.)

Farzaneh Sheikhnezhad Fard et Thomas P Trappenberg. A novel model for arbitration between planning and habitual control systems. *Frontiers in neurorobotics*, 13 :52, 2019. (Cité page 52.)

Amitai Shenhav, Sebastian Musslick, Falk Lieder, Wouter Kool, Thomas L Griffiths, Jonathan D Cohen, et Matthew M Botvinick. Toward a rational and mechanistic account of mental effort. *Annual review of neuroscience*, 40 :99–124, 2017. (Cité page 68.)

Olivier Sigaud et Olivier Buffet. *Markov decision processes in artificial intelligence*. John Wiley & Sons, 2013. (Cité page 7.)

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587) :484–489, 2016. (Cité pages 18 et 136.)

Satinder Pal Singh. Transfer of learning by composing solutions of elemental sequential tasks. *Machine Learning*, 8(3) :323–339, 1992. (Cité page 29.)

Burrhus Frederic Skinner. *The behavior of organisms : An experimental analysis*. BF Skinner Foundation, 2019. (Cité pages xii, 3, 29 et 42.)

Richard S Sutton. Reinforcement learning architectures. *Proceedings ISKIT*, 92, 1992. (Cité pages 5, 18, 19 et 27.)

Richard S Sutton et Andrew G Barto. A temporal-difference model of classical conditioning. Dans *Proceedings of the ninth annual conference of the cognitive science society*, pages 355–378. Seattle, WA, 1987. (Cité pages 14 et 47.)

Richard S. Sutton et Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st édition, 1998. ISBN 0262193981. (Cité pages 5, 9, 11, 12, 19, 20, 25, 33, 35, 66, 67, 68 et 82.)

Andrea L Thomaz et Cynthia Breazeal. Transparency and socially guided machine learning. Dans *5th Intl. Conf. on Development and Learning (ICDL)*, 2006. (Cité page 28.)

Andrea L Thomaz et Cynthia Breazeal. Experiments in socially guided exploration : Lessons learned in building robots that learn with and without human teachers. *Connection Science*, 20(2-3) :91–110, 2008. (Cité pages 28 et 29.)

Andrea Lockerd Thomaz, Cynthia Breazeal, et al. Reinforcement learning with human teachers : Evidence of feedback and guidance with implications for learning performance. Dans *Aaai*, volume 6, pages 1000–1005. Boston, MA, 2006. (Cité pages 30, 31, 32 et 33.)

Catherine A Thorn, Hisham Atallah, Mark Howe, et Ann M Graybiel. Differential dynamics of activity changes in dorsolateral and dorsomedial striatal loops during learning. *Neuron*, 66(5) :781–795, 2010. (Cité page 45.)

Edward Lee Thorndike. *Animal intelligence : Experimental studies*. Transaction Publishers, 1970. (Cité page 40.)

Edward C Tolman. Cognitive maps in rats and men. *Psychological review*, 55(4) :189, 1948. (Cité page 44.)

Meropi Topalidou, Daisuke Kase, Thomas Boraud, et Nicolas Rougier. 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)*, 2015. (Cité page 57.)

Meropi Topalidou, Daisuke Kase, Thomas Boraud, et Nicolas P Rougier. Dissociation of reinforcement and hebbian learning induces covert acquisition of value in the basal ganglia. *bioRxiv*, page 060236, 2016. (Cité page 58.)

Meropi Topalidou, Daisuke Kase, Thomas Boraud, et Nicolas P Rougier. A computational model of dual competition between the basal ganglia and the cortex. *eneuro*, 5(6), 2018. (Cité pages 49 et 53.)

J Gregory Trafton, Laura M Hiatt, Anthony M Harrison, Franklin P Tamborello, Sangeet S Khemlani, et Alan C Schultz. Act-r/e : An embodied cognitive architecture for human-robot interaction. *Journal of Human-Robot Interaction*, 2(1) :30–55, 2013. (Cité page 64.)

