



**HAL**  
open science

# Apprentissage par renforcement pour le pilotage énergétique de l'éclairage dans un bâtiment connecté

Nassim Haddam

► **To cite this version:**

Nassim Haddam. Apprentissage par renforcement pour le pilotage énergétique de l'éclairage dans un bâtiment connecté. Apprentissage [cs.LG]. Université Paris-Saclay, 2023. Français. NNT : 2023UP-ASG020 . tel-04106652

**HAL Id: tel-04106652**

**<https://theses.hal.science/tel-04106652v1>**

Submitted on 25 May 2023

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

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

# Apprentissage par renforcement pour le pilotage énergétique de l'éclairage dans un bâtiment connecté

*Reinforcement learning for lighting energy control in a  
connected building*

## Thèse de doctorat de l'université Paris-Saclay

École doctorale n° 580 : Sciences et technologies de l'information et de  
la communication (STIC)

Spécialité de doctorat : Informatique

Graduate School : Informatique et sciences du numérique, Référent :  
Université de Versailles-Saint-Quentin-en-Yvelines

Thèse préparée dans les unités de recherche **DAVID (Université  
Paris-Saclay, UVSQ)** et **LINEACT (CESI)**, sous la direction de **Dominique  
BARTH**, Professeur des universités, le co-encadrement de **Benjamin COHEN  
BOULAKIA**, Enseignant-Chercheur

Thèse soutenue à Versailles, le 23 mars 2023, par

**Nassim HADDAM**

### Composition du jury

Membres du jury avec voix délibérative

<b>Jean-Michel FOURNEAU</b> Professeur, DAVID, UVSQ	Président
<b>Lydia BOUDJELOUD-ASSALA</b> Maîtresse de conférence, LORIA, Université de Lor- raine	Rapporteur & Examinatrice
<b>Abder KOUKAM</b> Professeur, CIAD, SeT, Université de Technologie de Belfort-Montbéliard	Rapporteur & Examineur
<b>Lila BOUKHATEM</b> Maîtresse de Conférences, LISN, Université Paris Saclay	Examinatrice
<b>Latifa OUKHELLOU</b> Directrice de Recherche, GRETTIA, IFSTTAR	Examinatrice

**Titre :** Apprentissage par renforcement pour le pilotage énergétique de l'éclairage dans un bâtiment connecté

**Mots clés :** Bâtiment Intelligent, Apprentissage par renforcement, contrôle de l'éclairage, confort lumineux

**Résumé :** L'enjeu de cette thèse est de proposer et de valider des algorithmes permettant un pilotage énergétique intelligent capable de s'adapter aux usagers et leurs pratiques. Nous voulons contrôler spécifiquement l'éclairage d'une pièce dans laquelle l'utilisateur peut interagir librement avec le système. Ceci nous a amené à concevoir un modèle simulant les réactions de l'utilisateur face au changement du signal lumineux. Ce modèle n'est pas connu par l'algorithme de contrôle et sert uniquement à tester et à valider les algorithmes de contrôle que nous proposons. Le modèle de l'utilisateur a les propriétés suivantes :

- L'utilisateur réagit souvent pour les valeurs faibles du signal et peu souvent pour les valeurs fortes du signal.
- Les réactions de l'utilisateur dépendent du passé (non stationnarité) et elles sont dominées par les expériences les plus récentes.
- Si le signal a pris des valeurs faibles par le passé, l'utilisateur aura tendance à réagir plus souvent que si le signal avait pris des valeurs fortes.

Nous proposons des algorithmes capables de contrôler la lumière du bâtiment sans connaître au préalable le modèle de l'utilisateur. À cet effet, nous avons utilisé l'apprentissage par renforcement. Dans l'apprentissage par renforcement, un agent évolue dans un environnement donné et il veut effectuer une certaine tâche. L'agent apprend à effectuer cette tâche par essai et erreur en interagissant continuellement avec l'environnement. L'apprentissage par renforcement est a priori le seul modèle permettant d'optimiser un objectif donné sans connaissance de l'environnement.

Dans cette thèse, nous étudions plusieurs paradigmes pour le système de contrôle dans lesquelles

la décision de l'agent est modélisée de façon différente :

- Le paradigme par choix de variation du signal dans lequel le choix du système porte sur la vitesse de baisse du signal et la valeur du signal sur laquelle il faut s'arrêter. Le contrôle concerne l'ensemble de la variation du signal.
- Le paradigme par choix de valeur du signal dans lequel le choix de l'agent porte sur la valeur actuelle. L'agent choisit une valeur dans le voisinage de la valeur courante.

Les contributions principales de cette thèse consistent en la proposition d'algorithmes d'apprentissage par renforcement basés sur ces deux paradigmes et l'étude expérimentale et la comparaison de ces algorithmes, notamment face à différents paramétrages de l'agent et de l'utilisateur.

Le paradigme par choix de variation du signal a été implémenté avec l'apprentissage par renforcement sans états. Nous avons comparé les performances des algorithmes d'apprentissage par renforcement suivants : epsilon-greedy, LRI, LRP, la poursuite et la poursuite hiérarchique. Nous nous sommes intéressés spécifiquement à l'évolution au cours du temps de l'énergie, de la récompense, et de l'état de l'utilisateur. Bien entendu, la vitesse de convergence et la qualité de convergence ont aussi été prises en compte.

Le paradigme par choix de valeur du signal a été implémenté avec l'apprentissage par renforcement basé sur les états. Nous avons étudié les performances des algorithmes les plus populaires : SARSA et Q-learning. La convergence des algorithmes, l'énergie consommée ainsi que la récompense et l'état de l'utilisateur ont été pris en compte dans cette comparaison.

**Title :** Reinforcement learning for lighting energy control in a connected building

**Keywords :** Intelligent building, reinforcement learning, lighting control, lighting comfort

**Abstract :** The challenge of this thesis is to propose and validate algorithms allowing an intelligent energy control to adapt to users and their practices. In this thesis, we want to specifically control the light in a room in which the user can freely interact with the system. This led us to design a model simulating the user's reactions to the change of the light signal. This model is not known by the control algorithm and is only used to test and validate our proposed control algorithms. The user model has the following properties :

- The user reacts often for low signal values and infrequently for high signal values.
- user's reaction depends on the past (non-stationarity) and is dominated by the most recent experiences.
- the signal has taken low values in the past, the user will tend to react more often than if the signal had taken high values.

We propose algorithms able to control the building light without knowing the user's model. For this purpose, we used reinforcement learning. In reinforcement learning, an agent evolves in a given environment and wants to perform a certain task. The agent learns to perform this task through trial and error by continuously interacting with the environment. Reinforcement learning is a priori the only model allowing to optimize a given objective without knowledge of the environment.

We study several paradigms for the control system in which the agent's decision is modeled differently :

- The paradigm by choice of signal variation in which the choice of the system concerns the speed of the signal decrease and the value of the signal on which to stop. The control concerns the whole signal variation.
- The paradigm by choice of signal value in which the agent's choice concerns the current value. The agent chooses a value in the neighborhood of the current value.

The main contributions of this thesis have been to propose reinforcement learning algorithms to model these two types of systems and to compare the performances of these algorithms for different settings of the agent and the user.

The signal variation choice paradigm was formulated with stateless reinforcement learning. We compared the performance of the following reinforcement learning algorithms : epsilon-greedy, LRI, LRP, tracking, and hierarchical tracking. We were specifically interested in the evolution over time of energy, reward, and user state. Of course, the convergence speed and the convergence quality have also been taken into account.

The signal value choice paradigm has been formulated with state-based reinforcement learning. We studied the performance of the most popular algorithms : SARSA and Q-learning. The convergence of the algorithms, the energy consumed as well as the reward and the state of the user were taken into account in this comparison.



# Apprentissage par renforcement pour le pilotage énergétique de l'éclairage dans un bâtiment connecté

*Reinforcement learning for lighting energy control in a  
connected building*

Thèse préparée dans les unités de recherche **DAVID (Université Paris-Saclay, UVSQ)** et **LINEACT (CESI)**, sous la direction de **Dominique BARTH**, Professeur des universités, le co-encadrement de **Benjamin COHEN BOULAKIA**, Enseignant-Chercheur

**Nassim HADDAM**

## Composition du jury

### Membres du jury avec voix délibérative

<b>Jean-Michel FOURNEAU</b> Professeur, DAVID, UVSQ	Président
<b>Lydia BOUDJELOUD-ASSALA</b> Maîtresse de conférence, LORIA, Université de Lorraine	Rapporteur & Examinatrice
<b>Abder KOUKAM</b> Professeur, CIAD, SeT, Université de Technologie de Belfort-Montbéliard	Rapporteur & Examineur
<b>Lila BOUKHATEM</b> Maîtresse de Conférences, LISN, Université Paris Saclay	Examinatrice
<b>Latifa OUKHELLOU</b> Directrice de Recherche, GRETTIA, IFSTTAR	Examinatrice

# Remerciements

Je souhaite remercier en premier lieu mon directeur de thèse Dominique Barth. Je lui suis reconnaissant pour le temps conséquent qu'il m'a accordé, ses qualités pédagogiques et scientifiques, sa franchise et sa sympathie. J'ai beaucoup appris à ses côtés et je lui adresse ma gratitude pour tout cela. J'adresse de chaleureux remerciements à mon co-encadrant de thèse Benjamin Cohen Boulakia, pour son attention de tout instant sur mes travaux, pour ses conseils avisés et son écoute qui ont été prépondérants pour la bonne réussite de cette thèse. J'ai pris un grand plaisir à travailler avec mes encadrants. Leurs conseils m'ont été d'une valeur incommensurable dans ma réflexion scientifique ainsi que pour le développement de mon esprit critique tout au long de cette thèse.

Je remercie également les membres de mon jury de thèse Jean-Michel Fourneau, Abder Koukam, Lydia Boudjeloud-Assala, Latifa Oukhellou et Lila Boukhatem pour avoir accepté de faire partie de mon jury. Leurs commentaires au travers de leurs rapports ou pendant la soutenance ont été très instructifs.

Je voudrais remercier tous les membres (anciens et actuels) du laboratoire DAVID de l'UVSQ et du laboratoire CESI LINEACT. Je ne pourrais faire justice à tout le monde dans ces quelques lignes, mais je mentionne particulièrement Anas Hossini, Aurélien Lepage, Bintou Fofana, Coline Gianfrotta, Dalil Tadjedine, Joseph Desquaires, Maël Guiraud, Maxime Tréca, Muriel Davies, Omar Gacem (rip), Wilfried Ehounou, Yann Richou et Youssef Ait El Mahjoub. Je remercie également tous mes enseignants de master pour m'avoir aidé à renforcer la réflexion algorithmique nécessaire à cette thèse (David Auger, Franck Quessette, Leila Kloul, Pierre Couchenay, Sandrine Vial, Thierry Mautor, Yann Strozecki, ...).

Je souhaite remercier ma famille, ma mère, mon père, mon frère et mon cousin pour leur soutien émotionnel à toute épreuve et sans qui rien de cela ne serait possible. Merci !





# Table des matières

<b>1</b>	<b>Aperçu sur les méthodes de contrôle de bâtiment et de prédiction de confort des usagers</b>	<b>13</b>
1.1	Introduction	13
1.2	Confort des usagers	13
1.2.1	Confort thermique	14
1.2.2	Confort visuel	18
1.3	Méthodes de contrôle lumineux dans un bâtiment	24
1.4	Méthodes de contrôle thermique dans un bâtiment	26
1.5	Méthodes de contrôle thermique utilisant le RL	30
1.5.1	Approches tabulaires	31
1.5.2	Approches utilisant les fonctions d'approximation	31
<b>2</b>	<b>Apprentissage par renforcement</b>	<b>35</b>
2.1	Les approches d'apprentissage par renforcement sans états	35
2.1.1	Méthodes basées sur la politique	35
2.1.2	Les méthodes basées sur les valeurs	41
2.1.3	L'algorithme $\epsilon$ -Greedy	43
2.1.4	Exploration de Boltzmann	44
2.1.5	Méthodes basées sur les valeurs : optimisme face à l'incertitude	45
2.2	Les méthodes d'apprentissage par renforcement basé sur les états	49
<b>3</b>	<b>Modèle des algorithmes de contrôle énergétique de bâtiment</b>	<b>53</b>
3.1	Contexte	53
3.2	Paradigmes	54
3.3	Le contrôle du point de vue du système	56
3.4	Modèle de l'utilisateur	59
<b>4</b>	<b>Modèle par politique de variation du signal</b>	<b>67</b>
4.1	Le système de contrôle	68
4.1.1	Formalisme	68
4.1.2	L'agent des pentes	71
4.1.3	L'agent des valeurs d'arrêt	75
4.1.4	Le système complet	76
4.2	Évaluation du système de contrôle	80
4.2.1	Notions sur l'évaluation des performances des algorithmes d'apprentissage par renforcement sans états	81
4.2.2	Évaluation de l'agent des pentes	83
4.2.3	Évaluation du système de contrôle	98
4.2.4	Comparaison des algorithmes d'apprentissage par renforcement sur le système complet	101
4.3	Comparaison des algorithmes avec durée maximale non limitée	101
<b>5</b>	<b>Modèle par choix de valeur du signal</b>	<b>107</b>
5.1	Formalisme	108
5.1.1	La représentation de l'état	108
5.1.2	L'action	110
5.1.3	La récompense	111
5.1.4	Politique de l'agent et apprentissage	111

5.1.5	Profil de l'objectif et paramétrage de la fonction de récompense . . . . .	116
5.2	Évaluation des algorithmes à états . . . . .	121
5.2.1	Métrique pour l'évaluation de la vitesse et de la qualité de convergence . . . . .	123
5.2.2	Comparaison entre l'algorithme de SARSA et de Q-learning . . . . .	124
5.3	Discrétisation . . . . .	128
5.3.1	Discrétisation de l'espace-temps . . . . .	129
5.3.2	Discrétisation du temps sur la fréquence de décision . . . . .	132
5.4	Comparaison entre le système par choix de variation du signal et le système par choix de valeur du signal . . . . .	134
<b>A</b>	<b>Méthodes classiques d'apprentissage par renforcement</b>	<b>145</b>
A.1	Apprentissage par renforcement sans états . . . . .	145
A.2	Apprentissage par renforcement basé sur les états . . . . .	148
A.2.1	Interface entre l'agent et environnement . . . . .	150
A.2.2	L'environnement : processus de décision Markovien . . . . .	153
A.2.3	Tâches et objectif en apprentissage par renforcement . . . . .	155
A.2.4	Politique optimale . . . . .	157
A.2.5	Architecture et types d'algorithmes d'apprentissage par renforcement . . . . .	159
A.2.6	Méthodes par valeurs : La programmation dynamique . . . . .	163
A.2.7	Problème de prédiction : Monte-Carlo et Différence Temporelle . . . . .	167
A.2.8	Problème de contrôle : SARSA et Q-learning . . . . .	174
A.2.9	Les méthodes d'approximation . . . . .	175
A.2.10	Les méthodes basées sur les politiques : policy-gradient et acteur-critique . . . . .	178
<b>B</b>	<b>Profil du système par choix de variation du signal</b>	<b>181</b>
B.1	L'agent des pentes . . . . .	181
B.2	Le système complet . . . . .	181

# Introduction

Le secteur du bâtiment représente une part importante de la consommation d'énergie dans le monde, et ce, sur tout le cycle de vie du bâtiment, de la phase de construction, de l'usage à la fin de vie, et au recyclage. En effet, les bâtiments représentent près de 36 % de l'énergie totale consommée et 37 % des émissions de CO<sub>2</sub> liées à l'énergie en 2020 (d'après l'Agence Internationale de l'Énergie [2, 43]). En France également, le secteur du bâtiment représente près de 44 % de l'énergie consommée (d'après le Ministère de la Transition écologique et de la Cohésion des territoires [27]) et il est l'un des principaux contributeurs aux émissions de gaz à effet de serre (26 % des émissions françaises d'après le Centre Interprofessionnel Technique d'Études de la Pollution Atmosphérique [21]), ce qui en fait l'un des domaines clés de la lutte contre le réchauffement climatique. De nombreux efforts ont été entrepris depuis plusieurs décennies par la majorité des pays industrialisés pour réduire significativement les impacts environnementaux de ce secteur, et en particulier pour améliorer l'efficacité énergétique. Nous pouvons citer, par exemple, la loi française sur la transition énergétique de 2015 (dont le premier objectif est de « *Réduire les émissions de gaz à effet de serre et la consommation énergétique du bâtiment* » d'après le Ministère de la Transition écologique et de la Cohésion des territoires [28]). De plus, dans un contexte géopolitique complexe, maximiser son indépendance énergétique représente pour les territoires un enjeu récent, mais crucial. La performance énergétique étant un objectif majeur dans l'exploitation de bâtiment, il est permis de s'interroger sur l'économie d'énergie pouvant être réalisée par une gestion intelligente des ressources du bâtiment. Les *bâtiments intelligents* semblent offrir des opportunités pour répondre à ces enjeux de manière nettement plus efficaces qu'avec un bâtiment classique. Par exemple, un bâtiment intelligent pourra analyser les données d'occupation ainsi que les données liées aux usages des équipements afin de les exploiter de manière à réaliser des économies d'énergie.

Le concept de *bâtiment intelligent* est un terme difficile à définir de manière formelle, tout comme l'intelligence elle-même est une notion n'ayant jamais fait l'objet d'un réel consensus scientifique. Pour cette raison, la définition communément admise d'un *bâtiment intelligent* a évolué au cours du temps [14]. De nos jours, pour qu'un bâtiment soit considéré comme intelligent, il faut que ce dernier, soit *communicant* [3], *actionnable* [3] et que tous ses sous-systèmes (matériels et logiciels) soient *interopérables* [50] (tel que montré dans la figure 1) :

- Un bâtiment est *communicant* s'il peut collecter les données sur son évolution et s'il permet aux usagers de les visualiser de manière à faciliter l'exploitation du bâtiment. Ces données sont souvent assez hétérogènes et peuvent intégrer des données de capteurs tel que l'évolution de la température, de l'humidité, de la consommation d'énergie, etc, ainsi que des données sur l'activité des usagers.
- Un bâtiment est *actionnable numériquement* s'il permet à l'utilisateur de le contrôler via une

interface à distance.

- Un bâtiment est *interopérable*, si les sous-systèmes le composant arrivent à communiquer entre eux. L'interopérabilité nécessite que les protocoles et les formats soient uniformisés.

Bien que ces éléments soient nécessaires à l'exploitation du bâtiment, et suffisent généralement à *automatiser* certaines tâches liées à sa gestion, ils ne semblent pas suffire à garantir qu'un tel bâtiment ait un *comportement intelligent*.<sup>1</sup> Nous proposons de considérer qu'un bâtiment est intelligent s'il a la capacité à anticiper les changements et à s'adapter aux situations inattendues (conformément à la proposition faite dans [14]). Ces changements peuvent être divers et ils dépendent notamment du type de bâtiment considéré. Certaines activités sont en effet de nature à dégager de la chaleur (du fait par exemple de la présence de machines), ou nécessitent une intensité d'éclairage particulière.

Généralement, on considère que l'exploitation du bâtiment regroupe les trois axes suivants : le pilotage énergétique, la maintenance du bâtiment ainsi que la supervision. De manière plus détaillée :

- Le pilotage consiste à exploiter au mieux les ressources du bâtiment. Le but du pilotage énergétique est de fournir un service qualitatif aux usagers du bâtiment, et de minimiser l'énergie nécessaire pour fournir ce service à travers la gestion automatique des équipements du bâtiment.
- La maintenance s'efforce de garantir la meilleure disponibilité de ces ressources.
- La supervision permet de connaître l'état du bâtiment. Il apparaît cependant discutable de considérer que la supervision est un objectif en soi, car elle reste un moyen d'atteindre les deux autres objectifs.

Comme nous l'avons mentionné auparavant, dans cette thèse, nous nous intéressons au pilotage énergétique du bâtiment, en particulier au contrôle de la lumière.

Depuis la fin des années 2000, les technologies permettant de capturer des données, de les stocker et de les traiter se sont développées à un rythme accéléré. Cet essor technologique survenu d'une part sur les capacités de stockage (avec l'avènement du Big Data), et d'autre part sur les capacités de traitement (notamment avec l'apparition de GPUs hautement performants) a eu un impact considérable sur l'apprentissage automatique et l'intelligence artificielle. La phase d'exploitation (dont le pilotage) fait partie des domaines dans lesquels les attentes vis-à-vis de ces méthodes d'apprentissage automatique sont très fortes, y compris au niveau industriel.

Beaucoup de travaux ont été menés pour le contrôle de l'énergie dans les bâtiments. La plupart se sont focalisées sur des méthodes classiques telles que les méta-heuristiques [93, 73, 58]. Ces méthodes visent à optimiser une fonction objective faisant intervenir la consommation énergétique et/ou le confort des usagers avec potentiellement des contraintes portant sur le confort. Cependant, d'autres méthodes se basent sur les données collectées par le bâtiment afin de les utiliser dans le cadre du pilotage énergétique du bâtiment.

Plus précisément, l'une des questions traitées par la communauté scientifique est de concevoir des systèmes de contrôle capables de contrôler l'environnement de l'utilisateur de manière à satisfaire

---

1. Par ailleurs, un bâtiment peut ne pas être considéré comme intelligent même s'il est équipé de dispositifs qui le sont et qui fonctionnent de manière adaptative (s'il est équipé d'un compteur intelligent par exemple).

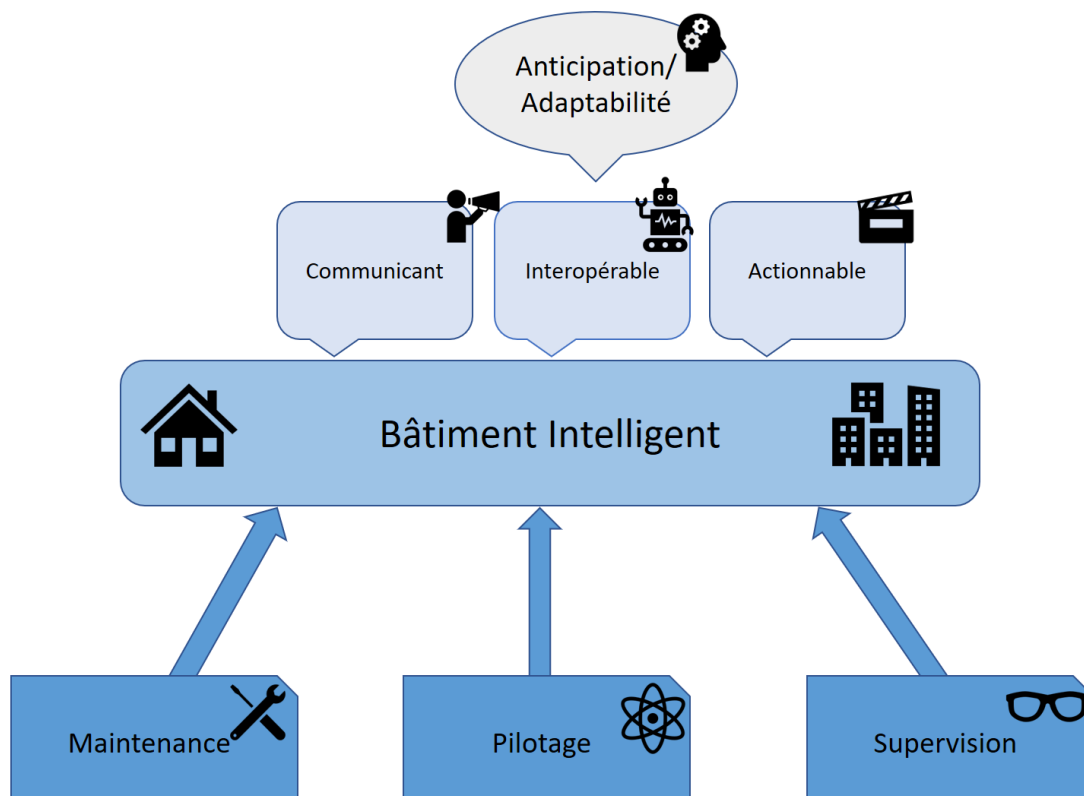


Figure 1 – Le bâtiment intelligent

leurs préférences tout en minimisant l'énergie nécessaire à cela. Cette question a attiré beaucoup d'attention ces dernières années, que ce soit sur le contrôle thermique [41, 76, 137], ou le contrôle de la lumière [20, 101]. Dans le cadre de notre thèse, nous proposons des algorithmes d'apprentissage par renforcement pour le contrôle lumineux de bâtiment connecté. Nous étudions et validons expérimentalement ces algorithmes, au travers de simulations basées sur des modèles comportementaux. Nous considérons un bâtiment connecté et un usager interagissant avec le bâtiment. Le problème que nous traitons est de savoir s'il est possible de contrôler le signal lumineux du bâtiment de façon à assurer le confort des usagers au moindre coût énergétique. Cet objectif représente un défi, car si l'usager se retrouve dans une situation d'inconfort, il pourra modifier la valeur du signal à sa convenance. Nous considérons par ailleurs que plus l'usager a réagi par le passé, plus il aura tendance à réagir dans le futur, même pour une valeur relativement forte d'intensité lumineuse. Ainsi, cette "mémoire du passé" dont l'utilisateur dispose rend l'environnement non stationnaire du point de vue du système, ce qui crée des difficultés algorithmiques du point de vue de l'apprentissage. Dans un tel contexte, de potentielles relations pourraient émerger entre le confort des usagers et la consommation énergétique.

Dans le cadre de cette thèse, nous présentons d'abord dans le chapitre 1 un aperçu sur les méthodes de contrôle de bâtiment classiquement considérées dans la littérature. Certaines de ces méthodes de contrôle s'appuient sur des indices censés mesurer le confort lumineux et thermique des usagers. Ces indices seront aussi cités dans le chapitre 1. Nous présentons dans le chapitre 2 les principes généraux des différentes méthodes d'apprentissage par renforcement que nous avons utilisé dans le cadre de cette thèse. Nous proposons ensuite dans le chapitre 3 un modèle comportemental de l'usager que nous utilisons pour étudier et valider expérimentalement les algorithmes que nous

proposons dans le chapitre suivant. Ce modèle sera étudié expérimentalement, nous en tirerons des conclusions importantes nous permettant de reformuler notre objectif d'apprentissage. À cet effet, nous présentons une étude expérimentale sur le comportement énergétique du système, qui donne lieu à une discussion et une interprétation concernant l'hypothèse d'un point d'équilibre entre énergie et confort. Toutes les hypothèses de ce modèle sont détaillées et justifiées sur la base des conclusions du chapitre 1. Sur la base de ces travaux, nous proposons et appliquons deux paradigmes pour représenter la décision du système : le paradigme par choix de variation du signal qui sera présenté dans le chapitre 4 et le paradigme par choix de valeur du signal qui sera présenté dans le chapitre 5. Dans le paradigme par choix de variation du signal, la décision du système est représentée par un couple composé de la vitesse de baisse du signal ainsi que de la valeur d'arrêt du signal. Cette décision porte sur l'ensemble du tracé des valeurs du signal au cours du temps. Dans le paradigme par choix de valeur du signal, la décision porte sur la valeur courante du signal et dépend d'information sur l'utilisateur ; elle s'effectue étape par étape. Nous comparons les performances de ces deux paradigmes selon plusieurs critères dans le chapitre 5. Enfin, nous passons en revue les contributions ainsi que les perspectives de ces travaux dans le chapitre 5.

# Chapitre 1

## Aperçu sur les méthodes de contrôle de bâtiment et de prédiction de confort des usagers

### 1.1 Introduction

L'objectif de ce chapitre est de donner un aperçu des méthodes de contrôle utilisées dans la littérature. Nous présentons aussi les différentes tentatives de modélisation de confort sur lesquelles certaines de ces méthodes se basent. Ce travail nous permettra de guider nos choix en rapport à la modélisation du système de contrôle que nous concevons et sur sa validation. Nous décrivons dans la section 1.2 les modèles proposés dans la littérature pour prédire le confort des usagers. Ces modèles incluent les modèles de confort thermique présentés dans la sous-section 1.2.1 et les modèles de confort visuel présentés dans la sous-section 1.2.2. Même si les approches de confort thermique ne constituent pas le sujet de notre thèse, nous avons jugé pertinent de les inclure dans notre travail compte tenu de la généralité de certaines de ces approches et la littérature abondante sur ce sujet. À cet égard, la conception du modèle de l'utilisateur que nous proposons dans le chapitre 3 se fonde sur des hypothèses caractérisant ces approches.

Nous présentons les méthodes appliquées au contrôle lumineux dans la section 1.3. Nous présentons ensuite les méthodes de contrôle thermique dans la section 1.4. Nous verrons que les méthodes de contrôle thermiques se basent classiquement sur des approches dites "strictes" qui exploitent un modèle de l'environnement thermique pour contrôler le signal. En outre, certaines de ces méthodes intègrent le confort des usagers à travers les modèles décrits dans la sous-section 1.2.1. Finalement, compte tenu de l'importance des travaux appliquant l'apprentissage par renforcement au contrôle thermique ainsi que leur proximité à nos travaux, la section 1.5 détaillera ces méthodes ainsi que le domaine d'application de celles-ci.

### 1.2 Confort des usagers

Le confort est défini comme étant l'état d'esprit dans lequel l'utilisateur est entièrement satisfait de son environnement physique. Le confort est donc une notion fondamentalement subjective et qu'il est difficile d'évaluer. Toutefois, la prise en compte du confort des usagers dans le pilotage énergé-

tique apparait nécessaire pour plusieurs raisons. D'une part parce qu'en l'absence de contrainte à ce niveau, minimiser l'énergie de fonctionnement revient bien souvent à interrompre strictement le service (ce qui reste l'action la plus économe en énergie dans ce cas). Ceci sans compter que dans le cas de bâtiments à usage tertiaire, des lois imposent en général un niveau minimum de confort. D'autre part, l'absence de prise en compte du confort des usagers pourrait entraîner une hausse inattendue de la consommation. Typiquement, dans un bâtiment trop peu chauffé, les usagers utilisent souvent un radiateur d'appoint, dont l'usage cumulé représente une surconsommation qui pourrait sans doute être évitée ou minimisée.

Dans cette section, nous traitons des différentes notions et des méthodes touchant au confort des usagers dans un bâtiment. Comme nous l'avons mentionné auparavant, nous nous pencherons sur le confort visuel, mais aussi sur le confort thermique en raison de la littérature abondante sur ce sujet et la généralité de certaines approches de confort thermique qui pourraient également s'appliquer au confort visuel. Nous verrons notamment que dans le contexte d'un système intelligent contrôlant le bâtiment, la notion de *confort adaptatif* [26] abordée plus loin implique que le comportement de l'utilisateur change dynamiquement en fonction du comportement du système. De plus, les facteurs influençant le confort ne se limitent pas à des éléments purement objectifs et mesurables. La sensibilité ainsi que le type d'activité que l'utilisateur est en train d'effectuer peuvent aussi changer son expérience du confort.

En outre, le confort visuel a été généralement pris en compte dans une approche qui privilégie les aspects énergétiques, et où ce confort joue un rôle assez marginal. Quelques auteurs [91] s'inspirent des travaux réalisés sur le confort thermique adaptatif pour proposer un système de gestion d'éclairage dynamique prenant en compte le temps, l'espace et les besoins des usagers. Ils soutiennent l'idée que le confort lumineux des usagers est adaptatif. Étant donné que la notion de confort adaptatif a été étudiée par la communauté scientifique du confort thermique, une grande partie cet état de l'art sur le confort de l'utilisateur se focalise sur les différents aspects du confort thermique.

### 1.2.1 Confort thermique

La sensation thermique est la conséquence de l'excitation du système nerveux cutané suite à des influences de l'environnement physique. Pour cette raison, les premières tentatives de modélisation du confort thermique utilisent des lois formulées mathématiquement afin de prédire le confort des usagers [37]. En effet, ces lois se basent sur des notions physiques et physiologiques, permettant de prédire le confort d'un individu compte tenu des conditions environnementales dans lesquelles il se trouve. Cette famille de modèles est appelée *modèles rationnels* et le modèle le plus populaire de cette catégorie est le modèle de Fanger et ses variantes [37, 35], qui ont été adoptés par l'ASHRAE et l'ISO pour formuler des normes portant sur les technologies de Chauffage, Ventilation et Climatisation (CVC).

Le modèle de Fanger est un modèle physiologique basé sur les échanges de chaleur et l'équilibre thermique qui a été formulé afin de prédire le confort des usagers dans un bâtiment [37]. Le modèle de Fanger évalue le confort d'un environnement physique sur l'*échelle de sensation thermique* (Thermal Sensation Scale [37]). Dans cette échelle (montrée dans la figure 1.1), le confort est évalué sur



sept niveaux (de -3 à 3), dans laquelle -3 correspond à "très froid", 3 correspond à "très chaud" et la sensation thermique neutre à 0 (0 correspond aussi à la sensation de confort). Le modèle de Fanger inclut une équation qui permet de calculer ce qui est appelé le *vote prédit moyen* dans une pièce donnée (*Predicted Mean Vote* ou PMV). Le vote prédit moyen correspond à la moyenne de vote des usagers de cette pièce sur l'échelle de sensation thermique donnée pour des conditions environnementales et individuelles fixées.

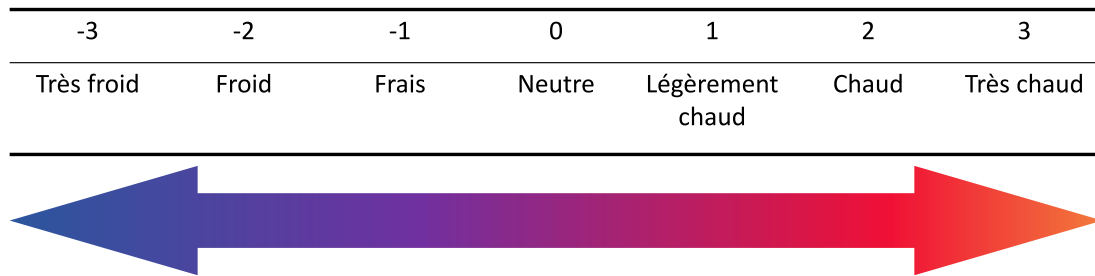


Figure 1.1 – Illustration de l'échelle de sensation thermique [37].

Dans le modèle de Fanger, les variables environnementales comprennent la température de l'air entourant le corps de l'utilisateur, la température radiative moyenne (qui représente les échanges de chaleurs dus aux radiations du soleil), la vitesse relative de l'air et l'humidité relative [56]. Quant aux variables individuelles, celles-ci sont le niveau d'activité des usagers ainsi que l'isolation thermique de leurs vêtements [56]. La vitesse relative de l'air et l'humidité relative peuvent affecter l'intensité des sueurs, ce qui affecte aussi les sensations thermiques. Le niveau d'activité des usagers et le niveau d'isolation de leurs vêtements agissent aussi sur le confort thermique, le premier par le biais de la chaleur émanant du corps humain (qui est liée au taux métabolique [8]) et le deuxième en limitant les échanges de chaleur entre la peau et l'environnement extérieur. En plus du PMV, le modèle de Fanger introduit un autre concept qui est le *pourcentage de personnes insatisfaites* (*Percentage of People Dissatisfied* ou PPD) [37]. Comme son nom l'indique, le PPD est la proportion de personnes qui sont insatisfaites de l'environnement thermique dans lequel elles se trouvent. Selon le modèle de Fanger, la PPD peut être calculée uniquement à partir du PMV de la façon suivante :

$$PPD = 100 - 95 \times e^{-(0.03353 \times PMV^4 + 0.2179 \times PMV^2)}$$

La qualité des modèles de confort basés sur le modèle de Fanger a été régulièrement remise en question par la littérature. Plusieurs auteurs ont noté des biais au niveau de ce modèle, soutenant ainsi l'idée que ses prédictions ne peuvent s'appliquer en pratique [17, 31, 32, 84]. En effet, le modèle de Fanger a été formulé à partir d'une population d'étudiants nordiques en milieu de laboratoire où l'environnement thermique est homogène et stationnaire. De ce fait, diverses critiques ont été formulées à son encontre, portant notamment sur le modèle dans sa globalité, son champ d'application géographique, son application à divers types de bâtiments, ainsi que sur les paramètres d'entrée du modèle.

Par exemple, dans [31], Kahkonen a effectué une étude dans laquelle le confort thermique a été évalué dans 17 entreprises sur 129 sites de travail, dans des boutiques, des magasins et des bureaux.

L'estimation du PMV indiquait que l'environnement thermique était trop chaud (les PMVs calculés étaient généralement inférieurs à ceux estimés). L'étude proposée par Fountain et al. [44] a montré qu'il existe des variations sur le confort thermique dans un groupe d'individus (d'une unité environ), ainsi que pour un même individu plusieurs jours de suite.

Des faiblesses du modèle de Fanger ont également été soulevées pour les climats chauds, dans les bâtiments à ventilation naturelle ou semi-contrôlée [24]. Par exemple, dans [24], de Dear a montré que pour les bâtiments à ventilation naturelle, la température intérieure considérée comme la plus confortable est significativement plus forte dans les climats chauds par rapport aux climats froids. La raison à cela peut être attribuée à l'augmentation du taux métabolique du corps humain en climat chaud, mais aussi au fait que les bâtiments à ventilation naturelle présentent souvent des situations thermiques transitoires (passage du chaud au froid ou du froid au chaud) et non homogènes, situations pour lesquelles le modèle de Fanger est inadapté [123]. L'échelle proposée par le modèle de Fanger a également été critiquée, dans la mesure où la sensation thermique neutre ne correspond pas nécessairement au confort thermique [25]. Par ailleurs, le modèle de Fanger ne prend pas en compte certains facteurs qui jouent un rôle non négligeable dans le confort thermique tel que l'âge et le sexe [60].

Une vision alternative à la modélisation rationnelle des sensations thermiques a été proposée par les partisans du *confort adaptatif* [26]. Le confort adaptatif repose sur le principe selon lequel

*si des changements de l'environnement se produisent et conduisent à un inconfort chez les usagers, les usagers agissent de manière à rétablir leur confort* [26].

Le corollaire de ce principe est donc « qu'un environnement confortable est un environnement que les usagers ne tentent pas de changer ». Dans la logique du confort adaptatif, l'utilisateur n'est pas passif dans son environnement, mais interagit constamment avec celui-ci afin de créer les conditions thermiques qui lui conviennent. L'utilisateur ne se limite pas à subir les conditions environnantes, mais agit de façon à créer ou rétablir les conditions qui le satisfont. L'environnement change aussi en réaction à l'utilisateur. L'utilisateur et l'environnement influent l'un sur l'autre de manière continue, en un phénomène comparable à une boucle de rétroaction. L'utilisateur est donc vu comme un système de contrôle thermique adaptatif (qui régule la température par rapport à son confort). Il peut répondre de plusieurs manières face à un changement indésirable : il peut agir de façon directe ou indirecte, consciente ou inconsciente. Plus exactement, l'utilisateur peut agir [60] : physiologiquement (par les tremblements, la sueur, les changements métaboliques, l'acclimatation, etc.), par des actions conscientes affectant son environnement (ouvrir les fenêtres, activer la ventilation ou la climatisation) ou par des actions sur lui-même (changer d'habits, de posture, d'activité) ainsi que par des phénomènes d'adaptation psychologiques. La réaction d'une personne à une température qui n'est pas parfaite dépend beaucoup de ses attentes, de sa personnalité et de son activité [83]. À côté de cela, il existe également une multitude de facteurs qui limitent les opportunités de contrôle et d'adaptation de l'utilisateur [60] : la pression sociale, la culture, la santé physique et mentale, l'affluence, etc.

Originellement, l'idée du modèle adaptatif avait pour but de montrer que le confort des utilisateurs peut être assez variable et qu'il dépend notamment de l'environnement dans lequel l'utilisateur évolue ainsi que de facteurs personnels le concernant. Or, une conséquence assez naturelle de ce point est

que les usagers ont tendance à préférer des températures plus élevées dans les climats chauds [26], ce qui vient contredire l'une des hypothèses fondamentales du modèle de Fanger (la sensation de confort ne change pas si les conditions environnementales ne changent pas). Le confort adaptatif pourrait donc proposer un moyen simple pour prédire la température de confort. Dans le confort adaptatif, on considère souvent que la température de confort thermique à l'intérieur d'une pièce dépend linéairement de la température de l'air extérieur (à l'exclusion de tout autre paramètre). Le confort adaptatif ignore les autres paramètres influençant le confort de l'utilisateur, car on considère que ces paramètres dépendent d'une manière ou d'une autre de la température de l'air [40]. De plus, dans le confort adaptatif, dans la mesure où l'utilisateur s'adapte à son environnement, ce dernier peut se sentir en situation de confort même pour des intervalles de température assez larges [40]. Ceci implique qu'il est possible de créer des conditions confortables pour l'utilisateur en se reposant peu sur les systèmes CVC [46]. L'un des avantages à cela est qu'un même usager pourra tolérer différents types d'environnement thermiques. De plus, réduire l'utilisation des systèmes de CVC aura bien entendu pour effet de réduire la consommation d'énergie. Nous notons par ailleurs d'autres éléments intéressants en lien avec le confort adaptatif. Les travaux de Iggo [62] montrent que les thermorécepteurs cutanés sont sensibles aux taux de changement de température (du moins chez certaines espèces). Et d'autres travaux comme ceux de Humphreys [59] montrent que les réactions des individus ne dépendent pas uniquement de la température courante, mais aussi de la suite des valeurs de température menant à celle-ci (ce qui s'apparente à une forme de mémoire sensorielle, hypothèse que nous exploitons dans l'ensemble des travaux présentés dans la section 3.4).

Différents organismes ont adopté le confort adaptatif en proposant chacun leurs formulations du confort adaptatif ainsi que leurs standards (ASHRAE Standard 55, EN 15251, GB/T 50785). Le confort adaptatif a aussi été intégré dans le modèle de Fanger [35]. Au-delà du fait que ces formulations diffèrent en fonction des organismes qui les proposent, il y a beaucoup d'imprécisions sur l'usage de ces modèles, notamment sur les valeurs des paramètres à prendre : par exemple, le nombre de jours à considérer dans le calcul de la température extérieure moyenne ainsi que les seuils de signification des modèles statistiques sous-jacents à ces modèles [16].

Certains travaux ont proposé des modèles tout à fait différents [143, 144, 145], prenant en compte les échanges thermiques intervenant entre l'utilisateur et son environnement, mais aussi au sein du corps humain. A priori, ces approches semblent être les plus à même de donner de bons résultats, car elles sont celles qui intègrent le plus d'informations pertinentes sur le ressenti thermique de l'utilisateur. Néanmoins, ces modèles n'ont été intégrés par aucun organisme international, il n'y a donc pas d'autorité qui pourrait les valider. De plus, ils n'ont pas été comparés à ceux utilisés par les normes en vigueur que nous avons mentionnée.

Parallèlement au confort adaptatif, un nouveau paradigme très proche de ce dernier est en train d'émerger avec l'avènement de l'apprentissage automatique : c'est *le confort personnalisé* [128]. Dans ce paradigme, le confort des usagers est amené à une granularité plus fine dans le sens où il est représenté à travers la prise en compte des facteurs qui créent des différences au niveau individuel dans le confort des usagers. Cela signifie qu'il n'y a pas de *modèle de confort* des usagers, mais *un modèle spécifique* qui est ajusté pour tel type d'utilisateur, ou *un modèle* ajusté pour telle personne ou tel type de bâtiment, etc. L'implémentation de système prenant en compte les différences in-

dividuelles n'est envisageable qu'à la condition d'avoir accès aux données sur le comportement de l'utilisateur et d'avoir l'infrastructure informatique nécessaire pour stocker et exploiter ces données. Dans l'hypothèse où elles sont disponibles, ces données sont ensuite utilisées pour entraîner un système d'apprentissage automatique prédisant le comportement des usagers. Un tel système est entraîné de façon à prendre en compte les différences individuelles entre les usagers, formant ainsi un système de confort personnalisé. Nous pouvons citer plusieurs exemples de travaux [76, 38, 135] ayant proposé des modèles de confort thermique se basant sur des approches d'apprentissage automatique. Lee et al. [76] ont proposé une méthode d'apprentissage automatique basée sur les approches Bayésiennes [55], entraînée sur des données réelles pour prédire le confort des usagers d'immeuble de bureau. Nous citons également les travaux dans [135], dans lesquels Wu et al. proposent une méthode d'apprentissage basée sur le Bagging [55] afin de prédire les sensations thermiques des occupants d'un dortoir.

La conclusion qui émerge est qu'il ne semble pas y avoir de consensus sur l'approche à prendre pour prédire le confort thermique des usagers, et ce, malgré les multiples tentatives qui ont été entreprises à cet effet. Dans l'état actuel de nos connaissances, toutes ces tentatives se sont avérées infructueuses compte tenu des difficultés à modéliser formellement le confort. Le confort est subjectif par nature et ne peut être modélisé mathématiquement comme le suggèrent les travaux sur le *confort adaptatif* [26]. Dans cette approche, l'utilisateur est vu comme un élément actif de son confort, il interagit avec le bâtiment afin de créer les conditions qui lui sont favorables. Dans le contexte d'un système intelligent contrôlant le bâtiment, le confort adaptatif implique que le comportement de l'utilisateur change dynamiquement en fonction du comportement du système. Même si le confort adaptatif a été essentiellement étudié dans le contexte du confort thermique, certains travaux s'en sont néanmoins inspirés afin de concevoir des systèmes de contrôle de lumière [91] (qui font l'objet de la section suivante). Toutefois, précisons que la difficulté à concevoir un modèle *prédictif* de l'utilisateur n'empêche pas d'émettre des hypothèses sur les *tendances* régissant son *comportement*. Nous proposons notamment dans le chapitre 3 une hypothèse sur la mémoire de l'utilisateur et l'impact de cette mémoire sur son comportement vis-à-vis du confort (de façon similaire aux travaux de Humphreys [59]).

Finalement, notons que l'évaluation du confort sera toujours une difficulté fondamentale de tout système de contrôle : comme l'utilisateur peut être vu comme un système qui optimise son confort et s'adapte, il aura toujours tendance à rechercher plus de confort s'il le peut, ce qui rend difficile l'évaluation et la comparaison même des systèmes de contrôle en termes de préférences des usagers. Ce point impliquerait potentiellement que l'utilisateur aura toujours l'impression que le système ne "comprend" pas assez ses besoins.

## 1.2.2 Confort visuel

La vision est un élément important de l'expérience des usagers dans les bâtiments. De ce fait, énormément d'efforts ont été entrepris pour la conception d'un environnement lumineux adéquat dans le bâtiment, de la conception architecturale des bâtiments à la conception, la fabrication et l'installation des lampes. Les éléments architecturaux prennent en compte la disposition des bureaux par rapport aux fenêtres et doivent être abordés avec soin [18, 98]. Ainsi, les lampes doivent

être disposées de sorte à éclairer l'ensemble des pièces du bâtiment afin de satisfaire les usagers vis-à-vis de leurs occupations. Par ailleurs, l'utilisation croissante des technologies numériques dans les bureaux posent de nouvelles difficultés pour les employés de bureau. Ainsi, les yeux doivent s'adapter sans cesse aux changements de luminosité, de la distance par rapport aux objets et de contraste. Ces conditions visuelles changeantes peuvent provoquer des sensations d'éblouissement, des difficultés de concentration, voire des maux de tête. Des douleurs musculaires peuvent s'ajouter à ces symptômes lorsque les employés tentent de modifier leur posture afin d'éviter l'inconfort dû à de mauvaises conditions visuelles (par exemple suite aux reflets d'une source lumineuse sur l'écran).

Il existe déjà des normes sur les pratiques recommandées pour l'évaluation de la qualité de l'éclairage (concernant, par exemple, l'éblouissement, les distributions de luminance, etc.) qui sont publiées dans les manuels d'éclairage [29]. Le problème principal concernant ces normes est qu'elles portent essentiellement sur des facteurs physiologiques. Or, des effets psychologiques (ou indirects) peuvent aussi se produire, car l'éclairage peut affecter l'attention, la motivation ainsi que le comportement. Il y a encore très peu de recommandations sur la manière d'utiliser les informations d'ordre psychologique dans la conception et l'analyse des espaces d'éclairage [70].

Au-delà de la nécessité d'identifier les conditions idéales du confort visuel pour des raisons de bien-être et de santé, la lumière artificielle pose aussi des défis supplémentaires, notamment par rapport à la consommation de l'énergie. En effet, l'éclairage électrique représente une part non négligeable de la consommation d'énergie dans le bâtiment. Dans un rapport de l'AIE (l'Agence Internationale de l'Énergie) [100], il est attesté que dans les pays de l'OCDE (Organisation de Coopération et de Développement Économiques), l'éclairage électrique atteint 19 % de la consommation énergétique (dans les pays non membres de l'OCDE, ces parts sont généralement plus élevées). Les travaux de Pohl et Werner [103] font état de valeurs similaires. L'éclairage est en outre responsable d'une part importante des coûts de maintenance des bâtiments, et la plupart des installations d'éclairage actuelles (environ 90 %) ont plus de 20 ans (elles utilisent des équipements d'éclairage vieux et inefficaces [103]). Pour cette raison, il est nécessaire de pouvoir formuler avec précision les objectifs de performances visées par les concepteurs, et ce, de façon à prendre en compte les besoins de l'utilisateur. Pour ce faire, plusieurs indices ont été proposés afin d'évaluer le confort visuel [29].

Il existe principalement quatre types d'indices : les indices portant sur *la quantité de lumière* [15], les indices portant sur *l'uniformité* [4] de la lumière, les indices portant sur *l'éblouissement* [29], et les indices portant sur *la couleur* [15] de la lumière. Généralement, les indices portant sur la quantité de lumière indiquent si la lumière est suffisamment présente dans l'environnement de l'utilisateur. Les indices portant sur l'uniformité reflètent la mesure dans laquelle la répartition de la lumière dans une pièce d'un bâtiment influence les sensations de l'utilisateur. Finalement, les indices portant sur l'éblouissement représentent l'impact de la brillance d'un environnement lumineux sur le confort des occupants du bâtiment.

Commentons plus avant les différents types de confort lumineux. En premier lieu, l'éblouissement se produit généralement lorsque la lumière directe du soleil pénètre dans la pièce et éclaire directement les yeux des occupants (ou se reflète des surfaces environnantes). L'éblouissement provoque ainsi une gêne chez l'utilisateur, un inconfort ou une perte de la performance visuelle et de visibilité. Il

est possible de distinguer deux types d'éblouissement [15] :

- L'éblouissement incapacitant, qui arrive quand l'intensité lumineuse est beaucoup trop forte pour l'observateur, ce qui l'empêche d'avoir une vision correcte.
- L'éblouissement d'inconfort, qui survient quand il y a une gêne chez l'observateur qui n'entrave pas pour autant ses capacités visuelles. L'éblouissement d'inconfort est caractérisé par un fort contraste sur différentes parties du champ visuel de l'observateur.

Quant aux indices sur la qualité de lumière, ceux-ci portent spécifiquement sur le rendu d'une couleur. Les recherches montrent que les gens ont tendance à préférer la lumière naturelle (ce qui permettrait de faire des économies d'énergie). Même si les différents types d'indices de confort lumineux sont corrélés entre eux, ceux-ci ne concernent généralement qu'un seul des aspects cités précédemment. Notons néanmoins que les travaux de recherche sur le confort visuel se sont essentiellement portés sur les indices mesurant l'impact de la quantité de lumière et ceux mesurant le degré d'éblouissement. La raison principale à cela est que les indices de qualité et d'uniformité de lumière dépendent beaucoup trop du type d'activité effectué. Les usagers peuvent préférer une lumière de couleur atypique dans l'intention de créer une certaine ambiance dans leurs espaces de travail ou de repos. Les événements faisant impliquer une foule de personnes tels que les concerts ou les spectacles ou les événements sportifs peuvent aussi exiger l'usage de diverses couleurs de la lumière avec différentes uniformités, etc.

Les indices de confort visuel diffèrent non seulement par ce qu'ils mesurent, mais aussi par une multitude d'éléments liés à leur utilisation, à la portée de ces indices ainsi qu'au protocole servant à les définir. Les différences entre les indices de confort visuel (servant entre autres à classifier leur usage) sont précisées comme suit [15] :

- Le fait que l'indice soit prévu pour la lumière naturelle ou artificielle. En effet, les usagers ont tendance à préférer la lumière naturelle à la lumière artificielle [22], et ceci même si la lumière naturelle est de plus forte intensité.
- La durée pendant laquelle est calculé l'indice. Les indices qui sont calculés seulement pour un instant donné sont appelés indice à *court-terme* alors que les indices dont le calcul implique de plus grandes durées sont appelées indice à *long-terme*.
- L'étendue spatiale pour laquelle est calculé l'indice. Si un indice est calculé pour un point donné de l'espace, alors il est appelé indice *local* et s'il est calculé pour un espace étendu (pièce, bureau, etc.), il est appelé indice *zonal*.
- L'indice peut être *unilatéral* (ayant un seuil de confort) ou *bilatéral* (ayants deux seuils de confort, une limite inférieure et une limite supérieure).

Comme montré dans certains travaux ([15, 102]), les différentes manières de classifier les indices sont le résultat de considération pouvant faire changer l'adoption d'un indice par rapport à l'autre. Le consensus est par exemple beaucoup moins fort sur les valeurs de seuil prises pour les indices bilatéraux que pour les indices unilatéraux, car il est bien plus difficile d'être certain des valeurs de ces indices avec deux seuils différents par indice qu'avec un seul seuil. Il en est de même pour l'étendue spatiale et temporelle des indices. Il est plus difficile d'interpréter les indices zonaux (et à long-terme). En revanche, les indices locaux (et à court-terme) sont moins représentatifs. Précisons que différentes communautés ont tendance à utiliser différents indices (le *Daylight Factor* [127] est

très utilisé par les architectes alors que le *Daylight Autonomy* [7] est plutôt exploité par les électriciens). Finalement, précisons que l'une des difficultés principales de ces indices (et même des indices sur l'éblouissement) est qu'ils ne sont pas toujours accompagnés de standards servant à définir les zones de confort de l'utilisateur. Étant donné que le confort visuel dépend de myriade de facteurs (le type d'activité ainsi que des données physiologiques et psychologiques des occupants du bâtiment, etc.), certains auteurs ne se sont pas avancés sur les valeurs de seuil le définissant.

Dans cette section, nous présentons, un peu plus en détail, les indices de confort mesurant *la quantité de lumière* et les indices de confort mesurant *l'éblouissement*. Nous précisons que les indices de confort visuel ne se reposent généralement que sur quelques notions de photométrie [118] (les indices de confort étant généralement formulés en termes de ces concepts). La figure 1.2 illustre les concepts de photométrie servant à formuler ces indices.

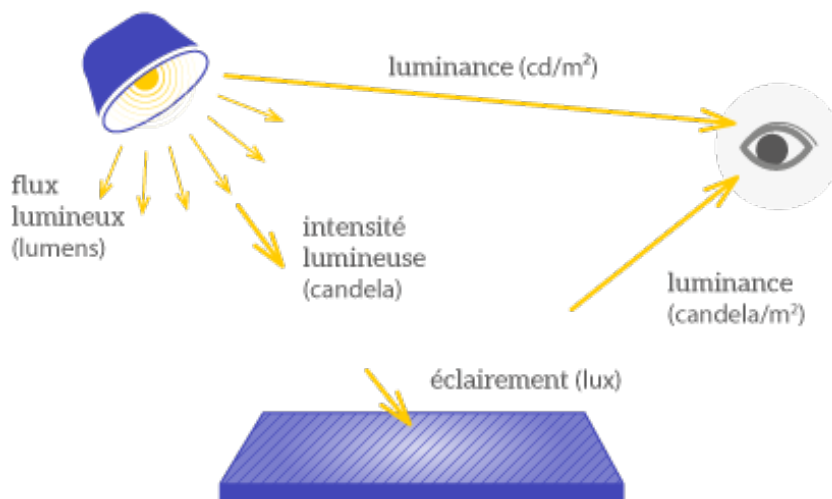


Figure 1.2 – Illustration des concepts de photométrie.

## L'éblouissement

Nous présentons dans ce qui suit les indices touchant *l'éblouissement* [29]. Les indices prévus pour prédire l'éblouissement sont nombreux, mais se reposent tous sur quelques concepts clé de photométrie (tel que la luminance [118]). Les premiers indices ayant été proposés par la communauté scientifique sont le *British Glare Index* [102] et le *Visual Comfort Probability* [52]. Ces indices représentent la base sur laquelle ont été conçus la majorité des indices d'éblouissement. Ces deux indices représentent deux manières différentes de formuler un indice d'éblouissement :

- La première partant sur une équation générale reliant la sensation d'éblouissement ressentie par

l'observateur à des éléments de son environnement, tel que le nombre de sources lumineuses, leurs intensités et leurs positions dans le champ de vision de l'observateur.

- La deuxième visant à prédire la sensation d'éblouissement d'un ensemble d'individus dans une pièce donnée en fonction de facteurs portant sur l'environnement lumineux de la pièce. Cette manière de procéder est très analogue à celle adoptée pour calculer le PMV (Predicted Mean Vote) associé au modèle de Fanger pour la prédiction du confort thermique [37].

Toutefois, ces deux indices sont connus pour mal évaluer les situations dans lesquelles la luminance est très intense, les situations dans lesquelles les sources lumineuses sont de grande taille, les situations pour lesquelles les luminaires ne sont pas accrochés au plafond, et même l'éblouissement dû à la lumière du jour. Pour répondre à ces difficultés, d'autres indices ont été formulés tels que le *CIE Glare Index* (CGI) [34], le *Unified Glare Index* (UGR) [77], et le *Daylight Glare Index* (DGI) [18].

Le *CIE Glare Index* (CGI) a été développé par un comité technique de la Commission Internationale de l'Éclairage (CIE) [34]. Essentiellement, cet indice est né d'efforts dont le but était de combiner les meilleurs points des principaux systèmes d'évaluation de l'éblouissement à l'époque. La formule du CGI est essentiellement divisée en deux parties : l'une décrivant l'environnement lumineux de la pièce et l'autre décrivant l'effet combiné de la luminance, de la taille des sources lumineuses et de la position des sources d'éblouissement.

Le CGI prend en compte l'adaptation de l'utilisateur aux sources lumineuses, contrairement au VCP et au BGI (sous forme de terme d'éclairement au niveau de l'œil). Ceci est crucial si des sources lumineuses de grande taille se trouvent dans le champ de vision de l'observateur. Dans ce cas, les sources de grande taille contribuent à l'adaptation visuelle de l'observateur, car elles recouvrent une portion plus grande du champ visuel (et se confondent en un sens avec l'arrière-plan).

Le CGI a le défaut d'avoir une formulation un peu complexe. Pour cette raison, le Unified Glare Index (UGR) a été développé à partir du CGI afin de produire une version simplifiée de cet indice [77]. Le UGR n'intègre plus la partie liée à l'adaptation (l'éclairement direct au niveau de l'œil), mais reste malgré cela assez adopté dans le monde. Tout comme les précédents indices, le UGR ne prend pas en compte les sources non uniformes, les sources indirectes et les sources complexes de lumière.

La majorité des indices d'éblouissement sont développés pour des conditions d'éclairage électrique et ne sont pas applicables aux situations où l'environnement est éclairé par la lumière du jour. La raison principale à cela est que, généralement, la taille des fenêtres est trop grande pour que ces indices puissent prédire de manière fiable l'éblouissement ressenti par les usagers. La communauté scientifique a donc proposé un autre indice, le Daylight Glare Index (DGI) qui est assez proche du BGI et qui tient compte de la lumière du jour [18]. Toutefois, cette variante a aussi été critiquée par les travaux d'Iwata et Tokura [63] qui attestent que le DGI a tendance à mésestimer l'effet d'éblouissement de la lumière du jour. En outre, certains chercheurs ([97, 63]) ont montré que différents partitionnements d'une source d'éblouissement uniforme aboutissent à différentes prédictions, bien que la source lumineuse soit perçue comme une seule source d'éblouissement par l'observateur (ce problème est retrouvé aussi pour le VCP, le BGI).



**Critiques** Il existe des travaux dont le but est d'évaluer les situations d'éblouissement en milieu de travail. Les employés peuvent faire face à des environnements lumineux très différents au cours d'une journée en raison du caractère variable de la lumière du jour et des conditions extérieures. En outre, de grandes variations ont été observées chez les individus en termes de sensation d'éblouissement (surtout pour l'éblouissement d'inconfort). Par exemple, dans les travaux de Luckiesh et Guth [79], une variation de luminance de ratio 5 :1 a été trouvée chez les sujets (les sujets ajustaient la luminance de la source pour indiquer leur limite entre le confort et l'éblouissement). D'autre part, dans les travaux de Osterhaus et Bailey [99], il a été constaté que le sujet le moins sensible avait besoin d'augmenter la luminance d'environ 100 fois pour obtenir le même niveau d'éblouissement subjectif que le sujet le plus sensible (l'expérience consiste à demander aux sujets de régler la luminance des sources d'éblouissement entourant un écran d'ordinateur afin d'obtenir un même niveau d'éblouissement). De plus, les réponses des sujets individuels étaient souvent incohérentes lors de l'évaluation d'une même situation.

Les études d'évaluation de l'expérience des usagers (permettant aux occupants d'un bâtiment d'exprimer leurs opinions sur les conditions d'éclairage et la conception globale du bâtiment) pourraient bien contribuer à combler les lacunes dans la conception et l'analyse de l'éclairage naturel des bâtiments. Mais en raison des changements fréquents des conditions de lumière du jour (même dans des intervalles de temps très courts), il est difficile de contrôler suffisamment l'environnement pour pouvoir comparer expérimentalement les réponses des occupants. Les études d'évaluation de l'expérience des usagers nécessitent également un nombre relativement important de sujets pour aboutir à des résultats statistiquement significatifs, ce qui rend tout travail de recherche sur ce sujet difficile à effectuer.

Même en l'absence de plaintes ponctuelles ou récurrentes de la part des occupants du bâtiment, il est souvent possible d'apporter des améliorations significatives de l'environnement lumineux des usagers. Des études confirment que le niveau de contrôle de bâtiment doit être assez fin afin de réaliser les avantages en termes d'économie d'énergie et de confort des occupants [11].

## Indices de confort visuel : la quantité de lumière

Dans ce qui suit, nous présentons les indices sur le confort visuel mesurant la quantité de lumière. Tout comme pour les indices d'éblouissement, la majorité des indices ayant été proposés pour mesurer la quantité de lumière se reposent sur quelques indices et notions photométriques de base, nous notons particulièrement *l'éclairement* [118], le *Daylight Factor* [127] et le *Daylight Autonomy* [7].

Le *Daylight Factor* [127] d'un point  $P$  est l'éclairement reçu sur le voisinage de ce point dans le plan horizontal le coupant divisé par l'éclairement horizontal au niveau de ce point si le ciel n'est pas obstrué par le toit. Il est utilisé par les architectes pour décider des tailles des fenêtres, du type de vitrage ainsi que du type de luminaires utilisés dans les pièces des bâtiments. Il a aussi été adopté par la Commission Internationale de l'Éclairage (CIE) en 1955 [82]. Le *Daylight Autonomy* a été proposé initialement par l'*Association Suisse des Électriciens* en 1989 [7]. Cet indice représente la proportion de temps occupé du bureau pendant lequel l'intensité lumineuse atteint ou dépasse un certain niveau.

Tous les indices de confort visuel mesurant la quantité de lumière se basent sur une variante des indices présentés précédemment. Par exemple, le *sDA* (Spatial Daylight Autonomy) [61] est défini comme étant la proportion d'un espace donné dans lequel la Daylight Autonomy *DA* dépasse un seuil donné. Nous pouvons citer aussi la *UDI* (Useful Daylight Illuminance) [89] qui est définie comme étant la fraction de temps dans une année où l'éclairement naturel horizontal, en un point donné d'une pièce, se situe dans une plage de valeurs donnée. L'intensité de l'inconfort visuel (IVD) prend en compte l'aire du tracé de l'évolution de l'éclairement au cours du temps qui est en dehors de seuils prédéfinis [110]. Il est néanmoins difficile de les exploiter en pratique, car les seuils proposés pour tous ces indices diffèrent assez souvent d'auteur en auteur. Et cette difficulté est accrue si l'on veut évaluer la quantité de lumière dans l'ensemble d'une pièce (et non pas sur un seul point). Enfin, nous pouvons conclure qu'il n'existe pas de norme universelle en matière de confort lumineux. La raison, une fois de plus, en est que le confort visuel (en termes de quantité de lumière et même d'éblouissement) varie d'individu en individu, qu'il dépend fortement de l'activité de l'utilisateur et de facteurs dynamiques et changeants (par exemple la position de la tête, la distance à la source lumineuse, l'angle de vue, etc.). Par ailleurs, les éléments (notamment sur le champ visuel de l'utilisateur) permettant de déterminer le niveau de confort visuel ne sont pas toujours disponibles, et mettre en place l'infrastructure pour les obtenir peut même être perçue comme étant beaucoup trop intrusive.

### 1.3 Méthodes de contrôle lumineux dans un bâtiment

Les premiers systèmes de contrôle de lumière étaient équipés de capteurs de mouvement PIR (Passive Infrared Sensor) et se contentaient d'allumer la lumière lorsque la pièce était occupée et de l'éteindre dans le cas contraire ([94, 45, 67]). Ces méthodes sont dites *réactives*, elles basent la prise de décision sur des informations en temps réel sans anticiper l'avenir. Pour décider si une pièce est inoccupée, ces systèmes considèrent un délai au-delà duquel la pièce est considérée comme telle si aucun mouvement n'est détecté par le capteur. C'est le *délai d'attente* (Time Delay ou TD). Bien entendu, certaines activités (comme la lecture) nécessitent un niveau de luminosité suffisamment élevé et peuvent être effectuées sans que l'utilisateur ait besoin d'être en mouvement pendant une longue période. Ceci est la raison pour laquelle ces méthodes basées sur les TDs sont aujourd'hui jugées inefficaces [90].

Pour pallier ce problème, certains travaux se basent sur des distributions des délais de repos des capteurs afin de décider si le système doit éteindre la lumière. Les travaux de Garg et Bansal [47] proposent d'intégrer l'activité des occupants par le biais d'une méthode statistique basée sur le calcul de distributions des délais d'attente. Le délai d'attente est adapté à chaque occupant et permet d'atteindre une économie d'énergie proche de 5 %. D'autres travaux comme ceux de Nagy et al. [90] se sont inspirés de Garg et Bansal [47] pour prendre en compte différents niveaux d'éclairage. Les travaux de Nagy et al. [91] intègrent aussi différents niveaux d'éclairage qui sont calculés à partir de distributions sur le niveau de luminosité choisi par les usagers. Notons que ces méthodes sont assez simples. Par ailleurs, elles utilisent des statistiques descriptives sur l'usage et le comportement des occupants (tel que la médiane, la moyenne, les intervalles de confiance, etc.)

pour contrôler le signal lumineux. Elles ne sont pas robustes aux changements, car elles évoluent de manière assez lente, même s'il y a une modification significative dans le comportement de l'utilisateur.

Une méthode plus directe pour contrôler la lumière consiste à optimiser une fonction objective donnée. Cette approche nécessite un modèle de confort visuel basé sur des connaissances sur les préférences des usagers. Plusieurs modèles ont été proposés pour représenter le confort des usagers en termes de confort visuel (voir la sous-section 1.2). Toutefois, la majorité de ces modèles présupposent la connaissance de certains éléments de l'environnement qui ne sont pas facilement accessibles. Un modèle peut par exemple avoir besoin d'informations sur l'inclinaison de la tête, le nombre de sources d'éblouissements dans le champ visuel de l'utilisateur, etc. Les études sur la fiabilité de ces prédictions sont rares. Mentionnons tout de même Meerbeek et al. [85] qui a étudié le comportement des usagers dans 40 bureaux d'un même bâtiment équipé de stores intelligents, montrant que la majorité (près de 77.5 %) des occupants du bâtiment finissent par désactiver la gestion automatique des stores.

Une alternative à cette approche consiste à utiliser l'apprentissage par renforcement pour le contrôle de la lumière. En effet, l'apprentissage par renforcement est une famille de méthodes qui permet au système d'apprendre un comportement spécifié par l'optimisation d'un signal de récompense provenant de l'environnement, même en l'absence de modèle explicite de l'environnement. Compte tenu du fait que les modèles de confort lumineux semblent peu fiables en plus d'être difficile à mettre en place, la communauté scientifique s'est penchée sur l'application des approches d'apprentissage par renforcement au contrôle lumineux. Nous notons deux travaux [20, 101] qui proposent une telle approche. Nous présentons ces travaux en détail dans les paragraphes suivants.

Dans les travaux de Cheng et al. [20], une méthode de contrôle basée sur Q-learning (voir l'annexe A pour une description détaillée de cette méthode) intégrant les retours des usagers est développée afin d'actionner conjointement les stores et les luminaires d'une pièce. La représentation de l'espace d'état se compose de l'état de confort de l'utilisateur (obscurité, confort ou éblouissement), de l'état de la lumière (nombre de luminaires allumés) et de l'état du store (l'angle et la position du store). La partie de l'espace d'état relative au confort de l'utilisateur est apprise à travers les retours de l'utilisateur sur une interface mobile. Les actions sont des incréments possibles dans le nombre de luminaires allumés ainsi que dans la position et l'angle d'ouverture des stores. La récompense consiste en une moyenne pondérée entre une partie représentant l'énergie et une partie représentant le confort. La partie représentant le confort est calculée en utilisant le modèle de confort appris à travers les retours de l'utilisateur. La partie énergétique est linéaire en fonction de l'énergie électrique consommée (en raison du fonctionnement des luminaires). L'une des améliorations de cette méthode par rapport au Q-learning classique est de mettre à jour la politique uniquement sur les états "intéressants" (définis suivant les réactions de l'utilisateur). De plus, les décisions ne sont pas prises à chaque étape, mais à chaque fois que l'utilisateur envoie une plainte ou qu'un délai prédéfini prend fin.

Dans les travaux de Park et al. [101], un contrôleur d'éclairage basé sur l'apprentissage par renforcement et prenant en compte les retours de l'utilisateur a été introduit et validé expérimentalement en utilisant, entre autres, le LCR (*Light to Comfort Ratio*), une métrique équilibrant confort et énergie proposée par les auteurs. L'utilisateur est situé dans une pièce où il peut librement allumer et

éteindre l'interrupteur de lumière. L'espace d'état comprend l'occupation de la pièce (pièce occupée ou inoccupée), la position de l'interrupteur (allumé ou éteint), le niveau de lumière intérieure (confort, sombre ou lumineux) et la période de la journée (lumière croissante, milieu de journée, lumière décroissante et sombre). Les actions disponibles pour le système sont l'allumage et l'extinction de la lumière. Chaque jour, les données relatives à l'éclairage et aux transitions sont enregistrées. En utilisant ces données, le système apprend la partie de la représentation de l'état qui inclut les niveaux de lumière et la période de la journée. Les probabilités de transition sont également apprises et le modèle de transition est utilisé pour évaluer les valeurs d'action et mettre à jour la politique à utiliser le jour suivant. Le système choisit des stratégies chaque minute et suspend le système de contrôle pendant 30 minutes chaque fois que l'utilisateur intervient.

La critique principale pouvant être faite à l'égard des travaux de Cheng et al. [20] et de Park et al. [101] est qu'ils ne proposent que deux niveaux de luminosité (allumé ou éteint). Ces méthodes sont donc limitées, à la fois en termes de confort pouvant être apporté à l'utilisateur et en termes d'économie d'énergie.

## 1.4 Méthodes de contrôle thermique dans un bâtiment

Les méthodes de contrôle thermique dans le bâtiment recouvrent une large gamme de techniques. Elles ont été beaucoup plus étudiées que les méthodes de contrôle lumineux. Nous pouvons isoler différentes manières de les classer. Nous distinguons :

- Les méthodes dont le principe est d'atteindre une valeur de référence de la manière la plus performante énergétiquement [65, 139]. Cette valeur dite de *consigne* peut être ou non modifiée par l'utilisateur. Ces méthodes peuvent être vues comme des méthodes d'automatisation.
- Les méthodes qui prennent en charge le confort ou le comportement des utilisateurs [78] dans le contrôle thermique. Cela revient à déterminer la valeur de référence, qui n'est pas un paramètre d'entrée comme dans les méthodes précédentes.

Remarquons que ces deux approches ne sont pas mutuellement exclusives. Il est possible de déterminer une valeur de consigne algorithmiquement, puis de s'appuyer sur une méthode de contrôle pour s'efforcer d'atteindre cette valeur.

Il est aussi possible de catégoriser ces méthodes suivant un angle différent : le type d'approche choisi pour contrôler le signal. Dans le cadre de cette thèse, nous présentons un aperçu des méthodes de contrôle de bâtiment selon le type d'approche retenue pour le contrôle. Ces approches sont synthétisées dans la table 1.1. Cette classification retrace le développement historique de ces méthodes. Ainsi, les **méthodes classiques de contrôle** sont les premières méthodes à être proposées. Elles sont aussi les plus utilisées actuellement dans le monde industriel. Nous distinguons aussi les **méthodes de contrôle strict** qui sont un peu moins utilisées dans la pratique, mais qui permettent de modéliser des situations plus complexes. Et finalement, le reste des méthodes constituent les **méthodes de contrôle souple** dont l'usage est encore largement limité dans le contexte de travaux de recherche.

Type d'approche	Méthodes
Contrôle classique	Méthode on/off Méthode par PID
Contrôle strict	MPC Contrôle optimal
Contrôle souple	Métaheuristiques Apprentissage supervisé Apprentissage par renforcement

Table 1.1 – Aperçu sur les approches de contrôle thermique.

Dans les systèmes les plus anciens proposés dans la littérature [71, 65, 139], et que l'on retrouve dans de nombreux bâtiments en exploitation, le contrôle énergétique (plus exactement le niveau d'activation des équipements et le moment où ces équipements sont activés) est effectué avec des méthodes dont le principe est très simple. Nous retrouvons notamment les méthodes de contrôle on/off et les méthodes de contrôle par correcteur PID (Proportionnel, Intégral, Dérivé). Ces deux types de méthode constituent les **méthodes classiques de contrôle**. Généralement, ces méthodes servent à atteindre une valeur de référence de la manière la plus performante énergétiquement.

Dans le contrôle on/off, l'élément contrôlé a deux états possibles : *activé* et *désactivé*. Le contrôle se base sur un ensemble de règles et de conditions déterminant si le système est activé. Le principe du contrôle PID (reposant sur les phénomènes de l'hystérésis) est de réduire l'écart entre les valeurs retournées par le système de contrôle et les valeurs souhaitées [71, 65, 139]. Toutefois, l'inconvénient principal de ces méthodes est qu'elles sont beaucoup trop rudimentaires : tout d'abord, elles ne permettent pas d'optimiser directement la consommation d'énergie, elles ne font que suivre une valeur de consigne en se basant sur un modèle énergétique à court-terme. Compte tenu des caractéristiques non-linéaires de l'environnement thermique des bâtiments, les solutions prodigués par ces méthodes sont loin d'être optimales. Par ailleurs, elles ne tiennent pas compte du confort de l'utilisateur.

D'autres méthodes intégrant des modèles relativement évolués sur la dynamique thermique des bâtiments ont été proposées dans la littérature : nous pouvons citer les méthodes d'optimisation continues qui regroupent le contrôle optimal [104] ainsi que les méthodes par MPC (*Model Predictive Control*) [12]. Ces méthodes tentent de donner des solutions exactes à des problèmes d'optimisation en se basant le plus souvent sur un modèle explicite de l'évolution de l'environnement (ce modèle étant généralement un modèle basé sur la thermodynamique du bâtiment). Pour cette raison, ces méthodes sont dénommées **méthodes de contrôle strict**. Dans les travaux utilisant des techniques de contrôle optimal, la fonction objective est parfois l'énergie consommée, mais elle peut aussi intégrer le confort thermique, ainsi que la réduction des coûts [142]. Les méthodes basées sur le MPC contiennent classiquement trois éléments : un modèle prédictif (représenté explicitement), une fonction objective, ainsi qu'une loi de contrôle. En général, la fonction objective fait intervenir l'énergie consommée ainsi que le coût. Plusieurs travaux [81, 131] ont appliqué le MPC au contrôle thermique. D'autres variantes de ces deux types d'approches ont été appliquées dans le contrôle thermique. Citons le *contrôle robuste* [80] (basé sur une prise en compte des incertitudes des systèmes) et les *méthodes de contrôle non-linéaires* [53] (l'aspect non linéaire concerne ici la fonction objective).

En théorie, les **méthodes de contrôle strict** devraient être celles pouvant donner les meilleurs résultats [19, 132]. Toutefois, les performances et la fiabilité des méthodes de contrôle strict dépendent fortement de la précision du modèle thermodynamique du bâtiment. Ce modèle doit également être exploité de manière efficace à l'aide d'outils mathématiques et numériques afin de pouvoir contrôler le bâtiment en temps réel. Des choix doivent être faits par rapport aux éléments à intégrer au modèle ainsi qu'aux hypothèses qui en forment la base. Toutefois, la température du bâtiment (cela vaut aussi pour les autres facteurs influençant la consommation d'énergie et le confort des usagers) est affectée par de nombreux facteurs. Nous pouvons citer notamment la structure et les matériaux du bâtiment, l'environnement (par exemple, la température ambiante, l'humidité et l'intensité du rayonnement solaire) et les gains de chaleur internes des occupants, des systèmes d'éclairage et d'autres équipements. En conséquence, la température du bâtiment présente souvent des comportements imprévisibles dans le cadre d'une modélisation incomplète.

À cet égard, les bureaux d'étude travaillent en général avec des modèles de granularité assez large (essentiellement des modèles basés sur des équations décrivant les interfaces thermiques entre les différents espaces d'un bâtiment). Ces modèles sont assez simplifiés, et dans l'ensemble, il est souvent assez difficile de développer un modèle de la dynamique thermique du bâtiment qui soit à la fois précis et qui puisse être exploité de façon à contrôler de manière optimale le chauffage, la ventilation et la climatisation (CVC) [19, 132]. Au-delà de ce point, ce modèle ne serait plus valable dès que le bâtiment connaîtrait des changements structurels assez significatifs (travaux, changements des équipements de CVC, etc.), ce qui rajoute des coûts et des délais de réalisation supplémentaires. D'autre part, les travaux récents de la communauté scientifique autour de la modélisation thermique se penchent sur des modélisations fines et réalistes physiquement, basés sur les réseaux de neurones [95]. Dans une approche par méthode de contrôle strict, l'usage de ces modèles fins apparaît préférable. Leur usage n'est pas encore répandu au niveau industriel, mais nous pouvons nous attendre à ce que cela évolue dans les années à suivre. Ces méthodes nécessiteront tout de même une expertise importante pour être utilisées dans le contexte de l'exploitation de bâtiment. Cela implique, au-delà du travail scientifique et technologique, une vraie évolution des métiers liés au bâtiment. Malgré leur efficacité, les méthodes de contrôle strict sont donc assez coûteuses à implémenter et à maintenir (en termes de temps et de ressources financières et humaines) et elles sont, à ce jour, peu robustes aux changements [19, 132].

Face aux difficultés des approches basées sur le contrôle strict, il existe d'autres méthodes dont le principe n'est plus de trouver des solutions exactes à l'aide d'un modèle précis de l'environnement, mais simplement de contrôler l'environnement thermique à l'aide d'heuristiques reflétant ses tendances les plus saillantes ou en exploitant les données collectées sur son évolution. L'objectif de ces approches est de trouver une manière de contrôler l'environnement thermique qui ne soit pas optimale (mais qui s'en rapproche tout de même). Les approches qui se fondent sur ce principe sont appelées **méthodes de contrôle souple**. Ces approches regroupent l'apprentissage automatique ainsi que les approches par logiques floues et les métaheuristiques.

Différentes métaheuristiques ont été appliquées dans le domaine du CVC pour l'optimisation de la consommation énergétique tel que les algorithmes génétiques [93], ainsi que l'optimisation par essaim de particules [73]. Toutefois, ces méthodes se basent sur la recherche itérative de solutions et

sont assez consommatrices en termes de temps et de ressources de calcul. Elles ne peuvent donc pas être appliquées directement en pratique. En outre, il n'y a pas de manière canonique pour intégrer des connaissances à priori à ces algorithmes. Par ailleurs, les métaheuristiques exigent un environnement de simulation dans lequel elles devront opérer au préalable. Ces approches [93, 73] nécessitent souvent un modèle rationnel du confort de l'utilisateur (voir la sous-section 1.2.1) afin de calculer la fonction objective des solutions.

Afin de traiter les inconvénients des métaheuristiques, des méthodes de contrôle par logique floue ont été utilisées dans le contexte du CVC [64]. Le contrôle par logique floue [96] tente de déduire des inférences utiles à partir de représentations de connaissances humaines qui sont traitées par la machine. Les méthodes de contrôle par logique floue sont constituées de trois étapes :

1. L'étape de *fuzzification* qui consiste à convertir les données brutes perçues par le système (la température) en concepts abstraits (ou "flous") interprétables par l'être humain (chaud/froid). La *fuzzification* renvoie pour chaque concept une valeur de vérité qui est entre 0 et 1 (et non des valeurs binaires).
2. L'étape d'inférence qui consiste à inférer algorithmiquement (avec des instructions conditionnelles) ce que devra faire le système compte tenu des valeurs en entrée. Dans cette étape, l'action prend aussi la forme de concepts flous (par exemple, chauffer "juste un peu") représentées sous forme de valeurs entre 0 et 1.
3. La dernière étape qui est la *défuzzification*, convertit les actions de contrôle "floues" en actions physiquement interprétables. Finalement, cette action est appliquée dans l'environnement.

Il est manifeste que les méthodes de contrôle par logique floue sont très similaires aux systèmes experts, à la différence que les inférences se font sur des concepts flous de façon analogue à la manière dont l'être humain raisonne. Même si les méthodes de contrôle par logique floue sont très utilisées dans différents systèmes de contrôle, le fonctionnement de ces méthodes exige tout de même des données venant d'experts sur les systèmes physiques (ainsi que sur les usagers dans le cas d'un système de contrôle thermique).

Les méthodes par apprentissage automatique peuvent permettre de contourner les inconvénients des métaheuristiques et des méthodes par logique floue (sous réserve de la disponibilité des données). À cet égard, ces approches sont basées sur les données, ce qui signifie que leurs performances s'améliorent à priori à mesure que l'algorithme traite plus de données. Parmi les méthodes d'apprentissage qui ont été appliquées au contrôle thermique, les plus populaires sont celles basées sur les *réseaux de neurones* [49]. Leurs performances sont accrues quand les données et les ressources de calculs sont disponibles en abondance, et permettent d'avoir des systèmes de plus en plus performants.

Pour rappel, la majorité des méthodes d'apprentissage utilisées en pratique se basent sur *l'apprentissage supervisé* [49]. Le principe général de l'apprentissage supervisé consiste à apprendre à faire des prédictions à partir d'exemple annotées. Plus exactement, l'objectif est d'apprendre à paramétrer, de manière automatique, une fonction d'approximation de façon à minimiser l'écart entre les valeurs estimées et ces annotations. Les réseaux de neurones [66] ont été utilisés pour le contrôle thermique dans lequel le confort a été prédit en utilisant le modèle de Fanger (voir la sous-section 1.2.1).

Nous insistons sur le fait que l'inconvénient des méthodes par apprentissage supervisé est qu'elles exigent des données en grande quantité, et que par ailleurs, ces données doivent impérativement être étiquetées. De plus, elles sont mal adaptées aux tâches qui requièrent d'optimiser un objectif donné, elles sont plus adaptées à la prédiction. Pour pouvoir apprendre une politique qui optimise un objectif donné, il faut disposer en amont de données sur les performances de cette politique optimale (ou d'une politique proche de l'optimale), ce qui est rarement faisable en pratique (notamment dans le cas d'un bâtiment intelligent). Il est d'autant plus difficile de faire du contrôle avec de l'apprentissage supervisé qu'il faut que les données soient assez diverses afin de trouver la meilleure politique (alors que les méthodes d'apprentissage supervisé sont prévues pour les environnements stationnaires). Elles sont aussi très énergivores, et leur usage dans un objectif de réduction de consommation énergétique peut sembler contre-productif. L'apprentissage supervisé apparaît donc difficile à mettre en place pour le pilotage thermique du bâtiment au vu des raisons susmentionnées.<sup>1</sup>

Il est donc nécessaire de considérer des approches qui peuvent s'améliorer dans le temps et qui sont aussi capables de gérer des fonctions objectives diverses. Pour cela, de plus en plus de travaux récents utilisent l'apprentissage par renforcement pour le contrôle thermique [140, 146, 132]. L'apprentissage par renforcement permet d'apprendre une politique de contrôle par interaction avec l'environnement sans modèle préalable de ce dernier. Il permet aux concepteurs de s'affranchir du besoin de collecter et d'étiqueter les données de contrôle. Le lecteur souhaitant revenir sur les fondements des approches basées sur l'apprentissage par renforcement pourra consulter l'annexe A. Nous présentons dans la section qui suit les travaux utilisant l'apprentissage par renforcement dans le contexte du contrôle thermique.

## 1.5 Méthodes de contrôle thermique utilisant le RL

L'apprentissage par renforcement (voir l'annexe A) a connu un succès fulgurant et a permis à la machine de conquérir énormément de domaines qui lui étaient jusqu'à présent inaccessibles, tel que les jeux vidéo d'arcade [87], le jeu de go [111], le jeu starcraft [126]. Il a été utilisé aussi pour découvrir des algorithmes efficaces pour le produit matriciel [39]. Le succès des algorithmes d'apprentissage par renforcement dans de nombreuses applications a abouti à une question ouverte sur la façon de concevoir et de mettre en œuvre ces algorithmes dans les systèmes de contrôle des bâtiments. Les méthodes d'apprentissage par renforcement n'ont pas été examinées de manière exhaustive dans le contexte des systèmes CVC [54]. Le fait que les bâtiments numériquement opérables ne sont pas encore très nombreux, le manque de données d'exploitation de bâtiment, le fait que la plupart des recherches ne se sont focalisées que sur l'aspect physique du contrôle de bâtiment (et non pas sur le comportement des occupants), et l'explosion combinatoire du nombre de couples d'état-actions, ont fait que les méthodes d'apprentissage par renforcement n'ont commencé à être appliquées au contrôle de bâtiment que récemment.

---

1. Il est toutefois possible d'utiliser l'apprentissage supervisé pour implémenter une méthode de contrôle en l'hybridant avec une des méthodes traditionnelles présentées auparavant (au risque de perdre les avantages fournis par l'utilisation de l'apprentissage).



Les avantages des méthodes d'apprentissage par renforcement sont qu'elles n'exigent pas que les données soient indépendantes et identiquement distribuées, qu'elles sont plus flexibles, qu'elles intègrent l'action dans leur modèle, là où d'autres techniques d'apprentissage ne s'intéressent qu'à la prédiction, et qu'elles peuvent être combinées avec l'apprentissage profond pour une convergence plus rapide et efficace quand le nombre d'états-actions est trop grand. Nous présentons dans ce qui suit les différentes méthodes d'apprentissage par renforcement ayant été appliquées dans le contexte du bâtiment, en précisant à chaque fois l'objectif visé ainsi que le domaine d'application.

### 1.5.1 Approches tabulaires

Ce sont les méthodes qui maintiennent les estimations associées aux couples d'états-actions dans la mémoire. La méthode la plus utilisée dans ce contexte est le Q-learning [130]. Les méthodes qui ont été utilisées visent à réduire le nombre d'états-actions du modèle. Une première stratégie a été d'utiliser un environnement simulé [107, 108], d'autres méthodes sont basées sur la pré-estimation de l'ensemble des états-actions [51], l'utilisation du Q-learning avec trace d'éligibilité [116], ou encore l'utilisation de FSD (fuzzy state discretization) [141].

Comme nous l'avons précisé dans la section 1.4, la consommation énergétique du bâtiment est influencée par de très nombreux facteurs (particulièrement si le confort thermique est aussi intégré dans l'objectif du système de contrôle). De ce fait, pour permettre un contrôle thermique performant, il faut que le système de contrôle puisse prendre en compte ces facteurs. Malheureusement, l'intégration de ces facteurs nécessite que l'algorithme de contrôle soit en mesure de gérer l'explosion combinatoire qui en résulte. Pour cette raison, des méthodes d'approximation ont été utilisées dans le contexte du contrôle thermique.

### 1.5.2 Approches utilisant les fonctions d'approximation

Nous invitons le lecteur à se référer à l'annexe A pour plus de détails sur les méthodes d'approximation. Les méthodes d'approximation sont utilisées quand il y a beaucoup trop de couples d'états-actions, l'objectif étant de tirer parti des données collectées sur les trajectoires déjà visitées afin d'accélérer la convergence. Dans les méthodes basées sur les valeurs, les q-valeurs (les valeurs des actions) sont approchées en supposant que celles-ci dépendent d'un paramètre  $w$  ( $q'(s, a, w) \approx q_\pi(s, a)$  où  $\pi$  est la politique actuelle de l'agent). Ce paramètre  $w$  est modifié légèrement à chaque itération de descente de gradient de sorte à mieux estimer les q-valeurs. Dans les méthodes basées sur la politique, la fonction d'approximation concerne la politique de l'agent. La politique est améliorée en modifiant de façon itérative, par descente de gradient, le paramètre  $\theta$  dont la politique dépend [105].

Concernant les méthodes basées sur l'approximation de la fonction de valeur, deux types d'approches sont à considérer : les méthodes incrémentales qui modifient le paramètre transition par transition et les méthodes par batch qui prennent en considération un ensemble de transitions avant de modifier le paramètre. Parmi les méthodes incrémentales, Approximate SARSA( $\lambda$ ) [116] est un exemple classique. Ces méthodes ont été appliquées dans les travaux de Gracia et al. [1] afin d'optimiser la consommation énergétique ainsi que dans ceux de Dalamagkidis et al. [68] qui prennent en compte le confort thermique (en s'appuyant sur le modèle de Fanger présenté dans la sous-section

1.2.1). Les travaux de Xu et al. [136] utilisent la méthode RLS-TD qui repose sur le principe de différence temporel (Temporal Difference) et modifie les poids par la méthode des moindres carrés. Concernant les méthodes par batch, nous pouvons citer Ruelens et al. [36] qui utilisent fitted Q-iteration [23] afin de contrôler un chauffe-eau électrique de façon à optimiser les coûts liés à sa consommation électrique.

Les méthodes d'apprentissage par renforcement intégrant l'apprentissage profond comme fonction d'approximation ont été appliquées dans le domaine du contrôle thermique. Nous présentons dans ce qui suit quelques exemples de ces méthodes. Celles-ci ont été comparées aux méthodes classiques (basées sur des règles décrites explicitement), et elles semblent les surpasser en termes de performances énergétiques.

Nous notons dans les travaux de Wei et al. [132] l'utilisation du Deep Q-learning (DQN) [86] dans le cadre du contrôle thermique d'un bâtiment en environnement de simulation. L'objectif est de contrôler une pompe à chaleur d'une pièce de façon à trouver un équilibre entre confort et énergie. Dans ce système, la récompense est composée d'un terme représentant la consommation d'énergie (l'énergie est considérée comme linéaire en fonction du débit d'air de la pompe) et d'un second terme représentant le confort de l'utilisateur (à quel point la température à l'instant courant est en dehors d'un intervalle de confort considéré).

Nous notons aussi les travaux de Zhang et al. [146] où une implémentation d'une version parallèle de l'acteur-critique (appelée *Asynchronous Advantage Actor Critic* ou A3C) [72] a été entraînée par simulation à contrôler l'environnement thermique d'un bâtiment. L'état est constitué des dernières observations de caractéristiques décrivant l'environnement intérieur et extérieur du bâtiment. La récompense inclut comme précédemment une composante représentant l'énergie et une composante représentant le confort (là encore se basant sur le modèle de Fanger 1.2.1). La particularité de cette approche est qu'elle considère aussi différents seuils de température quand le bâtiment est inoccupé.

Finalement, nous notons dans les travaux de Yu et al. [140] l'utilisation d'une méthode innovante de contrôle thermique d'un bâtiment commercial basée sur le soft acteur-critique [5] dont l'apprentissage est guidé par un mécanisme d'attention. Chaque zone est contrôlée par un agent dont l'objectif intègre un terme de pénalité dû à l'énergie, un terme de pénalité qui est actif quand la température est en dehors d'un intervalle prédéterminé et un terme de pénalité de concentration de CO<sub>2</sub> qui est actif quand celle-ci est en dehors d'un intervalle lui aussi prédéterminé.

## Conclusion

Nous avons décrit dans ce chapitre les modèles proposés dans la littérature pour prédire le confort des usagers. Nous avons abordé les modèles de confort lumineux, mais aussi les modèles de confort thermique. Nous en concluons qu'il est difficile de proposer des modèles de confort assez précis pour pouvoir être exploités efficacement par un système de contrôle énergétique. Puisque le confort lumineux dépend d'énormément de facteurs (la position des sources lumineuses par rapport au champ de

vision de l'observation, l'activité de l'utilisateur, le type de lumière, etc.) et en raison des changements fréquents des conditions de la lumière du jour (même dans des laps de temps très courts), il est ardu d'étudier de manière complète le comportement des utilisateurs. En effet, une telle étude nécessiterait un nombre relativement important de sujets pour aboutir à des résultats statistiquement significatifs, ce qui rend tout travail de recherche sur ce sujet difficile à effectuer. De plus, la sensation de confort est subjective, elle varie en fonction des individus et au cours du temps. Le confort adaptatif est une notion découlant de la littérature sur le confort thermique, elle vient apporter des réponses par rapport à la conception des systèmes de contrôle ainsi que leur relation à l'utilisateur. Parce que le confort adaptatif stipule que l'utilisateur réagit à son environnement en situation d'inconfort, l'une des conséquences les plus importantes de cette approche est qu'il est possible d'avoir des informations sur l'état de l'utilisateur à partir de ses interactions avec le système. À cet égard, la conception de notre modèle de l'utilisateur présenté dans le chapitre 3 se fonde sur des hypothèses caractérisant ces approches.

Au cours de ce chapitre, nous avons aussi décrit les différentes méthodes utilisées dans le contrôle thermique et lumineux. Nous en concluons qu'il est nécessaire de considérer des approches qui peuvent s'améliorer dans le temps et qui sont aussi capables de gérer des fonctions objectives diverses. Pour cette raison, les méthodes les plus utilisées actuellement dans le contrôle thermique se basent sur l'apprentissage par renforcement profond [140, 146, 132]. Malheureusement, ces méthodes sont très exigeantes en termes de données et de ressources de calcul. Compte tenu du manque de données d'exploitation des bâtiments actuels, il apparaît pertinent de proposer des systèmes plus simples offrant ainsi la possibilité de générer des politiques initiales pertinentes ou de fournir un environnement de simulation réaliste qui servira de base pour des méthodes plus complexes. D'autant qu'il apparaît paradoxal de minimiser la consommation énergétique d'un bâtiment avec des approches réputées pour être assez énergivores, y compris dans leur phase d'exploitation.



# Chapitre 2

## Apprentissage par renforcement

Dans ce chapitre, nous présentons les méthodes d'apprentissage par renforcement que nous avons considérées dans le cadre des travaux de cette thèse. Nous expliquons les approches d'apprentissage par renforcement sans états, puis nous détaillons les algorithmes d'apprentissage basés sur les états. Si le lecteur souhaite avoir plus de détails sur les notions, mécanismes et variantes de fonctionnement sur lesquels s'appuieront les méthodes d'apprentissage par renforcement utilisées dans le cadre de cette thèse, il pourra se référer à l'annexe A. Le lecteur pourra aussi se référer au livre de Sutton et Barto [116] pour approfondir ses connaissances sur l'apprentissage par renforcement.

### 2.1 Les approches d'apprentissage par renforcement sans états

Dans cette partie, nous expliquons dans le détail les approches d'apprentissage par renforcement sans états. Notons que les algorithmes présentés ici ne forment pas la totalité du corps de littérature d'apprentissage par renforcement sans états, mais plutôt les méthodes les plus couramment utilisées dans la pratique et que nous avons nous-mêmes utilisées dans le cadre de ces travaux de thèse. Nous présentons le principe général de chacune de ces approches, ainsi que les notions théoriques sous-jacentes sur lesquelles elles reposent.

Dans un premier temps, nous présentons les algorithmes d'apprentissage basés sur la politique, avant d'aborder les deux approches basées sur les valeurs : les approches naïves et les approches basées sur l'optimisme face à l'incertitude, pour finir avec les méthodes à espace d'états d'information.

#### 2.1.1 Méthodes basées sur la politique

Dans l'apprentissage par renforcement, les algorithmes basés sur la politique et le cadre formel dans lequel leurs propriétés ont été étudiées proviennent de la littérature sur les automates d'apprentissage [92]. En particulier, les *automates à structure variable* [92] permettent de formaliser une situation dans laquelle un agent interagit avec son environnement, où le choix des actions est fait de manière stochastique et où la probabilité de sélection des actions change en fonction du temps. Il est donc naturel que les *automates à structure variables*, servent à modéliser le fonctionnement

des *méthodes basées sur la politique* (méthodes dans lesquelles la politique de l'agent est représentée avec un vecteur de probabilité, voir l'annexe A). Cette sous-section présente les algorithmes les plus courants parmi les *méthodes basées sur la politique* en utilisant les concepts des *automates à structure variables* [92]. Précisons que cette terminologie est empruntée de la littérature de l'apprentissage par renforcement à base d'états.<sup>1</sup>

Dans tous les algorithmes à base *d'automates à structure variable* [92], l'apprentissage se déroule comme suit :

1. L'agent choisit une action donnée à l'aide de sa *politique* qui est représentée par un vecteur de probabilité, puis l'applique au niveau de l'environnement.
2. L'environnement retourne une réponse aléatoire après application de l'action courante, c'est *la récompense* de l'action.
3. La récompense est ensuite utilisée par l'agent afin de mettre à jour le vecteur de probabilité. La mise à jour se fait de sorte à *renforcer* les bonnes actions en incitant l'agent à les choisir plus souvent.

Au fil du temps, l'agent affecte une probabilité de plus en plus importante aux actions les plus favorables jusqu'à converger vers la meilleure action.

Avant de présenter les algorithmes, nous allons commencer par introduire des critères de performance utilisés dans la littérature des automates à structure variable. Ces critères expriment, à partir du comportement asymptotique d'un agent, des règles pour juger si cet agent arrive effectivement à apprendre. Ces critères varient dans la littérature, et il est nécessaire de les mentionner pour rendre compte des différences qualitatives entre les différentes approches d'apprentissage par renforcement.