Elizabeth Tricomi, Bernard W Balleine, et John P O'Doherty. A specific role for posterior dorsolateral striatum in human habit learning. *European Journal of Neuroscience*, 29(11) :2225–2232, 2009. (Cité page 45.)

Ryan J Urbanowicz et Jason H Moore. Learning classifier systems : a complete introduction, review, and roadmap. *Journal of Artificial Evolution and Applications*, 2009, 2009. (Cité page 5.)

Vivian V Valentin, Anthony Dickinson, et John P O'Doherty. Determining the neural substrates of goal-directed learning in the human brain. *Journal of Neuroscience*, 27(15) :4019–4026, 2007. (Cité page 45.)

Jessica Van Brummelen, Marie O'Brien, Dominique Gruyer, et Homayoun Najjaran. Autonomous vehicle perception : The technology of today and tomorrow. *Transportation research part C : emerging technologies*, 89 : 384–406, 2018. (Cité page 75.)

Laurens Van Der Maaten, Eric Postma, et Jaap Van den Herik. Dimensionality reduction : a comparative. *J Mach Learn Res*, 10(66-71) :13, 2009. (Cité page 16.)

David Vernon, Giorgio Metta, et Giulio Sandini. The icub cognitive architecture : Interactive development in a humanoid robot. Dans *2007 IEEE 6th International Conference on Development and Learning*, pages 122–127. Ieee, 2007a. (Cité page 64.)

David Vernon, Giorgio Metta, et Giulio Sandini. A survey of artificial cognitive systems : Implications for the autonomous development of mental capabilities in computational agents. *IEEE transactions on evolutionary computation*, 11(2) :151–180, 2007b. (Cité page 64.)

Guillaume Viejo, Mehdi Khamassi, Andrea Brovelli, et Benoît Girard. Modeling choice and reaction time during arbitrary visuomotor learning through the coordination of adaptive working memory and reinforcement learning. *Frontiers in behavioral neuroscience*, 9 :225, 2015. (Cité pages 49, 53 et 69.)

Anna-Lisa Vollmer, Jonathan Grizou, Manuel Lopes, Katharina Rohlfing, et Pierre-Yves Oudeyer. Studying the co-construction of interaction protocols in collaborative tasks with humans. Dans *4th International Conference on Development and Learning and on Epigenetic Robotics*, pages 208–215. IEEE, 2014. (Cité page 28.)

Richard Volpe, Issa Nesnas, Tara Estlin, Darren Mutz, Richard Petras, et Hari Das. The claraty architecture for robotic autonomy. Dans *Aerospace Conference, 2001, IEEE Proceedings.*, volume 1, pages 1–121. IEEE, 2001. (Cité page 63.)

C. Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, UK, 1989. (Cité page 15.)

Christopher JCH Watkins et Peter Dayan. Q-learning. *Machine learning*, 8 (3-4) :279–292, 1992. (Cité page 15.)

John B Watson. Psychology as the behaviorist views it. *Psychological review*, 20(2) :158, 1913. (Cité page 40.)

Steven D Whitehead. A complexity analysis of cooperative mechanisms in reinforcement learning. Dans *AAAI*, pages 607–613, 1991. (Cité pages 30 et 31.)

Marco Wiering et Martijn Van Otterlo. Reinforcement learning. *Adaptation, learning, and optimization*, 12(3), 2012. (Cité page 23.)

Marco A Wiering et Hado Van Hasselt. Two novel on-policy reinforcement learning algorithms based on td (λ)-methods. Dans *2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, pages 280–287. IEEE, 2007. (Cité page 23.)

Marco A Wiering et Hado Van Hasselt. Ensemble algorithms in reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 38(4) :930–936, 2008. (Cité pages 23, 24 et 25.)

Eric Wiewiora. Potential-based shaping and q-value initialization are equivalent. *Journal of Artificial Intelligence Research*, 19 :205–208, 2003. (Cité page 22.)