L'action appliquée à l'étape courante  $n$  est notée  $\alpha(n)$ . La récompense à l'étape  $n$  retournée après application de cette action est notée  $\beta(n)$ . Nous représentons la politique de l'agent à l'étape  $n$  par le vecteur stochastique  $p(n)$ . L'espérance de la récompense associée au vecteur stochastique à l'instant  $n$  est notée  $W(n)$  et s'écrit

$$\begin{aligned}
 W(n) &= \mathbb{E}[\beta(n)|p(n)] \\
 &= \sum_{i=1}^r p(\beta(n) = 1 | \alpha(n) = \alpha_i) \cdot P(\alpha(n)) \\
 &= \sum_i d_i p_i(n)
 \end{aligned} \tag{2.1}$$

avec :

- $P(\alpha(n))$  représente la probabilité de choisir l'action  $\alpha(n)$  à l'instant  $n$ .
- $d_i$  est l'espérance de la récompense obtenue suite à l'application de l'action  $\alpha_i$ .

---

1. Comme expliqué dans l'annexe A, le terme *méthodes basées sur la politique* concerne les approches d'apprentissage par renforcement basées sur les états dans lesquels la politique est représentée explicitement (généralement à l'aide d'une loi de probabilité). Le paramètre de la politique permet d'avoir des ajustements différents sur les actions devant être choisies par l'agent. À cet égard, le cas le plus simple est celui dans lequel il y a un seul état seulement et où la politique est représentée par une loi de probabilité catégorielle. Nous retrouvons ainsi les algorithmes à base d'automates à structure variable étudiée par la théorie des automates.

—  $p_i$  est la probabilité de choix de l'action  $\alpha_i$ .

Si les actions sont sélectionnées de manière uniforme, la récompense moyenne est notée  $W_0$  et prend la valeur suivante  $W(n) = W_0 = 1/r \sum_i d_i$ . Le vecteur stochastique  $p(n)$  est considéré comme une variable aléatoire et la séquence  $\{p(n)\}_n$  est un processus stochastique. Le critère d'apprentissage le moins strict qu'il est possible de considérer est celui pour lequel l'agent arrive à apprendre une meilleure politique que la politique uniforme (en termes de  $W(n)$ ). Nous disons dans ce cas que l'algorithme arrive à apprendre de l'environnement. Un automate vérifiant cette condition est au moins aussi bon qu'un automate qui n'a aucune connaissance sur l'environnement. Cela signifie que l'automate aura, a minima, appris à améliorer sa politique de départ. Dans ce cas, on dit que l'automate est *Expédient*. Dans le cas idéal, l'espérance de la récompense moyenne converge vers la valeur de la meilleure action et l'automate est alors appelé automate *Optimal*. Il n'existe à ce jour aucun automate dont on a démontré qu'il est *Optimal*. Dans la pratique, on opte souvent pour une variante relaxée du critère mentionnée auparavant. Si la récompense moyenne de l'automate converge à un seuil arbitrairement petit vers la récompense de la meilleure action, on dit que l'automate est  $\epsilon$ -*Optimal*. Un automate est *Absolument Expédient* si et seulement si la récompense moyenne s'améliore au cours du temps.

Nous présentons de manière plus formelle les concepts précédents [92] :

**Definition 2.1.1.** *Un automate est dit Expédient SSI  $\lim_{n \rightarrow \infty} E[W(n)] > W_0$ .*

**Definition 2.1.2.** *Un automate est dit Optimal SSI  $\lim_{n \rightarrow \infty} E[W(n)] = d_m$  où  $\max_i \{d_i\}$ . Une définition équivalente à celle-ci est de dire que  $\lim_{n \rightarrow \infty} p_m(n) = 1$ .*

**Definition 2.1.3.** *Un automate est dit  $\epsilon$ -Optimal s'il satisfait la condition suivante  $\lim_{n \rightarrow \infty} E[W(n)] > d_m - \epsilon$  pour tout  $\epsilon$  si on ajuste les paramètres de l'automate.*

La dernière définition est celle d'un automate *Absolument Expédient*.

**Definition 2.1.4.** *Un automate est dit Absolument Expédient SSI  $E[W(n+1)|p(n)] > W(n)$  pour tout  $n$ , pour tout  $p_i(n) \in [0, 1]$  et pour tout ensemble possible  $\{d_i\}$ ,  $i \in \{1, 2, \dots, r\}$ .*

La dernière définition est particulièrement utile, car, si un automate est  $\epsilon$ -Optimal alors, il est Absolument Expédient pour les environnements non-stationnaires [119]. Elle est équivalente à des conditions relativement simples à vérifier concernant uniquement les mises à jour effectuées par l'algorithme de renforcement (et pas seulement sur son comportement asymptotique). La condition principale doit être vérifiée pour les actions autres que l'action choisie à l'étape courante. Cette condition assure que les probabilités (des actions autres que l'action courante) restent toutes dans le même ordre de grandeur. Plus exactement, elle stipule que la proportion entre la quantité rajoutée par l'algorithme de renforcement et la probabilité des actions est la même pour toutes les actions (autres que l'action courante). Cette condition force l'algorithme à changer faiblement la probabilité des actions qui ont une faible chance d'être choisies, et à changer fortement les probabilités des actions qui ont une forte chance d'être choisies. Cette condition permet de bien équilibrer l'*exploration* et l'*exploitation* de l'automate (ces deux termes sont expliqués dans l'annexe A).

Les algorithmes les plus connus de la théorie des automates sont l'algorithme de LRI (*Linear Reward Inaction*) [119] et une de ses variantes, l'algorithme du LRP (*Linear Reward Penalty*) [119].

Comme nous l'avons dit plus tôt, ces algorithmes maintiennent un vecteur stochastique qui est mise à jour à chaque étape. Pour LRI, la mise à jour est faite de manière à augmenter la probabilité de l'action sélectionnée par l'agent linéairement à sa récompense. L'algorithme LRP dérive de LRI en y apportant une modification. Le principe de LRP est de baisser la probabilité des actions qui n'ont conduit à aucune récompense (en plus d'augmenter la probabilité des actions menant à une bonne récompense comme LRI). L'algorithme 1 illustre le principe du LRI et du LRP et de leurs variantes. L'algorithme de LRP possède deux taux d'apprentissage  $a$  et  $b$ . Le paramètre  $a$  est associé au renforcement des actions menant à une récompense, le paramètre  $b$  sert à pénaliser les actions ne menant pas à une récompense en décrémentant leurs probabilités.

---

**Algorithme 1** Algorithme de mise à jour du LRx (Linear Reward -) à l'étape  $n$ .

---

**Inputs :**  $p(n)$  - vecteur stochastique à l'étape courante  $n$

$\beta(n)$  - la récompense à l'étape courante  $n$

$i$  - indice de l'action courante

$a$  - taux d'apprentissage associé aux actions récompensées

$b$  - taux d'apprentissage associé aux actions pénalisées

```

1: if  $\beta(n) = 1$  then
2:    $p_i(n + 1) = p_i(n) + a(1 - p_i(n))$ 
3:    $p_j(n + 1) = (1 - a)p_j(n) \quad \forall j \neq i$ 
4: else if  $\beta(n) = 0$  then
5:    $p_i(n + 1) = b(1 - p_i(n))$ 
6:    $p_j(n + 1) = b/(r - 1) + (1 - b)p_j(n) \quad \forall j \neq i$ 
7: end if

```

---

L'algorithme du LRP est obtenu à partir de l'algorithme 1 avec  $a = b$ , et l'algorithme LRI est obtenu à partir du même algorithme avec  $b = 0$ . L'algorithme du LRI est  $\epsilon$ -Optimal dans tout environnement non-stationnaire, il a donc de très bonnes garanties de convergence. L'inconvénient majeur de LRI est qu'il possède des états absorbants (plus formellement, c'est la chaîne de Markov  $\{p(n)\}_n$  qui possède des états absorbants [13]). Sachant qu'un état absorbant est un état dont on ne peut sortir et comme l'algorithme de LRI incrémente seulement les probabilités des actions ayant une récompense, tous les vecteurs de base (ayant une valeur unitaire dans une des composantes et zéro dans les composantes restantes) sont des états stationnaires de LRI. Ceci signifie que dès que la probabilité d'une action atteint 1, elle ne baissera pas, même si l'action en question n'est pas la meilleure. L'algorithme de LRI est donc généralement mauvais pour les environnements non stationnaires, car l'algorithme ne s'adaptera pas en cas de changement dans l'environnement.<sup>2</sup> Aussi, pour forcer l'algorithme à converger vers la meilleure action, il est nécessaire de choisir des valeurs de  $a$  relativement faibles ralentissant ainsi l'apprentissage de l'algorithme. Puisque le LRP fait baisser les probabilités des mauvaises actions, il n'a pas d'états absorbants comme LRI [92] et il peut être librement paramétré afin de le rendre plus rapide que ce dernier. En revanche, LRP n'a pas les mêmes garanties de convergence que LRP, en effet, l'algorithme du LRP est seulement Expédient

---

2. Précisons tout de même que ce résultat ne s'applique pas forcément à notre cas d'usage où la non stationnarité de l'environnement a une structure : elle dépend du comportement de l'utilisateur en réaction au système.



(pour les environnements non-stationnaires). Nous pouvons modifier l'algorithme 1 afin d'obtenir un algorithme ayant de meilleures propriétés de convergence et pas d'états absorbants. Effectivement, si la valeur de  $b$  est assez faible ( $b \ll a$ ), on obtient un algorithme  $\epsilon$ -Optimal [75]. Cette variante de l'algorithme est appelée Linear Epsilon Reward Penalty et est notée  $L_{R-\epsilon P}$ .

Une autre famille d'algorithmes analogue à LRI sont les *algorithmes d'estimation* [92]. L'idée de ces algorithmes est d'utiliser les récompenses reçues par l'agent à l'issue de l'interaction de celui-ci avec l'environnement, afin de mieux guider son apprentissage. L'algorithme de *la poursuite* [69] est un *algorithme d'estimation* dont le principe est de renforcer non pas l'action choisie à l'instant présent, mais l'action qui a la meilleure estimation parmi les actions de l'agent. Pour cela, l'algorithme devra garder en mémoire les estimations construites à partir des récompenses obtenues par chaque action. La probabilité de l'action qui a la meilleure estimation est augmentée linéairement. Si nous reprenons les mêmes notations que précédemment, nous pouvons résumer les opérations de l'algorithme de la poursuite par les formules qui suivent. Tout d'abord, nous définissons les estimations des actions :

$$\hat{d}_i = R_i(n')/Z_i(n')$$

avec :

- $R_i(n')$  : la somme des récompenses obtenues par l'action  $a_i$  d'indice  $i$  jusqu'à l'étape  $n'$ .
- $Z_i(n')$  : est le nombre de fois où l'action  $a_i$  a été sélectionnée jusqu'à l'étape  $n'$ .
- L'indice correspondant à l'action dont l'estimation de la récompense est maximale est  $h = \operatorname{argmax}_i \{\hat{d}_i\}$ .

En notant par  $pr_k$ , le vecteur stochastique obtenu après l'application de la  $k$ 'ième action, la mise à jour de la politique dans l'algorithme de la poursuite est

$$pr_{k+1} := pr_k + \alpha \cdot (e_h - pr_k)$$

où  $e_i$  est le vecteur de base ayant une valeur 1 à la position d'indice  $i$ . L'algorithme de la poursuite consiste en l'application de trois étapes de manière itérative : la sélection puis l'exécution de l'action, la réception de la récompense, et enfin la mise à jour des estimations. De manière plus détaillée :

- L'algorithme choisit l'action courante, qui sera ensuite appliquée dans l'environnement.
- L'environnement retourne la récompense associée à l'action dernièrement appliquée.
- Finalement, la probabilité  $p_h$  de l'action ayant la meilleure estimation est incrémentée de manière proportionnelle au taux d'apprentissage  $\alpha$ .

L'algorithme de *la poursuite hiérarchique* [138], est une variante de l'algorithme de la poursuite classique où la politique est représentée par un arbre dans lequel chaque nœud représente un agent d'apprentissage par renforcement (utilisant l'algorithme de la poursuite pour l'apprentissage), et où les feuilles représentent les actions dont l'agent dispose pour interagir avec l'environnement.

Le processus de sélection des actions de l'algorithme de la poursuite hiérarchique est exécuté comme suit : chaque agent (en commençant par la racine) sélectionne un agent de niveau inférieur dans l'arbre parmi les agents auxquels il est connecté (ses enfants), puis l'active pour prendre le

contrôle. L'agent activé choisit à son tour un de ses fils pour le sélectionner et ainsi de suite jusqu'aux feuilles (les actions). La mise à jour du vecteur de probabilité est effectuée sur chaque agent selon le principe de l'algorithme de la poursuite (en suivant le chemin de la meilleure action actuelle jusqu'à la racine — donc en sens inverse).

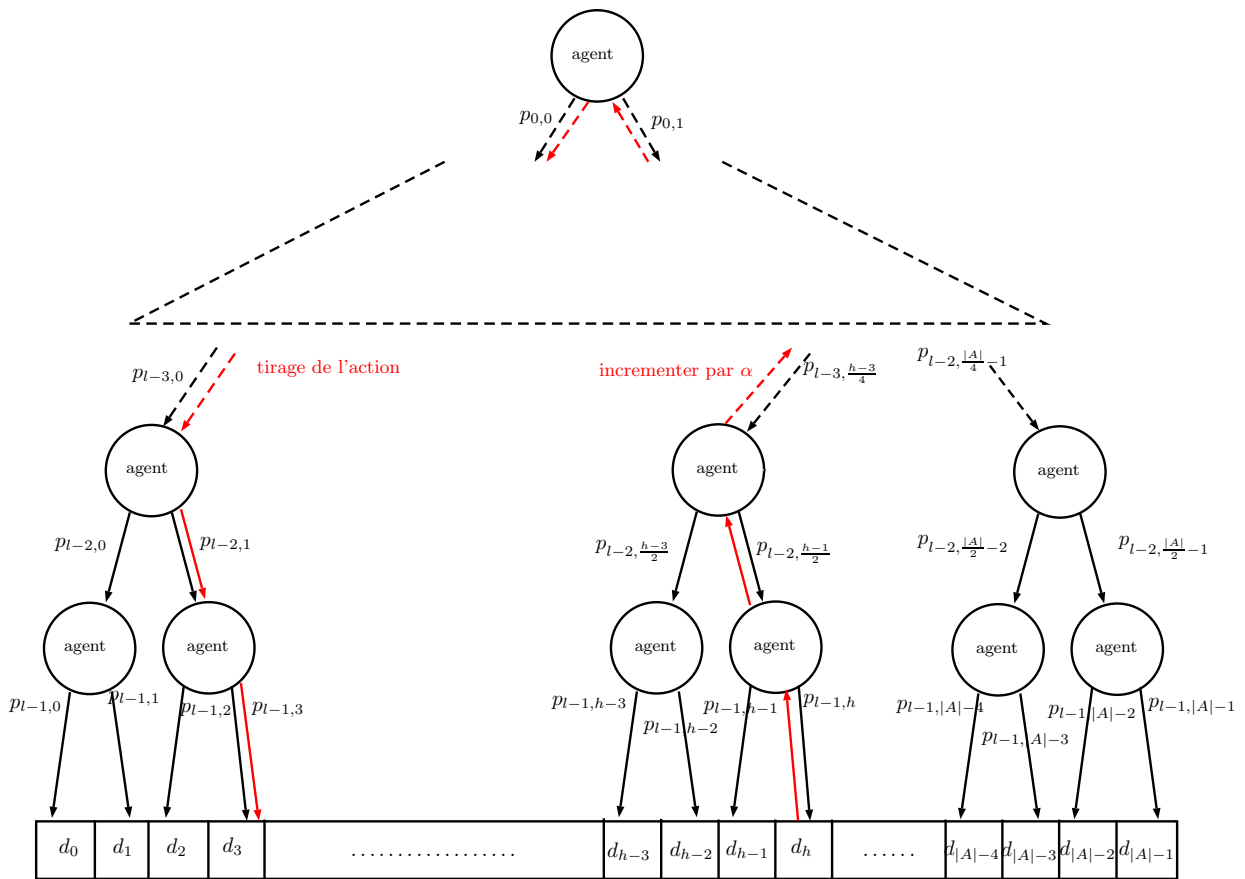


Figure 2.1 – Exemple illustrant l'algorithme de la poursuite hiérarchique.

Dans la figure 2.1, un exemple de l'algorithme de la poursuite hiérarchique est montré. Dans cet exemple, l'agent est représenté par un arbre binaire ayant  $l \in \mathbb{N}^*$  niveaux. L'ensemble des actions est noté  $A$ . Elles sont indexées de 0 à  $|A| - 1$  et constituent les feuilles de cet arbre. Les feuilles de l'arbre sont étiquetées par les estimations des actions leur correspondant. Chaque nœud interne de cet arbre représente un agent possédant un vecteur stochastique servant à choisir un agent parmi ses fils. Ce vecteur n'est pas représenté explicitement dans l'exemple 2.1. Chaque arc reliant un nœud à son fils est pondéré par la probabilité que ce nœud sélectionne ce fils. À chaque fois qu'une action doit être sélectionnée, l'agent correspondant au nœud racine est le premier à faire un tirage aléatoire parmi ses fils, puis vient le tour de l'agent choisi par la racine et ainsi de suite. Les agents associés au niveau le plus bas tirent leurs fils parmi un sous-ensemble contigu d'actions de  $A$ .

Dans l'exemple de la figure 2.1,  $l$  est le nombre de niveaux de l'arbre. Pour chaque niveau  $i$  de l'arbre, la probabilité de tirer le  $j^{ième}$  nœud (en partant de la gauche) est notée  $p_{i,j}$ . Puisque l'arbre de la figure 2.1 est binaire, pour chaque niveau  $i$  le nombre d'agents présents dans ce niveau est de  $\frac{|A|}{2^i}$ . Le chemin en rouge allant de la racine aux actions représente le tirage d'une action (le tirage

de l'action d'indice 2). Nous pouvons donc voir que la probabilité de tirer une action est le produit des poids du chemin reliant la racine à la feuille.

Dans l'algorithme de la poursuite hiérarchique (tout comme dans l'algorithme de la poursuite), après chaque choix et exécution d'action, la mise à jour de la politique va affecter uniquement la probabilité de choix de l'action ayant la meilleure estimation. Dans l'arbre de la figure 2.1, le chemin en rouge allant de l'action  $a_h$  à la racine correspond au choix de la meilleure action (selon les estimations de l'agent). Par conséquent, chaque probabilité de ce chemin est incrémentée de  $\alpha$  dans la mise à jour de la politique de l'algorithme de la poursuite hiérarchique.

Finalement, notons le fait que, de la même manière qu'il y a différents types d'algorithmes dans l'apprentissage par renforcement, il y a également différents types d'environnement étudiés qui sont généralement classifiés en termes de domaine de définition de la fonction de récompense. Dans les formulations classiques des algorithmes basés sur la politique, la récompense fait partie de l'ensemble binaire. Cependant, il est possible de voir la récompense comme une variable aléatoire continue prenant ses valeurs dans l'intervalle  $[0,1]$ . Dans ce cas, l'agent doit trouver l'action  $\alpha_i$  qui maximise l'espérance de la récompense qui est  $s_i = E[\beta(n)|\alpha(n) = \alpha_i]$ . Tous les algorithmes présentés dans cette sous-section peuvent être adaptés à ces cas en remplaçant les récompenses binaires par leurs homologues continues.<sup>3</sup>

## 2.1.2 Les méthodes basées sur les valeurs

Dans cette sous-section, nous présentons l'autre type de méthodes généralement utilisées dans l'apprentissage par renforcement. Cette terminologie est aussi empruntée à la littérature de l'apprentissage par renforcement à base d'états (comme pour les méthodes basées sur la politique), mais nous la reprenons néanmoins afin de rendre l'exposé des méthodes clair et cohérent (voir l'annexe A pour plus de détails).

Dans ce type de méthodes, l'agent estime directement la récompense de chaque action et se base sur ces estimations pour sélectionner l'action courante. Comme nous le verrons dans les algorithmes qui suivent, l'agent a besoin d'effectuer de l'exploration (donc randomiser le choix des actions) afin d'apprendre les bonnes actions. Les algorithmes de cette catégorie de méthodes se différencient essentiellement par la manière de choisir les actions à partir de leurs valeurs. La question qui se pose pour ces approches est : quels sont les éléments qui pourraient aider l'exploration à partir des valeurs des actions (ou d'éléments supplémentaires) ?

La manière la plus simple d'introduire de l'exploration est de construire une loi de probabilité à partir des valeurs des actions. On peut distinguer deux manières simples pour le faire :  $\epsilon$ -Greedy [116] et l'exploration de Boltzmann [116]. D'autres méthodes plus sophistiquées [9, 30, 48] tentent d'estimer l'incertitude des actions afin de tester plus souvent les actions qui ne sont pas assez explorées. Ces estimations sont alors intégrées à la politique afin de diversifier le choix des actions. Ces approches se basent sur le principe de l'*optimisme face à l'incertitude*.

---

3. Il est possible de voir que les récompenses doivent être normalisées entre 0 et 1 dans le cas où la récompense est en dehors de cet intervalle.

Une manière d'évaluer théoriquement ces algorithmes avec des notions unifiées est de les comparer à l'aide de leurs *regret*. Le regret est l'espérance de l'écart entre la récompense de la meilleure action et la récompense des actions choisies effectivement par l'agent durant l'apprentissage. Elle est définie mathématiquement à chaque étape  $n$  par :

$$L_n = \mathbb{E}\left[\sum_{k=1}^n (v^* - q(a_k))\right]$$

où  $q(a_n)$  représente l'espérance de la récompense de l'action  $a_n$  exécutée à l'étape  $n$  et  $v^*$  représente l'espérance de la récompense de l'action optimale. Le regret représente donc la perte d'opportunité due au choix des actions non optimales par rapport au gain rapporté par le choix de l'action optimale tout au long de l'apprentissage. Le regret d'un algorithme d'apprentissage par renforcement permet de quantifier dans quelle mesure le dilemme exploration-exploitation est bien adressé par l'algorithme. Nous allons présenter dans ce qui suit les algorithmes de la famille d'apprentissage par renforcement basé sur les valeurs. Mais avant de détailler ces algorithmes, nous présentons l'expression du regret ainsi que sa valeur dans le pire cas. Pour commencer, on peut effectuer les dérivations suivantes sur le regret  $L_n$ ,

$$L_n = \mathbb{E}\left[\sum_{k=1}^n (v^* - q(a_k))\right] = \sum_{a \in A} \mathbb{E}[N_n(a)](v^* - q(a)) = \sum_{a \in A} \mathbb{E}[N_n(a)]\Delta_a \quad (2.2)$$

avec

- $N_n(a)$  est le nombre de fois où l'action  $a$  a été choisie jusqu'à l'instant  $n$ .
- $\Delta_a = v^* - q(a)$  est l'écart entre la récompense de la meilleure action et la récompense de l'action  $a$ .

Le regret est dans le pire cas asymptotiquement linéaire en fonction du nombre d'étapes de l'apprentissage. En effet, considérons l'algorithme qui choisit toujours la pire action dont l'indice est noté  $f$ . Nous voyons bien que le regret à l'instant  $n$  vaut  $L_n = \sum_{a \in A} \mathbb{E}[N_n(a)]\Delta_a = \mathbb{E}[N_n(f)]\Delta_f = n \cdot \Delta_f$ . Tout algorithme ayant des performances linéaires en fonction du nombre d'étapes est mauvais, car il aurait les mêmes performances que le pire algorithme possible.

Comme nous l'avons dit en annexe A, un bon algorithme d'apprentissage par renforcement est un algorithme qui équilibre bien exploration et exploitation. Toujours exploiter est une mauvaise idée, car le regret est aussi linéaire dans ce cas. L'idée la plus simple pour implémenter une stratégie qui exploite seulement est de se baser sur des estimations obtenues sur un nombre fixe d'étapes afin d'isoler l'action optimale. Cette méthode est dite *Greedy*, et consiste à d'abord estimer les actions pendant un certain nombre d'essais, puis à choisir l'action qui a eu la meilleure récompense. Le nombre d'essais étant toujours fini, l'algorithme a toujours une probabilité non nulle de prendre une mauvaise action une fois les essais effectués. Au vu de ce fait, le regret est forcément linéaire avec l'algorithme *Greedy*. L'approche *Greedy* mise donc beaucoup trop sur l'exploitation.

Cette dernière remarque motive l'utilisation de l'apprentissage par renforcement, notamment quand il est difficile d'avoir des connaissances *a priori* sur la qualité des actions. Nous présentons dans

la sous-section suivante la méthode la plus simple pour créer de l'exploration en modifiant légèrement l'algorithme *Greedy*. Précisons que le meilleur regret réalisable par un algorithme d'apprentissage par renforcement est logarithmique en fonction du nombre d'étapes de l'algorithme comme le stipule le théorème suivant :

**Théorème 1.** [74] *La limite à l'infini du regret d'un algorithme est au moins logarithmique en fonction du nombre d'étapes de l'apprentissage :*

$$\lim_{n \rightarrow \infty} L_n \geq \log(n) \cdot \sum_{a|\Delta_a > 0} \frac{\Delta_a}{KL(R^a || R^{a^*})}$$

où  $R^a$  est la distribution des valeurs de la récompense de l'action  $a$  et  $R^{a^*}$  est la distribution des valeurs de la récompense de l'action optimale. Aussi  $KL(P||Q)$  est la divergence de Kullback-Leibler entre les distributions  $P$  et  $Q$  [55] qui sert à quantifier la ressemblance entre deux distributions de probabilité.

### 2.1.3 L'algorithme $\epsilon$ -Greedy

Le principe de  $\epsilon$ -Greedy [116] consiste à choisir la plupart du temps la meilleure action d'après les connaissances de l'agent, mais aussi de choisir avec une faible chance une des actions aléatoirement parmi toutes les actions disponibles à l'agent. L'approche  $\epsilon$ -Greedy dépend du paramètre  $\epsilon$  qui est la probabilité de choix aléatoire des actions. L'inconvénient de cette méthode est que le regret est linéaire en fonction du nombre d'étapes de l'apprentissage [116]. En effet, à chaque étape, la probabilité de choisir une action différente de la meilleure action est non nulle. Donc, sur un nombre élevé d'étapes, il est très probable que l'algorithme sélectionne une action non optimale une proportion fixe du nombre d'étapes total (cette proportion dépend bien entendu de  $\epsilon$ ). L'algorithme de  $\epsilon$ -Greedy a donc, en théorie, de très mauvaises performances (aussi mauvaises que le pire algorithme possible). L'algorithme de  $\epsilon$ -Greedy a le défaut de toujours explorer (même quand l'algorithme a assez d'information sur l'environnement, la probabilité de choix des mauvaises actions est toujours non nulle). Néanmoins, l'algorithme de  $\epsilon$ -Greedy a certains avantages clairs : il est simple à implémenter, ce qui en fait un algorithme assez robuste (même si les propriétés de l'environnement sont différentes de celles attendues).

Afin d'accélérer la convergence de  $\epsilon$ -Greedy, on peut faire diminuer la probabilité de sélection aléatoire des actions au cours du temps. De cette façon, l'algorithme pourra explorer au début de l'apprentissage, puis exploiter les bonnes actions plus tard quand l'algorithme sera assez certain sur les valeurs des estimations des actions. En diminuant la valeur de  $\epsilon$  de la bonne manière, il est possible d'obtenir un regret logarithmique avec  $\epsilon$ -Greedy (cette variante est appelée *Decaying  $\epsilon_t$ -Greedy*). Mais la manière de décrémenter la valeur de  $\epsilon$  dépend des écarts à la meilleure récompense. En effet, elle requiert la connaissance du plus petit écart possible  $d = \min_{a|\Delta_a > 0} \Delta_a$ . Et la manière de réduire la valeur de  $\epsilon$  pour avoir un regret logarithmique est la suivante :

$$\epsilon_n = \min\left\{1, \frac{c.a}{d^2.n}\right\}$$

$$d = \min_{a|\Delta_a>0} \Delta_a$$

$$c > 0$$

$a$  est le nombre d'action de l'agent

$n$  est l'itération courante

Le désavantage de *Decaying  $\epsilon_t$ -Greedy* est qu'il nécessite d'avoir des connaissances au préalable sur l'environnement (la valeur de  $d$ ) afin d'avoir un bon regret. Une alternative simple à prendre en compte est de combiner  $\epsilon$ -Greedy avec une approche encourageant l'exploration très tôt durant l'apprentissage. Il est possible d'atteindre cet objectif d'exploration en initialisant les valeurs des actions avec de fortes valeurs. Cette approche est appelée *initialisation optimiste* [116]. Le principe sous-jacent est de supposer le meilleur jusqu'à preuve du contraire. Même si en pratique, l'initialisation optimiste peut donner de bons résultats, elle a aussi un regret linéaire dans le temps.

Il existe néanmoins des algorithmes capables de trouver l'action optimale en un temps logarithmique sans s'appuyer sur des connaissances *a priori* de l'environnement et qui se reposent sur des idées très proches de *l'initialisation optimiste* [116]. Pour ce faire, certaines méthodes quantifient explicitement les connaissances acquises par l'agent suite à son interaction avec l'environnement. Le principe est d'essayer plus souvent les actions qui ont été choisies peu souvent par l'algorithme, ces actions étant considérées comme ayant plus de potentiel que les autres actions. C'est le principe de *l'optimisme face à l'incertitude* qui sera présenté dans une section prochaine.

En plus des variantes de  $\epsilon_t$ -Greedy, il existe un moyen de créer de l'exploration dans le choix des actions autrement qu'en affectant une probabilité explicite d'exploration. Par exemple, il est possible d'affecter à chaque action un poids dépendant de son estimation et normaliser les poids afin d'avoir un vecteur stochastique qui va constituer la politique de l'agent. L'exploration de Boltzmann [116] en est un exemple et nous la présentons dans la sous-section suivante.

## 2.1.4 Exploration de Boltzmann

Bien que la sélection d'actions par la méthode  $\epsilon_t$ -Greedy soit un moyen efficace et connu d'équilibrer l'exploration et l'exploitation dans l'apprentissage par renforcement, l'un de ses inconvénients est que dans la partie exploration d' $\epsilon_t$ -Greedy, toutes les actions non optimales sont choisies sans préférence de certaines sur les autres. Ceci signifie qu'il est tout aussi probable que l'action ayant la pire estimation soit choisie plutôt que celle qui est la plus proche de l'action optimale. Dans les applications critiques, cela peut être très insatisfaisant d'avoir à jouer les mauvaises actions aussi souvent que les bonnes. La solution la plus simple est de lier les probabilités de choix des actions à leurs récompenses estimées. Toutes les actions sont classées et pondérées suivant l'estimation de leurs récompenses, ce qui signifie que la probabilité la plus élevée est toujours attribuée à l'action ayant la plus grande estimation. C'est ce qu'on appelle la sélection d'action par *softmax* [116]. La méthode *softmax* la plus courante utilise une distribution de Gibbs, ou de Boltzmann. L'exploration de Boltzmann [116] est la méthode *softmax* quand la distribution choisie pour construire la politique est la distribution de Boltzmann [116]. Dans ce cas, la probabilité de choix de chaque action

à l'étape  $n$  est décrite comme suit,

$$p_n(a) = \frac{e^{q_n(a)/\tau}}{\sum_{b \in A} e^{q_n(b)/\tau}}$$

où  $\tau \in \mathbb{R}$  est un paramètre appelé la *température*. Ce paramètre décrit dans quelle mesure l'agent préfère l'exploration à l'exploitation en se basant sur les valeurs estimées des actions  $q_n(a)$ . Quand la valeur de  $\tau$  tend vers l'infini, la politique obtenue par la distribution de Boltzmann ressemble de plus en plus à une politique uniforme. Inversement, quand la température tend vers 0, la politique a de plus en plus tendance à préférer la meilleure action (la politique de l'agent tend vers la politique *Greedy*). En somme, le paramètre  $\tau$  détermine quel est le nombre d'actions que l'agent s'autorise à jouer en se basant sur l'estimation de leurs récompenses. La température permet de régler l'équilibre entre exploration et exploitation, il faut donc diminuer la température tout au long de l'apprentissage.

Le désavantage de l'exploration de Boltzmann est que le comportement de l'agent (et donc la mise à jour de la politique) dépend beaucoup des valeurs numériques (l'échelle) des estimations des récompenses. Par exemple, si les récompenses des actions ont des valeurs trop proches, leurs estimations le seront probablement aussi et l'algorithme risque de trop explorer même si la température est faible. Et ceci car le vecteur stochastique dépend directement des estimations des récompenses des actions. Et si, dans le même exemple, la température est très forte, et que les actions un peu plus mauvaises donnent par chance une bonne récompense, l'algorithme risque d'être bloqué pendant plusieurs itérations sur les mauvaises actions (également parce que l'algorithme se base sur les estimations des récompenses pour le calcul du vecteur stochastique). Il faut donc réduire progressivement la valeur de  $\tau$  au cours du temps pour obtenir le bon dosage entre exploration et exploitation. Il faut aussi déterminer les valeurs de départ et de fin de cette température. Cela ne peut se faire qu'en ayant une idée sur l'échelle des valeurs des récompenses des actions, car elles déterminent la manière dont devrait être baissée la température afin d'assurer un bon équilibre entre exploration et exploitation.

Étant donné la difficulté de choisir la bonne valeur de  $\tau$ , et comme le regret de l'exploration de Boltzmann est linéaire [116], il est généralement préférable d'utiliser  $\epsilon$ -*Greedy* plutôt que l'exploration de Boltzmann comme stratégie d'exploration.

### 2.1.5 Méthodes basées sur les valeurs : optimisme face à l'incertitude

Comme il a été dit précédemment, le principe de cette famille de méthodes consiste à mesurer la précision des estimations des récompenses des actions et préférer non seulement les bonnes actions, mais aussi les actions dont le résultat est plus incertain (qui ont été choisies moins souvent par l'algorithme). Le regret de cette famille de méthodes est logarithmique. L'idée est que les actions les moins testées ont une plus grande chance de se révéler meilleures.

#### UCB : Upper Confidence Bound

Dans l'algorithme de UCB [9] une borne supérieure de la valeur de chaque action est estimée à partir des interactions de l'agent avec son environnement. À chaque étape de l'apprentissage,

l'agent choisit l'action ayant la borne la plus élevée. La borne sert de bonus aux actions qui ont été testées peu souvent par l'agent. Elle dépend du nombre de fois où l'action a été choisie : plus une action est choisie, plus son estimation devient certaine et donc plus sa borne se resserre autour de son estimation. L'algorithme de UCB est présenté dans Algorithme 2. Le regret de UCB est logarithmique en fonction du nombre d'étapes de l'algorithme.

---

**Algorithme 2** Algorithme de mise à jour de UCB à l'étape  $n$ .

---

**Variables :**  $n_a$  - nombre de fois où l'action  $a$  est testée.

$\beta(n)$  - la récompense à l'étape courante  $n$ .

$n_a$  - nombre de fois où l'action  $a$  est testée.

$\bar{x}_a$  - moyenne des récompenses obtenues par l'action  $a$ .

$n$  - la somme des  $n_a$  pour toutes les actions  $a$ .

**Initialisations :** jouer toutes les actions, initialiser chaque action  $a$  avec sa récompense

$r_a$ .

1:  $a \leftarrow \operatorname{argmax}_{a \in A} [\bar{x}_a + \sqrt{\frac{2 \ln(n)}{n_a}}]$

2: Jouer l'action  $a$  et recevoir sa récompense  $r_n$

3:  $n_a \leftarrow n_a + 1$

4:  $n \leftarrow n + 1$

5:  $\bar{x}_a \leftarrow \bar{x}_a + \frac{1}{n_a} (r_n - \bar{x}_a)$

---

Le théorème suivant donne une borne supérieure sur le regret de UCB

**Théorème 2.** [9] *La limite à l'infini du regret de l'algorithme UCB est logarithmique en fonction du nombre d'étapes :*

$$\lim_{n \rightarrow \infty} L_n \leq 8 \log(n) \sum_{a | \Delta_a > 0} \Delta_a$$

Les bornes des actions utilisées dans UCB sont dérivées d'inégalités assez générales sur la théorie des probabilités. Ces inégalités ont besoin de très peu d'hypothèses pour être vérifiées (nous pouvons citer l'inégalité d'Hoeffding [57]), l'approche est donc essentiellement fréquentiste.

Il existe par ailleurs une variante bayésienne de UCB [6]. Dans cette approche, chaque action a une distribution sur les valeurs possibles qu'elle peut prendre. Ces distributions reflètent les connaissances a priori de l'agent sur les estimations réelles que pourraient avoir les actions. L'agent met à jour ces distributions suivant les récompenses des actions. Tout comme dans UCB l'action choisie par l'algorithme est celle qui maximise la somme de l'estimation et de la borne de l'action. La différence principale avec UCB classique est que pour UCB Bayésien, l'estimation est déduite à partir des distributions des actions. Cette borne est généralement paramétrée par un niveau de confiance qui influence implicitement la tendance de l'algorithme à exploiter ou explorer. Nous présentons dans ce qui suit les approches Bayésiennes.



## Approches Bayésiennes et approches similaires

Les approches que nous avons présentées jusqu'ici sont limitantes quand il s'agit d'intégrer des connaissances préalables sur le problème à résoudre. Les approches bayésiennes permettent d'intégrer explicitement des hypothèses préalables sur la distribution de la récompense de chaque action, et d'exploiter ces hypothèses pour arriver à apprendre la meilleure action. L'agent utilise ces distributions (qui portent généralement sur les valeurs possibles de la récompense des actions), afin de choisir l'action qui devra être prise. Ces distributions sont décrites par des paramètres qui sont mis à jour à chaque fois que l'agent applique l'action choisie et reçoit sa récompense. Dans cette sous-section, nous présentons le principe général des algorithmes basés sur les approches bayésiennes en commençant par poser les concepts communs à toutes ces approches. Généralement, les approches bayésiennes contiennent les éléments suivants :

- Un paramètre  $\tilde{\mu}$  associé à chaque action représentant la récompense moyenne/variance de l'action.
- Une distribution antérieure (a priori) supposée  $P(\tilde{\mu})$  pour chacun des paramètres. Cette distribution reflète les croyances de départ de l'algorithme sur les valeurs des actions.
- L'historique  $D$  des récompenses observées par l'agent durant l'apprentissage.
- Une fonction de probabilité supposée  $P(r/\tilde{\mu})$  associée à chaque action et qui donne la probabilité de recevoir une récompense sachant la valeur du paramètre.
- Une distribution postérieure (obtenue après observation des données) pour chaque action  $P(\tilde{\mu}/D) = P(D/\tilde{\mu})P(\tilde{\mu})$  où  $P(D/\tilde{\mu})$  est la fonction de vraisemblance des données [42].

À chaque étape, l'algorithme choisit l'action courante en se basant sur ses croyances concernant les estimations des actions. Ces croyances traduisent quelles sont les valeurs que l'algorithme suppose être les plus probables pour les récompenses de chaque action (elles sont modélisées par les distributions antérieures). L'algorithme reçoit la récompense de l'action et met à jour les paramètres de la distribution en utilisant la formule de Bayes. La mise à jour utilise le principe de *l'estimateur du maximum a posteriori (MAP)* [55]. L'estimateur du maximum a posteriori est généralement connu pour les distributions classiques.

L'agent utilise le principe de *l'optimisme face à l'incertitude* pour agir dans l'environnement. En revanche, les points abordés jusqu'ici sur les méthodes bayésiennes nous révèlent seulement comment sont modélisées les connaissances de l'agent ainsi que la manière dont l'expérience de l'agent interagissant avec l'environnement affecte ses connaissances. Mais nous n'avons pas encore montré comment l'agent exploite ces connaissances pour choisir l'action la plus adéquate. Somme toute, les principes employés pour modifier la politique de l'agent découlent de manière assez naturelle des concepts posés auparavant sur les approches bayésiennes : si l'agent est peu certain de la valeur de la récompense d'une action ou s'il pense que l'action va mener à une bonne récompense, alors il aura plus de chance d'essayer l'action en question. L'agent privilégie les actions sur lesquelles il a le moins de recul ainsi que les actions amenant les meilleures récompenses.

Comme nous l'avons dit précédemment, l'UCB Bayésien est une approche bayésienne. De ce fait, l'algorithme d'UCB Bayésien modélise également l'environnement avec des distributions de probabilité sur les valeurs des récompenses et met à jour ses distributions en utilisant *l'estimateur*

du *maximum a posteriori* (MAP). Mais avant tout, l'UCB Bayésien repose sur l'utilisation *explicite* du principe de l'optimisme face à l'incertitude pour choisir l'action à prendre. D'autres approches bayésiennes sont cependant différentes de l'UCB Bayésien dans le sens où elles utilisent *implicitement* l'optimisme face à l'incertitude dans le choix des actions. Parmi ces approches, nous pouvons distinguer les approches de *matching de probabilité* [30].

Le principe du *matching de probabilité* [30] est de sélectionner une action en fonction de la probabilité qu'elle soit la meilleure action, compte tenu de l'expérience passée de l'agent. Cette probabilité peut, théoriquement, être calculée à chaque étape à partir des distributions sur les valeurs des actions. Le *matching de probabilité* implémente implicitement l'idée d'optimisme face à l'incertitude, car il y a une plus grande incertitude sur les actions les moins prises par l'agent, ce qui signifie qu'elles ont une plus grande chance de se révéler meilleures que les autres actions. Néanmoins, il est difficile de calculer les probabilités de *matching*, nous devons, pour ce faire, considérer chaque combinaison des valeurs des actions, calculer sa probabilité et déterminer la meilleure action à jouer pour cette combinaison, puis sommer ces probabilités pour chaque action en fonction des paramètres. Pour cette raison, il est préférable de l'estimer sur un échantillon tiré des distributions des valeurs des actions. L'*échantillonnage de Thompson* [120, 121] utilise ce principe.

Ceci nous mène aux approches se basant sur le *bootstrapping* [33] qui est l'une des techniques les plus classiques de rééchantillonnage. L'idée est d'évaluer la précision des estimations des récompenses des actions en échantillonnant à nouveau les distributions des récompenses reçues de chaque action. Avoir une meilleure précision permet à l'agent de mieux choisir les actions à prendre. L'échantillonnage de Thompson est une méthode qui se marie très bien avec le *bootstrapping* car il est basé sur des estimations issues d'un échantillon. Pour cette raison, le *bootstrapping* est généralement utilisé avec l'échantillonnage de Thompson dans la littérature de l'apprentissage par renforcement [114].

La dernière famille d'approches que nous présentons et qui utilisent implicitement le principe d'optimisme face à l'incertitude sont les méthodes à base *d'espace d'état de l'information* [48]. Ces méthodes visent à explorer correctement les états sur lesquelles l'agent a peu d'information tout en privilégiant les bonnes actions et le font en modélisant explicitement l'information que possède l'agent sur son environnement. Assurément, l'exploration est utile, car elle permet d'informer l'agent sur l'utilité des actions. Mais elle est coûteuse dans le sens où elle peut amener à des pertes d'opportunité si l'algorithme choisit trop souvent les mauvaises actions. La question qui pourrait alors se poser est de savoir s'il est possible de quantifier explicitement la valeur de l'information en termes de récompense. Dans ce cas, la question de savoir comment faire un bon compromis entre exploration et exploitation se traduit par le fait de comparer entre le gain apporté en jouant la meilleure action et la récompense sacrifiée menant à l'obtention de l'information sur les actions incertaines. En tout cas, le gain en information est estimé plus grand dans les situations les plus incertaines. C'est ce qui pousse l'agent à explorer encore plus les situations pour lesquelles l'agent connaît peu d'éléments.

Pour ce faire, l'idée de base des approches par espace d'états d'information est d'intégrer les informations de l'agent dans une représentation à état artificielle. Naturellement, ces états ne correspondent pas à des situations différentes de l'environnement, mais bien aux informations de l'agent sur l'environnement. Dans cette représentation, chaque état correspond à des valeurs possibles pour

les paramètres des distributions de probabilité des récompenses des actions. En ce sens, les méthodes par espace d'état d'information sont assez similaires aux approches bayésiennes car elles modélisent explicitement les informations de l'agent en termes de distribution de probabilité sur les récompenses des actions. La différence principale entre ces deux approches est que les distributions de probabilité ne servent pas seulement à modéliser le choix des actions, mais sont aussi la base sur laquelle l'algorithme choisit l'action courante. En effet, puisque les distributions font partie de l'état, l'agent devra apprendre à trouver pour chaque état d'information quelle serait l'action menant à la récompense à long terme la plus grande.

Les méthodes d'exploration par espace d'états sont théoriquement les meilleures, elles trouvent la meilleure action en utilisant un minimum d'échantillons réels d'expérience, mais elles s'appuient fortement sur les échantillons simulés d'expérience obtenus de l'espace à états d'information. L'espace des états peut être arbitrairement grand et l'apprentissage dans cet espace peut utiliser d'énormes quantités de ressources. Pour cette raison, elles ont été écartées dans le cadre de cette thèse. Dans la section qui suit, nous traitons des méthodes d'apprentissage par renforcement basé sur les états.

## 2.2 Les méthodes d'apprentissage par renforcement basé sur les états

Dans les méthodes basées sur les valeurs, la politique n'est pas représentée explicitement (voir l'annexe A), l'agent n'a que les valeurs estimées des actions (ou des états) et les actions sont choisies uniquement à l'aide de ces estimations. Dans le cadre de notre travail, nous avons choisi les méthodes basées sur les valeurs, ces méthodes étant très populaires. Elles ont la capacité de bien gérer de faibles quantités de données et offrent ainsi la possibilité de faire de l'apprentissage *off-policy* (voir l'annexe A). Nous ne nous intéresserons pas aux méthodes basées sur les fonctions d'approximation. En effet, les bâtiments intelligents actuels ne possèdent souvent pas encore suffisamment de ressources en termes de données et de capacité de calcul permettant de tirer parti pleinement des avantages de l'apprentissage. Ainsi, dans le cadre de cette thèse, nous nous restreignons uniquement à la question de savoir dans quelle mesure les algorithmes sans fonction d'approximation arrivent à apprendre par interaction avec l'utilisateur un comportement énergétiquement optimal. Nous ne nous intéresserons pas non plus aux méthodes basées sur la politique. En effet, celles-ci sont connues pour nécessiter de grandes quantités de données. De plus, choisir ces méthodes présuppose de connaître la forme de la distribution adéquate pour représenter la politique, ce qui n'est pas le cas ici, puisque nous considérons l'hypothèse que l'agent ne connaît rien de son environnement.

Ceci étant dit, nous présentons les algorithmes d'apprentissage par renforcement que nous avons utilisé (compte tenu de leur popularité) : SARSA [116] et Q-learning [130]. Ces deux algorithmes reposent sur le principe de différence temporelle, mais l'utilisent de manière assez différente. L'algorithme SARSA représente l'implémentation la plus directe du principe de différence temporelle. En partant de l'algorithme 11, SARSA est obtenu simplement en remplaçant les estimations des valeurs par les estimations des paires d'états-action, puis en rajoutant un mécanisme d'exploration au niveau de l'algorithme. L'algorithme du Q-learning est similaire à SARSA à la différence notable que dans

Q-learning l'agent ne tente pas d'évaluer la politique courante, mais plutôt d'estimer directement la meilleure politique.

Dans SARSA, au début de chaque itération, l'agent aura déjà choisi (à l'itération précédente) l'action  $A$  qu'il va prendre dans l'état  $S$  où se trouve actuellement l'environnement. Il applique l'action  $A$ , l'environnement retourne le prochain état  $S'$  ainsi que la récompense  $R$ . L'agent choisit ensuite la prochaine action  $A'$  dans le nouvel état  $S'$  où se trouve l'environnement (en anticipation pour l'itération suivante). L'estimation de l'agent pour l'action  $A$  changera alors en fonction de la différence temporelle entre la paire de l'état-action courante  $(S, A)$  et celle qui la suit  $(S', A')$ . Comme montré dans l'équation suivante :

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)) \quad (2.3)$$

Le lecteur constatera que l'équation correspond exactement à celle de la différence temporelle. Précisons finalement que l'action est choisie selon une politique  $\epsilon$ -greedy, tout ceci est précisé dans l'algorithme 3.

---

**Algorithme 3** L'algorithme tabulaire de SARSA pour le calcul de la meilleure politique  $\pi^*$ .

---

**Inputs :**  $Q(s, a)$  - les valeurs des paires d'états-action initialisées arbitrairement sauf dans l'état final  $Q(F, \cdot) = 0$ .

$\alpha$  - Le taux d'apprentissage (le pas de l'évaluation).

$\epsilon$  - Probabilité de prendre la meilleure action.

```

1: for  $i$  allant de 1 à  $N$  do
2:   Initialiser l'état courant  $S$ 
3:   Choisir l'action  $A$  partant de  $S$  suivant l'algorithme  $\epsilon$ -greedy sur  $Q$ 
4:   while  $S \neq F$  do
5:     Pendre l'action  $A$ , observer  $R, S'$ 
6:     Choisir l'action  $A'$  partant de  $S'$  suivant l'algorithme  $\epsilon$ -greedy sur  $Q$ 
7:      $Q(S, A) \leftarrow Q(S, A) + \alpha(R + \gamma Q(S', A') - Q(S, A))$ 
8:      $S \leftarrow S'$ 
9:      $A \leftarrow A'$ 
10:  end while
11: end for

```

---

L'algorithme SARSA est un algorithme *on-policy*, les données utilisées pour l'apprentissage proviennent de la politique que l'agent apprend. L'algorithme SARSA se base sur *l'itération de la politique généralisée* où *l'évaluation* se fait par différence temporelle sur les estimations des paires d'états-action (voir la sous-section A.2.6 pour plus de détails). L'*amélioration* se fait quand la valeur estimée de la meilleure action change dans la politique  $\epsilon$ -greedy. L'algorithme de SARSA converge avec une probabilité de 1 vers la fonction de valeur des actions correspondant à la politique optimale, tant que toutes les paires d'états-action sont visitées une infinité de fois et que la politique converge à la limite vers la politique gloutonne (ceci peut être le cas si on a  $\epsilon = \frac{1}{t}$  où  $t$  est le nombre d'étapes de l'algorithme) [116].

L'algorithme Q-learning est un algorithme assez similaire à SARSA. La différence principale est que Q-learning tente d'estimer directement la meilleure politique. L'équation de mise à jour de Q-learning peut être obtenue en partant de celle de SARSA et en remplaçant l'estimation de la paire d'état-action suivante par l'estimation maximale des actions dans l'état suivant. Ceci donne lieu à l'équation suivante,

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)) \quad (2.4)$$

La différence entre SARSA et Q-learning est montrée dans la figure 2.2. Le déroulé de l'algorithme change aussi, l'agent choisit l'action selon une politique  $\epsilon$ -greedy dans l'état courant. L'agent observe la prochaine récompense ainsi que le prochain état. Ensuite, il corrige l'estimation actuelle par différence temporelle en utilisant l'équation 2.4. Tout cela est montré dans l'algorithme 4.

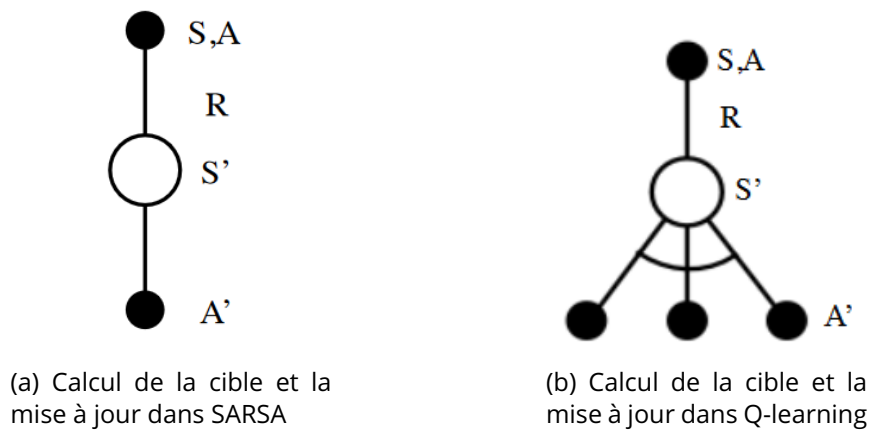


Figure 2.2 – Schémas représentant le calcul de la cible et la mise à jour dans les méthodes de contrôle basées sur la différence temporelle [116].

---

**Algorithme 4** L'algorithme tabulaire de Q-learning pour le calcul de la meilleure  $\pi^*$ .

---

**Inputs :**  $Q(s, a)$  - les valeurs des paires d'états-action initialisées arbitrairement sauf dans l'état final  $Q(F, \cdot) = 0$ .

$\alpha$  - Le taux d'apprentissage (le pas de l'évaluation).

$\epsilon$  - Probabilité de prendre la meilleure action.

- 1: **for**  $i$  allant de 1 à  $N$  **do**
  - 2: Initialiser l'état courant  $S$
  - 3: **while**  $S \neq F$  **do**
  - 4: Prendre l'action  $A$ , observer  $R, S'$
  - 5: Choisir l'action  $A'$  partant de  $S'$  suivant l'algorithme  $\epsilon$ -greedy sur  $Q$
  - 6:  $Q(S, A) \leftarrow Q(S, A) + \alpha(R + \gamma \max_a Q(S', a) - Q(S, A))$
  - 7:  $S \leftarrow S'$
  - 8: **end while**
  - 9: **end for**
- 

L'algorithme de Q-learning est un algorithme *off-policy*, les données utilisées dans l'apprentissage ne proviennent pas de la politique qui est estimée. En effet, les données sont générées par la politique

$\epsilon$ -Greedy sur les estimations actuelles de l'agent (car c'est cette politique qui est exécutée). En revanche, ce que l'agent veut réellement estimer, c'est la *politique greedy (gloutonne)* en fonction des estimations de l'agent (car c'est elle qui approche la politique optimale). Comme les politiques greedy et  $\epsilon$ -greedy sont assez proches, l'algorithme arrive à isoler les bonnes actions avec suffisamment de temps. Effectivement, ces deux politiques deviennent de plus en plus proches à mesure que l'algorithme interagit et apprend de son environnement. L'algorithme de Q-learning converge vers la fonction de valeurs des actions correspondant à la politique optimale  $q^*$  tant que toutes les paires d'états-action continuent à être mises à jour et que la séquence des paramètres du pas est décrétementée suivant les conditions classiques d'approximation stochastique [129].

## Conclusion

Dans ce chapitre, nous avons présenté les méthodes les plus classiques des approches d'apprentissage par renforcement. Ces approches incluent l'apprentissage par renforcement sans états ainsi que l'apprentissage par renforcement basé sur les états.

Dans l'apprentissage par renforcement sans état, nous distinguons les méthodes basées sur la politique qui représentent explicitement la politique de l'agent à l'aide d'une loi de probabilité, et les méthodes basées sur les valeurs qui la représente implicitement. Les approches basées sur les valeurs regroupent les méthodes qui représentent explicitement les incertitudes sur les estimations des actions et les approches qui ne représentent pas explicitement ces incertitudes.

Les éléments présentés sur les méthodes d'apprentissage par renforcement nous permettent d'orienter notre choix sur l'algorithme à prendre pour implémenter le système par choix de variation de signal et le système par choix de valeur du signal. Nous implémentons le système par choix de variation de signal avec l'apprentissage par renforcement sans états et le système par choix de valeur du signal par l'apprentissage par renforcement basé sur les valeurs. Ainsi, pour la suite de notre travail, nous avons choisi au moins un représentant pour chaque type d'apprentissage par renforcement basé sur les états : les algorithmes  $\epsilon$ -greedy, UCB ainsi que LRI et ses variantes. Concernant les approches d'apprentissage par renforcement basé sur les états, notre choix s'est porté sur les méthodes populaires SARSA et Q-learning.

# Chapitre 3

## Modèle des algorithmes de contrôle énergétique de bâtiment

Dans ce chapitre, nous commençons par poser les différents éléments composant le contexte de la thèse. Nous exposons ensuite les paradigmes que nous étudions et qui sont le paradigme dans lequel la décision du système porte sur la variation du signal, et celui dans lequel la décision porte sur le choix de la valeur du signal. Finalement, nous détaillons le modèle du système et de l'utilisateur. Ce modèle de l'utilisateur a été conçu sur la base de l'état de l'art présenté dans le chapitre 1 et servira de base pour tester et valider expérimentalement nos algorithmes.

### 3.1 Contexte

Dans ce qui suit, nous décrivons la situation que nous considérons tout au long de notre travail. Nous avons un bâtiment constitué d'une seule pièce qui est occupée par un utilisateur. La pièce est équipée d'un système qui permet de contrôler sa luminosité (c'est le système de contrôle). Dans ce contexte, l'utilisateur a besoin d'un certain niveau de luminosité afin de satisfaire ses besoins, et le système de contrôle aura besoin de consommer de l'énergie afin de maintenir une intensité lumineuse suffisante pour l'utilisateur. Nous considérons que le coût énergétique augmente proportionnellement à la valeur du signal, mais que le confort diminue en même temps. De ce fait, nous postulons l'existence d'une valeur sur laquelle le signal doit être réglé afin d'équilibrer entre les performances énergétiques et le confort de l'utilisateur. Évidemment, le signal devra être diminué afin d'arriver à la valeur fixée par le système. Cependant, la diminution du signal pourrait avoir un impact négatif sur l'utilisateur et l'amener à intervenir plus souvent sur le système pour remonter l'intensité du signal, augmentant ainsi la consommation d'énergie. Il y a donc un compromis que le système devra apprendre à régir entre consommation énergétique et confort de l'utilisateur. L'objectif du système de contrôle est de piloter le signal de façon à assurer cet équilibre, supposé, entre énergie et confort de l'utilisateur.

En outre, nous considérons que l'utilisateur peut interagir quand il veut avec le système et qu'il a une liberté totale pour fixer la valeur du signal lumineux. Ainsi, même si le système règle le signal lumineux à une certaine valeur, l'utilisateur a toujours la possibilité de la changer. Nous supposons, d'ailleurs, que le signal prend immédiatement la valeur choisie par l'utilisateur. Nous considérons aussi qu'à chaque fois que l'utilisateur intervient, il met le signal à sa valeur maximale. L'hypothèse que nous avons prise sur la liberté dont jouit l'utilisateur sur le choix de la valeur du signal lumineux dé-

coule de nos conclusions tirées de la littérature sur le confort lumineux et thermique de la section 1.2.

Dans le cadre de notre travail, nous considérons que le temps est discret et qu'il est divisé en étapes. À chaque étape, le signal prend une valeur donnée. Seule l'intervention de l'utilisateur fait partie de l'entrée du système. De plus, nous considérons que la fréquence d'intervention de l'utilisateur reflète son confort : plus l'utilisateur a tendance à réagir pour une valeur donnée, moins cette valeur lui est confortable. Nous considérons aussi que l'utilisateur se trouve dans une situation complètement homogène (pas de changement de luminosité à l'extérieur du bâtiment, pas de changement de l'activité de l'utilisateur, etc.). Finalement, nous considérons que les réactions de l'utilisateur dépendent de l'historique complet du signal et pas seulement de sa valeur courante. Ainsi, si les valeurs prises par le signal sont basses, l'utilisateur pourra devenir sensible à une valeur qui serait confortable si le signal avait pris des valeurs plus grandes. Ce point est d'importance cruciale dans notre travail, nous le détaillerons plus loin dans la section 3.4 où nous décrivons le modèle de l'utilisateur servant à tester les méthodes de contrôle énergétique du signal lumineux que nous proposons. Dans ce qui suit, nous décrivons les différents paradigmes que suivent ces méthodes.

## 3.2 Paradigmes

Comme dit précédemment, notre objectif est de concevoir un système de contrôle du signal lumineux tenant compte de la consommation d'énergie et du confort de l'utilisateur. Dans cette section, nous présentons les différents paradigmes que nous étudions dans le cadre de cette thèse ainsi que les spécificités caractérisant notre problématique.

Le type d'algorithme à utiliser pour le système de contrôle forme le point essentiel de cette thèse. À ce propos, nous distinguons, dans le cadre de notre travail, différents paradigmes concernant *la décision du système* et qui sont :

- Le paradigme dans lequel le système choisit la pente que prend le signal lorsque celui-ci diminue ainsi que la valeur sur laquelle le signal doit s'arrêter. Dans ce paradigme, chaque décision du système a une durée donnée et le système change éventuellement de décision à chaque fois que l'utilisateur intervient (ou quand une certaine durée est atteinte sans que l'utilisateur intervienne). Une analogie utile pour comprendre cette approche est que le système a un ensemble de tracés possibles pour les valeurs du signal au cours du temps et il devra trouver quel est le tracé satisfaisant ses critères de performances. Nous nommons ce paradigme **choix de variation du signal**, nous le détaillerons dans le chapitre 4.
- Le paradigme dans lequel le système décide de la valeur sur laquelle doit être fixé le signal. Les valeurs choisies par le système peuvent être maintenues pour une durée donnée et sont toujours dans le voisinage de la valeur actuelle. Ce paradigme est nommé **choix de valeur du signal**, nous le détaillerons dans le chapitre 5.

Ces deux approches sont naturelles à considérer et nous semblent pertinentes, elles feront l'objet d'une étude détaillée dans notre travail. La vitesse de convergence des algorithmes, les gains en matière de coûts énergétiques ainsi que l'impact sur le comportement de l'utilisateur formeront des



points essentiels de cette étude. Dans les paragraphes qui suivent, nous précisons certains éléments concernant les connaissances du système ainsi que le caractère non stationnaire du problème.

Compte tenu du fait qu'on ne peut pas prédire le comportement de l'utilisateur (selon les conclusions tirées du *confort adaptatif*), nous considérons que le système de contrôle énergétique ne connaît rien des principes régissant le comportement de l'utilisateur. Le système de contrôle apprendra par interaction avec l'utilisateur comment contrôler le signal de façon à équilibrer entre énergie et confort. De cette façon, le système s'améliore au fur et à mesure et identifie progressivement quelles sont les stratégies à prendre. Nous disons que le système est *model-free* (c.-à-d. qu'il n'a pas de connaissances *a priori* sur l'utilisateur).

Comme nous l'avons dit précédemment, la sensibilité de l'utilisateur change avec le temps et dépend des valeurs passées du signal. Puisque l'utilisateur se comporte différemment suivant l'historique du signal, cela veut dire que l'utilisateur est *non-stationnaire* du point de vue du système. D'autant plus, la non stationnarité de l'utilisateur dépend du comportement passé du système, créant ainsi une boucle à rétroaction dans notre problématique. Les actions du système sont choisies sur la base des réactions passées de l'utilisateur et conditionnent à leur tour son comportement future puisque celui-ci va s'adapter au système. La figure 3.1 illustre l'interaction entre le système et l'utilisateur.

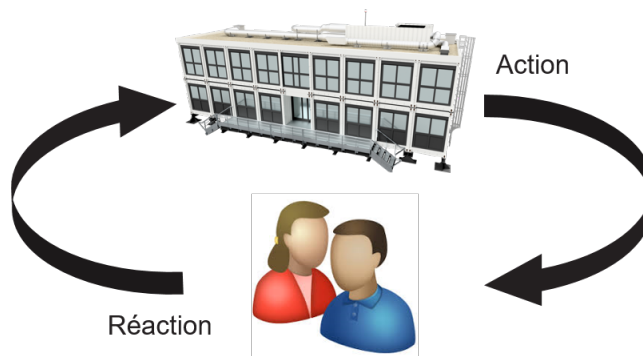


Figure 3.1 – Interaction entre bâtiment intelligent et usager.

Compte tenu du fait qu'on ne peut pas modéliser le comportement de l'utilisateur et que celui-ci est non stationnaire, l'utilisation de l'apprentissage nous apparaît l'approche la mieux adaptée si l'on veut que le système arrive à contrôler efficacement le signal lumineux. Le point à traiter ensuite serait d'identifier quel type d'apprentissage devra être choisi pour le système. Comme nous l'avons montré dans la partie de l'état de l'art, il existe plusieurs alternatives potentielles dans l'apprentissage statistique. Toutefois, la direction semblant être la plus naturelle et pertinente pour concevoir le système de contrôle nous paraît être l'apprentissage par renforcement. Le problème qu'on se propose de résoudre nécessite que l'agent apprenne par interaction avec l'environnement les actions adéquates à prendre. Par ailleurs, dans la mesure où l'apprentissage par renforcement permet de formuler des algorithmes permettant à un agent agissant dans un environnement d'apprendre par essai et erreur à atteindre un objectif dont il ne connaît rien au préalable, l'apprentissage par renforcement semble être l'approche la plus convenable répondant à notre problématique.

En considérant l'apprentissage par renforcement dans le contexte du contrôle énergétique de bâtiment, le système peut être considéré comme l'agent et l'utilisateur comme (faisant partie de) l'en-

vironnement [20, 101]. Le système pourra ainsi apprendre de proche en proche les actions à prendre permettant d'équilibrer entre énergie et confort. Le système le fera en essayant diverses actions et en observant les réactions de l'utilisateur pour ensuite ajuster le choix des actions en fonction de celui-ci. La figure 3.2 montre le parallèle entre l'apprentissage par renforcement et l'interaction entre usager et bâtiment.



Figure 3.2 – Parallèle entre l'apprentissage par renforcement et l'interaction entre usager et bâtiment.

Nous proposons donc d'utiliser les principes de l'apprentissage par renforcement pour concevoir le *système de contrôle*. Ce point est intéressant, à fortiori, car nous considérons plusieurs paradigmes dans la thèse, et il y a aussi plusieurs types d'apprentissage par renforcement. La question est de savoir quel type d'algorithme d'apprentissage par renforcement correspond le mieux à quel paradigme. Dans la section suivante, nous présentons une discussion sur l'apprentissage par renforcement dans le contexte du contrôle du signal lumineux dans un bâtiment. Nous détaillerons, pour chaque paradigme, le type d'apprentissage par renforcement le plus approprié en vue de formuler l'algorithme de contrôle. Nous présentons le contrôle ainsi que des éléments formels composant notre problème. Finalement, nous décrivons dans la section qui suit le modèle de l'utilisateur sur lequel nous nous sommes appuyés pour valider nos simulations.

### 3.3 Le contrôle du point de vue du système

En utilisant l'apprentissage par renforcement de façon classique, les changements potentiels affectant l'environnement (l'utilisateur) sont modélisés comme un ensemble fini et discret d'états, où chaque action de l'agent provoque une transition de l'état courant vers un autre état. Comme nous l'avons dit dans l'état de l'art, ce modèle d'apprentissage par renforcement est nommé apprentissage par renforcement *basé sur les états*. La sensibilité de l'utilisateur étant inconnue du système, le système devra l'estimer à travers les interactions passées entre le système et l'utilisateur. Cette estimation de la sensibilité de l'utilisateur servira de base sur laquelle le système se repose pour apprendre à agir dans l'environnement. Elle a la même fonction que l'état (même si elle ne nous renseigne pas sur l'état réel de l'utilisateur), nous l'appellerons donc simplement état. Néanmoins, comme nous n'avons pas accès à une représentation d'états pour laquelle l'environnement est parfaitement connu, il faudra faire attention à en choisir une qui puisse permettre à l'agent d'apprendre à agir correctement dans l'environnement. L'apprentissage par renforcement à base d'états correspond bien au paradigme **choix de valeur du signal** (voir la section 3.2), l'action correspond à la prochaine valeur choisie par le système et l'état correspond à l'élément en fonction duquel le système choisit cette valeur. Ainsi, nous choisissons de prendre l'apprentissage basé sur les états pour formuler notre système par choix de valeur. Nous notons néanmoins que l'apprentissage basé sur les états est connu pour avoir

une combinatoire qui rend l'apprentissage assez long.

Cependant, il existe d'autres approches d'apprentissage par renforcement n'utilisant pas d'états (ce qu'on appelle *l'apprentissage par renforcement sans états* comme mentionné dans l'état de l'art). Et comme il est difficile de représenter avec précision l'évolution du comportement de l'utilisateur avec un ensemble fini d'états de taille raisonnable, nous proposons de considérer les approches sans états pour notre problème. Nous pouvons voir que les approches d'apprentissage par renforcement sans états correspondent bien au paradigme par **choix de variation du signal** (voire section 3.2). Dans le cas où l'apprentissage par renforcement sans états est utilisé pour le paradigme par **choix de variation du signal**, l'agent devrait choisir la meilleure action parmi un ensemble d'actions donné. L'action est, dans ce cas-là, la manière dont le système devra baisser la valeur du signal. Ce qui signifie que la décision de l'agent ne porte pas uniquement sur la prochaine valeur du signal, mais également sur les valeurs futures. Puisque les futures valeurs sont déjà choisies par le système, nous pouvons entrevoir que l'agent n'aura pas nécessairement besoin d'état pour guider ses décisions. L'action à elle seule déterminerait l'état réel de l'utilisateur. De ce fait, l'hypothèse qu'une action unique est la meilleure quel que soit le comportement de l'utilisateur semble a priori raisonnable. Même si le système ne connaît rien de l'utilisateur, l'évolution des performances sera essentiellement déterminée par l'action appliquée et de ce fait l'apprentissage sans états pourrait parfaitement convenir pour le modèle de **choix de variation de signal**. En revanche, la conception de la fonction de récompense représentera un élément essentiel de ces approches qu'il va falloir considérer avec précaution : elle devra refléter les performances énergétiques du système, mais aussi prendre le confort de l'utilisateur en considération. Notons tout de même que l'apprentissage basé sur les états permet d'avoir une combinatoire sur les actions qui est plus simple à gérer pour les algorithmes.

Dans les deux types d'approches, le système de contrôle devra maximiser une fonction d'utilité pour équilibrer au mieux entre consommation d'énergie et le confort de l'utilisateur. La fonction d'utilité détermine la qualité d'une action vis-à-vis du contrôle efficace du signal et elle doit guider correctement le système de contrôle vers ce but.

Dans ce qui suit, nous définissons les éléments communs aux deux types de modèles étudiés dans notre travail. Nous supposons que le temps et les valeurs du signal sont discrétisés. La valeur du signal à l'étape  $t$  est désignée par  $v_t$ . L'ensemble des valeurs que le signal peut prendre est  $S \subseteq \{s \in \mathbb{R} \mid s_{min} \leq s \leq s_{max}\}$  où  $s_{min}$  et  $s_{max}$  sont respectivement les valeurs minimale et maximale du signal. Le signal évolue dans le temps suite aux actions prises par le système et nous considérons l'aire délimitée par l'évolution du signal et l'axe temporel comme critère pour évaluer la consommation énergétique du système. Dans toutes les expériences que nous considérerons par la suite, le signal est initialisé à sa valeur maximale  $s_{max}$ . Le système de contrôle choisira alors l'action à prendre suivant sa politique. Comme nous l'avons dit dans l'état de l'art, la politique est l'algorithme servant à choisir l'action à prendre. La politique peut prendre diverses formes (nous détaillerons ce point dans les chapitres suivants), mais dans tous les cas, l'agent choisit toujours l'action suivant sa politique. Dans la majorité des algorithmes considérés dans ce travail, la politique initiale n'a pas d'*a priori* sur la qualité des actions (par exemple, si une politique est représentée par un vecteur stochastique, la probabilité de choisir chaque action est la même pour toutes les actions). Les stratégies choisies au départ seront a priori peu efficaces, l'agent apprendra au fur et à mesure

quelles sont les bonnes actions à prendre. De même que le type de décision de l'agent peut varier suivant le paradigme système (système par variation ou système par valeur), les algorithmes aussi pourront varier dans un même type de système. La figure 3.3 montre un exemple du déroulé du contrôle effectué par le système.

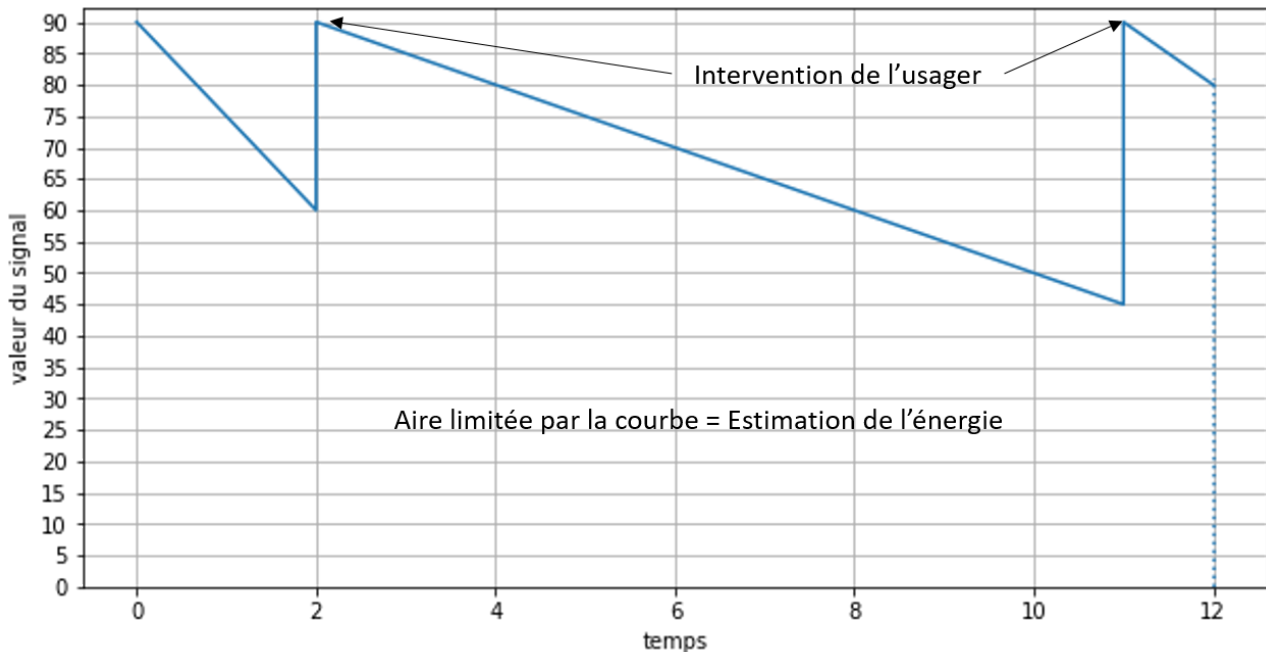


Figure 3.3 – Graphique représentant la réaction de l'utilisateur et son impact énergétique.

L'exemple présenté dans la figure 3.3 illustre comment le système contrôle la valeur du signal et la régule en fonction des réactions de l'utilisateur. Dans cet exemple, nous supposons que l'action est la pente que prend le signal quand il est diminué. Le signal est initialisé à la valeur maximale  $s_{max} = 90$ , la valeur minimale est la valeur nulle ( $s_{min} = 0$ ). Le système baisse la valeur du signal sur les deux premières étapes en diminuant à chaque fois le signal de 15 unités. La valeur en  $t = 2$  vaut alors 60. À cette étape, l'utilisateur intervient sur la valeur du signal pour la remettre à sa valeur maximale  $s_{max} = 90$ . Après cela, le système baisse plus doucement la valeur du signal (5 unités par pas temps) jusqu'à ce qu'elle atteigne la valeur 45 à l'étape 11. L'utilisateur réagit ensuite et remet le signal à sa valeur maximale, puis le système rebaisse la valeur du signal de 10 unités à l'étape suivante dans laquelle le signal prend la valeur 80.

L'exemple ci-dessus ne montre que l'évolution du signal au cours du temps, il précise à quel moment l'utilisateur intervient sur le signal et comment le système régule la valeur du signal en réponse aux réactions de l'utilisateur. L'exemple ne détaille ni la politique de l'agent, ni comment la mettre à jour, ni l'évolution de l'utilisateur. Ces éléments seront détaillés dans des sections prochaines. Dans le but de tester les approches que nous proposons, nous avons conçu un modèle comportemental de l'utilisateur permettant de décrire ses réactions face à un changement sur les valeurs du signal. Nous précisons que le modèle présenté dans cette partie servira seulement à tester la pertinence des méthodes qui seront proposées dans le présent travail et aussi à les valider. En effet, toutes les expériences que nous présentons ne supposent rien sur l'utilisateur ni la façon dont le contrôle du signal lumineux affecte son confort. Ce modèle d'utilisateur ne sera donc pas exploité par les algorithmes (ni

aucun modèle). En effet, nous insistons sur le fait que, dans le cadre de notre travail, nous aborderons seulement l'apprentissage sans modèle (*model-free* [116]) dans lequel l'agent est censé apprendre sans aucun modèle explicite de la dynamique de l'environnement, ce modèle servira seulement à faire des simulations permettant d'évaluer les algorithmes que nous proposons.

### 3.4 Modèle de l'utilisateur

Dans la section précédente, nous avons posé les notions essentielles permettant de décrire le système et l'évolution du signal. Néanmoins, comme nous l'avons précisé, afin d'analyser l'efficacité du système de contrôle, nous aurons besoin d'un modèle simulé de l'utilisateur qui nous permettra d'évaluer si la qualité du contrôle est satisfaisante. Dans cette section, nous commençons par présenter les hypothèses dont nous nous sommes servis pour définir notre modèle de l'utilisateur. Nous soulignons encore une fois le fait que les hypothèses sur la base desquelles nous formulons notre modèle s'accordent bien avec les principes du confort adaptatif (vu dans l'état de l'art), puisque l'évolution de l'utilisateur est principalement déterminée par les actions du système. Dans cette section, nous formulons un modèle à la fois simple et analytique tenant compte de ces hypothèses. Ce modèle sert à évaluer les algorithmes d'apprentissage par renforcement et ne sera pas exploité par l'apprentissage.

Dans le cadre de notre travail, nous prenons en compte les hypothèses suivantes :

1. **L'hypothèse de réactivité** : L'utilisateur a tendance à réagir très souvent pour les valeurs au-dessous de sa valeur de confort et peu souvent pour les valeurs au-dessus de sa valeur de confort. Cette hypothèse implique que la probabilité de réaction de l'utilisateur augmente quand la valeur du signal est inférieure à la valeur de confort.
2. **L'hypothèse de mémoire** : L'utilisateur est réactif non seulement aux conditions actuelles, mais aussi aux variations passées du signal. L'utilisateur a une mémoire du passé qui est dominée par les expériences les plus récentes (similairement aux travaux de Humphreys [59]). Les expériences récentes influencent le plus la sensibilité ainsi que le comportement de l'utilisateur.
3. **L'hypothèse des tendances dues au passé** : Plus l'utilisateur a réagi par le passé en réponse au contrôle effectué par le système, plus il aura tendance à réagir rapidement à des valeurs de signal qui lui sont inconfortables dans le présent. Cette hypothèse implique que l'utilisateur a tendance à réagir plus rapidement si les valeurs passées du signal sont faibles, même quand la valeur actuelle du signal est élevée. Les changements de l'utilisateur s'opèrent comme si sa valeur de confort avait changé.

Après avoir présenté les hypothèses concernant le comportement de l'utilisateur, nous entamons maintenant les détails de la conception du modèle de l'utilisateur. En premier lieu, nous considérons que les réactions de l'utilisateur sont décrites par une loi de probabilité qui varie dans le temps et qui dépend de la valeur courante du signal. Formellement, à chaque étape  $t \in \mathbb{N}$ , l'utilisateur a une probabilité notée  $p_t(v_t)$  d'agir sur la valeur du système, où  $v_t$  est la valeur du signal choisie par le système à l'étape  $t$ . Nous considérons par ailleurs que la probabilité d'intervention de l'utilisateur pour une valeur donnée du signal est déterminée par le confort de l'utilisateur quand le signal prend cette valeur. Plus exactement, nous supposons que la sensibilité de l'utilisateur augmente (resp. diminue) lorsque la valeur du signal diminue (resp. augmente).

Afin de tenir compte de l'**hypothèse de réactivité** présentée précédemment, nous postulons que, à chaque étape  $t$ , la probabilité d'intervention est monotone décroissante sur la valeur du signal. Afin de tenir compte de l'**hypothèse des tendances dues au passé**, la mise à jour des paramètres du modèle de l'utilisateur doit faire intervenir les valeurs passées de ces paramètres et dépendre de l'intensité de la fréquence d'intervention de l'utilisateur. De plus, pour tenir compte de l'**hypothèse de mémoire**, la mise à jour devrait dépendre plus fortement des valeurs passées récentes des paramètres de l'utilisateur.

La fonction la plus utilisée pour décrire la probabilité de réaction de l'utilisateur est la fonction sigmoïde ([124]). Cette fonction nous permet de distinguer des zones dans l'intervalle de valeurs du signal pour lesquelles l'utilisateur réagit différemment :

- Une zone dans laquelle l'utilisateur intervient beaucoup, appelée "zone d'inconfort". Dans la figure 3.4, cette zone pourrait être représentée par l'intervalle  $[0, 15]$  des valeurs du signal.
- Une zone dans laquelle l'utilisateur intervient rarement qu'on appelle la "zone de confort". Dans la figure 3.4, cette zone pourrait être représentée par l'intervalle  $[55, 90]$  des valeurs du signal.
- Une zone pour laquelle les réactions de l'utilisateur sont incertaines. Dans la figure 3.4, cette zone pourrait être représentée par l'intervalle  $[15, 55[$  des valeurs de signal.

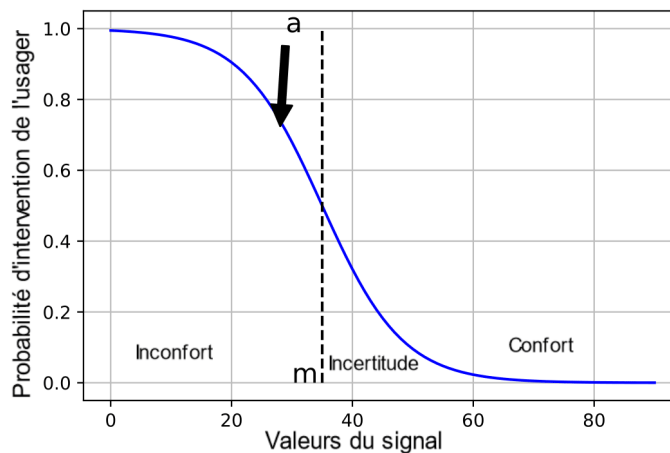


Figure 3.4 – Exemple de la fonction sigmoïde qui retourne pour chaque valeur la probabilité d'intervention de l'utilisateur.

La probabilité d'intervention est, à chaque étape  $t$ , une fonction sigmoïde déterminée par deux paramètres  $m_t$  et  $a_t$ . Le paramètre  $m_t$  représente la valeur en dessous de laquelle l'utilisateur intervient très souvent, nous nommons ce paramètre le *seuil d'intervention*. Le paramètre  $a_t$  représente comment l'intensité de la valeur du signal discrimine les réactions de l'utilisateur, ce paramètre influence la pente de la fonction sigmoïde et reflète à quelle mesure il y a de l'aléatoire au niveau des réactions de l'utilisateur.

La fonction sigmoïde et les paramètres servant à décrire l'utilisateur sont illustrés dans la figure 3.4. Plus la valeur de  $m_t$  est grande, plus l'utilisateur devient sensible à la diminution du signal. Quand la valeur de  $a_t$  est grande, la fonction sigmoïde prend une forme de fonction d'escalier, et l'utilisateur

intervient toujours pour les valeurs faibles du signal et il n'intervient jamais pour les valeurs fortes. Et quand la valeur de  $a_t$  est faible, la fonction sigmoïde devient lisse et la probabilité d'intervention baisse doucement pour des valeurs de plus en plus faibles du signal. Les paramètres  $m_t$  et  $a_t$  de la fonction sigmoïde -servant à modéliser l'utilisateur- évoluent en fonction du contrôle passé (ces paramètres sont dynamiques).

L'utilisateur a une probabilité variable dans le temps d'augmenter la valeur du signal. Dans le cadre de notre travail, nous faisons l'hypothèse que quand la sensibilité de l'utilisateur augmente, la valeur de  $a_t$  augmente aussi. Formellement, la probabilité de réaction à l'étape  $t$  est définie avec la fonction sigmoïde de la façon suivante,

$$p_t = 1 - \frac{1}{1 + e^{-a_t(v_t - m_t)}} = \frac{1}{1 + e^{a_t(v_t - m_t)}}$$

Les valeurs initiales des paramètres décrivant l'utilisateur sont notées  $a_0$  et  $m_0$ . La valeur  $m_0$  représente la valeur de confort de l'utilisateur. Nous considérons que ces valeurs sont les valeurs minimales pour chacun de ces paramètres respectivement et qu'elles représentent un utilisateur dans son état le moins sensible, indépendamment de l'effet de la variation du signal sur lui. Ceci signifie que l'utilisateur ne peut se trouver à un état plus confortable que dans celui auquel il est au moment où il rejoint la pièce au tout début du contrôle. À chaque étape, les paramètres  $a_t$  et  $m_t$  sont mis à jour en fonction des variations présentes et passées du signal.

Le paramètre  $a_t$  est mis à jour comme détaillé dans les équations suivantes :

$$\begin{aligned} \text{— } effPreVal &= \frac{s_{max} - v_t}{s_{max} - s_{min}} \\ \text{— } effPasVar &= Sl_t = pe \cdot \frac{sl_t}{sl_{max}} + (1 - pe) \cdot Sl_{t-1} \\ \text{— } a' &= \frac{effPreVal + effPasVar}{2} \\ \text{— } a &= \frac{1}{1 - a'} - 1 + a_0 \end{aligned}$$

Les valeurs  $effPreVal$  et  $effPasVar$  représentent respectivement l'effet de la valeur actuelle et des valeurs passées du signal sur le paramètre  $a_t$ . Le paramètre  $pe$  représente le poids donné à l'observation actuelle dans la dynamique de l'utilisateur ( $pe \in [0, 1]$ ,  $1 - pe$  est donc le poids attribué au passé). Par conséquent, le paramètre  $pe$  représente la mémoire de l'utilisateur sur l'historique des valeurs du signal. Lorsque  $pe$  est proche de 0, les réactions de l'utilisateur dépendent principalement de la valeur actuelle du signal et lorsque  $pe$  est proche de 1, les réactions de l'utilisateur ont tendance à dépendre de l'historique complet du signal. Par conséquent, le paramètre  $pe$  nous permet de définir une mémoire des valeurs passées du signal pour l'utilisateur. Différentes valeurs de  $pe$  correspondant à différentes tailles de la mémoire de l'utilisateur. Dans cette thèse, nous considérons trois valeurs du paramètre  $pe$  qui sont 0.05, 0.35, et 0.95 qui correspondent respectivement à un utilisateur avec une mémoire à long terme, un utilisateur avec une mémoire à moyen terme et un utilisateur avec une mémoire à court terme.

Nous avons aussi  $a'$  qui est la moyenne entre les deux valeurs  $effPreVal$  et  $effPasVar$ . La valeur  $effPreVal$  représente l'écart entre la valeur actuelle du signal et sa valeur maximale normalisée entre 0 et 1. Et  $effPasVar$  est la moyenne mobile exponentielle des variations passées du signal normalisées aussi entre 0 et 1. Nous pouvons voir dans la mise à jour de  $a_t$  que

lorsque  $a'$  s'approche de 1, le paramètre  $a_t$  s'approche de l'infini. Cela peut être interprété comme le fait que plus le contrôle est drastique pour l'utilisateur, moins il y a de bruit dans les réactions de l'utilisateur.

Nous désignons par  $\bar{p}_t$  la moyenne mobile exponentielle des probabilités d'intervention des  $t + 1$  premières étapes. Le paramètre  $m_t$  est mis à jour, à chaque étape, de sorte que la différence entre la nouvelle valeur de  $m_t$  et sa valeur initiale  $m_0$  soit proportionnelle aux trois grandeurs suivantes :

- la probabilité d'intervention moyenne actuelle  $\bar{p}_t$ ,
- la différence séparant  $m_0$  et la valeur maximale du signal  $s_{max}$ ,
- et le poids donné à l'effet du présent sur l'utilisateur  $pe$ .

Cette observation signifie que pour une même valeur du signal, l'utilisateur aurait tendance à intervenir plus souvent si les valeurs passées du signal étaient plus faibles. Le modèle proposé pour simuler l'utilisateur ainsi que les équations de mise à jour de ce modèle représentent une manière de créer des effets de mémoire au niveau du comportement de l'utilisateur tout en liant les réactions passées de l'utilisateur à sa réactivité présente. Tout ceci est décrit dans les équations suivantes :

- $\bar{p}_t = pe.p_t + (1 - pe).\bar{p}_{t-1}$
- $m_t = m_0 + \bar{p}_t.(s_{max} - m_0)$

La figure 3.5 montre une évolution potentielle de l'utilisateur vers un état de plus en plus sensible suite à une baisse du signal provoquée par le système. Nous remarquons bien que les valeurs de  $a_t$  de  $m_t$  s'accroissent avec le temps (la valeur de  $m_t$  passe de 30 à 50 puis 70). Il est aussi possible de voir que l'utilisateur devient plus sensible quand  $m_t$  devient plus grande, car la surface qui est entre la courbe et le segment des valeurs s'agrandit avec l'accroissement de  $m_t$ . L'accroissement de  $a_t$  ne change pas la surface sous la courbe, mais change la mesure dans laquelle les valeurs du signal discriminent le comportement de l'utilisateur.

L'algorithme 5 décrit comment est simulé le comportement de l'utilisateur selon le modèle de l'utilisateur qui a été proposé. Chaque itération dans la boucle **while** de l'algorithme 5 correspond à une étape. À chaque étape, le système de contrôle choisit une valeur. Juste après, l'utilisateur choisit de réagir ou non en suivant le modèle sigmoïdal. Les paramètres du modèle sont alors mis à jour comme expliqué ci-dessus, traduisant un changement potentiel du comportement de l'utilisateur. Nous explicitons dans les paragraphes qui suivent les fonctions utilisées dans l'algorithme 5.

Dans l'algorithme 5, la fonction *controlStep()* est la fonction qui retourne la valeur courante choisie par le système  $v$  ainsi que la pente courante  $sl$ . La pente  $sl$  est la différence entre la valeur courante et la valeur précédente. Dans toutes nos simulations, la valeur courante  $v$  est choisie par les algorithmes que nous proposons dans le cadre de cette thèse. Dans cette sous-section, nous ne détaillons ni le choix de la valeur courante ni la partie renforcement des algorithmes de contrôle. Précisons aussi que dans toutes les simulations, un nombre fini d'étapes est effectué. L'algorithme 5 récapitule les opérations effectuées durant la mise à jour du modèle de l'utilisateur.

La fonction *lectureProbaInterv(a, m, v)* retourne la probabilité  $p$  d'intervention de l'utilisateur suivant le seuil courant d'intervention  $m$ , la valeur courante du paramètre de discrimination  $a$ , ainsi que



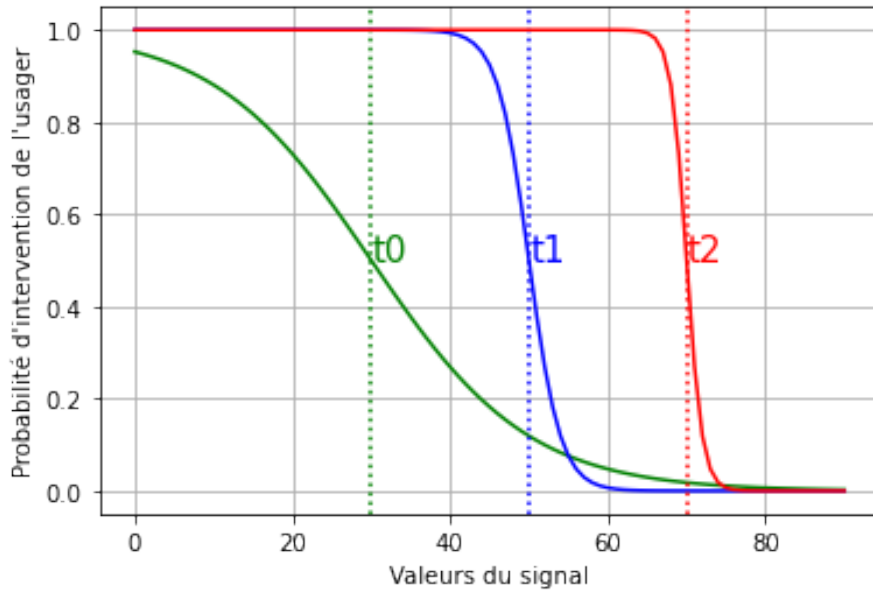


Figure 3.5 – Exemple de l'évolution de l'utilisateur sur les trois premières étapes ( $t_0, t_1, t_2$ ). Les changements de l'utilisateur s'opèrent comme si sa valeur de confort avait changé.

la valeur courante du signal  $v$ . La probabilité est retournée par le modèle sigmoïdal de paramètres  $a$  et  $m$  introduits auparavant.

La fonction  $userInterv(p)$  prend en entrée la probabilité d'intervention de l'utilisateur à l'étape courante  $p$  et retourne la valeur choisie par l'utilisateur s'il intervient et la valeur courante sinon. Comme nous l'avons dit précédemment, cette valeur est la valeur maximale  $s_{max}$ . L'intervention de l'utilisateur est simulée par tirage aléatoire d'une loi de Bernoulli de paramètre  $p$ . Comme nous l'avons mentionné, la valeur de  $p$  dépend de la valeur courante décrite grâce à la fonction sigmoïde.

Considérons quelques exemples afin d'illustrer comment l'algorithme 5 change l'état de l'utilisateur. La figure 3.6 représente des évolutions potentielles que l'algorithme 5 pourrait prendre avec les paramètres initiaux de la simulation suivants  $v_0 = s_{max} = 90$ ,  $s_{min} = 0$ ,  $m_0 = 35.0$  et  $a_0 = 0.1$ . Dans 3.6, nous supposons que le système de contrôle dispose deux actions possibles : baisser la valeur du signal de 5 unités ou de 30 unités (dénotées -5 et -30 respectivement).

L'arbre de la figure 3.6 montre les évolutions possibles de l'utilisateur pour les cinq premières étapes ( $t \in \{0, 1, 2, 3, 4\}$ ). Dans ce qui suit, nous considérons un exemple d'exécution (étiqueté "Scénario"). Dans cet exemple, nous nous intéressons aux stratégies de l'agent et la manière dont l'utilisateur change et réagit à ces stratégies, nous regardons également quelles sont les alternatives qu'aurait pu prendre le système ainsi que leur impact sur l'utilisateur.

À la première étape, l'utilisateur reste dans un état d'inaction quelle que soit la stratégie prise par le système. En effet, nous voyons bien que la valeur de  $m_1$  reste très faible pour les deux stratégies du système (baisser la valeur du signal de 30 unités ou de 5 unités). Dans le cas où le système baisse la valeur du signal de 30 unités au tout début, la décision du système devient alors déterminante sur l'état de l'utilisateur. Nous voyons bien que si le système baisse encore une fois l'intensité du signal de

---

**Algorithme 5** Algorithme de mise à jour des paramètres  $a$  et  $m$  liés aux probabilités de réaction de l'utilisateur.

---

```

1:  $a \leftarrow a_0$ 
2:  $m \leftarrow m_0$ 
3:  $v \leftarrow s_{max}$ 
4:  $v_{pre} \leftarrow s_{max}$ 
5:  $v_{ac} \leftarrow s_{max}$ 
6:  $sl \leftarrow 0$ 
7: while True do
8:    $v \leftarrow controlStep()$   $\triangleright$  Fonction renvoyant la valeur choisie par le système.
9:    $sl \leftarrow v - v_{pre}$ 
10:   $p \leftarrow lectureProbaInterv(a, m, v)$   $\triangleright$  Fonction qui retourne la probabilité d'intervention de l'usager.
11:   $effPreVal \leftarrow \frac{s_{max}-v}{s_{max}-s_{min}}$ 
12:   $sl_{adj} \leftarrow \min(sl, 0)$   $\triangleright$  Quand le système monte la valeur du signal l'effet de la variation du signal n'est pas considérée.
13:   $effPasVar \leftarrow pe \cdot \frac{sl_{adj}}{sl_{max}} + (1 - pe) \cdot effPasVar$ 
14:   $a_{inter} \leftarrow \frac{effPreVal + effPasVar}{2}$ 
15:   $a \leftarrow \frac{1}{1-a_{inter}} - 1 + a_0$ 
16:   $\bar{p} \leftarrow pe \cdot p + (1 - pe) \cdot \bar{p}$ 
17:   $m \leftarrow m_0 + \bar{p} \cdot (s_{max} - m_0)$ 
18:   $v_{pre} \leftarrow v_{ac}$ 
19:   $v_{ac} \leftarrow userInterv(a, m)$   $\triangleright$  Fonction qui renvoie la valeur fixée par l'usager s'il intervient et la valeur actuelle sinon.
20: end while

```

---

30 unités, l'utilisateur se retrouve dans un état assez défavorable ( $m_2 = 53.77$  en orange). En revanche, si la valeur du signal est baissée de cinq unités, l'utilisateur reste toujours dans un état confortable ( $m_2 = 35.96$  en bleu).

Comme nous le voyons, si le système décide de baisser la valeur du signal de 30 unités une troisième fois, l'utilisateur se retrouve dans un état inconfortable quelque soit les stratégies choisies par la suite. D'ailleurs, nous voyons bien que la probabilité d'intervention de l'utilisateur est de 1.0 à ce moment-là, ce qui signifie que l'utilisateur va forcément intervenir à l'étape suivante.

Partant de l'arc étiqueté "Intervention", nous avons les évolutions possibles des stratégies du système dans le cas où l'utilisateur intervient à la troisième étape. Contrairement au tout début où l'utilisateur est dans un état de repos pour les deux stratégies, l'état de l'utilisateur diffère selon qu'on prend la stratégie baissant le signal de 5 unités ou la stratégie baissant le signal de 30 unités ( $m_3 = 55.44$  (en orange) pour la stratégie -5, contre  $m_3 = 74.69$  (en rouge) pour la stratégie -30. De ce fait, même si l'intensité du signal est à son maximum, l'utilisateur est cependant plus sensible qu'auparavant, car

les valeurs prises par le signal avaient été trop faibles et ne semblaient pas satisfaire ses besoins en termes de confort.