Eric Wiewiora, Garrison W Cottrell, et Charles Elkan. Principled methods for advising reinforcement learning agents. Dans *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 792–799, 2003. (Cité page 22.)

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4) :229–256, 1992. (Cité page 18.)

Daniel M Wolpert et Mitsuo Kawato. Multiple paired forward and inverse models for motor control. *Neural networks*, 11(7-8) :1317–1329, 1998. (Cité page 23.)

Cuebong Wong, Erfu Yang, Xiu-Tian Yan, et Dongbing Gu. Autonomous robots for harsh environments : a holistic overview of current solutions and ongoing challenges. *Systems Science & Control Engineering*, 6(1) : 213–219, 2018. (Cité page 2.)

David Wood, Jerome S Bruner, et Gail Ross. The role of tutoring in problem solving. *Journal of child psychology and psychiatry*, 17(2) :89–100, 1976. (Cité page 28.)

Klaus Wunderlich, Peter Dayan, et Raymond J Dolan. Mapping value based planning and extensively trained choice in the human brain. *Nature neuroscience*, 15(5) :786–791, 2012. (Cité page 46.)

Henry H Yin et Barbara J Knowlton. The role of the basal ganglia in habit formation. *Nature Reviews Neuroscience*, 7(6) :464–476, 2006. (Cité pages 45 et 49.)

Henry H Yin, Barbara J Knowlton, et Bernard W Balleine. Lesions of dorsolateral striatum preserve outcome expectancy but disrupt habit formation in instrumental learning. *European journal of neuroscience*, 19 (1) :181–189, 2004. (Cité page 45.)

Henry H Yin, Shweta Prasad Mulcare, Monica RF Hilário, Emily Clouse, Terrell Holloway, Margaret I Davis, Anita C Hansson, David M Lovinger, et Rui M Costa. Dynamic reorganization of striatal circuits during the acquisition and consolidation of a skill. *Nature neuroscience*, 12(3) : 333–341, 2009. (Cité page 45.)

Henry H Yin, Sean B Ostlund, Barbara J Knowlton, et Bernard W Balleine. The role of the dorsomedial striatum in instrumental conditioning. *European Journal of Neuroscience*, 22(2) :513–523, 2005. (Cité page 45.)

Abolfazl Zaraki, Mehdi Khamassi, Luke J Wood, Gabriella Lakatos, Costas Tzafestas, Farshid Amirabdollahian, Ben Robins, et Kerstin Dautenhahn. A novel reinforcement-based paradigm for children to teach the humanoid kaspar robot. *International Journal of Social Robotics*, pages 1–12, 2019. (Cité page 32.)

Notations

MDP	Markov decision process
AR	Apprentissage par renforcement
VI	Value Iteration
TD	Temporal difference
SARSA	State-Action-Reward-State-Action
DQN	Deep Q-Nework
ApD	Apprentissage par la démonstration
ApE	Apprentissage par les rétroactions évaluatives
BG	Basal ganglia
DMS	Dorsomedial striatum
DLS	Dorsolateral striatum
PFC	Prefrontal cortex
MB	Model-based
MF	Model-free
MC	Méta-contrôleur
SLAM	Simultaneous localization and mapping
EC	Entropy and Cost
RnD	Random

Ce document a été préparé à l'aide de l'éditeur de composition typographique
LaTeX 2ε en ligne Overleaf

Titre Architecture cognitive générique pour la coordination de stratégies d'apprentissage en robotique.

Résumé

Mots-clés Architecture de contrôle robotique ; Méta-contrôle ; Robotique bio-inspirée ; Apprentissage par renforcement ; Comportement instrumental ; Sélection de l'action ; Navigation autonome ; HRI

Title Generic cognitive architecture for learning strategies coordination in robotics.

Abstract

Keywords Robotic control architecture ; Meta-control ; Bio-inspired robotics ; Reinforcement learning ; Instrumental behavior ; Action selection ; Autonomous navigation ; HRI