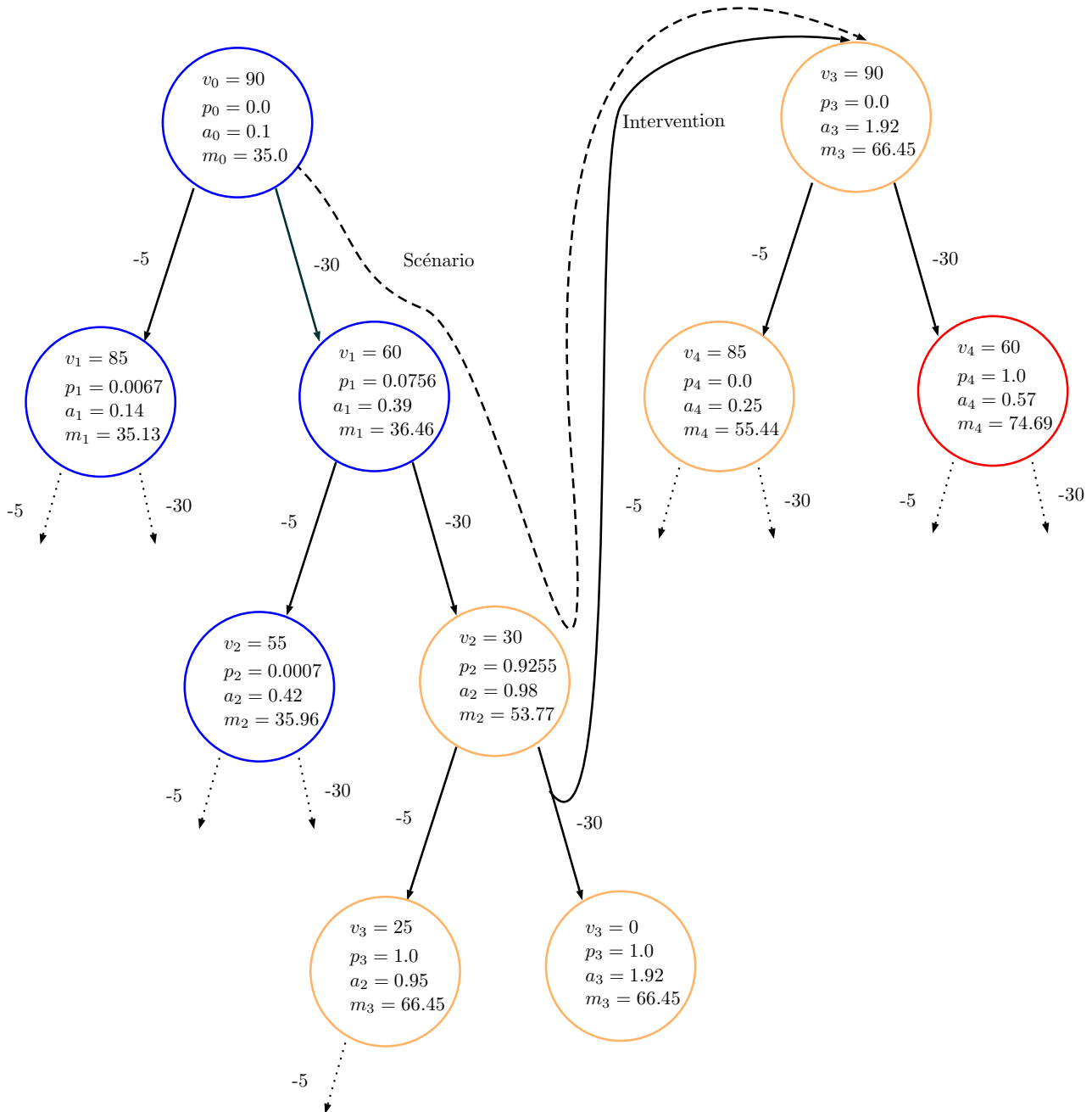


Figure 3.6 – Quelques évolutions possibles de l'utilisateur sur les cinq premières étapes ( $t = 0$ ,  $t = 1$ ,  $t = 2$ ,  $t = 3$ ,  $t = 4$ ). Chaque nœud de cet arbre représente un état potentiel de la simulation à l'étape courante et contient des informations sur l'utilisateur ainsi que la valeur du signal. Les informations contenues dans chaque nœud sont la valeur courante du signal, la probabilité d'intervention de l'utilisateur, la valeur de  $a$  ainsi que la valeur de  $m$ . Les arcs sont étiquetés par la stratégie choisie par le système. Pour chaque arc reliant deux sommets, la simulation va de l'état correspondant au sommet source vers l'état correspondant au sommet destination quand la stratégie associée à l'arc est appliquée par le système. L'arc étiqueté "Intervention" reflète une évolution possible de la simulation dans laquelle l'utilisateur intervient sur la valeur du signal. L'arc en pointillé étiqueté "Scénario" représente le scénario que nous considérons dans cet exemple et que nous commentons dans le texte.

Avant de clore ce chapitre, commentons davantage le fonctionnement ainsi que les propriétés de la mémoire dans le modèle de l'utilisateur, en commençant par préciser quelques points de terminologie que nous utiliserons dans notre travail. Dans le reste de notre travail, nous utilisons le terme *sensibilité de l'utilisateur* pour désigner plusieurs choses. Premièrement, nous pouvons considérer que la sensibilité de l'utilisateur est la probabilité d'intervention de l'utilisateur à l'étape courante pour la valeur courante qui est  $p_t(v_t)$ . Effectivement, si nous partons de ce principe, nous voyons bien que plus  $p_t(v_t)$  s'approche de 1 plus l'utilisateur est à même d'intervenir sur le signal, ce qui implique donc que la valeur actuelle du signal ne lui convient pas. D'un autre côté, plus  $p_t(v_t)$  s'approche de 0 moins il y a de chance que l'utilisateur intervienne, ce qui veut dire que l'utilisateur se trouve dans un état de confort. Par ailleurs, nous pouvons aussi dire que la sensibilité de l'utilisateur correspond à sa réactivité pour le signal dans sa globalité. Par exemple, nous pouvons considérer qu'un utilisateur est plus sensible qu'un autre si le seuil d'intervention du premier est supérieur au seuil d'intervention du second. Puisque la valeur de  $m_t$  est linéaire à  $\bar{p}_t$ , nous pouvons considérer que la sensibilité de l'utilisateur correspond à la moyenne mobile exponentielle des probabilités d'intervention de l'utilisateur. Dans ce cas, la sensibilité de l'utilisateur dépend de sa mémoire. Puisque la mémoire de l'utilisateur conditionne son comportement, nous aurons besoin d'avoir une compréhension approfondie de son effet sur l'utilisateur. Ceci nous aidera à orienter correctement nos interprétations du comportement des algorithmes de contrôle que nous testons.

## Conclusion

Dans ce chapitre, nous avons commencé par présenter les hypothèses concernant le contexte du problème ainsi que l'utilisateur en suivant le principe du confort adaptatif retrouvé dans la littérature. Nous avons ensuite présenté les différents modèles du système de décision traités dans notre travail. Ces modèles sont le modèle par choix de variation du signal et le modèle par choix valeur du signal. Le principe du premier modèle est de choisir les valeurs futures du signal parmi un ensemble d'évolutions futures possibles du signal, la seconde choisit seulement la valeur courante du signal. Nous avons ensuite précisé qu'il existe deux approches d'apprentissage par renforcement, la première se basant sur les états et la seconde n'ayant pas d'états. Nous avons proposé et justifié l'idée que le modèle par variation du signal sera formulé avec l'apprentissage par renforcement sans état et que le modèle par valeur du signal sera formulé par l'apprentissage par renforcement à base d'état. Nous avons argumenté pourquoi ces choix nous semblaient être les plus pertinents compte tenu de la nature de l'un et de l'autre des deux modèles du système proposés. Ensuite, nous avons expliqué les concepts et terminologies utilisés pour décrire la situation étudiée dans notre travail. Finalement, nous avons formulé un modèle de l'utilisateur en nous basant sur les hypothèses émises à son sujet et qui servira d'environnement d'expérimentation pour nos simulations. Dans le chapitre suivant, nous présentons en détail le paradigme par choix de variation du signal.

# Chapitre 4

## Modèle par politique de variation du signal

Comme précisé auparavant, notre objectif est de contrôler le signal en tenant compte de la consommation d'énergie et du confort de l'utilisateur. Dans cette partie, nous présentons une méthode qui devra baisser la valeur du signal afin d'arriver à la valeur idéale pour l'utilisateur tout en réduisant la consommation énergétique. Par conséquent, la conception la plus naïve pour le *système de contrôle* serait de considérer l'action comme le couple formé par la pente choisie pour diminuer la valeur du signal et la valeur à laquelle le signal se stabilise. Et étant donné la stochasticité des réactions de l'utilisateur (supposée dans notre modèle, car le confort de l'utilisateur est subjectif et indéterminable), le système aura besoin de faire plusieurs tirages de chaque action afin d'évaluer précisément l'impact des actions sur l'utilisateur. La taille de l'espace d'action est donc cruciale afin déterminer rapidement la meilleure stratégie de contrôle. La conception naïve du *système de contrôle* - qui consiste à concevoir un agent unique choisissant une stratégie parmi un ensemble de stratégies - pourrait conduire à des temps de convergence démesurés puisque le nombre d'actions pour l'agent serait le produit entre le nombre de pentes et le nombre de valeurs d'arrêt. Afin d'éviter ce phénomène d'explosion combinatoire des politiques, nous proposons une solution alternative qui consiste à considérer que le système de contrôle du signal est composé de deux agents : un agent qui choisit la pente pour diminuer le signal et un autre agent qui choisit la valeur d'arrêt du signal. Dans le cas où nous avons deux agents distincts coopérant pour contrôler le signal, le temps de convergence de l'algorithme dépendrait au pire cas du double du nombre maximum d'actions des deux agents (le double, car le choix de la valeur d'arrêt vient toujours après le choix de la pente). Le système de contrôle diminue le signal en fonction de la pente choisie par le système des pentes jusqu'à ce que la valeur choisie par le système des valeurs d'arrêt soit atteinte. Si l'utilisateur intervient (ou si la valeur minimale est atteinte), l'action est arrêtée puis le système de contrôle procède au choix d'une nouvelle stratégie de contrôle. Le choix de cette conception se justifie par le fait que le système à pente seule prend déjà énormément de temps pour converger comme présenté dans la section 4.2.

Dans cette partie, nous définissons le système de contrôle conçu avec le paradigme de variation du signal (voir section 3.2). Ensuite, nous exposerons une comparaison entre les différents algorithmes d'apprentissage par renforcement sans états que nous avons testés ainsi que les techniques d'amélioration que nous avons proposées.

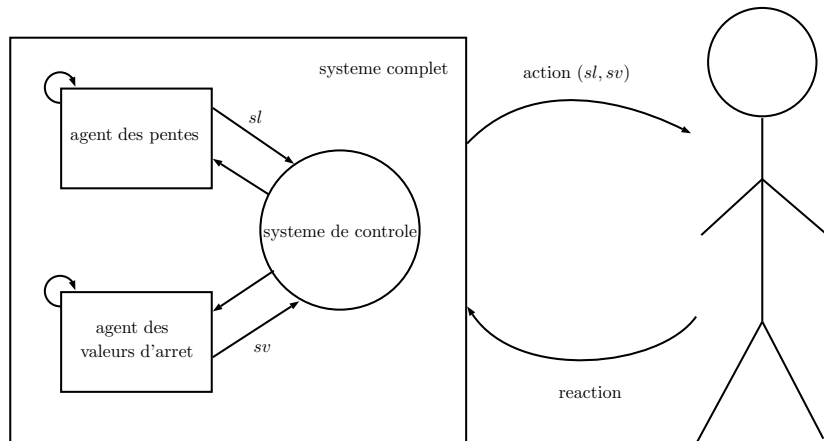


Figure 4.1 – Schémas montrant le système de contrôle.

## 4.1 Le système de contrôle

Dans ce qui suit, nous allons maintenant montrer comment utiliser l'apprentissage par renforcement sans états pour contrôler le signal, d'une part en termes de vitesse de diminution du signal (pentes), d'autre part en termes de valeur d'arrêt. Comme indiqué précédemment (voir le début du chapitre 4), le système de contrôle est composé de deux agents d'apprentissage par renforcement : l'un qui choisit la pente pour diminuer le signal et l'autre qui choisit la valeur du signal sur laquelle le signal est stabilisé. L'objectif du système est d'apprendre à la fois la pente et la valeur d'arrêt les plus économes en énergie, vérifiant les contraintes imposées à la satisfaction de l'utilisateur.

Dans cette section, nous décrivons formellement notre système, les algorithmes d'apprentissage utilisés pour notre travail et l'objectif utilisé pour l'apprentissage. L'agent des pentes et l'agent des valeurs d'arrêt seront décrits dans des sections séparées, en commençant par l'agent des pentes. Notre travail est particulier dans la mesure où le système de contrôle est composé de deux agents qui apprennent indépendamment et simultanément de l'environnement. Le système de contrôle composé uniquement de l'agent des pentes représente un système plus simple et sert de modèle pour faire ressortir la particularité de notre travail. Nous décrivons ensuite le système dans sa globalité en précisant les détails le composant.

### 4.1.1 Formalisme

Dans cette sous-section, nous décrivons conceptuellement les éléments composant le système de contrôle. Nous nous reposerons sur ces éléments afin de présenter le système de contrôle.

Rappelons que le système baisse le signal suivant une vitesse qu'il choisit préalablement, puis le stabilise à la valeur d'arrêt. La contrainte qui émerge de cela est que la valeur d'arrêt doit être inférieure à la valeur actuelle. Notons alors la valeur actuelle  $v$ . Nous notons  $SL \subset \mathbb{R}_*^-$  l'ensemble des pentes et  $SV(v) \subset \mathbb{R}^+$  l'ensemble des valeurs d'arrêt disponibles pour les agents (respectivement l'agent des pentes et l'agent des valeurs d'arrêt). Nous précisons que les ensembles  $SL$  et  $SV(v)$  sont finis, et nous notons l'ensemble contenant toutes les valeurs d'arrêt possible par  $SV = \cup_{s_{min} \leq v \leq s_{max}} SV(v)$ .

L'ensemble des actions (ou stratégies)  $AC$  disponibles pour le *système de contrôle* est donc formellement défini comme  $AC = \{(sl, sv) \in SL \times SV(v) \mid sl_{max} \leq sl \leq sl_{min}, sv_{min} \leq sv < v\}$  où  $sl_{min}$  et  $sl_{max}$  sont respectivement les pentes les plus douces et les plus raides disponibles pour l'agent des pentes et  $sv_{min}$  et  $sv_{max}$  sont respectivement les valeurs minimales et maximales que l'agent des valeurs d'arrêt peut choisir. La durée maximale d'une action notée  $max\_dur$  est fixée avant le début du contrôle. Lors de l'application d'une action, si l'utilisateur intervient ou s'il n'intervient pas pendant  $max\_dur$  étapes, l'action choisie par l'agent est arrêtée et l'agent met à jour sa politique.

Nous proposons plusieurs façons de fixer la durée maximale des actions. La première façon est évidemment d'imposer une durée maximale qui n'est bornée par aucune valeur. Dans ce cas, le système n'a pas son mot à dire sur la durée d'une action et seules les interventions de l'utilisateur comptent. La deuxième façon est d'imposer une durée maximale fixe. Dans ce cas, la durée maximale de toutes les actions est la même et elle ne change pas avec le temps (nous l'appelons la *durée maximale statique*). La troisième façon est de changer la durée maximale au cours du temps. Nous appelons cette façon de fixer la durée maximale des actions *durée maximale dynamique*. Dans ce cas, la durée maximale est la même pour toutes les actions, mais change en fonction du comportement passé de l'utilisateur. L'idée est que l'action courante peut être désactivée pour en choisir une autre si l'utilisateur prend beaucoup plus de temps que d'habitude à intervenir. Nous proposons deux règles de mise à jour de la *durée maximale dynamique*. Dans le paragraphe suivant, nous expliquons ces deux règles après avoir défini le concept de *temps d'intervention d'une action*.

Nous considérons que le temps d'intervention d'une action est le temps entre l'étape où l'action est choisie et l'étape où l'utilisateur intervient. Dans ce que nous proposons, les règles de mise à jour de la *durée maximale dynamique* dépendent à la fois de la moyenne et de l'écart-type du temps d'intervention. Dans ce cas, l'action à prendre pourrait être choisie à nouveau si la durée de l'action courante dépasse la durée moyenne de l'intervention de l'utilisateur d'un facteur de l'écart-type du temps d'intervention ( $temps\ de\ l'action\ courante = temps\ d'intervention\ moyen + const \times \text{écart-type du temps d'intervention}$ ). La première règle de mise à jour de la *durée maximale dynamique* consiste à la définir comme étant la somme de la moyenne du temps d'intervention et d'un facteur de l'écart type du temps d'intervention (nous la nommons *update\_mean\_var\_interv*). La deuxième règle de mise à jour est la même que la première règle, mais en supposant que la loi de probabilité du temps d'intervention de l'utilisateur suit une loi de probabilité exponentielle (même si l'hypothèse sans mémoire de la distribution exponentielle ne tient pas dans le contexte d'un utilisateur non stationnaire). La dernière règle de mise à jour donne un moyen d'approcher l'écart-type et évite de devoir mémoriser les temps d'intervention passés, ce qui pourrait, au moins, réduire les temps de calcul. Pour cette règle, l'écart-type est donc la racine carrée de la moyenne, et nous l'appelons *update\_mean\_interv*.

Sachant cela, le contrôle se déroule comme suit : le système de contrôle interroge l'agent des pentes et l'agent des valeurs d'arrêt pour recevoir une pente (notée  $sl$ ) et une valeur d'arrêt (notée  $sv$ ) qui forment la stratégie de contrôle (la paire  $(sl, sv)$ ) à appliquer à l'environnement. Le système de contrôle procède alors à l'application de la stratégie de contrôle retournée, pas à pas, en baissant le signal en fonction de la pente choisie jusqu'à atteindre la valeur d'arrêt choisie précédemment. Si l'utilisateur intervient, si la valeur minimale du signal est atteinte, ou si la durée maximale des actions

est atteinte, l'action choisie par l'agent est arrêtée et les agents mettent à jour leur politique afin de choisir plus souvent de bonnes stratégies de contrôle. Éventuellement, l'agent met à jour la durée  $max\_dur$  suivant la règle de mise à jour qui est adoptée. Nous proposons de récompenser les deux agents avec la même récompense dès que l'utilisateur intervient sur la valeur du signal ou que la durée limite est atteinte. Un exemple sur le fonctionnement du système par choix de variation du signal est montré dans la figure 4.2.

L'agent des pentes et l'agent des valeurs d'arrêt apprennent avec l'apprentissage par renforcement sans états. Or, ces deux agents n'auront pas nécessairement le même algorithme d'apprentissage par renforcement. Par conséquent, nous considérons, que tout agent apprenant avec l'apprentissage par renforcement (que nous nommons *agent d'apprentissage par renforcement*) a les trois éléments suivants qu'on retrouve dans l'apprentissage par renforcement sans états :

- L'action dénotée  $a$  faisant partie d'un ensemble prédéfini d'actions  $A$  (que ce soit pour l'agent des pentes où l'agent des valeurs d'arrêt).
- La récompense  $r$  qui est un nombre réel dans l'intervalle  $[0, 1]$ .
- La politique  $pol$  qui est une structure de donnée contenant la politique de l'agent. Cette structure de donnée représente soit un vecteur stochastique soit des estimations sur les valeurs des actions ou alors une combinaison des deux.

L'*agent d'apprentissage par renforcement* choisit une action, il met à jour sa politique et applique l'action au niveau de l'environnement. Ces propriétés correspondent aux fonctions et procédures suivantes que nous définissons :

- La fonction  $Choose(pol)$  retourne l'action que choisit l'agent à l'aide de la politique. Elle a en entrée la politique  $pol$  et retourne une action  $a \in A$ .
- La procédure  $Update(pol, a, r)$  met à jour la politique  $pol$  en se reposant sur la récompense  $r$  obtenue précédemment et l'action  $a$  associée à la récompense.

Avant d'aborder les aspects formels du système de contrôle par choix de variation du signal, nous commentons dans le paragraphe qui suit la figure 4.2 qui montre un exemple de son fonctionnement. Dans l'exemple de la figure 4.2, nous considérons que la valeur maximale du signal est de 100 et que la valeur minimale du signal est de 0. Le système a le choix entre deux pentes : baisser la valeur du signal de 10 unités par étape ou alors baisser la valeur du signal de 15 unités par étape. Les valeurs d'arrêt possibles sont 40 et 50. L'exemple montre l'application répétée de deux stratégies, une représentée en orange et une autre représentée en bleu. La stratégie représentée en orange consiste à faire baisser la valeur du signal de 15 unités par étape jusqu'à la valeur 40. Et la stratégie représentée en bleu consiste à faire baisser la valeur du signal de 10 unités par étape jusqu'à la valeur 50. Les cercles rouges représentent les étapes pendant lesquelles l'utilisateur réagit (en remettant la lumière au maximum). Comme la stratégie représentée en orange est plus agressive pour l'utilisateur que celle représentée en bleu (la pente de la première est plus raide et sa valeur d'arrêt est plus basse), l'utilisateur intervient plus tôt pour celle-ci.

La table 4.1 résume les notations qui seront utilisées tout au long de ce chapitre. Nous présentons dans le reste de ce qui suit, l'agent des pentes et l'agent des valeurs d'arrêt.



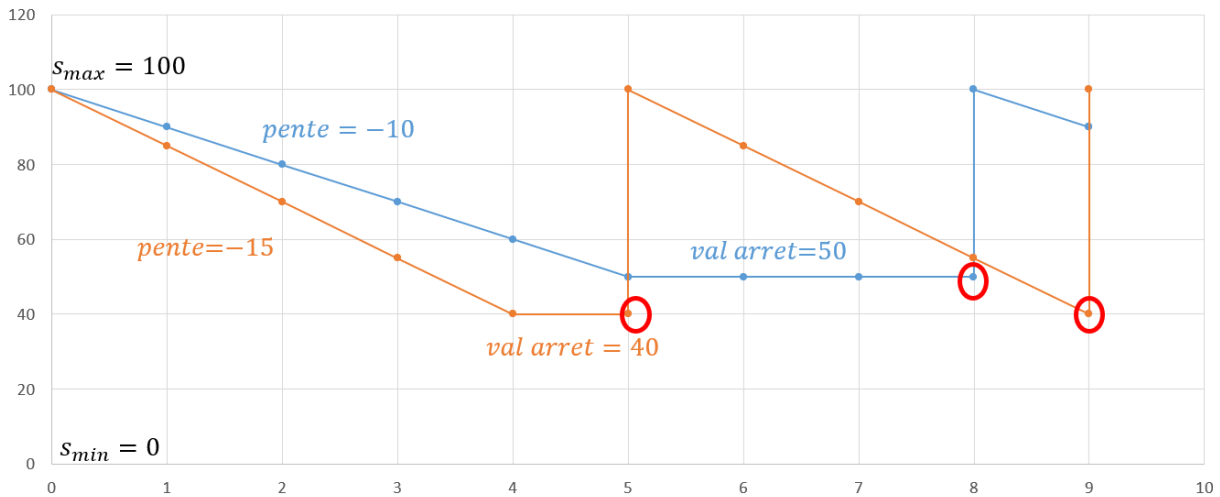


Figure 4.2 – Exemple de l'application du système de contrôle par choix de variation du signal

$SL$	L'ensemble des pentes.
$SV(v)$	L'ensemble des valeurs d'arrêt que l'agent peut prendre si la valeur actuelle est $v$ .
$SV$	L'ensemble de toutes les valeurs d'arrêt possibles.
$AC$	L'ensemble d'actions accessibles au système de contrôle (les couples de pentes et de valeurs d'arrêt).
$sl_{min}$	La pente la plus douce.
$sl_{max}$	La pente la plus raide.
$sv_{min}$	La valeur minimale du signal.
$sv_{max}$	La valeur maximale du signal.
$max\_dur$	La durée maximale d'une action.
$update\_mean\_var\_interv$	La durée maximale d'une action mise à jour avec la durée moyenne de réaction de l'utilisateur et son écart type.
$update\_mean\_interv$	La durée maximale d'une action mise à jour en partant du principe que le temps de réaction de l'utilisateur suit une loi exponentielle.
$Choose\_pol$	La fonction retournant l'action choisie par l'agent (des pentes ou des valeurs d'arrêt).
$Update$	La fonction permettant de mettre à jour la politique de l'agent.
$p_k$	Le vecteur stochastique associé à l'agent des pentes (après application de $k$ actions).
$ev_k$	Le vecteur des estimations des actions de l'agent des pentes (après application de $k$ actions).
$q_k^*$	Le vecteur stochastique associé à l'agent des valeurs d'arrêt et la valeur actuelle $v$ (après application de $k$ actions).
$ew_k$	Le vecteur des estimations des actions de l'agent des valeurs d'arrêt (après application de $k$ actions).
$ren_{n,n'}$	La récompense de l'action associée à l'énergie (appliquée entre l'étape $n$ et $n'$ ).
$r_{cs_{n,n'}}$	La récompense de l'action associée au confort (appliquée entre l'étape $n$ et $n'$ ).
$rn, n'$	La récompense complète de l'action (appliquée entre l'étape $n$ et $n'$ ).
$FC$	Le rapport entre le nombre de fois où l'algorithme converge vers la meilleure action et le nombre total d'exécutions.
$FCVO$	Le rapport entre le nombre de fois où l'algorithme converge dans le voisinage de l'action optimale et le nombre d'exécutions total.
$FMVO$	La fréquence de choix des actions dans le voisinage de la meilleure action (sans qu'il ait convergence vers les actions du voisinage).
$TMC$	Le temps moyen de convergence sur l'ensemble des exécutions d'un algorithme.

Table 4.1 – Liste des principales notations concernant les algorithmes et notions du modèle par choix de variation du signal.

## 4.1.2 L'agent des pentes

Dans ce qui suit, nous détaillons le comportement de l'agent des pentes.

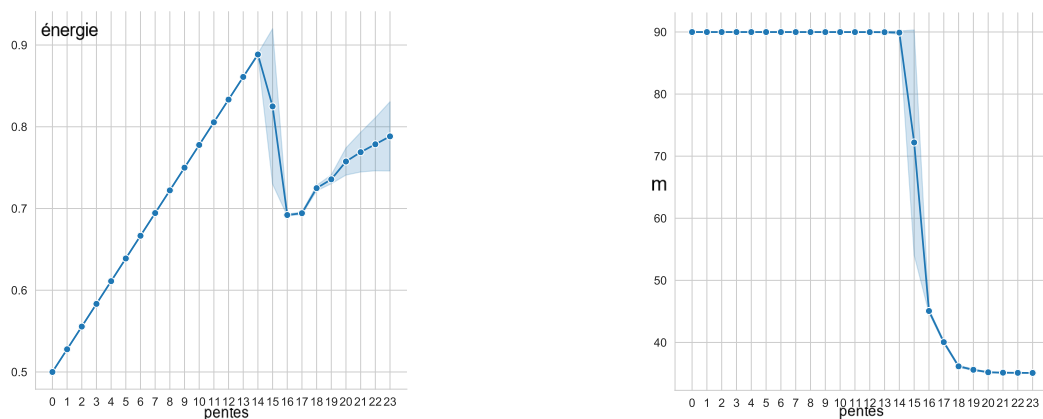
### Comportement énergétique de l'agent des pentes

Afin de contextualiser notre approche, nous allons considérer que l'agent des valeurs d'arrêt choisit toujours comme valeur d'arrêt la valeur minimale. L'objectif général de notre travail est d'apprendre des politiques de contrôle optimales étant donné une dynamique environnementale qui change en fonction des politiques passées. Le but est d'apprendre une situation qui prend en compte l'énergie et le confort afin d'atteindre une situation d'équilibre entre les deux.

Dans cette sous-section, nous allons montrer comment le fait de jouer différentes actions influence le comportement énergétique du système de contrôle. Nous montrons comment ces ten-

dances caractérisent un objectif énergétique que le système doit optimiser. Dans l'expérience que nous considérons dans cette partie, nous supposons que les actions sont représentées par des *pent*es qui sont des nombres négatifs (c'est-à-dire que les valeurs d'arrêt de toutes les actions correspondent à la valeur minimale). Ainsi, jouer une action représentée par une pente signifie que le système diminue le signal dans le temps selon la pente qu'il choisit jusqu'à atteindre la valeur minimale  $s_{min}$  ou jusqu'à ce que l'utilisateur intervienne.

Afin d'évaluer comment le système pourrait réagir au changement de pente, nous faisons une expérience dans laquelle nous jouons un ensemble de pentes et calculons les performances énergétiques et le paramètre usager (le paramètre  $m$ ) de chaque pente. Nous laissons le système appliquer chaque pente un certain nombre de fois, et calculons pour chaque pente l'énergie moyenne et la valeur moyenne de  $m$  obtenues sur plusieurs exécutions. Pour chaque exécution, l'énergie est calculée comme étant l'aire sous le signal délimité par l'axe temporel, le début de l'exécution et sa fin, divisée par sa durée. La figure 4.3 représente les résultats obtenus à partir de cette expérience avec les paramètres  $m_0 = 35$ ,  $a_0 = 0.1$ ,  $pe = 0.35$  (voir section 3.4) pour l'utilisateur. Nous calculons l'énergie moyenne et la valeur moyenne de  $m$  de chaque pente pour 30 exécutions. Précisons que ce calcul est irréaliste dans la pratique et sert essentiellement à comprendre le comportement du système et de l'utilisateur. Chaque exécution est composée de 10000 étapes. Les pentes sont triées de gauche à droite, de la plus raide à la moins raide respectivement. Ces pentes sont -90, -85, -80, -75, -70, -65, -60, -55, -50, -45, -40, -35, -30, -25, -20, -15, -10, -5, -1, -1/2, -1/7, -1/12, -1/17, -1/22 par étape. Les valeurs minimales et maximales du signal sont 0 et 90.<sup>1</sup>



(a) Énergie moyenne des pentes.

(b) Valeur moyenne de  $m$  pour les pentes.

Figure 4.3 – Performances des pentes

Nous remarquons dans les figures 4.3a et 4.3b que les valeurs données par les courbes sont des valeurs stationnaires de l'énergie et du paramètre  $m$ . Tout d'abord, nous voyons qu'il y a deux parties dans le profil. La première partie (linéaire) correspond aux pentes pour lesquelles l'utilisateur intervient (ou finit par intervenir) à chaque étape sur le système (les pentes pour lesquelles  $m$  prend sa valeur maximale). Aucune des pentes de cette partie ne doit être prise comme objectif, car l'utilisateur ne cesse d'intervenir pour ces pentes, ce qui implique une situation d'inconfort totalement inacceptable. La seconde partie (convexe) correspond aux pentes pour lesquelles l'utilisateur n'intervient pas à chaque étape (c'est-à-dire les pentes pour lesquelles le seuil d'intervention  $m$  n'a que des valeurs modérées).

1. À partir de maintenant, nous ferons référence aux pentes par leurs indices allant de 0 à 23.

Nous l'appelons donc la plage "acceptable" des pentes que le système pourrait jouer. Dans cette plage, nous observons que le fait de jouer des pentes raides amène l'utilisateur à intervenir trop souvent sur le signal, ce qui a un impact négatif sur les performances énergétiques. À l'inverse, si le système choisit des pentes trop douces, il consomme plus d'énergie que nécessaire, car le signal a plus souvent des valeurs élevées. Par conséquent, la courbe induit une pente énergétiquement optimale (dans la partie convexe) qui n'est ni trop raide pour l'utilisateur ni trop douce pour le système. Cette pente représente l'équilibre parfait entre performances énergétiques et confort de l'utilisateur. L'existence d'un tel optimum suggère que l'optimisation des performances énergétiques sur un système dans lequel l'utilisateur a la possibilité d'intervenir pourrait suffire à fournir des conditions suffisamment confortables à l'utilisateur.

Pour définir correctement notre problème, nous devons connaître la formulation adéquate de l'objectif que l'algorithme de contrôle doit optimiser. Cet objectif *doit* d'une manière ou d'une autre prendre en compte le confort de l'utilisateur (implicitement ou explicitement). Ce point est problématique, car il pourrait impliquer la nécessité d'avoir des connaissances préalables sur le comportement de l'utilisateur (une telle hypothèse est exclue, voir section 1.2) afin de définir correctement l'objectif d'optimisation. La courbe précédente pourrait indiquer qu'il existe une politique optimale définie *indépendamment* des connaissances des réactions de l'utilisateur parce que nous avons considéré une pente objective *purement* énergétique sur un *sous-ensemble* de pentes.<sup>2</sup>

L'objectif de l'agent des pentes peut être formulé comme suit. Nous disposons d'un ensemble de politiques dont certaines sont acceptables et d'autres inacceptables (comme défini précédemment). Nous ne savons pas quelle politique est acceptable et quelle politique ne l'est pas. Le problème est de concevoir un algorithme d'apprentissage sans modèle qui peut trouver la meilleure politique de contrôle dans le sous-ensemble des politiques acceptables.

## Représentation de la politique de l'agent des pentes

Suivant l'algorithme d'apprentissage par renforcement qui est utilisé, la politique de l'agent des pentes est représentée explicitement à l'aide d'un vecteur stochastique ou implicitement avec les estimations des valeurs des actions (voire avec les vecteurs stochastiques et les estimations). La politique de l'agent est mise à jour à chaque fois que l'action se termine (et donc à chaque fois que l'utilisateur intervient ou que la durée maximale des actions  $max\_dur$  est atteinte).

Notons par  $k$  le nombre de fois où une pente est appliquée par le système de contrôle depuis le début de l'apprentissage. Dans le cas où un vecteur stochastique est utilisé pour représenter la politique de l'agent des pentes, cette politique est notée  $p_k \in [0, 1]^{|SL|}$  où  $p_k[i]$  est la probabilité de choisir la  $i$ -ième pente la moins raide de  $SL$  (voire section 4.1.1). Si les estimations des récompenses des actions sont utilisées par l'agent des pentes, la politique est représentée par un vecteur contenant les estimations des actions qui est noté  $ev_k \in [0, 1]^{|SL|}$  où  $ev_k[i]$  est l'estimation de la  $i$ -ième pente la moins raide de  $SL$ .

---

2. Cette conclusion est valable pour différentes valeurs de la mémoire de l'utilisateur en incluant les valeurs d'arrêt comme le montre l'annexe B.

## La fonction de récompense

La récompense d'une action appliquée par l'agent des pentes est décomposée en deux parties : une partie reflétant la performance énergétique et une partie reflétant la contrainte que nous avons identifiée sur les pentes afin qu'elles puissent être considérées comme "acceptables". La partie énergétique d'une stratégie est calculée comme étant le rapport entre l'économie d'énergie moyenne de la stratégie actuelle et l'économie d'énergie maximale.

Mathématiquement, pour une stratégie  $sl$  (une pente) commençant à l'étape  $n$  et se terminant à l'étape  $n'$ , la partie de la récompense reflétant la consommation d'énergie est définie comme suit :

$$ren_{n,n'} = \frac{E_{max} - E_{n,n'}}{E_{max}} \quad (4.1)$$

Nous avons  $E_{n,n'}(sl) = \sum_{i=n+1}^{n'} (v_{i-1} + v_i) / 2 \cdot (n' - n)$  est l'énergie moyenne de l'action choisie  $sl$  pendant son exécution et que  $E_{max} = s_{max} - s_{min}$  est l'énergie maximale qui peut être achevée. La partie de la récompense reflétant les contraintes sur les pentes est définie comme suit :

$$rcs_{n,n'} = \frac{n' - n - 1}{n' - n} \quad (4.2)$$

Après avoir défini les récompenses partielles, nous définissons la récompense totale comme étant la fonction

$$r_{n,n'} = ren_{n,n'} \cdot rcs_{n,n'}^\gamma \quad (4.3)$$

où  $\gamma$  est un nombre réel strictement positif. La formule signifie que le bénéfice d'une stratégie en termes d'énergie est diminué par le terme dépendant de l'intervention de l'utilisateur. Le paramètre  $\gamma$  représente l'intensité avec laquelle nous récompensons le système de contrôle pour avoir satisfait l'utilisateur. Le choix de la valeur du paramètre  $\gamma$  a été fait après quelques expérimentations.

Dans ce qui suit, nous argumentons de manière détaillée le choix de la récompense suivant le même protocole expérimental que pour la figure 4.3. Cela concerne le cas de l'utilisateur ayant une mémoire à moyen terme avec  $m_0 = 35$  et  $a_0 = 0.1$ . La valeur de  $\gamma$  vaut 1.0. La figure 4.4 montre la récompense de chaque pente prise individuellement après l'avoir exécutée plusieurs fois et calculer la moyenne des moyennes de la récompense de chaque exécution (comme nous l'avons fait dans l'expérience précédente).

Dans la figure 4.4, nous voyons bien que la meilleure pente (en termes de récompense) est très proche de la pente correspondant à l'équilibre entre énergie et confort de l'utilisateur. En effet, la pente énergétiquement optimale est la pente d'indice 16 (dans la région creuse), tel que le montre la figure 4.3a, et la pente ayant la meilleure récompense est la pente 17 tel que montré dans la figure 4.4. Nous constatons aussi que la valeur de la récompense est nulle pour les pentes de la région linéaire. D'ailleurs, nous voyons bien que la partie confort  $rcs_{n,n'}$  de la récompense (voir la formule 4.3), vaut zéro lorsque l'utilisateur intervient à chaque étape pour remonter la valeur du signal, c'est la raison pour laquelle la récompense est nulle pour ces pentes-là. Pour les autres pentes, la région concave dans la courbe des récompenses des pentes correspond très bien à la région convexe de la courbe

de l'énergie des pentes (voir les figures 4.4 et 4.3). Nous pouvons alors en conclure que la fonction de récompense que nous avons proposée traduit parfaitement les tendances essentielles concernant l'équilibre à trouver entre confort et énergie caractérisant notre problématique de thèse.

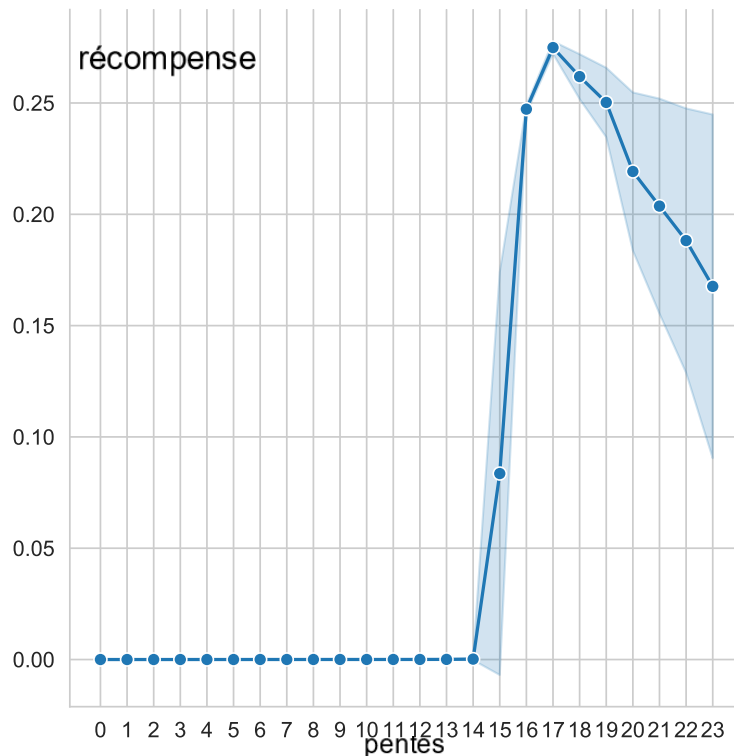


Figure 4.4 – Récompenses  $r_{n,n'}$  des pentes.

### 4.1.3 L'agent des valeurs d'arrêt

De même que pour l'agent des pentes, la politique de l'agent des valeurs d'arrêt est représentée soit par un vecteur stochastique, soit par les estimations des valeurs des actions et elle est mise à jour à chaque fois que l'action se termine.

Supposons que  $k$  actions ont été choisies par l'agent des valeurs d'arrêt depuis le début de l'apprentissage. Supposons que la valeur courante est  $v$ . Si la politique de l'agent des valeurs d'arrêt est représentée par un vecteur stochastique, celui-ci est noté  $q_k^v \in [0, 1]^{|SV(v)|}$  où  $q_k^v[i]$  est la probabilité de choisir la  $i$ -ième valeur la moins grande de  $SV(v)$ . Pareillement, si la méthode utilisée est par valeur, alors les estimations des récompenses des actions sont représentées par le vecteur  $ew_k \in [0, 1]^{|SV|}$  où  $ew_k[i]$  est l'estimation de la récompense associée à la  $i$ -ième valeur la moins intense de  $SV$ .

La récompense est partagée entre l'agent des pentes et l'agent des valeurs d'arrêt. Le calcul de l'énergie moyenne et de la fréquence d'intervention dans la récompense porte sur l'ensemble de la stratégie choisie par le système de contrôle (la pente suivie de la valeur d'arrêt) et ce dans les deux agents. L'agent des pentes et l'agent des valeurs d'arrêt sont mis à jour en utilisant la récompense partagée.

Comme nous l'avons dit dans la partie de l'agent des pentes, la fonction de récompense est censée inciter l'agent à choisir les bonnes actions en termes de consommation énergétique et de confort de l'utilisateur, le système visant à équilibrer proprement ces deux éléments. Étant donné la nécessité d'inclure ces deux éléments dans la fonction de la récompense, il est naturel de considérer la récompense de l'agent des pentes pour l'agent des valeurs d'arrêt. Le point qui devra être traité est de savoir à quel point cette fonction de récompense reflète l'objectif d'équilibre. Nous détaillerons ce point dans la prochaine sous-section où nous abordons le système complet.

#### 4.1.4 Le système complet

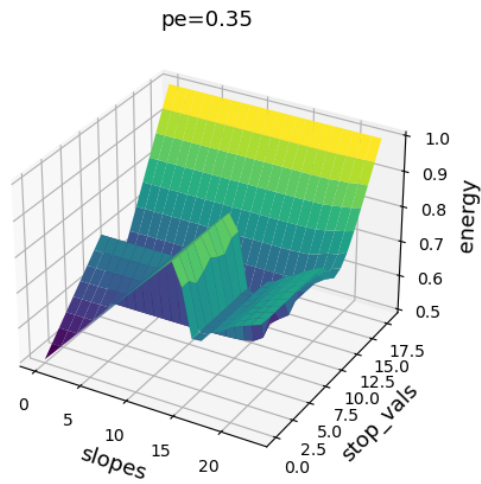
Dans cette sous-section, nous formulons le système complet. Nous commençons par faire le lien entre l'objectif du système des pentes et l'objectif du système complet, puis nous montrons comment nous formulons l'apprentissage pour le système complet.

##### Profil énergétique et fonction de récompense

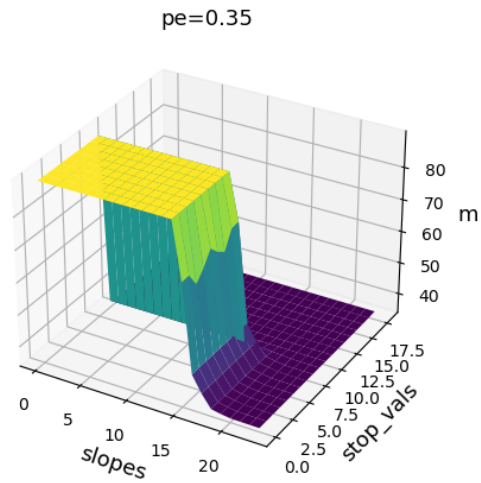
Nous effectuons et montrons dans cette sous-section les résultats de la même expérience que celle présentée dans les sous-sections 4.1.2 et 4.1.2. Nous montrons à travers ces simulations les performances des stratégies  $(sl, sv)$  disponibles au système et comment les conclusions tirées des sous-sections 4.1.2 et 4.1.2 concernant le système à pentes tiennent même pour le système complet et viennent confirmer nos analyses ayant trait du lien entre le système et l'utilisateur.

L'expérience est faite pareillement que celle illustrée par la figure 4.4, nous exécutons chaque stratégie (couple de pente et de valeur d'arrêt) de manière répétée pendant 10000 étapes sur 30 exécutions. Durant cette simulation, nous calculons l'énergie moyenne, la récompense moyenne ainsi que la valeur moyenne de  $m$  sur toutes les simulations de chaque stratégie. Le résultat est montré dans les courbes de la figure 4.5. L'expérience montrée ici correspond à l'utilisateur dont la mémoire est à moyen terme (avec  $pe = 0.35$ ). Dans l'ensemble des stratégies considérées, l'écart entre deux pentes successives est de cinq, et l'écart entre deux valeurs d'arrêt successives est de cinq aussi. Dans les deux paragraphes qui suivent, nous commençons par analyser, dans l'ensemble, les courbes de la figure 4.5.

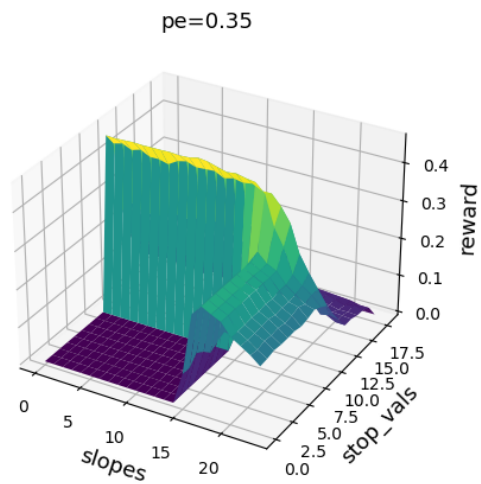
Comme nous l'avons fait pour l'agent des pentes dans la figure 4.3, nous identifions, dans ce qui suit, les différentes régions (associées aux stratégies) en mettant en relief le lien entre le comportement de l'utilisateur et le profil énergétique du système pour les stratégies de ces régions. Tout comme pour le système à pentes, l'utilisateur a tendance à réagir tout le temps pour les stratégies avec lesquelles la pente est trop raide et où la valeur d'arrêt est trop basse. La région où se trouvent ces stratégies (qui sont inacceptables pour l'utilisateur) est nommée la *région inacceptable*. La région inacceptable correspond aux stratégies pour lesquelles le paramètre  $m$  (le seuil d'intervention) a une valeur maximale (voir les figures 4.5a et 4.5b). Si nous excluons les stratégies de la région inacceptable, nous remarquons, en dehors de cette région, l'existence d'un ensemble de stratégies pour lesquelles la valeur de l'énergie est minimale (nous nommons cette région la *région optimale*). Ces stratégies forment la région d'équilibre entre énergie et confort de l'utilisateur : si nous prenons des pentes plus



(a) Énergies moyennes des stratégies du système



(b) Valeurs moyennes de  $m$  des stratégies du système



(c) Moyennes des récompenses des stratégies du système

Figure 4.5 – Performances des stratégies du système (couple de pente et de valeur d'arrêt)

raides (ou des valeurs d'arrêt plus basses) que celles correspondant aux stratégies de la région optimale, l'énergie consommée est dans ce cas plus conséquente. La raison expliquant ce phénomène est que le système consomme plus d'énergie que nécessaire quand les stratégies sont beaucoup trop agressives pour l'utilisateur, car elles l'incitent à intervenir plus fréquemment sur la valeur du signal. D'autre part, même quand les stratégies sont trop douces (quand la pente choisie est douce et/ou la valeur d'arrêt est trop élevée), l'énergie consommée est plus grande qu'il n'en faut pour satisfaire l'utilisateur (car le signal a des valeurs élevées plus longtemps que pour les meilleures stratégies). Le système devra être capable d'apprendre l'équilibre entre ces deux extrêmes qui conduisent inévitablement à un excès dans l'usage énergétique.

Par ailleurs, nous remarquons, qu'au-delà d'une certaine valeur d'arrêt (la valeur d'indice 10), la majorité des stratégies ont les mêmes performances énergétiques quelle que soit la pente qui est prise. La valeur d'arrêt semble avoir plus d'influence sur les performances énergétiques du système que le choix de la pente. Ce point semble avoir beaucoup de sens, car quand la valeur d'arrêt est trop grande, les variations du signal ne peuvent pas être gênantes pour l'utilisateur vu qu'elles auront forcément de faibles écarts entre elles et qu'elles (les pentes) porteront sur des valeurs qui sont déjà assez élevées pour satisfaire l'utilisateur. Par conséquent, si le système opte pour une valeur d'arrêt élevée, l'utilisateur interviendra forcément moins durant la baisse du signal. Globalement, nous pouvons dire que c'est principalement la valeur d'arrêt qui a le plus d'influence sur le comportement de l'utilisateur et donc sur la consommation énergétique du système. Dans les deux paragraphes qui suivent, nous analysons les résultats de la façon qui suit : nous choisissons une valeur d'arrêt, nous examinons le profil de pentes obtenu par l'intersection entre le plan correspondant à cette valeur d'arrêt et la courbe de la figure 4.5a. Nous faisons de même pour les pentes, nous fixons la pente, nous prenons l'intersection entre le plan correspondant à cette pente et la courbe de la figure 4.5a. Cette façon de faire nous permet d'étudier le profil énergétique et l'état de l'utilisateur en fixant la valeur d'arrêt (respectivement la pente) et en faisant varier la pente (respectivement la valeur d'arrêt).

Quand la valeur d'arrêt est fixée à des valeurs relativement faibles, nous retrouvons approximativement le profil linéaire suivi de la partie creuse que l'on a eu dans le cas où l'on a uniquement l'agent des pentes (voir la figure 4.3). Si le signal est baissé avec une même pente, faire varier la valeur d'arrêt semble aussi donner approximativement le même profil que dans le cas où l'on a uniquement l'agent des pentes. Néanmoins, là encore, nous observons certains segments pour lesquels l'énergie semble être stable. De plus, nous constatons que plus la pente est douce, plus grand est le segment de valeurs d'arrêt pour lesquelles l'énergie est stable. Pour chaque pente fixée, ces valeurs d'arrêt (celles ayant la même énergie) font toujours partie des valeurs les plus faibles ; elles ne sont donc probablement jamais atteintes, car l'utilisateur intervient toujours avant ce moment. Ceci est la raison pour laquelle certaines valeurs d'arrêt ont toutes la même consommation énergie (quand la pente est fixée).

## **Apprentissage du système complet**

Comme nous l'avons dit plus tôt, le système de contrôle est composé de deux agents : l'agent des pentes et l'agent des valeurs d'arrêt. Les deux agents choisissent la stratégie de contrôle (la pente



et la valeur d'arrêt) et la renvoient au système de contrôle qui fait évoluer le signal conformément à cette stratégie. Comme nous l'avons dit précédemment, lorsqu'une action est appliquée à l'environnement, celui-ci renvoie une récompense au système de contrôle (qui est reçue par l'agent des pentes et l'agent des valeurs d'arrêt). Celle-ci est ensuite utilisée pour mettre à jour les politiques du système de contrôle. Le fonctionnement du système complet est détaillé dans l'algorithme 6.

Les agents des pentes et des valeurs d'arrêt sont modélisés par des *agents d'apprentissage par renforcement* (voire sous-section 4.1.1). Pour l'agent des pentes, la politique est notée  $pol\_sl$  et l'action est notée  $sl \in SL$ . Pour l'agent des valeurs d'arrêt, la politique est notée  $pol\_sv$  et l'action est notée  $sv \in SV$ . Notons que ce ne sont pas les agents qui appliquent les stratégies, mais bien le système de contrôle qui le fait. La récompense  $r$  de la stratégie (pente et valeur d'arrêt) sera retournée et utilisée pour la mise à jour des agents. Précisons aussi que nous n'intégrons pas l'usager dans l'algorithme 6, nous l'avons déjà détaillé dans le chapitre 3. Dans cette sous-section, nous nous limitons à détailler comment le système utilise l'apprentissage pour contrôler le signal.

---

**Algorithme 6** Algorithme de contrôle du système complet.

---

**Inputs :**  $pol\_sl$  - la politique de l'agent des pentes.

$pol\_sv$  - la politique de l'agent des valeurs d'arrêt.

```

1: while True do
2:   ▷ Choix de la stratégie
3:    $sl \leftarrow Choose(pol\_sl)$ 
4:    $sv \leftarrow Choose(pol\_sv)$ 
5:   ▷ Application de la stratégie
6:    $r \leftarrow$  La récompense due à l'application de la stratégie  $(sl, sv)$ .
7:   ▷ Mise à jour de la stratégie
8:    $Update(pol\_sl, sl, r)$ 
9:    $Update(pol\_sv, sv, r)$ 
10: end while

```

---

L'implémentation des fonctions  $Choose()$  et  $Update()$  dans l'agent des pentes et l'agent des valeurs d'arrêt varient selon l'algorithme d'apprentissage par renforcement qui est utilisé. Nous présentons dans ce qui suit seulement l'implémentation de  $Update()$  pour les algorithmes du LRI [119], LRP [119],  $LR_{R-\epsilon P}$  [119]. Les implémentations des fonctions des agents d'apprentissage par renforcement suivent un principe similaire pour tous les algorithmes d'apprentissage par renforcement.

Pour l'algorithme du LRI et ses variantes (LRP et  $LR_{R-\epsilon P}$ ) l'implémentation de la fonction  $Update()$  est donnée dans l'algorithme 7. Dans l'algorithme du LRI et ses variantes, la politique est représentée à l'aide d'un vecteur stochastique (comme indiqué dans la section 2.1). Nous supposons donc que  $pol$  est une structure de données où  $pol.vec$  réfère au vecteur stochastique et  $pol.size$  réfère à la taille de  $pol.vec$ . Les paramètres  $\alpha$  et  $\beta$  des variantes de LRI sont des constantes. Nous notons  $e_i$  le vecteur unitaire de valeur unitaire dans la composante d'indice  $i$ . Le vecteur  $[val]_0^{l-1}$

est le vecteur de taille  $l$  dont tous les éléments sont  $val$ .

---

**Algorithme 7** L'implémentation de la fonction  $Update()$  pour l'algorithme LRI et ces variantes (LRP et  $LR_{R-\epsilon P}$ ) et de ses variantes.

---

**Inputs :**  $pol\_sl$  - la politique de l'agent.

$a$  - l'action dont on veut mettre à jour la probabilité.

$r$  - la récompense obtenue par la dernière action.

1: **if**  $r > 0$  **then**

2:  $pol.vec \leftarrow pol.vec + \alpha(e_{i[a]} - pol.vec)$

3: **else**

4:  $pol.vec \leftarrow pol.vec + \beta(\lfloor \frac{\beta}{pol.size-1} \rfloor_0^{pol.size-1} - pol.vec)$

5: **end if**

---

L'algorithme  $Update()$  implémente LRI si  $\beta = 0$ , LRP si  $\alpha = \beta$  et  $LR_{R-\epsilon P}$  si  $\beta \ll \alpha$ . La fonction  $Choose(pol)$  ne fait que tirer aléatoirement une action par le vecteur stochastique  $pol.vec$ .

Supposons que le système de contrôle joue sa  $k$ 'ième stratégie  $(sl, sv)$  à l'étape  $n$  et que cette stratégie a pris fin à l'étape  $n'$  (quelle que soit la raison de l'arrêt). Afin de mettre à jour les politiques des agents (pente et valeurs d'arrêt) le système de contrôle fait un appel à la fonction  $Update(pol\_sl)$  et  $Update(pol\_sv)$  avec  $pol\_sl.vec = p_k$  et  $pol\_sv.vec = q_k$ . Dans le cas où les estimations des actions sont utilisées par les agents de pente et de valeurs d'arrêt (ex. pour l'algorithme de la poursuite), les politiques  $pol\_sl$  et  $pol\_sv$  doivent aussi prendre en compte les estimations  $ev_k$  et  $ew_k$  respectivement, en plus des vecteurs stochastiques.

Précisons finalement que dans les algorithmes 6 et 7, nous ne présentons pas certains détails de notre travail afin de ne pas encombrer l'exposé de nos méthodes. Nous n'avons pas inclus la mise à jour ainsi que les réactions de l'utilisateur. Nous n'avons pas non plus détaillé comment nous implémentons l'application de pentes inférieures à 1 (pour une pente  $p < 1$ , la valeur courante est décrétementée toutes les  $1/p$  étapes). Par ailleurs, nous n'avons pas inclus la gestion de la durée maximale des actions ainsi que les contraintes potentielles que cette gestion pourrait amener (par exemple, quand la durée maximale est excédée, la valeur prise à l'instant suivant doit être inférieure à la valeur courante). Ces éléments sont, bien entendu, intégrés dans notre travail et leurs effets sur les performances du système sont étudiés en détail.

## 4.2 Évaluation du système de contrôle

Dans cette section, nous tâchons d'évaluer le système de contrôle en commençant par étudier l'agent des pentes, puis nous aborderons le système complet sous l'angle de l'agent des valeurs d'arrêts. Dans chaque étude, nous considérons différentes façons pour améliorer les performances de l'algorithme. L'évaluation du système porte sur le temps nécessaire pour que l'algorithme converge, mais aussi sur la fréquence de fois où il converge. Nous nous intéressons notamment à évaluer comment l'algorithme arrive à équilibrer entre énergie et confort. Avant d'aborder le cœur de la

partie évaluation, nous présentons donc les notions qui serviront à évaluer les performances des algorithmes.

### 4.2.1 Notions sur l'évaluation des performances des algorithmes d'apprentissage par renforcement sans états

Les performances des algorithmes d'apprentissage par renforcement peuvent être vues selon plusieurs angles. Dans notre cas, l'efficacité d'un système à contrôler le signal lumineux dépend de la qualité des politiques apprises par le système. Or, l'efficacité du système dépend principalement de sa capacité à rechercher un équilibre entre confort et consommation énergétique. Le point central dans nos discussions sur les performances des algorithmes doit majoritairement porter sur la convergence du système.

Nous avons deux éléments à prendre en compte concernant la convergence, premièrement, nous avons *la vitesse* de convergence qui reflète la vitesse à laquelle l'algorithme arrive à trouver un comportement *adéquat* à l'application visée (en faisant un bon compromis entre énergie et confort). Deuxièmement, nous avons *la qualité* de convergence qui informe à quel point ce comportement s'approche du comportement optimal. Ces deux éléments sont à prendre en compte, car ils nous permettraient de distinguer les méthodes capables de trouver rapidement un comportement *adéquat* (même si imparfait) et se reposer sur celui-ci pour l'améliorer de proche en proche et trouver la meilleure solution. Dans les paragraphes qui suivent, nous définissons des métriques sur la qualité de convergence ainsi que sur la vitesse de convergence.

Avant de détailler les métriques sur lesquelles se basent nos comparaisons, précisons d'abord comment se fera l'évaluation de nos algorithmes. L'évaluation des critères d'un paramétrage donné d'un algorithme est obtenu à partir d'un ensemble d'exécutions de cet algorithme pour ce paramétrage. Chaque exécution porte sur un nombre limité d'étapes. Comme les durées des exécutions sont en général assez longues, nous subdivisons la durée de l'apprentissage en sous-intervalles de longueur égale que nous nommons *bins*. Dans le cadre de notre travail, nous mesurons les performances de l'agent en nous basant sur ses choix passés. Nous calculons le nombre de fois où l'agent choisit chaque action, les performances de l'algorithme dépendent alors de la fréquence où l'agent a un comportement désiré (par exemple en termes de fréquence de choix de l'action optimale, nous expliquons ce point plus en avant dans le paragraphe suivant).

Un vecteur est maintenu dans lequel chaque case correspond à une action et contient la fréquence de sélection de cette action durant l'apprentissage. Dans ce vecteur, les actions sont ordonnées de la même manière que dans la politique de l'agent. À chaque fois qu'une action est sélectionnée, les fréquences de choix des actions sont mises à jour. L'évolution de ce vecteur reflète l'évolution de la politique de l'agent indépendamment de la manière dont elle est représentée. Par ailleurs, nous disons qu'un algorithme *converge* pour une *exécution* et un *seuil* donné, si la fréquence de sélection de l'action optimale atteint ce seuil durant l'apprentissage. Nous disons aussi qu'un algorithme *converge dans le voisinage de l'action optimale* pour une *exécution* et un *seuil* donnés, si la somme de la fréquence de sélection de l'action optimale et de l'un de ses deux voisins atteint ce seuil durant l'exécution.

Concernant la *qualité* d'apprentissage, nous proposons la métrique qu'on nomme la *fréquence de convergence* (qu'on abrège en FC) qui est le rapport entre le nombre de fois où l'algorithme converge vers la meilleure action et le nombre total d'exécutions. Nous proposons une autre métrique qu'on nomme la *fréquence de convergence dans le voisinage de l'action optimale* (qu'on abrège en FCVO). La FCVO est le rapport entre le nombre de fois où l'algorithme converge dans le voisinage de l'action optimale et le nombre d'exécutions total. Cette métrique est une version relâchée de la FO. Afin d'avoir une idée sur le comportement des algorithmes, même dans le cas où ceux-ci ont une mauvaise convergence, nous avons besoin d'une métrique qui nous dit dans quelle mesure l'algorithme a tendance à choisir l'action optimale (ou des actions proches de l'action optimale). À cet effet, nous proposons une troisième métrique que nous nommons la *fréquence du mode dans le voisinage de l'action optimale* (qu'on abrège en FMVO). La FMVO est le rapport entre le nombre de fois où l'action ayant été choisie le plus souvent se trouve dans le voisinage de l'action optimale (sans qu'il y ait convergence dans le voisinage de l'action optimale) sur le nombre d'exécutions. Ainsi, nous avons  $FCVO + FMVO = 1$  au maximum. Afin de définir la FO, la FCVO ainsi que la FMVO, nous aurons besoin des notations suivantes. En considérant un agent  $A$  disposant de l'ensemble d'actions  $A$  nous avons :

- $t_f$  et  $t_b$  sont respectivement la durée totale d'une exécution et la taille d'un sous-intervalle de temps (bin).
- $n_e$  est le nombre d'exécutions de la simulation.
- $n_{i,j}(t)$  représente le nombre de fois où l'action d'indice  $j$  a été appliquée durant l'exécution  $i$  jusqu'à l'étape  $t$  de cette exécution.
- $f_{i,j}(t) = \frac{n_{i,j}(t)}{\sum_{j \in A} n_{i,j}(t)}$  est la fréquence à laquelle l'action  $j$  est appliquée durant l'exécution  $i$  jusqu'à l'étape  $t$  de cette exécution.
- $id_{max}$  est l'indice de l'action ayant la récompense la plus grande (la meilleure action) parmi les actions de l'agent.
- $argmax_{j \in A} f_{i,j}(t)$  est l'action estimée comme étant la meilleure par l'agent à l'étape  $t$  pendant l'exécution  $i$ .
- $tb_k = k \cdot \frac{t_b}{t_f}$  est la dernière étape du  $k$ 'ième sous-intervalle (bin).

Nous définissons aussi les ensembles suivants :

- $EC(t, sc) = \{i : f_{i, id_{max}}(t) > sc\}$  est l'ensemble des exécutions pour lesquelles la fréquence de choix de la meilleure action dépasse le seuil  $sc$  à l'étape  $t$ .
- $ECVOD(t, sc) = \{i : f_{i, id_{max}}(t) + f_{i, id_{max}+1}(t) > sc\}$  est l'ensemble des exécutions pour lesquelles la somme de la fréquence de choix de la meilleure action et la fréquence de choix de l'action se trouvant à sa droite dépasse  $sc$  à l'étape  $t$ .
- $ECVOG(t, sc) = \{i : f_{i, id_{max}}(t) + f_{i, id_{max}-1}(t) > sc\}$  est l'ensemble des exécutions pour lesquelles la somme de la fréquence de choix de la meilleure action et la fréquence de choix de l'action se trouvant à sa gauche dépasse  $sc$  à l'étape  $t$ .
- Les notions précédentes nous permettent de définir l'ensemble  $ECVO(t, sc) = EC(t, sc) \cup ECVOD(t, sc) \cup ECVOG(t, sc)$  qui contient les exécutions pour lesquelles la fréquence de sélection de l'action optimale et de ses voisins dépasse le seuil  $sc$  à l'étape  $t$ .

- Nous définissons aussi l'ensemble  $EMVO(t, sc) = \{i : \operatorname{argmax}_{j \in Af_{i,j}(t)} \in \{id_{max}, id_{max} + 1, id_{max} - 1\} \wedge i \notin ECVO(t, sc)\}$ .

Ceci nous permet de définir les métriques mesurant la qualité de convergence :

- La fréquence de convergence est définie pour un seuil donné  $sc$  comme étant  $FC(sc) = \frac{\sum_{i=1}^{n_e} \mathbb{I}_{EC}(t_f, sc)(i)}{n_e}$ .
- La fréquence de convergence dans le voisinage de l'action optimale est définie pour un seuil donné  $sc$  comme étant  $FCVO(sc) = \frac{\sum_{i=1}^{n_e} \mathbb{I}_{ECVO}(t_f, sc)(i)}{n_e}$ .
- La fréquence du mode dans le voisinage de l'action optimale est définie pour un seuil donné  $sc$  comme étant  $FMVO(sc) = \frac{\sum_{i=1}^{n_e} \mathbb{I}_{EMVO}(t_f, sc)(i)}{n_e}$ .

Pour mesurer *la rapidité* de convergence, nous proposons la métrique du *temps moyen de convergence* (TMC). Nous précisons que le temps est mesuré en nombre de *bins* et aussi que la TMC est définie par rapport à un seuil donné. Le calcul de la TMC s'effectue comme suit, premièrement, nous calculons pour chaque exécution le temps nécessaire pour que l'exécution converge dans le voisinage de l'action optimale avec un certain seuil. La TMC est alors la somme de ces temps divisée par le nombre d'exécutions. Afin de définir la TMC formellement, nous aurons aussi besoin des notations suivantes :

- Le temps de convergence de la  $i$ 'ième exécution est défini pour un seuil  $sc$  comme étant  $tc_i(sc) = \min\{k : \forall k' \geq k, i \in ECVO(tb_{k'}, sc)\}$ .
- Le temps moyen de convergence de l'algorithme est alors défini pour un seuil  $sc$  comme étant  $tc(sc) = \frac{\sum_{i=1}^{n_b} tc_i(sc)}{n_e}$ .

## 4.2.2 Évaluation de l'agent des pentes

Dans cette sous-section, nous évaluons, à l'aide des critères proposés auparavant, les performances de l'agent des pentes pour différents algorithmes d'apprentissage par renforcement. Ce qui nous permet de sélectionner les meilleurs algorithmes d'apprentissage par renforcement. Finalement, nous présentons des méthodes afin de rendre l'apprentissage plus rapide pour les algorithmes sélectionnés.

### Évaluation des performances des algorithmes d'apprentissage par renforcement

Nous comparons les performances de ces algorithmes en termes de vitesse de convergence et de qualité de convergence, puis nous nous focaliserons sur les algorithmes qui nous semblent être les plus prometteurs (et qui sont, comme nous les verrons, les algorithmes à base *d'automates à structure variables*, ce sont les algorithmes du même type que LRI). L'objectif est d'étudier le comportement des algorithmes d'apprentissage par renforcement pour la tâche du contrôle de signal lumineux dans un contexte où la réactivité de l'utilisateur est non stationnaire.

**Comparaison des algorithmes classiques d'apprentissage par renforcement** Nous comparons différents algorithmes d'apprentissage par renforcement (sans état) afin d'identifier les algorithmes les plus adaptés à notre problème. Les algorithmes étudiés sont l'algorithme du LRI (Linear

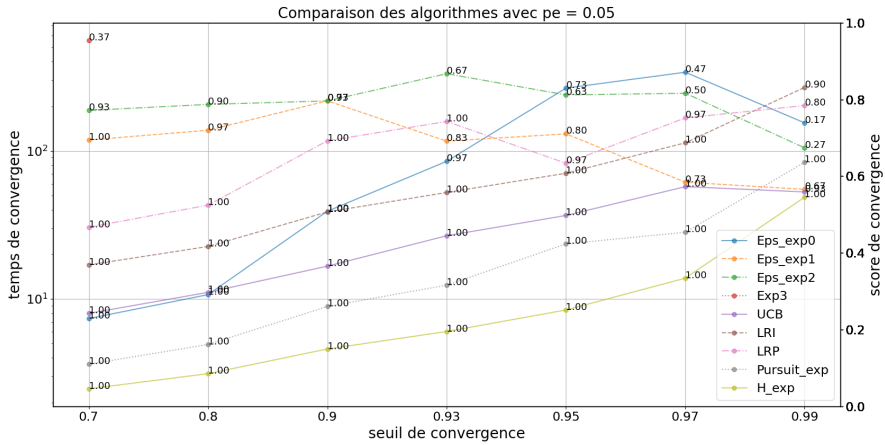
Reward Inaction) [119], l'algorithme du LRP (Linear Reward Penalty) [119], l'algorithme de la poursuite [69], l'algorithme de la poursuite hiérarchique [138], les trois variantes de l'algorithme de  $\epsilon$ -greedy [116], l'algorithme Exp3 [10] ainsi que l'algorithme de UCB (Upper Confidence Bound) [9]. L'expérience permettant d'effectuer cette comparaison ainsi que son analyse sont présentées dans ce qui suit.

Les algorithmes sont testés pendant  $3 \times 10^6$  étapes (subdivisés en intervalles de taille 5000 étapes) pour 30 exécutions pour différents ensembles d'hyperparamètres. L'objectif est de comparer la vitesse de convergence des algorithmes pour un ensemble de seuil de convergence afin d'avoir une idée sur les performances quantitatives des algorithmes. Pour ce faire, nous aurons d'abord besoin de sélectionner les meilleures valeurs des hyperparamètres de chaque algorithme afin de nous reposer ensuite sur ceux-ci pour faire notre comparaison.

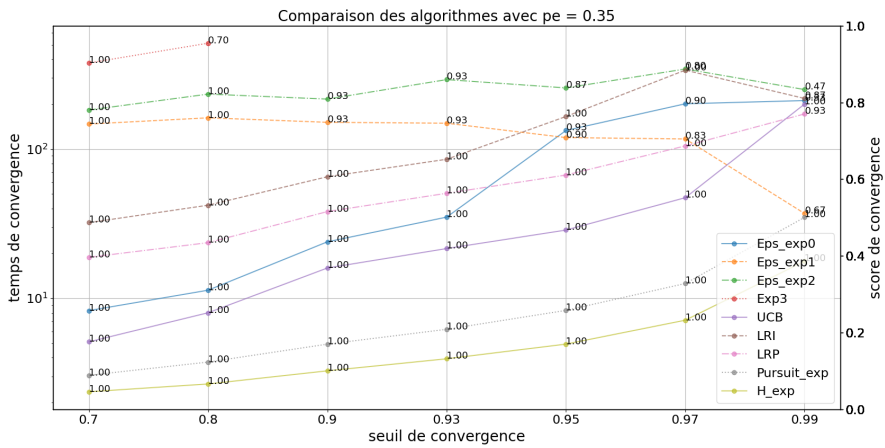
Chaque paramétrisation de l'algorithme est évaluée à l'aide d'un *score* constitué de trois valeurs : un critère sur la qualité de convergence qui est la FCVO de la paramétrisation, un critère sur la rapidité de convergence qui est le TMC de cette même paramétrisation et un dernier critère sur la qualité de convergence qui est la FMVO. Pour chaque algorithme, les meilleures valeurs des hyperparamètres sont choisies en prenant d'abord seulement celles pour lesquelles la FCVO est la plus grande parmi ces paramétrisations. De ce fait, nous aurons seulement les paramétrages pour lesquelles l'algorithme converge le plus souvent possible. Après avoir filtré les mauvaises valeurs des hyperparamètres par la FCVO, la meilleure paramétrisation est choisie en prenant ensuite la valeur des hyperparamètres atteignant le seuil de convergence le plus rapidement sur la base du TMC. Dans le cas où plusieurs paramétrisations ont le même TMC, un autre critère de qualité de convergence qui est la FMVO est utilisé pour choisir la meilleure paramétrisation.

Nous évaluons les performances des algorithmes tels que montré dans le paragraphe précédent pour différents seuils de convergence (défini dans 4.2.1). Ces valeurs du seuil de convergence sont 0.7, 0.8, 0.9, 0.93, 0.95, 0.97, 0.99. Nous prenons pour chacun des algorithmes les meilleures valeurs des hyperparamètres et les résultats sont présentés dans la figure 4.6. La figure 4.6 montre une comparaison sur la vitesse de convergence (mesurée par le TMC) des algorithmes mentionnés auparavant pour les valeurs de  $pe$  considérées (usager à long terme pour  $pe = 0.05$ , usager à moyen terme pour  $pe = 0.35$ , et usager à court terme pour  $pe = 0.95$  de gauche à droite). Les valeurs sur l'axe des ordonnées sont affichées en échelle logarithmique pour faciliter la visualisation des résultats. L'axe des ordonnées à droite de la figure correspond aux valeurs de la FCVO, cet axe est affiché en échelle linéaire. Les temps de convergence des algorithmes (les points de la figure 4.6) sont étiquetés par la valeur de la FCVO correspondant au paramétrage choisi pour chacun des algorithmes. Les algorithmes qui n'ont convergé pour aucune des valeurs testées des hyperparamètres n'ont pas été inclus dans la figure. Dans la figure 4.6, les différentes variantes des algorithmes Epsilon-Greedy sont notées  $Eps\_exp0$ ,  $Eps\_exp1$  et  $Eps\_exp2$  respectivement. L'algorithme de poursuite hiérarchique est noté  $H\_exp$ . Les valeurs des hyperparamètres testés sont montré dans le tableau 4.2.

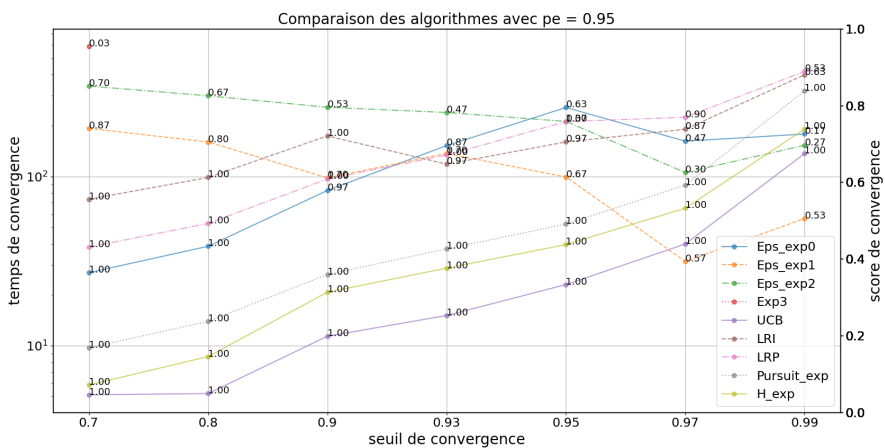
Nous constatons qu'il existe globalement cinq types d'algorithmes en termes de vitesse de convergence, ceux qui prennent beaucoup de temps à converger ou ne convergent pas du tout (Exp3), ceux dont la convergence est lente, mais stable (LRI, LRP), ceux dont la vitesse de convergence ainsi



(a) Comparaison avec  $pe = 0.05$ .



(b) Comparaison avec  $pe = 0.35$ .



(c) Comparaison avec  $pe = 0.95$ .

Figure 4.6 - Comparaison des algorithmes d'apprentissage par renforcement en termes de vitesse de convergence. L'axe des abscisses représente les seuils de convergence et l'axe des ordonnées représente le nombre d'intervalles nécessaires à chaque algorithme pour converger vers le seuil correspondant.

Algorithmes	Valeurs des hyperparamètres
<i>Eps_exp0, Eps_exp1, Eps_exp2</i>	{0.0025, 0.01, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1}
<i>UCB</i>	{0.01, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1, 0.11, 0.12, 0.13, 0.14, 0.15, 0.5, 1.0, 1.41}
<i>LRI, LRP, Pursuit, H_exp</i>	{0.00078125, 0.0015625, 0.003125, 0.00625, 0.0125, 0.025, 0.05, 0.1}
<i>Exp3</i>	{0.0025, 0.01, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1}

Table 4.2 – Liste des valeurs des hyperparamètres testées pour chaque algorithme.

que la qualité de convergence est erratique (Epsilon-Greedy et ses variantes), ceux qui convergent très bien sur tous les types d'utilisateurs (algorithmes de poursuite et de poursuite hiérarchique) et ceux dont le comportement dépend de la mémoire de l'utilisateur (UCB). Dans ce qui suit, nous proposons des hypothèses et des interprétations possibles sur les observations portant sur les performances des algorithmes.

En ce qui concerne la mémoire de l'utilisateur et sa relation avec la vitesse de convergence, on remarque que le cas d'un utilisateur avec une mémoire à moyen terme ( $pe = 0.35$ ) semble être le plus facile pour tous les algorithmes considérés. Nous pouvons voir cela en observant la qualité de convergence (FCVO) des algorithmes pour  $pe = 0.35$  qui approche 1 pour beaucoup d'algorithmes. Le cas de l'utilisateur avec une mémoire à court terme ( $pe = 0.95$ ) représente un défi pour tous les algorithmes (essentiellement en termes de stabilité de convergence), probablement parce que l'utilisateur change brusquement quand les actions qui se suivent sont beaucoup trop différentes (surtout pour les pentes raides). Le cas d'un utilisateur avec une mémoire à long terme ( $pe = 0.05$ ) présente un défi pour l'agent, car les réactions de l'utilisateur dépendent du comportement passé du système (et cette dépendance peut s'étendre très loin dans le passé). Effectivement, le comportement de l'utilisateur pourrait varier pour une même action suivant les actions passées choisies par le système.

L'algorithme de Exp3 semble avoir des performances médiocres. En effet, l'hypothèse de présence d'un adversaire sous-jacent à l'algorithme de Exp3 ne semble pas correspondre à notre problématique, et ce, malgré l'opposition apparente qui existe entre l'utilisateur dont les besoins sont énergivores et le système dont l'objectif est de réduire la consommation d'énergie.

La convergence de Epsilon-Greedy (toutes variantes confondues) est assez instable, et ce, pour tous les types d'utilisateur. Pour certains seuils, Epsilon-Greedy converge rapidement et très peu souvent, et pour d'autres paramétrages, il converge lentement et de manière assez stable (d'où la forme en dents de scie des performances des variantes de Epsilon-Greedy). Notons que l'algorithme de Epsilon-Greedy présente des performances instables, et ce même pour les variantes dans lesquelles la valeur de  $\epsilon$  est décrétementée au cours du temps. Ce point semble confirmer les observations de la section 2.1 sur la difficulté de choisir la bonne valeur de  $\epsilon$  et la manière de la décrétement.

Les algorithmes LRI et LRP convergent également assez lentement (de 50 à 110 bins). En ce qui concerne l'algorithme LRI, la raison la plus classique de la lenteur de convergence de cet algorithme est que le taux d'apprentissage doit être fixé à un niveau bas pour que l'agent ne reste pas bloqué sur une action sous-optimale. En revanche, dans l'algorithme LRP, puisque les probabilités des actions ne donnant pas de récompenses sont réduites, la lenteur de convergence pourrait être mieux expliquée par le fait que le vecteur stochastique virevolte beaucoup trop souvent avant de converger vers de



bonnes actions. Dans tous les cas, la mise à jour de la politique change de très peu les probabilités du vecteur stochastique pour de faibles valeurs des hyperparamètres, ce qui pourrait expliquer le fait que les variantes de LRI convergent de manière aussi stable. Le comportement de LRI est à mettre en parallèle avec celui de Epsilon-Greedy : en effet, dans Epsilon-Greedy la politique peut changer très rapidement durant l'apprentissage, car elle dépend de la meilleure action estimée par l'agent, et celle-ci change au cours du temps, ce qui n'est pas le cas de LRI, dans lequel la politique est affectée petit à petit au cours de l'apprentissage.

Le comportement de l'algorithme UCB est qualitativement différent pour les types d'usager que nous avons considéré. En effet, UCB est le meilleur algorithme dans le cas où la mémoire de l'usager est faible ( $pe = 0.95$ ) et ses performances sont moins bonnes lorsque  $pe$  prend une valeur faible ou modérée ( $pe = 0.05$  et  $pe = 0.95$ ). Les performances de UCB dépendent principalement de la façon dont la borne supérieure est calculée ainsi que de la mise à jour des estimations. Comme les intervalles de confiance des estimations dépendent énormément des statistiques sous-jacentes à l'environnement, il est difficile de choisir des bornes supérieures proches des vraies valeurs pour des environnements non stationnaires, ce qui pourrait expliquer les variations des performances de UCB. Nous détaillons dans ce qui suit une hypothèse pouvant expliquer ces variations.

Puisque les probabilités de réaction de l'usager dépendent des actions passées du système, on pourrait supposer que l'algorithme aurait besoin de mettre à jour ses bornes de manière différente en fonction du comportement passé de l'algorithme. Cependant, la courbe associée à UCB montre que l'algorithme a de bonnes performances (meilleures que la majorité des algorithmes testés) pour tous les types d'usager. À notre sens, ceci constitue une indication que l'homogénéisation de la mémoire joue un rôle important dans les performances de UCB. La mémoire aurait tendance à s'homogénéiser dès le début de l'apprentissage. Il est possible qu'UCB tende à préférer les politiques bonnes en énergie et mauvaises pour l'usager au tout début, puis partir de ces politiques pour converger progressivement vers les actions de la région creuse.

L'UCB est plus performant dans le cas d'un usager ayant une faible mémoire ( $pe = 0.95$ ), probablement parce que ce cas est le plus proche du cas stationnaire (le contexte pour lequel l'algorithme UCB a été conçu). En effet, pour un usager ayant une mémoire très courte, si l'agent a tendance à choisir les pentes qui ne changent pas (ou peu) l'état de l'usager (par exemple, prendre des pentes douces alors que l'usager est peu sensible), l'environnement devient alors stationnaire du point de vue de l'agent. Et si, comme nous l'avons dit dans le paragraphe précédent, l'agent a tendance à préférer les politiques douces vers la fin de l'apprentissage, la mémoire de l'usager aurait tendance à s'homogénéiser beaucoup plus rapidement si celle-ci est vraiment courte. Et ceci expliquerait pourquoi UCB est meilleur quand la mémoire de l'usager est courte. Il faudrait néanmoins fournir des expériences testant l'algorithme de manière très intensive afin de confirmer de manière rigoureuse ces hypothèses portant sur les performances de UCB.

Les algorithmes de poursuite et de poursuite hiérarchique convergent très rapidement et de manière très stable (particulièrement l'algorithme de la poursuite hiérarchique). Concernant l'algorithme de poursuite et ses variantes, nous rappelons que la mise à jour du vecteur stochastique est effectuée de manière à poursuivre la meilleure action (suivant les estimations de l'agent). Manifestement,

cette stratégie de mise à jour semble constituer un meilleur moyen de traiter le dilemme exploration versus exploitation que le LRI. En effet, l'action que l'agent considère comme étant la meilleure, change de moins en moins au fur et à mesure que l'algorithme acquiert des connaissances sur son environnement. Cette propriété nous permettrait de fixer des taux d'apprentissage plus élevés pour l'algorithme sans subir de perte de performances comme avec l'algorithme LRI. Finalement, nos expériences semblent indiquer que la famille des algorithmes de poursuite semble être la plus adéquate pour notre problème, indépendamment de la mémoire de l'utilisateur. Cependant, tout comme pour UCB, les interprétations portant sur ces algorithmes doivent être confirmées par des expériences supplémentaires dont la conception devra faire l'objet d'un travail de recherche particulier.

Le cas d'étude considéré dans la thèse est tel que l'environnement est non stationnaire, dans le sens où l'environnement change en réponse à l'agent, ce qui diffère sensiblement des cas les plus couramment étudiés. Nous tâchons d'être prudent par rapport aux conclusions tirées en soulignant le fait que celles-ci ne s'étendent pas nécessairement à tout type de non stationnarité. Par rapport aux algorithmes de type poursuite, nous précisons tout de même que, comme seule la probabilité de la *meilleure action selon l'agent* est mise à jour, l'agent pourrait aussi être tenté d'exploiter plus que nécessaire les actions déjà apprises (si on était dans un autre type d'environnement).

Dans cette sous-section, nous avons comparé le comportement de différents algorithmes d'apprentissage par renforcement. Nous en avons conclu que les algorithmes de type poursuite ont les meilleures performances. Ces algorithmes sont des algorithmes basés sur la politique, dérivés de LRI (plus précisément des algorithmes basés sur *les automates à structure variable*). Dans ce qui suit, nous allons étudier qualitativement le comportement de ces algorithmes en observant leurs performances en termes de vitesse de convergence et de qualité de convergence. Nous comparons deux à deux les algorithmes de LRI et de la poursuite puis ceux de la poursuite et de la poursuite hiérarchique. L'algorithme du LRI sera pris comme base de comparaison.

### **L'algorithme du LRI et l'algorithme de poursuite**

Dans ce qui suit, nous montrons quelques exemples de convergence des algorithmes de LRI et de la poursuite en comparant le comportement de ces deux algorithmes. Pour ce faire, nous effectuons une expérience dans laquelle nous exécutons l'algorithme de poursuite et le LRI 30 fois pour 1000000 étapes avec  $pe = 0.35$ . Nous sélectionnons l'utilisateur avec une mémoire à moyen terme ( $pe = 0.35$ ) afin d'éviter de tirer de conclusions générales sur des cas extrêmes (mémoire trop petite ou trop grande). Afin d'observer le comportement de l'algorithme, le taux d'apprentissage est fixé à 0.001. La valeur du taux d'apprentissage a été choisie comme étant la plus basse possible permettant à l'algorithme de converger en un temps raisonnable. Dans l'algorithme de poursuite, toutes les estimations des actions sont initialisées à 0 (un choix classique qui est pertinent ici, car l'agent est pessimiste, toutes les pentes sont supposées sur la "partie linéaire"). Comme précédemment, l'évolution de la récompense est observée pendant l'exécution. Nous calculons la moyenne de la récompense sur toutes les exécutions. Les résultats sont représentés sur les courbes (figure 4.7) décrivant l'évolution de ces valeurs moyennes. Dans la figure 4.7, les cartes thermiques représentent les vecteurs stochastiques, à la fin de chaque exécution. Dans ces cartes, l'axe des x est l'indice de la pente et l'axe des y est le nombre d'exécutions.

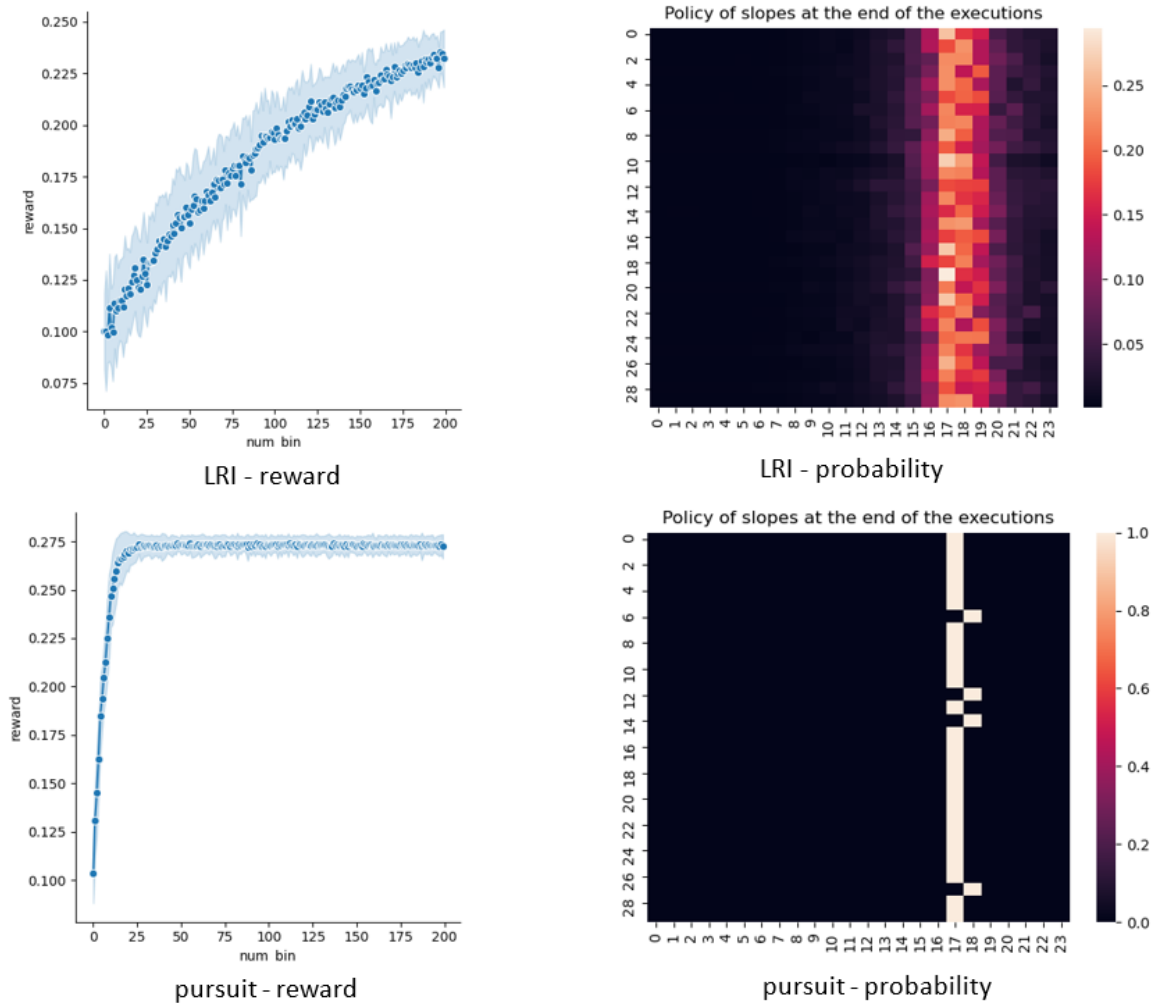


Figure 4.7 – Comparaison de LRI contre l’algorithme de poursuite - 1000000 étapes. Pour la colonne de gauche, l’axe des x est le temps, l’axe des y est la récompense. Pour la colonne de droite, l’axe des x est l’indice de la pente, l’axe des y est le nombre d’executions.

En observant les courbes de la figure 4.7, on remarque que l’algorithme de poursuite converge beaucoup plus vite que l’algorithme LRI (près de 10 fois plus vite). Nous remarquons également que l’algorithme de la poursuite converge presque à chaque fois. Ces résultats renforcent et affinent les conclusions de la comparaison précédente concernant les algorithmes de LRI et de la poursuite en mettant bien en relief que l’algorithme de la poursuite est plus rapide que LRI.

Comme l’algorithme de la poursuite estime directement la récompense des actions, il constitue une meilleure façon de traiter le dilemme exploration vs exploitation qui caractérise les problèmes d’apprentissage par renforcement, car au fur et à mesure que l’agent acquiert des connaissances sur l’environnement, l’action estimée comme étant la meilleure action change de moins en moins fréquemment et le vecteur stochastique converge plus rapidement vers elle. Nous pouvons voir à travers les cartes à chaleur dans 4.7 que l’algorithme de la poursuite est très stable en termes de convergence. En effet, contrairement à l’algorithme de LRI, l’algorithme de la poursuite attribue des probabilités élevées pour le choix des actions optimales et proches de l’optimale (l’action d’indice 17). Et comme nous l’avions supposé, on peut voir à travers ces cartes qu’on aurait pu se permettre

de choisir des taux d'apprentissage plus grand, car l'algorithme de la poursuite est plus certain des actions à prendre.

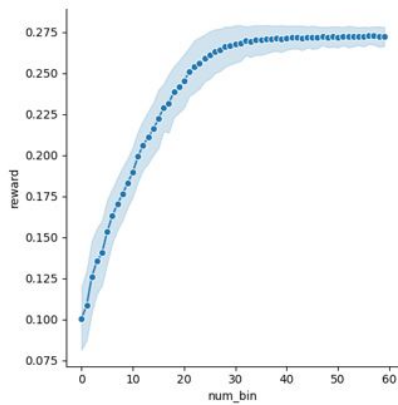
L'apport de l'expérience présentée ici est de montrer *en quoi et comment* l'algorithme de la poursuite est meilleur que l'algorithme du LRI. Dans ce qui suit, nous effectuons la même comparaison afin de nous renseigner sur des différences de comportement pouvant exister entre deux algorithmes du même type : la poursuite et la poursuite hiérarchique.

### **L'algorithme de la poursuite hiérarchique, une variante de l'algorithme de la poursuite**

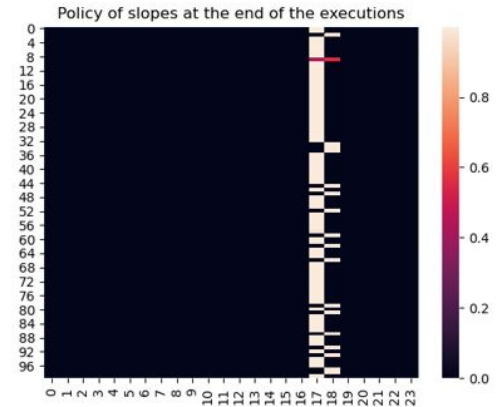
Tel que présenté dans l'article original [138], la force de l'algorithme de poursuite hiérarchique réside dans la capacité de cet algorithme à gérer de très grands espaces d'actions. Si l'agent dispose d'énormément d'actions, la probabilité de choisir chacune des actions sera très faible. Par conséquent, l'algorithme aura du mal à distinguer les actions qui devraient être explorées/exploitées, certaines actions pourraient ne jamais être choisies. Dans l'algorithme de la poursuite hiérarchique, la décision de choisir une action est décomposée et prise collectivement par une succession d'agents. Ces agents sont structurés dans un arbre. L'agent associé à la racine sélectionne un agent parmi ses fils, et ainsi de suite jusqu'à arriver au choix des actions. L'expérience faite en début de section sur la comparaison des algorithmes prouve bien que l'algorithme de la poursuite hiérarchique est plus rapide que la poursuite, en effet, nous avons bien vu dans les figures 4.6 que la poursuite hiérarchique surpasse son homologue classique sur la base de critères quantitatifs : sur la vitesse et la qualité de convergence. Le but de l'expérience faite ici est d'affiner encore plus ce résultat afin de proposer des interprétations possibles sur la différence de performances entre ces deux algorithmes, en examinant qualitativement le comportement de ces algorithmes.

Une comparaison entre l'algorithme de poursuite et l'algorithme de poursuite hiérarchique est présentée ici où nous avons  $pe = 0.35$  en suivant le même principe que l'expérience précédente. Nous calculons la moyenne de la récompense au cours du temps pour des intervalles de taille 5000. Nous exécutons 100 simulations de 300000 étapes, où nous calculons la moyenne de la récompense. Le taux d'apprentissage est fixé à 0.001. Les estimations initiales des actions sont fixées à zéro pour les algorithmes de poursuite et de poursuite hiérarchique. Pour l'algorithme de la poursuite hiérarchique, chaque agent dans la hiérarchie d'agents possède deux actions seulement comme dans [138]. Les résultats sont présentés dans les figures 4.8.

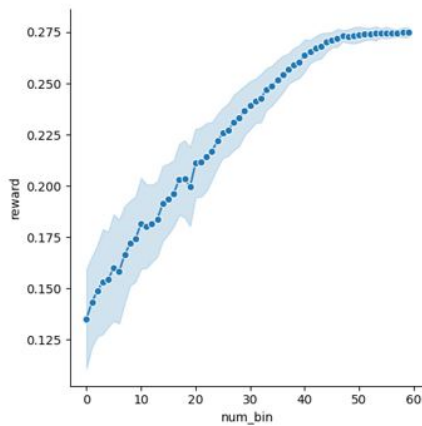
Les algorithmes de poursuite et de poursuite hiérarchique convergent tous deux vers la pente optimale (ou une pente proche de l'optimale) en moins de 300000 étapes. Bien que l'algorithme de poursuite semble plus rapide que sa version hiérarchique, ce dernier semble converger plus souvent vers la meilleure action. Cette constance dans la convergence de l'algorithme pourrait signifier qu'une amélioration supplémentaire de la vitesse de convergence est encore possible si nous augmentons suffisamment le taux d'apprentissage (au prix de ne pas converger exactement sur la meilleure action). Nous l'avons d'ailleurs observé dans la figure 4.6. En raison du fait que le choix d'une action est effectué de proche en proche, en faisant intervenir plusieurs agents, l'algorithme de la poursuite hiérarchique semble beaucoup privilégier l'exploration au début de l'apprentissage et mise sur l'exploitation vers la fin. En effet, l'algorithme de la poursuite hiérarchique a autant de chance de choisir



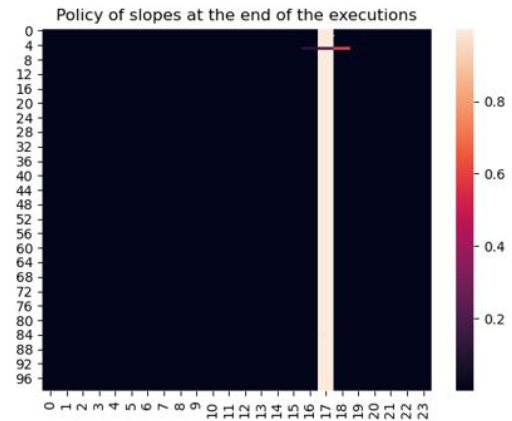
reward - pursuit



probabilities - pursuit



reward - hierarchical



probabilities - hierarchical

Figure 4.8 – L’algorithme de poursuite vs. l’algorithme de poursuite hiérarchique pour le système de choix de la pente - 300000 étapes. Pour la colonne de gauche, l’axe des x est le temps, l’axe des y est la récompense. Pour la colonne de droite, l’axe des x est l’indice de la pente, l’axe des y est le nombre d’exécutions.

les actions se trouvant d’un côté de l’espace d’action que de l’autre au tout début de l’apprentissage. De cette façon, même si la probabilité de sélection d’une action est beaucoup plus faible que la précision arithmétique, elle aura tout de même une chance d’être choisie par l’algorithme. Au cours de l’apprentissage, les régions ayant de mauvaises actions (voir la figure 4.3) auront de faibles chances d’être prises et seront donc écartées par l’algorithme qui ne laissera que les régions contenant les bonnes actions. L’algorithme de la poursuite hiérarchique, agissant ainsi à la façon d’une recherche binaire, est censé être beaucoup plus rapide que la poursuite classique quand le nombre d’actions est très grand. Remarquons finalement que l’algorithme de la poursuite semble converger rapidement vers de bonnes actions (sans nécessairement trouver à chaque fois l’action optimale), alors que l’algorithme de la poursuite hiérarchique converge moins rapidement, mais arrive toujours à trouver l’action optimale (comme le montre la figure 4.8).

L’apport de l’expérience présentée ici, (tout comme celle qui la précède) est de montrer *en quoi* et *comment* certains des algorithmes basés sur la politique peuvent être meilleurs ou plus rapides que d’autres. Nous avons, dans ce cas-là, montré des différences qualitatives entre les gains de la pour-

suite par rapport à LRI d'un côté et les gains de la poursuite hiérarchique par rapport à la poursuite de l'autre. Dans ce qui suit, nous montrons comment améliorer la convergence des algorithmes en commençant par préciser quelles sont les éléments qui influent sur le comportement des algorithmes au cours de l'apprentissage. Nous commençons par étudier l'effet du nombre d'actions, qui est bien entendu le point le plus critique dans la vitesse de convergence des algorithmes d'apprentissage par renforcement.

## Accélération des algorithmes par la durée maximale dynamique des actions

Nous proposons dans cette sous-section des moyens d'accélérer l'algorithme de la poursuite qui a été sélectionné par les expériences précédentes. Une approche possible pour améliorer considérablement la vitesse de convergence de l'algorithme consiste à faire varier la durée maximale des actions (discrétisation du temps). Une autre façon d'accélérer l'algorithme est de déterminer le meilleur ensemble d'actions disponibles pour l'agent (discrétisation de l'espace d'action). Nous montrons également comment l'ensemble des actions disponibles pour l'agent influence grandement la vitesse de convergence de l'algorithme.

### L'effet du nombre d'actions sur l'agent

Le premier élément qu'on considère et qui influence la convergence des algorithmes est le nombre d'actions dont dispose l'agent. Nous étudions, dans ce qui suit, son effet sur la vitesse de convergence de l'algorithme.

L'expérience que nous présentons consiste à lancer 30 fois le système de contrôle pour différents ensembles d'actions (pentes) correspondant à différents espaces d'action. Ces ensembles correspondent aux pentes où l'écart entre deux pentes successives est de 5, 10, 15 et 20. Avant de réaliser l'expérience, la meilleure pente a été calculée pour chaque ensemble d'actions et pour chaque valeur de  $pe$  en les répétant et en calculant la récompense moyenne, l'énergie moyenne et la valeur moyenne de  $m$  pendant les essais. L'analyse porte sur la récompense et l'énergie, car celles-ci sont connues du système et peuvent être estimées à partir de son comportement, l'état de l'utilisateur est montré pour mettre en relief d'éventuels compromis apparaissant entre la consommation de l'énergie et le confort de l'utilisateur.

Tout d'abord, nous précisons que l'action optimale en termes de récompense est la même pour l'ensemble des actions associées aux écarts 5 et 15. L'ensemble des actions associées aux écarts 10 et 20 ont des actions optimales différentes. L'algorithme utilisé est l'algorithme de la poursuite où le taux d'apprentissage est fixé à la valeur  $lr = 0.001$ . Chaque exécution a une durée de 1500000 étapes décomposées en intervalles de taille 5000 où l'on observe la valeur moyenne de l'énergie, la récompense et la valeur de  $m$ . Nous calculons ensuite la moyenne de ces valeurs sur toutes les exécutions. Les résultats sont présentés sur la figure 4.9 pour chaque valeur de  $pe$  (0.05, 0.35, 0.95). Chaque ligne correspond au résultat de l'expérience pour différentes valeurs de  $pe$  (on a  $pe \in \{0.05, 0.35, 0.95\}$ ). La colonne de gauche correspond à l'évolution dans le temps de la récompense pour toutes les valeurs de  $pe$ . La colonne du milieu correspond à l'évolution de l'énergie pour différentes valeurs de  $pe$ . Quant à la colonne de droite, elle correspond à l'évolution de la valeur de  $m$

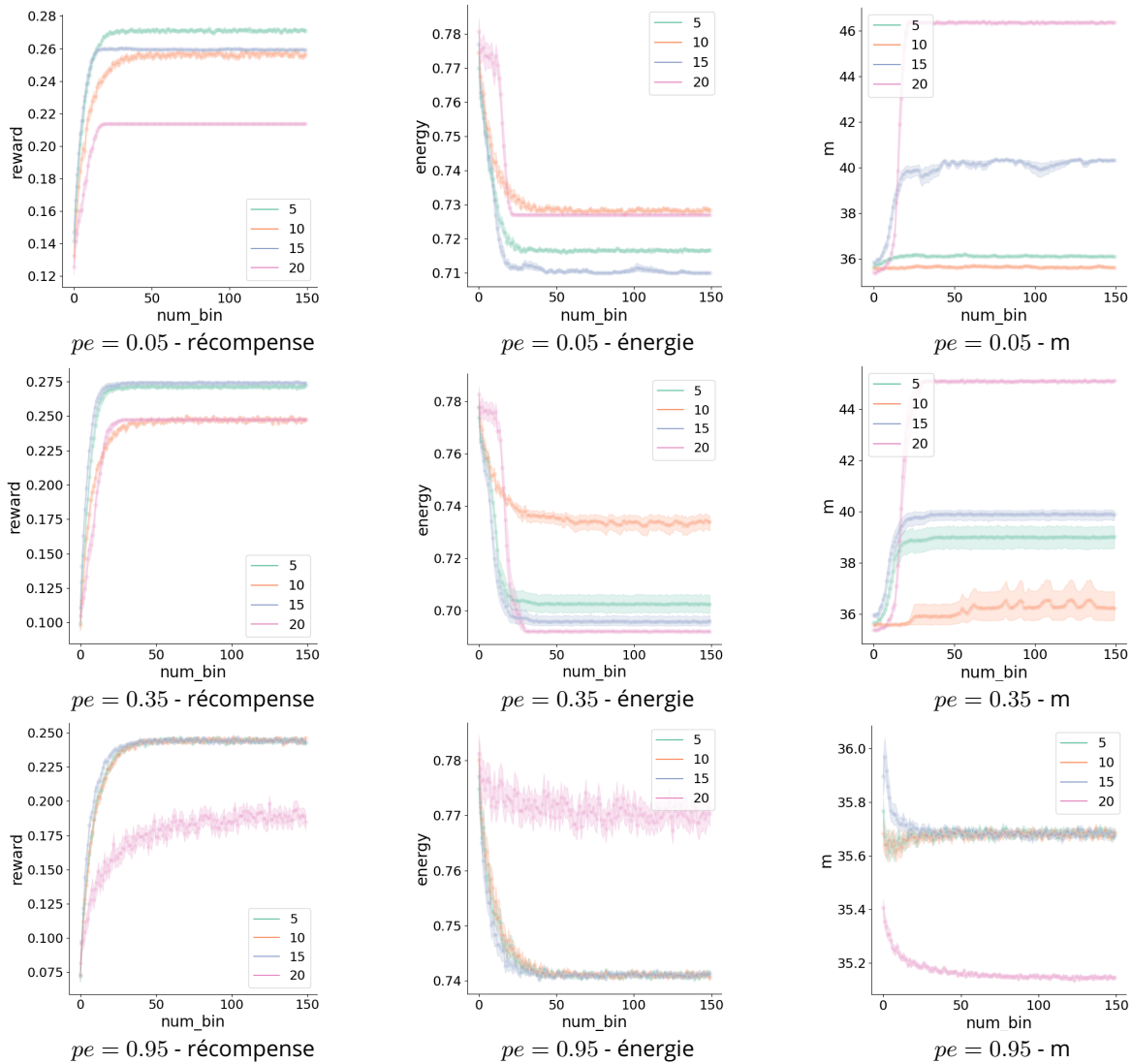


Figure 4.9 – L'évolution de la récompense (à gauche), de l'énergie (au milieu), et de l'état de l'utilisateur (à droite) pour un ensemble d'actions d'écart 5, 10, 15 et 20 pour les pentes.

pour  $pe = 0.35$ .

En observant la figure 4.9, nous constatons tout d'abord que le système s'améliore au cours du temps pour toutes les métriques, à l'exception de la valeur de  $m$  (mais l'utilisateur reste toujours dans la zone acceptable, car la valeur du paramètre  $m$  est proche de la valeur la plus basse  $m_0 = 35$  pour tous les types d'utilisateur, le lecteur pourra se référer à la figure 4.3 pour s'en convaincre). La deuxième observation que nous faisons est que les performances du système ne varient pas linéairement avec l'ensemble des actions. Ceci vaut pour tous les types d'utilisateurs, particulièrement pour l'utilisateur de mémoire à long terme où il est clair que le meilleur ensemble d'actions en termes de récompense est dans l'ordre décroissant 5, 15, 10, 20. Les mêmes observations peuvent être relevées concernant la vitesse de convergence. Il ne semble donc pas y avoir de relation simple entre les performances de l'algorithme d'apprentissage (en termes de récompense ou de vitesse de convergence) et la discrétisation de l'espace d'actions ainsi que la mémoire de l'utilisateur. Dans ce qui suit, nous analysons la discrétisation de chacun des types d'utilisateur pour enchaîner avec une conclusion globale concernant

cette étude.

Concernant l'utilisateur ayant une mémoire à moyen terme ( $pe = 0.35$ ), nous constatons que le meilleur ensemble d'actions pour ce type d'utilisateur est obtenu lorsque l'écart entre les actions successives est de 15. Nous constatons aussi que les performances diminuent au fur et à mesure que l'on s'éloigne de cet ensemble d'actions, tant en termes de vitesse de convergence que de la valeur de récompense obtenue par l'agent. Avec l'ensemble d'actions où l'écart est de 20, nous notons une très faible consommation d'énergie. D'autre part, l'utilisateur semble être dans un mauvais état par rapport aux autres discrétisations, nous le remarquons à la valeur de  $m$  qui approche 45 (la moitié de l'intervalle des valeurs du signal). L'ensemble d'actions associé à l'écart 10 obtient la même récompense que le précédent, mais nous pouvons voir que la consommation d'énergie est beaucoup plus élevée et que l'état de l'utilisateur est beaucoup plus faible qu'auparavant. Une autre observation que nous faisons est que même lorsque les actions optimales sont les mêmes pour différents ensembles d'actions (par exemple, les ensembles d'actions 5 et 15 ont des actions optimales similaires), le comportement de l'utilisateur et le comportement de l'algorithme changent qualitativement.

Pour l'utilisateur ayant une mémoire à long terme ( $pe = 0.05$ ), nous observons les mêmes tendances que pour l'utilisateur ayant une mémoire à moyen terme. Le meilleur ensemble d'actions correspond à l'ensemble où l'écart entre deux pentes successives est de 5, les performances se dégradent à mesure que l'on s'éloigne de cette discrétisation. Encore une fois, nous observons différents comportements pour des ensembles d'actions ayant des récompenses similaires. Pour l'utilisateur ayant une mémoire à long terme, cela concerne les discrétisations correspondant aux écarts 15 et 10. La consommation d'énergie est bien plus faible pour le cas où l'écart entre les pentes est de 15 par rapport au cas où l'écart est de 10, mais la relation inverse est retrouvée concernant l'état de l'utilisateur. Des différences qualitatives concernant la consommation d'énergie du système ainsi que l'évolution de l'état de l'utilisateur se trouvent dans les deux meilleures discrétisations (en 5 et en 15 comme montré dans la figure 4.9) alors qu'elles conduisent à des récompenses comparables et qu'elles ont la même action optimale.

Concernant l'utilisateur ayant une mémoire à court terme ( $pe = 0.95$ ), nous notons aussi d'énormes similitudes entre toutes les discrétisations sauf pour la discrétisation en 20. Notons également que l'utilisateur ayant une mémoire à court terme est celui qui apporte le plus grand défi pour les algorithmes d'apprentissage par renforcement, comme montré dans la figure 4.6. Ainsi, même si le choix de la discrétisation peut sembler, à première vue, moins conséquent que pour les deux premiers types d'utilisateur (ceux ayant une mémoire à moyen et à longs termes), il n'en est pas moins vrai que ce type d'utilisateur est le plus dur de tous comme l'a montré l'expérience sur la comparaison des algorithmes. De plus, on peut remarquer à travers les figures 4.9 que si l'on commence à s'écarter un peu trop des bonnes discrétisations, on risque d'observer d'énormes baisses en termes de performances (si l'on choisit la discrétisation en 20). Nous en concluons que le choix de la bonne discrétisation est aussi risqué pour l'utilisateur ayant une mémoire à court terme que pour les autres types d'utilisateur.

Une interprétation possible des observations mentionnées précédemment est qu'il y a un compromis à prendre en compte dans le choix de l'ensemble d'actions. Si nous prenons une discrétisation assez grossière de l'espace d'action, l'action optimale (sur toutes les discrétisations possibles) peut ne pas être sur l'ensemble d'actions sélectionné, ce qui conduit l'algorithme à apprendre des ac-



tions sous-optimales, voire à rendre l'apprentissage instable (car le système alterne entre des actions bonnes et mauvaises pour l'utilisateur). Mais si l'ensemble d'actions est trop dense, le système peut prendre beaucoup plus de temps pour apprendre la meilleure action parce que le système doit essayer toutes les actions suffisamment de fois pour identifier la meilleure action. L'ensemble d'actions sélectionné peut affecter la convergence de l'algorithme même si les actions optimales sont les mêmes pour différents ensembles d'actions. Une approche possible pour traiter ces problèmes est d'envisager l'utilisation de méthodes qui gèrent de grands espaces d'actions en filtrant rapidement les bonnes actions des mauvaises. Les méthodes hiérarchiques [138] présentées auparavant peuvent constituer une bonne alternative à explorer pour traiter cet inconvénient. Enfin, en nous basant sur l'évolution de la récompense, nous pouvons conclure que le meilleur ensemble d'actions est celui où l'écart entre pentes est de 5 pour l'utilisateur avec une mémoire à long terme. Pour l'utilisateur avec une mémoire à moyen terme, le meilleur ensemble d'actions correspond à un écart de 15, car la récompense est la plus grande, dans ce cas. Et pour un utilisateur avec une mémoire à court terme, nous préférons également choisir l'ensemble d'actions associé à l'écart 15.

Nous avons traité dans cette partie comment le choix des actions influence les performances du système ainsi que l'évolution de l'état de l'utilisateur. Notons par contre que les actions n'ont pas forcément les mêmes durées, la durée d'une action dépend notamment du comportement de l'utilisateur. Après avoir traité la question de la discrétisation de l'espace d'actions, nous traitons ensuite la question de savoir s'il est utile d'imposer une durée maximale pour les actions et s'il est utile de la faire varier (la question de la discrétisation du temps). L'étude sur la durée maximale des actions constitue le prolongement naturel des questions sur l'efficacité des algorithmes que nous avons examinés jusqu'ici.

### Durée maximale des actions

Dans l'optique d'accélérer la convergence des algorithmes d'apprentissage par renforcement dans notre problème, nous proposons d'étudier l'effet qu'a la durée maximale des actions sur l'apprentissage. À cet égard, nous mettons en place une expérience où nous exécutons l'algorithme de la poursuite 30 fois pendant 1500000 étapes. Le temps est subdivisé en sous-intervalles de taille 5000 dont on ne retient que la moyenne des valeurs calculées (récompense, énergie, valeur de  $m$ ). Le taux d'apprentissage est fixé à 0,002. L'expérience a été réalisée pour différentes valeurs du paramètre d'effet présent, à savoir 0,05, 0,35 et 0,95.

Les graphiques (sur la figure 4.10) montrent les performances du système de contrôle avec l'algorithme de la poursuite pour le cas où la décision de choisir une action est prise par le système à chaque intervention de l'utilisateur (nommé *base\_pur*), et pour le cas où celle-ci est fixée à une durée maximale fixe (nommé *freq\_dur*). Les graphiques de la figure 4.10 illustrent également les performances du système avec les deux façons d'utiliser la règle de mise à jour de la durée maximale dynamique des actions exposées dans la section 4.1. Le premier cas correspond à *update\_mean\_var\_interv*, et le second cas correspond à *update\_mean\_interv* (voir la sous-section 4.1.1). Dans la figure 4.10, chaque ligne correspond à une valeur de la mémoire de l'utilisateur (et sont de haut en bas  $pe = 0.05$ ,  $pe = 0.35$  et  $pe = 0.95$ ). Les colonnes représentent de gauche à droite l'évolution dans le temps de la valeur de la récompense, de l'énergie et de l'état de l'utilisateur (valeur de  $m$ ). Pour le cas *freq\_dur*, la durée est prise comme étant la meilleure parmi les valeurs  $\{1, 5, 10, 25, 50, 100\}$  pour chaque valeur de  $pe \in \{0.05, 0.35, 0.95\}$ . Pour les cas où la durée maximale des actions change, la constante

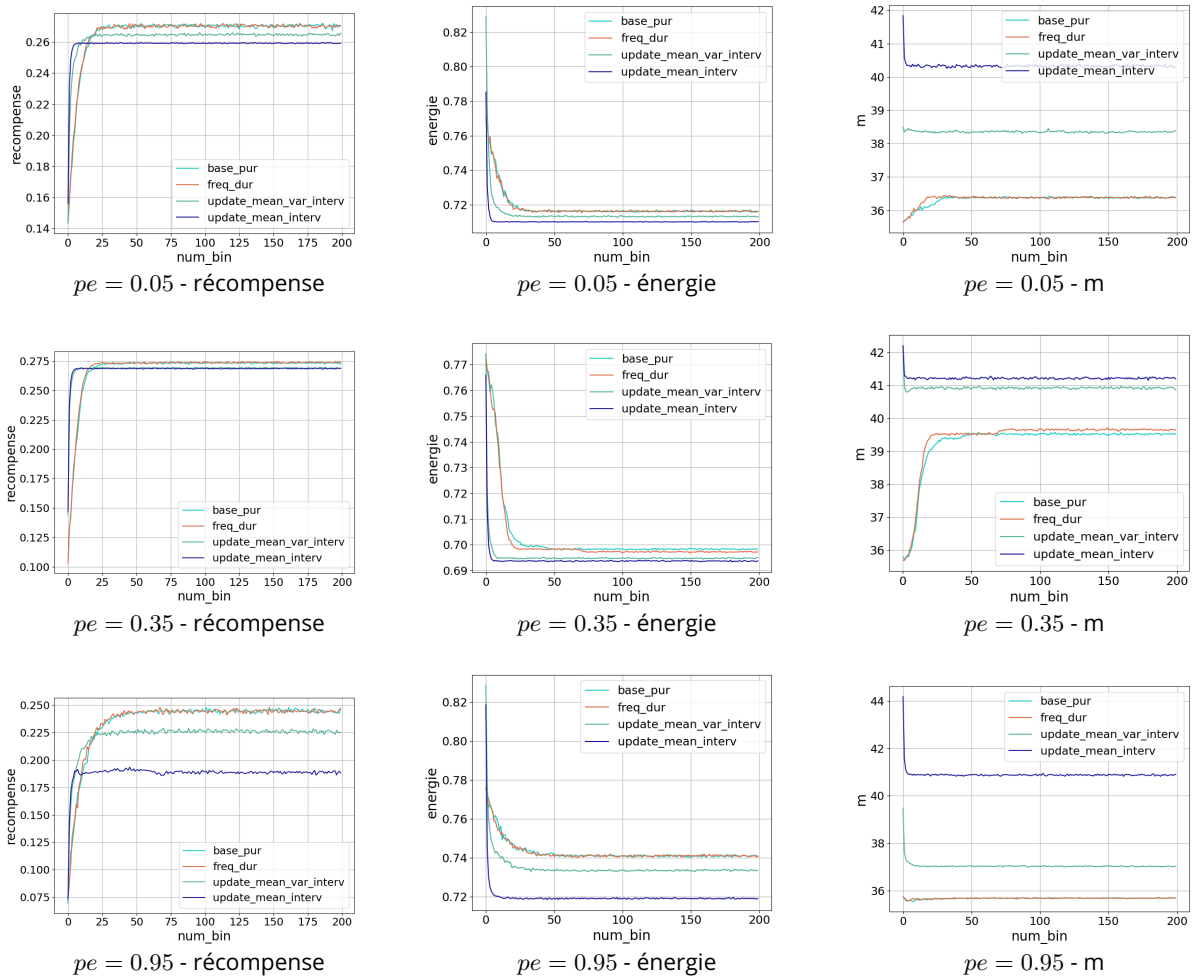


Figure 4.10 – L'évolution de la récompense (à gauche), de l'énergie (au milieu), et de l'état de l'utilisateur (à droite) pour un ensemble d'actions d'écart 5, 10, 15 et 20 pour les pentes.

qui est multipliée par l'écart-type dans le calcul de la durée maximale (voir 4.1.1), est également choisi comme étant la meilleure valeur parmi  $\{1, 2, 3, 4\}$  pour chaque  $pe \in \{0.05, 0.35, 0.95\}$ . Nous affichons également les cartes thermiques (dans la figure 4.11 pour le cas  $pe = 0.35$ ) des vecteurs stochastiques à la fin de l'apprentissage. Sur les cartes de chaleur, les axes sont étiquetés par les indices des pentes (axe des x) et par le numéro d'exécution (axe des y) (comme pour la figure 4.7).

D'après les courbes, nous pouvons observer que la récompense se stabilise le plus rapidement lorsque la règle de mise à jour de la durée maximale des actions est *update\_mean\_interv* pour toutes les valeurs de  $pe$ . Le cas *update\_mean\_var\_interv* se stabilise également rapidement pour tous les types d'utilisateur considérés. En outre, la valeur de la récompense est un peu plus élevée au début de l'apprentissage avec *update\_mean\_var\_interv*. Nous en concluons que faire varier la durée maximale des actions de manière dynamique (les cas *update\_mean\_interv* et *update\_mean\_var\_interv*), permet d'accélérer la convergence des algorithmes par rapport aux méthodes dans lesquelles la durée maximale est fixe ou illimitée (les cas *base\_pur* et *freq\_dur*).

Faire varier la durée maximale des actions permet de rendre l'apprentissage plus rapide, mais

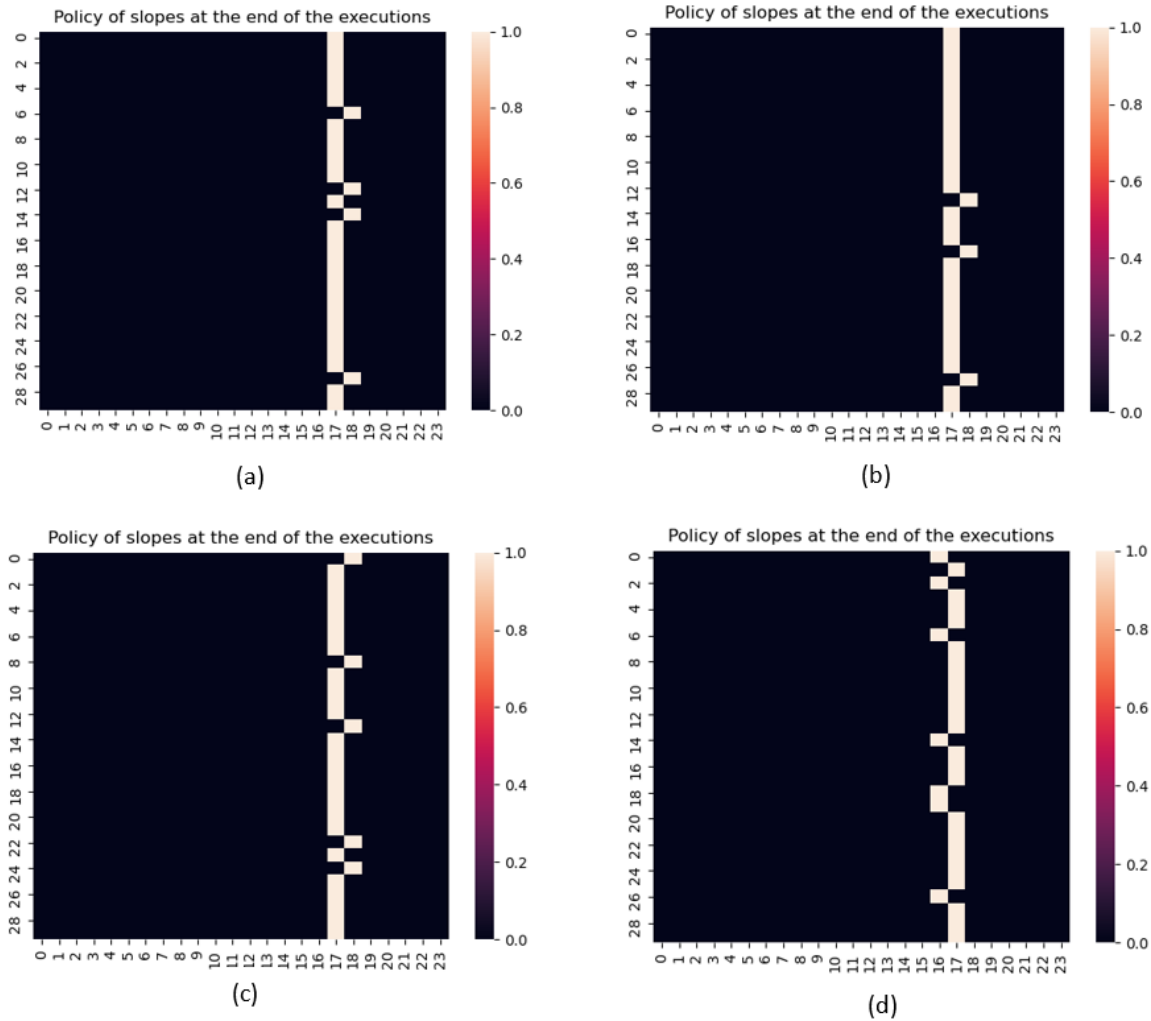


Figure 4.11 – Cartes thermiques des vecteurs stochastiques à la fin de chaque exécution avec  $pe = 0.35$ . (a), (b), (c) et (d) représentent respectivement les cas suivants : *base\_pur*, *freq\_dur*, *update\_mean\_var\_interv* et *update\_mean\_interv*. L'axe des x est l'indice de la pente, l'axe des y est le nombre d'exécutions.

cette rapidité de convergence s'obtient au prix d'une faible récompense à la fin de l'apprentissage (en particulier pour l'utilisateur avec une mémoire à court terme pour lequel l'état de l'utilisateur dépend principalement du présent). Cela peut s'expliquer par le fait que le principe de la *durée maximale dynamique* des actions est d'arrêter les pentes (principalement les pentes douces, car le signal baisse lentement pour celles-ci) avant que l'utilisateur n'intervienne, ce qui signifie que les pentes raides seront jouées un peu plus souvent. Le fait de jouer des pentes raides déclenche plus de réactions de la part de l'utilisateur (probablement encore plus souvent lorsque l'utilisateur a une courte mémoire). En revanche, ce phénomène ne représente qu'une gêne assez légère, car dans tous les cas, l'utilisateur est maintenu dans un état proche de son état le plus favorable ( $m = 41$  au pire sachant que  $m_0 = 35$ , cela représente 10% de la valeur maximale possible pour  $m$ ) et l'énergie utilisée pour ce faire est moindre pour toutes les règles de la durée maximale dynamique des actions. Nous voyons bien dans la figure 4.10 que ces conclusions sont les mêmes pour tous les types d'utilisateurs ( $pe = 0.05$ ,  $pe = 0.35$  et  $pe = 0.95$ ). Il est également possible de voir sur les cartes de chaleur associées à l'utilisateur ayant une mémoire à moyen terme (dans la figure 4.11) que l'algorithme converge vers la meilleure pente (la

penne 17 dans la figure 4.4) lorsque la durée maximale dynamique des actions est utilisée. Un autre inconvénient relativement léger de la durée maximale dynamique des actions est la convergence exacte vers l'action optimale : nous observons que le cas de base (celui dans lequel la durée des actions n'est pas limitée) converge plus souvent vers l'action optimale par rapport au cas où on fait varier la durée maximale des actions. Nous pouvons donc conclure que contrôler la durée maximale des actions permet d'augmenter considérablement la vitesse de convergence de l'algorithme à un coût très faible.

### 4.2.3 Évaluation du système de contrôle

Nous commençons l'évaluation du système de contrôle en étudiant la discrétisation des actions dans le système complet tel que nous l'avons fait pour le système à pentes, puis nous présentons une comparaison entre les différents algorithmes par apprentissage par renforcement que nous avons proposé et finalement, nous présentons cette même comparaison pour le cas où la durée maximale des actions est changée de manière dynamique.

#### Effet du nombre d'actions sur le système de contrôle

Dans ce qui suit, nous tâchons de voir si les phénomènes observés sur la relation entre la discrétisation de l'espace d'action et les performances de l'agent des pentes s'étendent bien au système complet. Pour ce faire, l'expérience que nous présentons reprend le même principe que l'expérience que nous avons fait dans le but d'étudier l'effet des actions dans l'agent des pentes (voir section 4.2.2). L'expérience consiste à lancer 30 fois le système de contrôle pour différents ensembles d'actions (pentes et valeurs d'arrêt) correspondant à différents espaces d'actions. Ces ensembles correspondent aux valeurs d'arrêt et aux pentes où l'écart entre deux valeurs d'arrêt successives (respectivement deux pentes successives) est de 5, 10, 15 et 20. Avant de réaliser l'expérience, les meilleurs couples (en termes de récompense) de pente et de valeurs d'arrêt ont été calculés pour chaque ensemble d'actions et pour chaque valeur de  $pe$  en les exécutant de manière répétée, puis en calculant la récompense moyenne, l'énergie moyenne et la valeur moyenne de  $m$  pendant les essais.

Étant donné que le système de contrôle est censé fixer la valeur du signal sur la valeur équilibrant le mieux entre énergie et confort de l'utilisateur, la pente n'est finalement qu'un moyen pour arriver à cette fin. De ce fait, nous nous intéresserons uniquement aux valeurs d'arrêt dans cette expérience. Nous précisons, à ce sujet, que les valeurs d'arrêt optimales en termes de récompense sont les valeurs 45, 50, 45, 40 pour les discrétisations où l'écart entre deux valeurs d'arrêt est de 5, 10, 15 et 20 respectivement.

L'algorithme utilisé est l'algorithme de la poursuite hiérarchique pour l'agent des pentes et l'algorithme de la poursuite pour l'agent des valeurs d'arrêt (voir la section 2.1.1). Le taux d'apprentissage est fixé à la valeur  $lr = 0.001$  pour les deux agents. La politique de l'agent de la poursuite hiérarchique est représentée sous forme d'arbre binaire. Chaque exécution a une durée de  $3 \times 10^6$  étapes décomposées en intervalles de taille 5000 où l'on observe la valeur moyenne de l'énergie, la récompense et la valeur de  $m$ . Nous calculons également la moyenne de ces valeurs pour toutes les exécutions. Les valeurs expérimentales sont présentées dans la figure 4.12 pour chacune des valeurs de  $pe$  (0.05, 0.35, 0.95). Notons que les discrétisations des pentes ont aussi été prises en compte

même si elles ne sont pas affichées, chaque courbe de la figure 4.12 montre les performances de l'agent des valeurs d'arrêt où la moyenne de chaque mesure de performances est prise sur toutes les discrétisations de l'agent des pentes.

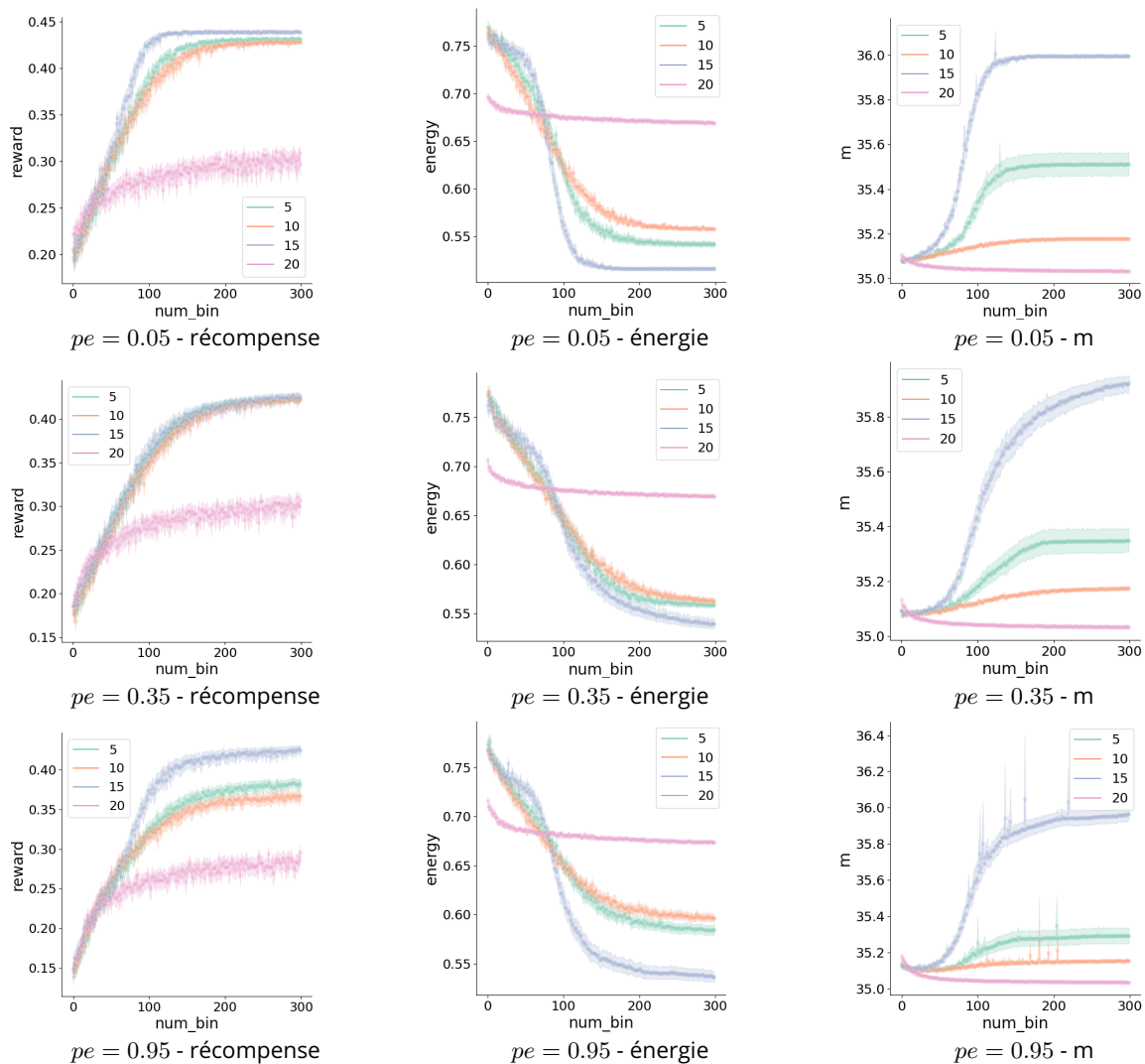


Figure 4.12 - L'évolution de la récompense (à gauche), de l'énergie (au milieu), et de l'état de l'utilisateur (à droite) pour les ensembles de valeurs d'arrêt avec un écart inter-valeurs de 5, 10, 15 et 20.

Dans la figure 4.12, chaque ligne correspond au résultat de l'expérience pour différentes valeurs de  $pe$  (0.05, 0.35, 0.95). La colonne de gauche correspond à l'évolution dans le temps de la récompense pour toutes les valeurs de  $pe$ . La colonne du milieu correspond à l'évolution de l'énergie pour différentes valeurs de  $pe$ . Quant à la colonne de droite, celle-ci correspond à l'évolution de la valeur de  $m$ .

Comme pour l'expérience sur les pentes, nous constatons tout d'abord que la récompense et l'énergie du système s'améliorent continuellement durant l'apprentissage, contrairement à l'état de l'utilisateur qui ne s'améliore pas toujours (précisons néanmoins que l'utilisateur est dans un état de confort, car la valeur maximale que prend le paramètre  $m$  est à 3 % de sa valeur maximale possible). Nous observons aussi que les performances du système en termes de récompense et de temps de convergence ne varient pas linéairement avec l'ensemble des actions. Cette observation vaut pour tous les

types d'utilisateurs, nous observons, par exemple, que pour l'utilisateur ayant une mémoire à long terme et l'utilisateur ayant une mémoire à court terme, les meilleures discrétisations en termes de récompense, sont, dans l'ordre décroissant, celles où l'écart inter-valeurs est de 15, 5, 10, 20. Le système complet ne semble pas exhiber de relations simples entre, d'un côté, les caractéristiques de l'espace d'action et de la mémoire de l'utilisateur, et de l'autre, ses performances en termes de récompense et de temps de convergence.

Concernant l'utilisateur ayant une mémoire à moyen terme ( $pe = 0.35$ ), nous constatons que le meilleur ensemble d'actions pour ce type d'utilisateur est obtenu lorsque l'écart entre les actions successives est de 15. Nous pouvons observer que les discrétisations associées à l'écart 10 et 20 possèdent différents comportements : la première est bonne en performance énergétique et relativement mauvaise en confort de l'utilisateur et la deuxième est relativement mauvaise en consommation énergétique et bonne en confort de l'utilisateur. Ici aussi, une autre observation que nous faisons est que même lorsque les actions optimales sont les mêmes pour différents ensembles d'actions (par exemple les discrétisations en 5 et en 15 ont des actions optimales similaires), le comportement de l'utilisateur et le comportement de l'algorithme changent qualitativement.

Pour l'utilisateur ayant une mémoire à long terme ( $pe = 0.05$ ), nous observons les mêmes tendances que pour l'utilisateur ayant une mémoire à moyen terme. Le meilleur ensemble d'actions correspond à l'ensemble où l'écart entre deux pentes successives est de 15, les performances se dégradent à mesure qu'on s'éloigne de cette discrétisation. Encore une fois, nous observons différents comportements pour des ensembles d'action ayant des récompenses similaires. Pour l'utilisateur ayant une mémoire à long terme, cela concerne les discrétisations correspondant aux écarts 15 et 10. La consommation d'énergie est bien plus faible pour le cas où l'écart entre les pentes est de 15 par rapport au cas où l'écart est de 10, mais la relation inverse est retrouvée concernant l'état de l'utilisateur. Des différences qualitatives concernant la consommation d'énergie du système ainsi que l'évolution de l'état de l'utilisateur se retrouvent dans les deux meilleures discrétisations (en 5 et en 15) alors que la meilleure valeur d'arrêt est la même pour les deux.

Concernant l'utilisateur ayant une mémoire à court terme ( $pe = 0.95$ ), nous observons les mêmes tendances que pour les deux premiers types d'utilisateurs. Certaines discrétisations sont bonnes en énergie, mais permettent d'assurer des conditions qui sont légèrement moins confortables pour l'utilisateur (la discrétisation où l'écart est de 5 et celle où l'écart est de 10) et d'autres où c'est le contraire (la discrétisation en 20). Nous observons aussi des comportements qualitativement différents pour les discrétisations ayant la même action optimale. Contrairement au système des pentes, le système des valeurs d'arrêt semble montrer plus de similitude entre l'utilisateur dont la mémoire est à court terme et l'utilisateur dont la mémoire est à long terme.

De la même manière que pour l'agent des pentes, une interprétation possible des observations mentionnées précédemment est qu'il y a un compromis à prendre en compte dans les différentes discrétisations de l'espace d'actions. Si la discrétisation choisie est trop éparse, l'algorithme risque de converger vers une action non optimale, car l'action optimale ne figurera pas parmi les actions disponibles pour l'agent. L'apprentissage pourrait ne converger sur aucune action en conséquence à l'instabilité de la valeur de  $m$  (l'état de l'utilisateur) qu'entraînerait cette discrétisation. A contrario,

si les actions sont trop nombreuses, l'algorithme pourrait mettre énormément de temps avant de converger. Encore une fois, l'ensemble d'actions qui est pris peut affecter la convergence de l'algorithme, même si les actions optimales sont les mêmes pour différents ensembles d'actions. Les méthodes hiérarchiques peuvent aussi être préconisées dans le cas du système des valeurs d'arrêt, car elles permettraient de gérer efficacement les espace d'actions de grandes tailles en zoomant rapidement sur les bonnes actions.

En conclusion, nous pouvons dire que l'ensemble d'actions devra être choisi avec une grande attention afin d'assurer un bon compromis entre vitesse de convergence et qualité de la solution trouvée. En outre, le nombre d'actions influence non seulement la convergence de l'algorithme, mais aussi le comportement de l'utilisateur et la consommation énergétique du système.

Dans les sous-sections qui suivent, nous comparons justement les performances des différents algorithmes d'apprentissage par renforcement pour le système complet, nous sélectionnerons ainsi les algorithmes les plus performants pour notre problématique et testerons la validité de l'hypothèse énoncée précédemment sur les performances des algorithmes hiérarchiques.

#### **4.2.4 Comparaison des algorithmes d'apprentissage par renforcement sur le système complet**

Nous comparons les performances des différents algorithmes d'apprentissage par renforcement présentés auparavant. Nous commençons par présenter cette comparaison dans le cas où la durée maximale des actions est infinie, et dans le cas où celle-ci varie au cours du temps.

Les comparaisons effectuées dans cette partie ne portent pas sur tous les couples d'algorithmes possibles entre l'agent des pentes et des valeurs. Procéder à une telle comparaison aurait pris beaucoup trop de temps. Dans ce qui suit, nous sélectionnons, pour l'agent des pentes, le meilleur algorithme de l'expérience de comparaison des algorithmes de l'agent des pentes (qui est la poursuite hiérarchique), puis nous comparons les performances du système complet pour différents algorithmes d'apprentissage par renforcement sans états pour l'agent des valeurs d'arrêt.

### **4.3 Comparaison des algorithmes avec durée maximale non limitée**

Les algorithmes pris pour l'agent des pentes dans le système complet, sont les mêmes que ceux sélectionnés précédemment dans l'étude de l'agent des pentes. Ces algorithmes sont l'algorithme du LRI (Linear Reward Inaction), l'algorithme du LRP (Linear Reward Penalty), l'algorithme de la poursuite, l'algorithme de la poursuite hiérarchique, les trois variantes de l'algorithme de  $\epsilon$ -greedy, l'algorithme Exp3 ainsi que l'algorithme de UCB (Upper Confidence Bound). Nous détaillons dans ce qui suit l'expérience permettant d'effectuer cette comparaison et aussi l'analyse de celle-ci.

Les algorithmes sont testés pendant  $4 \times 10^6$  étapes (subdivisés en intervalles de taille 5000

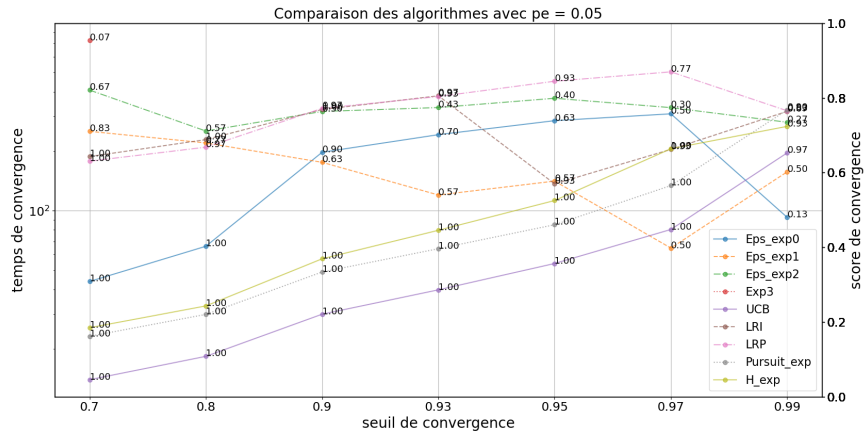
étapes) pour 30 exécutions pour différents ensembles d'hyperparamètres. L'expérience est réalisée selon le même schéma que l'expérience de comparaison des algorithmes pour l'agent des pentes de la section 4.2.2, nous sélectionnons pour chaque algorithme et pour chaque seuil les meilleures hyperparamètres afin de pouvoir comparer les performances des algorithmes. L'objectif est le même qu'auparavant, nous voulons comparer la vitesse de convergence des algorithmes pour un ensemble de seuils de convergence, tout en prenant en considération la proportion d'algorithmes qui convergent. Nous précisons que dans cette expérience, la durée maximale des actions n'est pas limitée. Par ailleurs, l'algorithme de l'agent des pentes est l'algorithme de la poursuite hiérarchique dans lequel la politique est représentée en arbre binaire et où le taux d'apprentissage à une valeur de 0.025.

Chaque paramétrisation de l'algorithme est évaluée à l'aide du même score utilisé dans l'étude de l'agent des pentes et qui est constitué de trois valeurs : la FCVO de la paramétrisation, la TMC de cette même paramétrisation suivie de la FMVO. Pour chaque algorithme, les meilleures valeurs des hyperparamètres sont choisies selon exactement le même principe que l'expérience de comparaison pour l'agent des pentes, en commençant par sélectionner les algorithmes suivant la FCVO, puis le TMC et ensuite la FMVO.

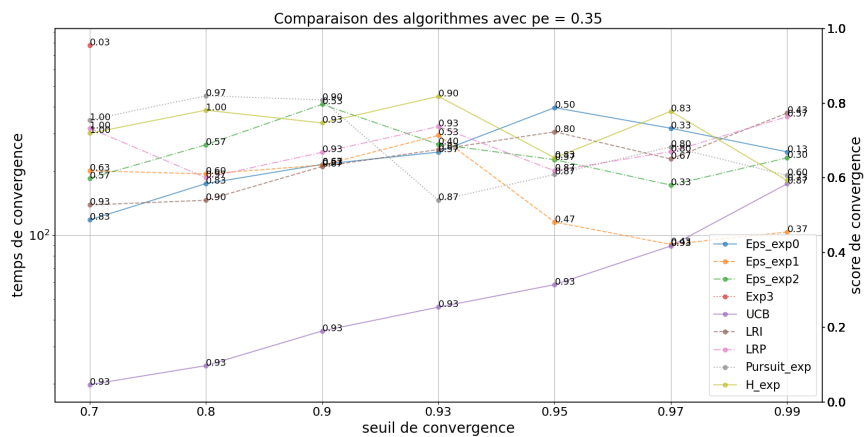
Les performances des algorithmes sont évaluées pour toutes les valeurs 0.7, 0.8, 0.9, 0.93, 0.95, 0.97 et 0.99 les résultats sont présentés dans la figure 4.13. La figure 4.13 montre une comparaison sur la vitesse de convergence (mesurée par le TMC) des algorithmes mentionnés auparavant pour les valeurs de  $pe$  considérées (usager à long terme avec  $pe = 0.05$ , usager à moyen terme avec  $pe = 0.35$ , et usager à court terme avec  $pe = 0.95$  de gauche à droite). Les temps de convergence des algorithmes sont étiquetés par la valeur de la FCVO correspondant au paramétrage choisi pour chacun des algorithmes. L'axe des abscisses représente les seuils de convergence et l'axe des ordonnées représente le nombre d'intervalles nécessaires à chaque algorithme pour converger vers le seuil correspondant. Les valeurs sur l'axe des ordonnées sont affichées en échelle logarithmique pour une bonne visualisation. L'axe des ordonnées à droite de la figure correspond aux valeurs de la FCVO, cet axe est affiché en échelle linéaire. Les algorithmes qui n'ont convergé pour aucune des valeurs testées des hyperparamètres n'ont pas été inclus dans la figure. Les légendes des différentes variantes des algorithmes Epsilon-Greedy sont notées  $Eps\_exp0$ ,  $Eps\_exp1$  et  $Eps\_exp2$  et correspondent au même ordre que celui dans lequel elles ont été présentées précédemment. La légende  $H\_exp$  est associée à l'algorithme de poursuite hiérarchique. Toutes les autres légendes sont explicites. Dans le tableau 4.2, nous montrons les valeurs des hyperparamètres testés pour chaque algorithme.

Contrairement à l'agent des pentes, nous constatons, dans cette expérience, qu'il y a globalement quatre types d'algorithmes en termes de vitesse de convergence, ceux qui prennent beaucoup de temps à converger ou ne convergent pas du tout (Exp3), ceux dont la convergence est lente, mais fréquente (LRI, LRP, et algorithmes de la poursuite et de la poursuite hiérarchique), ceux dont la vitesse de convergence ainsi que la qualité de convergence est erratique (Epsilon-Greedy et ses variantes), ceux qui convergent très bien sur tous les types d'utilisateurs (UCB). Contrairement au cas dans lequel on avait seulement l'agent des pentes, le cas où la mémoire est à long terme ( $pe = 0.05$ ) semble être le plus simple en termes de vitesse de convergence et de qualité de convergence. Nous pouvons voir cela en remarquant que les temps de convergence de la majorité des algorithmes sont au-dessus de 100 pour  $pe = 0.35$  et  $pe = 0.95$  (contrairement au cas  $pe = 0.05$ ) et que la FCVO

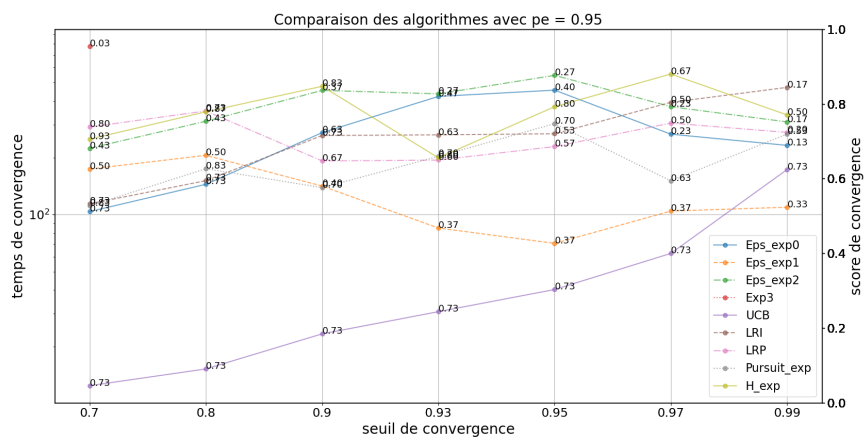




(a) Comparaison avec  $pe = 0.05$ .



(b) Comparaison avec  $pe = 0.35$ .



(c) Comparaison avec  $pe = 0.95$ .

Figure 4.13 – Comparaison des algorithmes d'apprentissage par renforcement pour l'agent des valeurs d'arrêt en termes de vitesse de convergence.

atteint 1 quand  $pe = 0.05$  pour quatre des algorithmes testés (contrairement au cas  $pe = 0.35$  et  $pe = 0.95$ ). Dans ce qui suit, nous proposons des hypothèses et des interprétations pouvant expliquer les observations portant sur les performances des algorithmes. Ces interprétations devront

être vérifiées par des expériences supplémentaires.

Comme la pente est inconnue de l'agent des valeurs d'arrêt, à chaque itération, le choix de la valeur d'arrêt est fait pour des pentes très différentes, qui auraient des effets différents sur le comportement de l'utilisateur. Ces variations au niveau des effets de l'utilisateur font que la stratégie qui devrait être jouée peut varier au cours du temps. Par exemple, l'agent des valeurs d'arrêt pourrait préférer des valeurs d'arrêt trop élevées, parce qu'auparavant, l'agent des pentes avait choisi des pentes trop raides qui avaient amené l'utilisateur à être beaucoup trop réactif sur le système. Puisque la mémoire de l'utilisateur change lentement quand  $pe$  est faible, les réactions de l'utilisateur ne vont pas changer considérablement si la stratégie de l'agent change, ce qui rendrait l'apprentissage plus lisse en rendant l'effet de la pente sur l'utilisateur plus cohérent avec l'effet de la valeur d'arrêt sur celui-ci.

L'algorithme de Exp3 semble être l'algorithme le moins performant pour l'agent des valeurs d'arrêt. L'hypothèse de présence d'adversaire sous-jacente à l'algorithme de Exp3 ne semble pas tenir pour le système complet.

Les mêmes conclusions sur la convergence de Epsilon-Greedy (toutes variantes confondues) tirées pour l'agent des pentes s'appliquent ici aussi pour le système complet. Pour certains paramétrages de Epsilon-Greedy, l'algorithme converge rapidement, mais très peu souvent, et pour d'autres paramétrages, il converge lentement, mais de manière assez stable (d'où la forme en dents de scie des performances des variantes de Epsilon-Greedy). L'algorithme Epsilon-Greedy, ainsi que ses variantes, semblent être difficile à paramétrer.

Les algorithmes de LRI et LRP convergent également assez lentement, mais de manière assez stable pour tous les types d'utilisateur, comme le montrent les scores de convergence. Pour plus de détails d'interprétations concernant ce résultat, veuillez vous référer à la section 4.2.2.

Contrairement au cas où l'on a seulement l'agent des pentes, les algorithmes de poursuite et de poursuite hiérarchique convergent très lentement (ces algorithmes sont plus rapides que LRI et LRP uniquement pour le cas  $pe = 0.05$ ). Néanmoins, ces deux algorithmes semblent montrer la même stabilité de convergence que LRI et LRP. Puisque la pente est inconnue de l'agent des valeurs d'arrêt, les mises à jour des estimations des récompenses associées aux valeurs d'arrêt sont effectuées indépendamment de la pente qui a été choisie. Les algorithmes de la poursuite et de la poursuite hiérarchique souffriraient alors de biais (au moins au début de l'apprentissage) les rendant aussi lent que LRI et LRP. De plus, l'algorithme de la poursuite hiérarchique semble être même plus lent que l'algorithme de la poursuite. Le biais, déjà présent dans les valeurs estimées des actions, semble dégrader encore plus les performances de l'algorithme de la poursuite hiérarchique vu que la sélection d'action y est très similaire à une recherche binaire. Néanmoins, la stabilité de convergence des algorithmes de poursuite et de poursuite hiérarchique prouve que ce biais s'estompe durant l'apprentissage, l'agent acquiert de l'expérience en interagissant avec son environnement et se base sur celle-ci pour corriger ses estimations et converger vers la meilleure action.

La grande différence qu'on peut observer sur les performances de l'agent des valeurs d'arrêt en comparaison avec les performances de l'agent des pentes se situe au niveau de l'algorithme de UCB.

En effet, l'algorithme de UCB est clairement le meilleur algorithme pour tous les types d'usager. Ici, l'effet de l'homogénéisation de la mémoire ne peut être invoqué seul, car les performances (en temps et en qualité de convergence) de UCB sont excellentes quelle que soit la mémoire de l'usager. Une analyse poussée sur le comportement de l'agent des pentes et des valeurs d'arrêt est nécessaire pour expliquer la supériorité de UCB en termes de performances. Toutefois, nous pouvons déjà proposer quelques pistes d'interprétations à ce phénomène. Déjà, la présence de biais au niveau des valeurs des actions semble inconséquente, car UCB prend en compte l'incertitude des estimations dans la sélection des actions, en plus des estimations elles-mêmes. De plus, comme il est dit dans la sous-section 4.1.4, la pente a moins d'importance sur les réactions de l'usager que la valeur d'arrêt. De ce fait, si la durée de maintien de la valeur d'arrêt est nettement supérieure au temps qu'il faut pour l'atteindre, c'est la valeur d'arrêt qui joue le plus grand rôle dans l'effet d'homogénéisation de la mémoire et qui va déterminer le plus le comportement de l'usager. Nous croyons que ces raisons amènent UCB à être plus performant que les autres algorithmes qui ont été testés.

Dans cette sous-section, nous avons comparé le comportement de différents algorithmes d'apprentissage par renforcement pour le système complet. Nous en avons conclu que l'algorithme de UCB est le meilleur pour l'agent des valeurs d'arrêt. Aussi, l'algorithme de poursuite hiérarchique et de poursuite semblent avoir de moins bonnes performances. Néanmoins, nous pouvons améliorer ces performances si nous modifions l'algorithme de poursuite et de poursuite hiérarchiques en le combinant avec UCB ou en prenant des éléments de l'agent des pentes afin de les intégrer aux estimations des actions.

## Conclusion

Dans cette partie, nous avons proposé une conception du système se basant sur le paradigme par choix de variation du signal. Dans ce paradigme, nous avons un agent qui choisit la pente et un agent qui choisit la valeur d'arrêt. Nous avons présenté une étude approfondie sur l'agent des pentes ainsi que sur le système complet (l'agent des pentes et l'agent des valeurs d'arrêt) en comparant les performances de différents algorithmes d'apprentissage par renforcement. À cet égard, les conclusions divergent en fonction du type de système considéré. Pour l'agent des pentes, le meilleur algorithme (en termes de vitesse de convergence et de qualité de convergence) semble être celui basé sur la poursuite hiérarchique. Nous jugeons ce résultat naturel étant donné que l'état de l'usager change continuellement quand la valeur du signal baisse. En effet, nous postulons que les performances des méthodes de type poursuite sont plus stables en raison du fait qu'elle repose sur l'optimisation locale. L'optimisation locale a l'avantage d'être assez robuste, même quand les hypothèses de l'environnement ne correspondent pas exactement aux cas pour lesquelles ces algorithmes ont été conçus. En ce qui concerne l'agent des pentes, nous avons aussi proposé un mécanisme permettant d'améliorer la vitesse de convergence en faisant varier la durée maximale des actions.

Concernant l'agent des valeurs d'arrêt, UCB semble donner de meilleurs résultats. Sachant que les expériences précédentes montrent que la valeur d'arrêt importe davantage que la pente sur le comportement de l'usager, nous pouvons déjà postuler des causes amenant UCB à être plus efficace

que les autres algorithmes. Quand l'agent commence à apprendre les stratégies pour lesquelles l'utilisateur réagit peu, la valeur d'arrêt choisie par le système sera inchangée pendant longtemps. Quand cette valeur est fixée, la mémoire s'homogénéise et l'environnement s'apparente à un environnement stationnaire.

Nous avons aussi étudié l'effet de la discrétisation sur les performances du système à pente et du système complet. Cette étude a montré qu'il y a un compromis à trouver entre le nombre d'actions d'un côté et la qualité de convergence ainsi que la vitesse de convergence de l'autre. Les expériences menées à cet effet montrent clairement qu'il est difficile de paramétrer le système d'apprentissage de façon à trouver ce compromis. De plus, le nombre d'actions influence non seulement la convergence de l'algorithme, mais aussi le comportement de l'utilisateur et la consommation énergétique du système. Notons que ces conclusions tiennent uniquement dans le cas que nous avons considéré et doivent être étudiées dans le cas où la luminosité extérieure change en intégrant également l'activité des utilisateurs.

# Chapitre 5

## Modèle par choix de valeur du signal

Nous avons étudié dans le chapitre 4 un type de modèle pour l'algorithme de contrôle énergétique de bâtiment que nous avons nommé *modèle par variation du signal*. Dans ce modèle, l'idée est que l'agent apprenne à choisir, parmi un ensemble prédéfini de politiques, celle équilibrant le mieux entre énergie et confort. La caractéristique principale de ce modèle est que la politique de baisse du signal recouvre plusieurs *étapes*. Dans ce chapitre, nous étudions le paradigme consistant à ce que la décision du système ne porte que sur l'étape courante. Dans ce modèle, le système apprend à choisir les valeurs du signal étape par étape de façon à minimiser l'énergie et satisfaire l'utilisateur. Les réactions de l'utilisateur dépendant effectivement de l'historique du signal, le système devra avoir une base sur laquelle il effectuera son choix. Cette base servira à guider le système vers un comportement énergétique adéquat. Pour ce faire, la valeur courante du signal ainsi que la sensibilité de l'utilisateur seront des éléments clé reflétant respectivement la consommation d'énergie et l'état de l'utilisateur.

Compte tenu des différents types d'apprentissage par renforcement, le modèle semblant être le plus adéquat dans ce contexte est l'apprentissage par renforcement à base d'états. Dans ce modèle, l'agent interagit avec l'environnement et cette interaction le conduit petit à petit à prendre les choix qu'il faut. Le modèle devra maximiser un signal cumulatif portant sur le présent, mais aussi sur le futur. Contrairement au modèle précédent, ici aussi la récompense ne recouvre qu'une seule étape (et pas toute la baisse du signal comme pour le modèle précédent). Tout comme dans le modèle précédent, les efforts de conception doivent porter sur cette notion de récompense qui reste difficile à formaliser vu le compromis entre performance du système et confort de l'utilisateur.

L'état est constitué de toute mesure représentant un élément situationnel sur la base duquel l'agent apprend à agir dans l'environnement. Vu le rôle prépondérant que prend l'état dans le comportement de l'agent, sa conception devra être faite avec soin. Dans le cadre de notre problème, l'état réel inclut la sensibilité de l'utilisateur (qui fait partie de l'environnement), mais celle-ci est inconnue du système. Par conséquent, il faudra trouver des éléments qui sont corrélés avec l'état et qui pourraient permettre d'identifier le comportement de l'utilisateur sans que son état réel soit complètement connu du système. À ce propos, l'état inclut la valeur courante du signal (considéré comme étant un proxy pour l'énergie consommée par le système) ainsi que la fréquence d'intervention de l'utilisateur (considéré comme étant un proxy pour la sensibilité de l'utilisateur).

Quant à la décision du système, elle porte sur la valeur suivante que devra prendre le signal.

Remarquons que dans une optique de conception réaliste du système, nous proposons que l'action soit un incrément par rapport à la valeur courante du signal. Ainsi, le système pourra uniquement prendre une valeur qui soit dans le voisinage de la valeur courante. Nous allons détailler tous ces éléments dans les sous-sections concernées. Pour une vision globale du fonctionnement du système tel qu'abordé dans ce chapitre, veuillez vous référer au schéma de la figure 5.1.

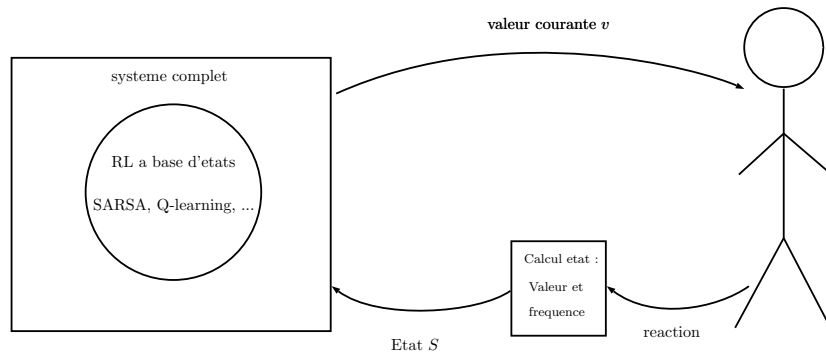


Figure 5.1 – Graphique représentant la réaction de l'utilisateur et son impact énergétique.

## 5.1 Formalisme

Dans cette sous-section, nous présentons les différentes composantes de notre système. Nous commençons par présenter les composantes de l'état, puis nous passons à l'action pour finir avec la politique de l'agent. Dans le système par choix de valeur du signal, l'ensemble des états est l'ensemble des couples constitués de la valeur courante du signal et de la fréquence d'intervention de l'utilisateur. L'action est représentée par un incrément sur la valeur actuelle du signal menant à une valeur dans son voisinage. Elle se traduit globalement par trois possibilités d'évolution du signal : soit la valeur du signal monte, soit elle descend, soit elle reste inchangée. À chaque fois que l'agent applique une action, l'agent n'est plus autorisé à incrémenter (ou décrémenter) le signal pendant une certaine durée. Contrairement au modèle par choix de variation du signal, l'effet d'une action change la valeur du signal par palier. La décision de l'agent ne porte pas sur la manière de faire varier le signal, l'agent décide essentiellement s'il faut changer la valeur du signal ou non. Dans ce qui suit, nous associons à chaque étape l'action qui a été appliquée, et ce même si l'action est maintenue pendant plus d'une étape. Ceci vaut également pour l'état et la récompense. La table 5.1 synthétise l'ensemble des notations utilisées dans ce chapitre.

### 5.1.1 La représentation de l'état

Comme nous l'avons dit auparavant, l'état fait intervenir la valeur courante du signal ainsi que la fréquence d'intervention de l'utilisateur. Toutefois, étant donné que les valeurs possibles du signal peuvent être nombreuses et que la fréquence d'intervention est continue, l'état ne peut être représenté en prenant naïvement les couples de toutes les valeurs du signal et de toutes les fréquences d'intervention. Pour cette raison, nous proposons d'*agréger* [116] les valeurs du signal ainsi que les valeurs de la fréquence d'intervention dans la représentation de l'état. De cette façon, les états qui sont assez proches seront considérés comme étant équivalents par l'agent, dans le sens où l'agent peut agir de la même façon pour tous ces états sans que cela nuise aux performances du système. Ce

$s_{min}$	La valeur minimale du signal.
$s_{max}$	La valeur maximale du signal.
$V$	L'ensemble des valeurs possibles du signal (l'intervalle d'entiers $[[s_{min}, s_{max}]]$ ).
$N_v$	Le nombre de sous-intervalles de valeurs du signal dans $V$ .
$FI$	L'intervalle des fréquences d'intervention (l'intervalle $[0, 1]$ ).
$N_f$	Le nombre de sous-intervalles de $FI$ .
$S$	L'espace d'état défini $[[0, N_v - 1]] \times [[0, N_f - 1]]$ .
$iv_n$	L'indice du sous-intervalle courant de valeurs du signal (à l'étape courante $n$ ).
$f_n$	L'estimation à l'étape $n$ de la fréquence d'intervention de l'utilisateur.
$if_n$	L'indice du sous-intervalle de l'estimation actuelle de la fréquence d'intervention de l'utilisateur.
$S_n$	L'espace à l'étape courante $n$ (le couple $(iv_n, if_n)$ ).
$A_n$	L'action choisie par l'agent à l'étape $n$ .
$chose\_val$	La fonction utilisée par l'agent pour choisir la valeur courante dans le sous-intervalle courant (de valeurs du signal).
$I_n$	Variable booléenne qui vaut 1 si l'utilisateur intervient à l'étape courante et qui vaut 0 sinon.
$dur\_sa$	La durée pendant laquelle l'agent reste sur le même sous-intervalle qu'il a choisi après application de l'action (la durée de maintien de l'action).
$val\_chos\_sys_n$	La valeur choisie par le système à l'étape $n$ .
$v_n$	La valeur effective que prend le signal à l'étape $n$ .
$ren_n$	La récompense due à l'énergie reçue à l'étape $n$ après application de l'action $A_n$ .
$r_{cs_n}$	La récompense associée au confort de l'utilisateur reçue à l'étape $n$ après application de l'action $A_n$ .
$r_n$	La récompense totale reçue à l'étape $n$ après application de l'action $A_n$ .
$\lambda_{intr}$	Le poids associé au confort de l'utilisateur dans le calcul de la récompense.
$Q$	La table contenant les estimations des valeurs des actions pour les différents états de l'environnement.
$\gamma$	Le facteur de <i>discount</i> .
$upd\_each\_time$	Variable booléenne valant 1 si la table $Q$ est mise à jour à chaque étape et 0 sinon.
$on\_policy$	Variable booléenne valant 1 si l'algorithme utilisée est SARSA et 0 si l'algorithme utilisée est Q-learning.
$Apply\_ac$	La fonction permettant à l'agent d'appliquer une action.
$RL\_func$	La fonction permettant d'appliquer une action.

Table 5.1 – Liste des principales notations concernant les algorithmes et notions du modèle par choix des valeurs du signal.

type de représentation constitue l'une des manières les plus simples permettant de gérer les espaces d'états de grande dimension et de mitiger le *fléau de dimensionnalité* ([116]). Nous présentons tous ces points de manière formelle dans les paragraphes qui suivent.

L'ensemble des valeurs possibles du signal noté  $V$  est représenté par un ensemble d'entiers compris entre deux valeurs  $s_{min}$  et  $s_{max}$  (un intervalle d'entier) s'écrivant  $V = [[s_{min}, s_{max}]]$  (un exemple est donné dans la figure 5.4). Comme nous l'avons mentionné, afin de gérer l'explosion combinatoire des états, nous subdivisons cet intervalle en plusieurs sous-intervalles. En considérant  $N_v \in \mathbb{N}$  sous-intervalles, nous aurons,

$$V = [[s_{min}, \frac{s_{max} - s_{min}}{N_v}]] \cup \cup_{i=1}^{N_v-1} [\frac{(s_{max} - s_{min})i}{N_v}, \frac{(s_{max} - s_{min})(i+1)}{N_v}]$$

De même que pour l'intervalle des valeurs du signal, l'intervalle des fréquences d'intervention  $FI = [0, 1]$  est subdivisé en plusieurs sous-intervalles qui sont au nombre de  $N_f$ . Formellement, ceci donne,

$$FI = [0, \frac{1}{N_f}] \cup \cup_{i=1}^{N_f-1} [\frac{i}{N_f}, \frac{(i+1)}{N_f}]$$

L'espace d'état est représenté par le produit des indices des sous-intervalles de  $V$  et de  $FI$  et qui s'écrit  $S = [[0, N_v - 1]] \times [[0, N_f - 1]]$ . L'état de l'environnement à l'étape  $n$  est noté  $S_n = (iv_n, if_n)$  où  $iv_n$  correspond à l'indice du sous-intervalle de la valeur courante du signal et  $if_n$  correspond à l'indice du sous-intervalle de l'estimation actuelle de la fréquence d'intervention de l'utilisateur. L'environnement transitionne d'un état à l'autre à chaque fois que l'agent choisit une valeur du signal et l'applique. L'indice du sous-intervalle de cette nouvelle valeur est alors calculé par la simulation et retourné à l'agent. De même, pour la fréquence moyenne d'intervention, celle-ci est mise à jour comme suit :

$$f_{n+1} \leftarrow f_n + \frac{1}{n+1}(I_n - f_n)$$

La simulation retourne ensuite l'indice du sous-intervalle associé à cette nouvelle valeur de la fréquence d'intervention. À ce niveau, l'état a effectivement finalisé sa transition de  $S_n = (iv_n, if_n)$  vers  $S_{n+1} = (iv_{n+1}, if_{n+1})$ . L'exemple montré dans la figure 5.4 illustre comment le système fait évoluer la valeur du signal. L'évolution de l'état peut être retrouvée partant de cet exemple en suivant l'évolution de  $iv_n$  (l'évolution de  $if_n$  suit le même principe que celle de  $iv_n$ ). Après avoir dit quelques mots sur l'environnement, nous abordons dans ce qui suit les éléments concernant l'agent, et qui sont l'action puis la politique.

### 5.1.2 L'action

Comme nous l'avons cité auparavant, l'action est représentée sous forme d'incrément dans l'espace des valeurs (effectué dans le voisinage de la valeur courante). Nous proposons d'exécuter ces incréments sur les sous-intervalles au lieu des valeurs, afin de réduire la dimensionnalité des paires d'états-action. L'action du système serait de décider si la valeur suivante du système serait dans le sous-intervalle courant, dans le sous-intervalle immédiatement supérieur au sous-intervalle courant ou alors dans le sous-intervalle immédiatement inférieur au sous-intervalle courant. De ce fait, à partir d'un sous-intervalle donné, l'état ne peut transitionner que vers un sous-intervalle qui est adjacent au premier (si l'utilisateur n'intervient pas). Dans notre modélisation de l'action (et de l'état) la magnitude des changements de valeurs sera toujours bornée, ceci a pour effet d'empêcher des changements dans la valeur du signal qui risquerait d'altérer fortement le confort de l'utilisateur. En dehors de la question sur *le fléau de dimensionnalité*, ce dernier point représente un avantage principal de notre modélisation.

L'ensemble des actions dont dispose l'agent dépend de l'indice du sous-intervalle courant. Si  $iv_n \in [[0, N_v - 1]]$  est l'indice de sous-intervalle courant, nous notons l'ensemble des actions accessibles à l'agent quand le signal fait partie de ce sous-intervalle, par  $\mathcal{A}(iv_n)$ . Formellement, nous avons :

$$\mathcal{A}(iv_n) = \begin{cases} \{0, 1\}, & \text{si } iv_n = 0 \\ \{-1, 0, 1\}, & \text{si } 0 < iv_n < N_v - 1 \\ \{0, -1\}, & \text{si } iv_n = N_v - 1 \end{cases}$$

En notant  $A_n$  l'action que l'agent prend à l'étape  $n$ , l'indice du sous-intervalle associé à la prochaine valeur est alors  $iv_n + A_n$  si l'utilisateur n'intervient pas, et cet indice est  $N_v - 1$  sinon. Nous considérons aussi que l'agent dispose d'une fonction *chose\_val()* qui a pour entrée l'indice  $iv_n + A_n$  du sous-intervalle choisi par l'algorithme et qui retourne une valeur parmi les valeurs possibles de ce sous-intervalle. Plusieurs politiques peuvent être proposées à cet égard : prendre une valeur type du sous-intervalle (la valeur minimale, maximale ou médiane) ou alors prendre une valeur aléatoire dans ce sous-intervalle. La politique de choix de valeur est un paramètre fixé avant l'apprentissage et il ne change pas au cours du temps. Après le choix d'une valeur du signal, l'utilisateur peut décider d'intervenir ou de ne pas intervenir. Le choix de l'intervention de l'utilisateur est représenté par une variable  $I_n$  valant 1 si l'utilisateur intervient à l'instant  $n$  et 0 sinon.

Dans notre modèle, il y a aussi un autre paramètre *dur\_sa* (qui signifie *duration of same action*) qui représente la durée pendant laquelle l'agent maintient la même action en cas d'absence de retour de la part de l'utilisateur. Quand l'agent choisit le sous-intervalle de valeur d'indice  $iv_n + A_n$ , l'agent est



forcé de rester sur le même intervalle pendant  $dur\_sa$  étapes. Bien entendu, si l'utilisateur intervient, l'agent choisit une action et le compteur associé à cette durée est remis à zéro. La durée  $dur\_sa$  doit être non nulle et très inférieure à la durée totale allouée à l'apprentissage. Le paramètre de durée de l'action ne change pas durant l'apprentissage et il est fixé avant que l'apprentissage ne commence. Pour plus de détails sur la sélection et l'exécution des actions, veuillez vous référer à l'exemple de la figure 5.4 (présenté plus loin).

### 5.1.3 La récompense

La qualité d'une action  $A_n$  appliquée à l'étape  $n$  dépend de l'énergie consommée et de la réactivité de l'utilisateur pendant l'application de cette action. Ainsi, la récompense aura deux parties, une partie reflétant la consommation de l'énergie et une autre reflétant le confort de l'utilisateur. Nous définissons, dans ce qui suit, l'expression de la récompense en apportant quelques précisions concernant notre modèle.

Nous notons  $val\_chos\_sys_n$  la valeur choisie par le système à l'étape  $n$ . Cette valeur est potentiellement différente de  $v_n \in V$  qui est la valeur que prend le signal après que l'utilisateur ait décidé s'il intervient ou non. Après que  $I_n$  ait pris sa valeur, nous aurons alors  $v_n = s_{max}$  si l'utilisateur intervient et  $v_n = val\_chos\_sys_n$  sinon. Dans ce cas, la partie énergie de la récompense de l'action  $A_n$  prise à l'étape  $n$  :

$$ren_n = 1 - \frac{v_{n-1} + val\_chos\_sys_n - 2s_{min}}{2(s_{max} - s_{min})}$$

Le terme  $1 - ren_n$  représente l'aire sous le signal durant l'action  $A_n$ , normalisée entre 0 et 1. Et  $ren_n$  représente l'économie d'énergie réalisée durant l'application de l'action  $A_n$ . La partie confort de la récompense est l'opposé de l'intervention de l'utilisateur  $I_n$ , elle s'écrit

$$rcs_n = 1 - I_n$$

La récompense est alors formée à partir de la partie associée à l'énergie auquel on rajoute la partie associée au confort de l'utilisateur multipliée par le facteur  $\lambda_{intr}$ . Le facteur  $\lambda_{intr}$  est un paramètre qui est fixe durant l'apprentissage. La forme générale de la récompense est donc

$$r_n = ren_n + \lambda_{intr}rcs_n$$

Nous montrerons comment fixer la valeur du paramètre  $\lambda_{intr}$  dans une section ultérieure. Dans la prochaine sous-section, nous montrons la représentation de la politique de l'agent. Nous indiquons aussi comment nous avons repris les algorithmes d'apprentissage par renforcement, Q-learning [130] et SARSA [116] pour les adapter à notre problème.

### 5.1.4 Politique de l'agent et apprentissage

Comme nous l'avons indiqué auparavant, nous utilisons les méthodes basées sur les valeurs : Q-learning et SARSA pour le contrôle du signal lumineux. Pour ces méthodes, la politique est représentée implicitement à l'aide des fonctions de valeur des paires d'états-action. L'agent maintient une table dans laquelle chaque entrée correspond à la valeur estimée de l'agent pour une paire

d'état-action. Ces estimations changent au cours du temps, nous notons cette table par  $Q$ . Dans la table  $Q$ , l'entrée  $Q(s, a)$  désigne l'estimation de l'agent de l'action  $a$  à l'état  $s$ .

Dans notre système, l'agent choisit l'action suivant une politique  $\epsilon$ -greedy sur les valeurs estimées des actions. Nous notons la valeur du paramètre d'exploration de la politique par  $\epsilon$ . Le taux d'apprentissage à l'étape  $n$  est notée  $\alpha$ . Concernant la mise à jour des estimations de  $Q$ , nous avons deux options possibles : mettre à jour les estimations de l'agent à chaque étape du contrôle que nous appelons **le cas continu**, ou alors effectuer cette mise à jour uniquement pour les étapes dans lesquelles la durée de maintien de l'action s'achève (quand l'utilisateur intervient ou après  $dur\_sa$  étapes sans retour de l'utilisateur) que nous appelons **le cas différé**. Ces deux cas correspondent à la variable  $upd\_each\_time$  qui vaut 1 pour le cas continu et 0 pour le cas différé. La variable  $on\_policy$  sert à paramétrer le choix de l'algorithme d'apprentissage par renforcement, sa valeur est Vrai si SARSA (voir l'algorithme 3) est utilisée et faux si Q-learning (voir l'algorithme 4) est utilisé. Dans ce qui suit, nous présentons la fonction  $Apply\_ac()$  et la procédure  $RL\_func()$  servant à implémenter les algorithmes d'apprentissage par renforcement. La fonction  $Apply\_ac()$  et la procédure  $RL\_func()$  sont schématisées dans les figures 5.2 et 5.3 respectivement.

Pour chaque étape  $n$ , l'agent choisit une action  $A_n$  qui sera appliquée dans l'environnement par appel de la fonction  $Apply\_ac()$ . La fonction  $Apply\_ac()$  est détaillée dans l'algorithme 8. La fonction  $Apply\_ac()$  a en entrée l'état courant  $S$ , l'étape  $st$  durant laquelle l'action courante débute ainsi que l'action choisie par l'agent  $A_{st}$  et la durée maximale de maintien de l'action  $dur\_sa$ . La fonction retourne l'état atteint après l'application de l'action ainsi que la récompense cumulative obtenue et la durée effective de l'action (ces deux éléments servent à mettre à jour la table  $Q_n$ ). L'état sur lequel l'action  $A_{st}$  est appliquée est stocké dans la variable  $BS$ . Pour chaque étape  $n$  avec  $st \leq n < st + dur\_sa$ , l'agent choisit la valeur courante qui est retournée par la fonction  $chose\_val()$ . L'agent reçoit ensuite l'état suivant, la récompense, la valeur suivante et  $interv$  (qui est une variable booléenne valant Vrai si l'utilisateur intervient à l'étape courante et qui vaut faux sinon). Ensuite, l'agent calcule la récompense cumulée associée à l'état courant (qui est contenue dans la variable  $cra$ ). Dans le cas où la mise à jour de la table des estimations est faite de manière continue, l'estimation de l'action  $A_{st}$  appliquée à l'état  $BS$  est alors mise à jour. La mise à jour de  $Q$  correspond à celle utilisée dans SARSA si  $on\_policy$  vaut Vrai et à Q-learning sinon. Après cela, la variable  $n$  est incrémentée et nous passons à l'état suivant. Afin de présenter les algorithmes de la manière la plus générale possible, la dernière mise à jour de la table  $Q$  sera effectuée dans la fonction  $RL\_func()$  en utilisant la récompense cumulée contenue dans la variable  $cra$ . Nous reviendrons sur ce point dans le paragraphe suivant.

La procédure  $RL\_func()$  (montrée dans l'algorithme 9) est générique. Elle implémente Q-learning et SARSA suivant la valeur de  $on\_policy$ . Si la variable  $on\_policy$  vaut Vrai alors la procédure  $RL\_func()$  correspond à l'algorithme SARSA et si elle est à faux alors cette même procédure correspond à l'algorithme Q-learning. La procédure  $RL\_func()$  a en entrée l'état initial, les estimations initiales, la variable  $on\_policy$  ainsi que la durée maximale de maintien de l'action. Dans la procédure  $RL\_func()$ , l'agent commence par initialiser la variable d'état  $S$ , l'étape initiale  $n$  ainsi que la première action. Dans la boucle **while** de la procédure  $RL\_func()$ , l'agent fait un appel à la fonction  $Apply\_ac()$  lui permettant d'exécuter l'action qui a été choisie à la fin de l'itération précédente (ou durant l'étape d'initialisation). Si la variable  $on\_policy$  vaut vrai, l'agent

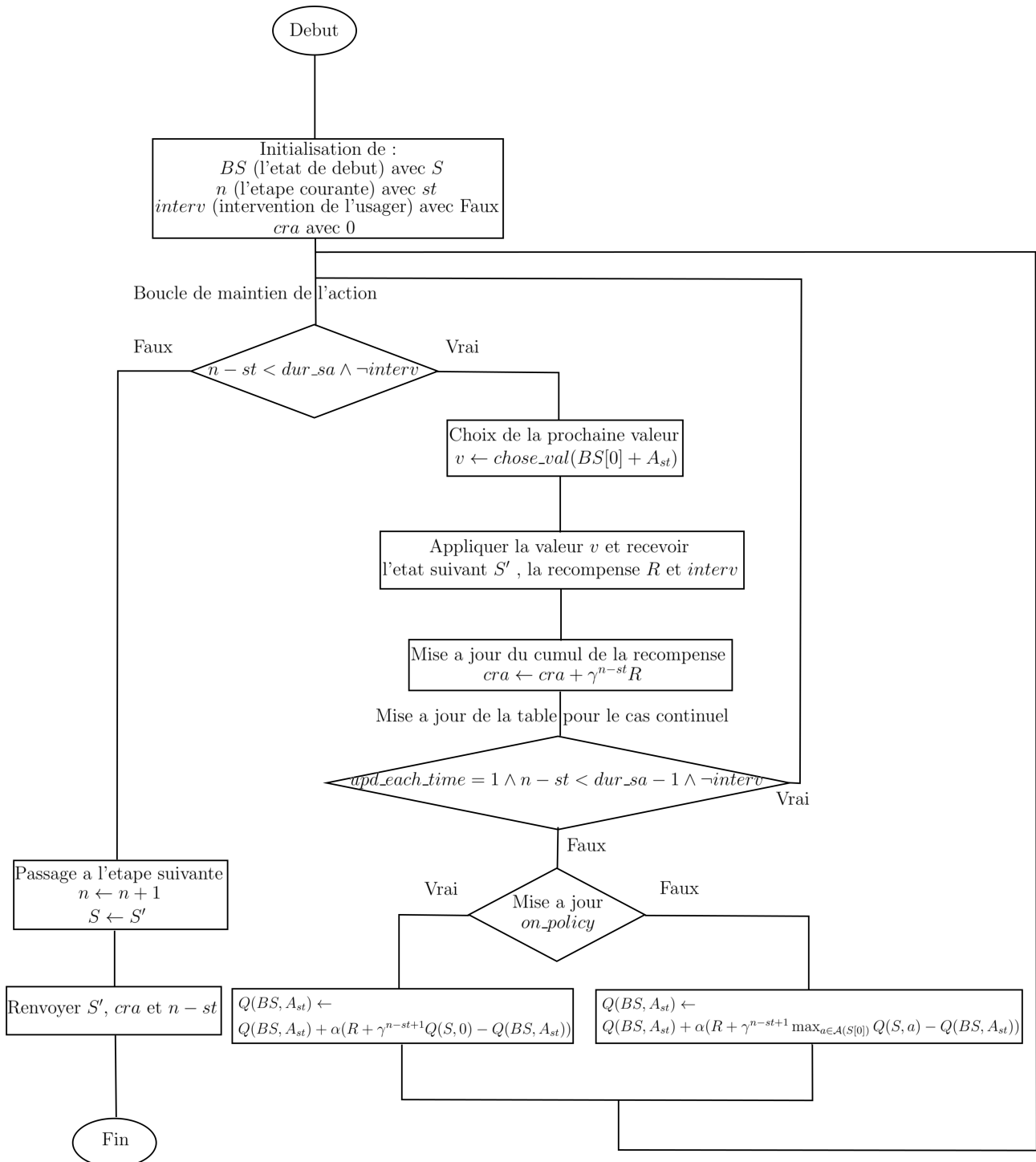


Figure 5.2 – Graphique représentant le fonctionnement de la fonction *Apply\_ac()*

choisit l'action  $A'$  qui sera exécutée à l'itération suivante, l'estimation  $Q(S, A)$  est alors mise à jour de la même manière que dans SARSA. En revanche, si *on\_policy* est à faux, l'estimation  $Q(S, A)$  est mise à jour comme dans l'algorithme Q-learning, et l'action  $A'$  qui sera exécutée dans l'itération suivante n'est choisi qu'après cette mise à jour. Finalement, l'agent met à jour l'état courant  $S$ , l'action courante  $A$  ainsi que l'étape  $n$  où cette action sera prise.

La figure 5.4 montre un exemple du déroulé de l'algorithme de contrôle basé sur le choix des valeurs du signal que nous avons proposé. Dans cet exemple, l'axe des abscisses représente le temps

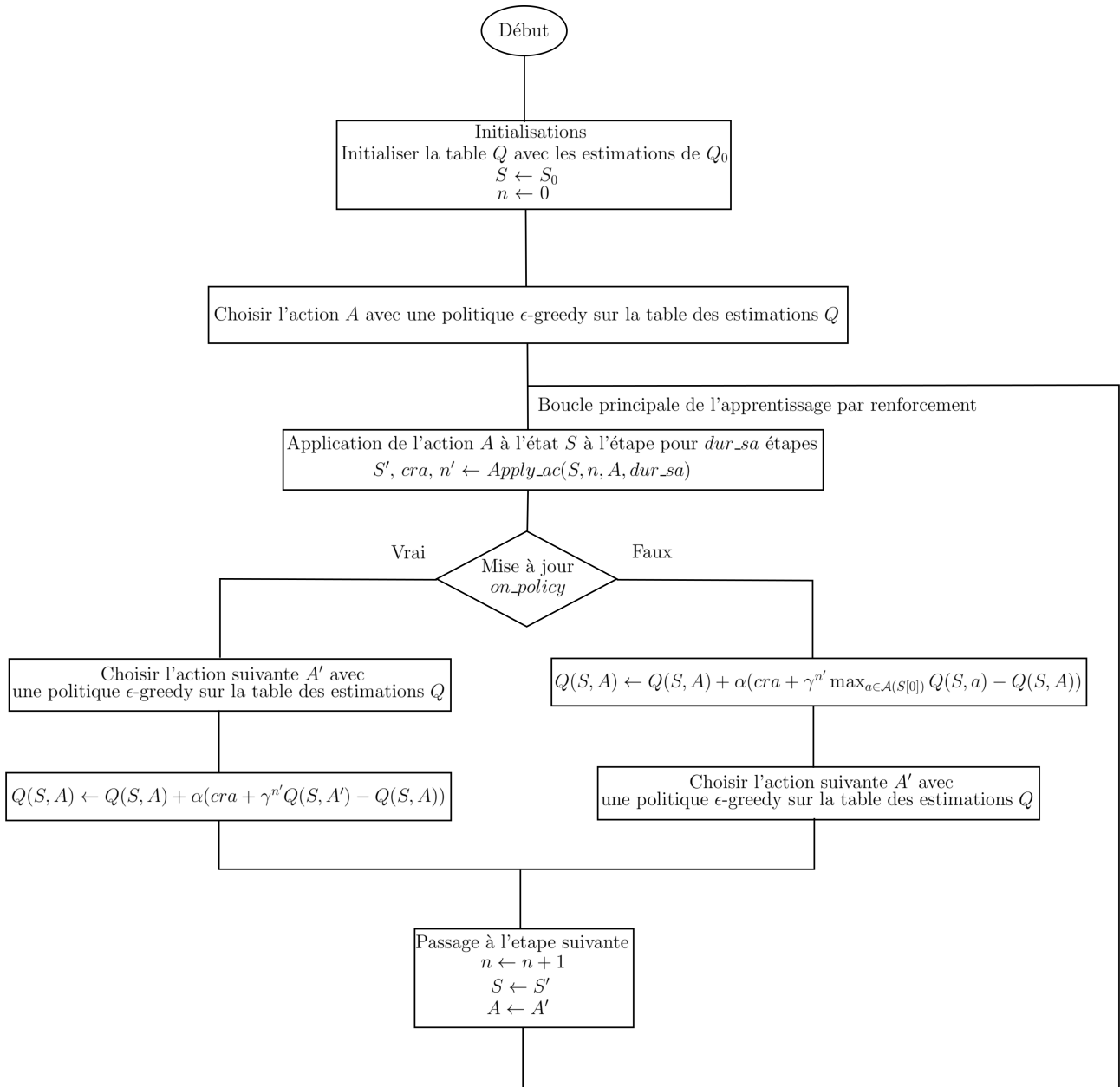


Figure 5.3 – Graphique représentant le fonctionnement de la fonction  $RL\_func()$

(en étapes), l'axe principal des ordonnées représente les valeurs du signal et l'axe secondaire des ordonnées (à droite de la figure) représente les indices des sous-intervalles. Les valeurs minimale et maximale du signal sont respectivement 0 et 100 ( $s_{min} = 0$  et  $s_{max} = 100$ ). Le nombre de sous-intervalles est de 10 ( $N_v = 10$ ). Comme précisé auparavant, le sous-intervalle où se trouve la valeur du signal à l'étape  $n$  est noté  $iv_n$ . Dans l'exemple de la figure 5.4, ces sous-intervalles sont indexés de 0 à 9. Tel que mentionné auparavant, à chaque étape  $n$ , l'agent a trois actions possibles : choisir le sous-intervalle au-dessus du sous-intervalle courant ( $A_n = 1$ ), choisir le sous-intervalle au-dessous du sous-intervalle courant ( $A_n = -1$ ) ou rester sur le même sous-intervalle ( $A_n = 0$ ). À chaque fois que le système choisit de faire baisser la valeur du signal, le signal est maintenu pendant  $dur\_sa = 3$  étapes consécutives après avoir été baissée. Dans l'exemple de la figure 5.4, la valeur initiale du signal est  $v_0 = s_{max}$  ce qui correspond au sous-intervalle  $iv_0 = 9$ . Le système décide alors de faire baisser le signal ( $A_0 = -1$ ). Par conséquent, le prochain sous-intervalle est  $iv_1 = iv_0 + A_0 = 9 - 1 = 8$  et

---

**Algorithme 8** Implémentation de la fonction *Apply\_ac()*.

---

**Inputs :**  $S$  - L'état courant.

$st$  - L'étape de début d'application de l'action.

$A_{st}$  - L'action choisie par l'agent devant être appliquée.

$on\_policy$  - Variable booléenne servant à paramétrer le choix de l'algorithme (vrai pour SARSA et faux pour Q-learning).

$dur\_sa$  - La durée maximale de l'application de l'action.

```
1:  $BS \leftarrow S$ 
2:  $n \leftarrow st$ 
3:  $interv \leftarrow False$ 
4:  $cra \leftarrow 0$ 
5: ▷ Boucle de maintien de l'action
6: while  $n - st < dur\_sa \wedge \neg interv$  do
7:    $v \leftarrow chose\_val(BS[0] + A_{st})$ 
8:   Appliquer la valeur  $v$  et recevoir l'état suivant  $S'$  ainsi que la récompense  $R$  et  $interv$ .
9:    $cra \leftarrow cra + \gamma^{n-st} R$ 
10:  ▷ Mise à jour de la table si la mise à jour est continue
11:  if  $upd\_each\_time = 1 \wedge n - st < dur\_sa - 1 \wedge \neg interv$  then
12:    if  $on\_policy$  then
13:       $Q(BS, A_{st}) \leftarrow Q(BS, A_{st}) + \alpha(R + \gamma^{n-st+1} Q(S, 0) - Q(BS, A_{st}))$ 
14:    else
15:       $Q(BS, A_{st}) \leftarrow Q(BS, A_{st}) + \alpha(R + \gamma^{n-st+1} \max_{a \in \mathcal{A}(S[0])} Q(S, a) - Q(BS, A_{st}))$ 
16:    end if
17:  end if
18:   $n \leftarrow n + 1$ 
19:   $S \leftarrow S'$ 
20: end while
21: return  $S', cra, n - st$ 
```

---

la prochaine valeur du signal  $v_1 = 85$  se trouve dans ce sous-intervalle. Cette valeur est maintenue pendant  $dur\_sa = 3$  étapes. À la troisième étape, l'agent décide encore une fois de baisser la valeur du signal ( $A_3 = -1$ ). L'indice du prochain sous-intervalle est donc  $iv_4 = iv_3 + A_3 = 7$  et la valeur du signal est  $v_3 = 75$ . Cette valeur est maintenue pendant  $dur\_sa = 3$  étapes. En suivant le même procédé, le système choisit la prochaine valeur du signal qui vaut 65 à partir de l'étape 6 puis 55 à partir de l'étape 9. La valeur 55 n'est pas maintenue longtemps, car l'utilisateur intervient à l'étape 11 en choisissant la valeur maximale ( $I_{11} = 1$ ). Ensuite, le système décide de rester sur le même sous-intervalle (d'indice 9), etc.

---

**Algorithme 9** Implémentation de la fonction  $RL\_func()$ .

---

**Inputs :**  $S_0$  - L'état initial.

$Q_0$  - La table contenant les estimations initiales des paires d'états-action.

$on\_policy$  - Variable booléenne servant à paramétrer le choix de l'algorithme (vrai pour SARSA et faux pour Q-learning).

$dur\_sa$  - La durée de maintien de l'action.

```
1: Initialiser la table  $Q$  avec les estimations de  $Q_0$ 
2:  $S \leftarrow S_0$ 
3:  $n \leftarrow 0$ 
4: Choisir l'action  $A$  avec une politique  $\epsilon$ -greedy sur la table des estimations  $Q$ 
5:  $\triangleright$  Boucle principale de l'apprentissage par renforcement
6: while  $True$  do
7:    $S', cra, n' \leftarrow Apply\_ac(S, n, A, dur\_sa)$ 
8:   if  $on\_policy$  then
9:     Choisir l'action suivante  $A'$  avec une politique  $\epsilon$ -greedy sur la table des estimations
        $Q$ 
10:     $Q(S, A) \leftarrow Q(S, A) + \alpha(cra + \gamma^{n'}Q(S, A') - Q(S, A))$ 
11:   else
12:     $Q(S, A) \leftarrow Q(S, A) + \alpha(cra + \gamma^{n'} \max_{a \in \mathcal{A}(S[0])} Q(S, a) - Q(S, A))$ 
13:    Choisir l'action suivante  $A'$  avec une politique  $\epsilon$ -greedy sur la table des estimations
        $Q$ 
14:   end if
15:    $n \leftarrow n'$ 
16:    $S \leftarrow S'$ 
17:    $A \leftarrow A'$ 
18: end while
```

---

### 5.1.5 Profil de l'objectif et paramétrage de la fonction de récompense

Dans cette sous-section, nous présentons le choix du paramétrage de la fonction de récompense en détaillant l'influence de la durée de maintien de l'action sur celle-ci. Pour ce faire, nous devons observer les performances du système pour différentes politiques et pour différentes valeurs de durée de maintien de l'action. Cependant, il n'est pas possible d'effectuer ces traitements en un temps raisonnable, en effet, le nombre de politiques est exponentiel en la dimensionnalité des paires d'états-action. Pour cette raison, nous nous limitons seulement à un sous-ensemble de politiques que nous jugeons représentatif des performances de l'ensemble des politiques disponibles pour l'agent. En rappelant que l'agent peut, à chaque instant où il reprend la main, baisser le signal, le remonter ou de le laisser tel quel. Les politiques que nous considérons sont celles qui consistent à faire baisser le signal avant de s'arrêter sur une zone qu'on aurait préalablement choisi. Ces politiques peuvent être identifiées par l'indice du sous-intervalle dans lequel la valeur du signal se stabilise. Dans ce qui

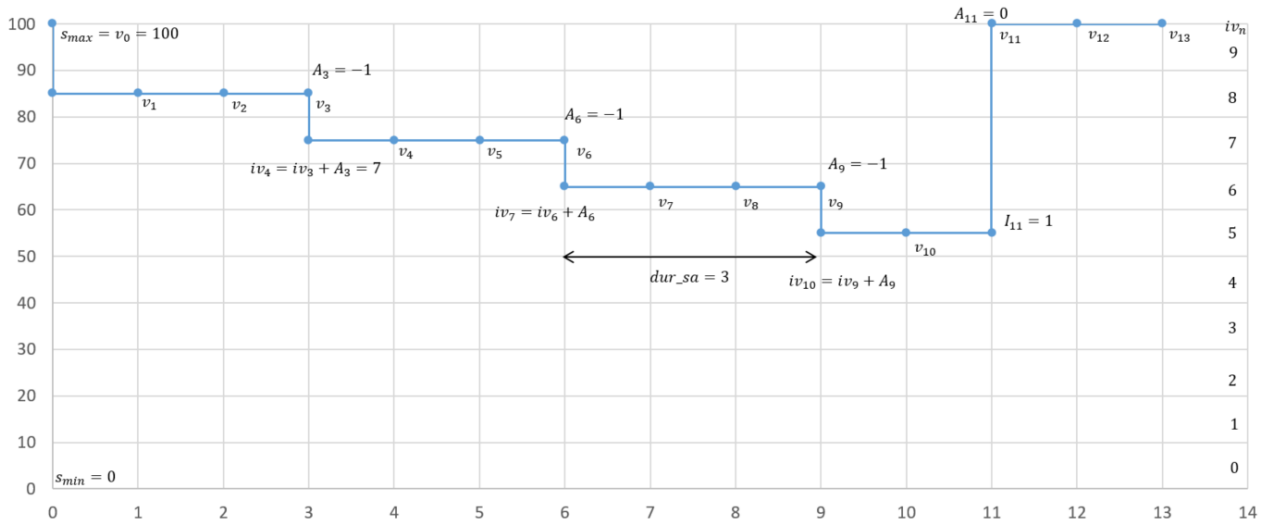


Figure 5.4 – Exemple de l’application de l’algorithme de contrôle basé sur le choix des valeurs du signal.

suit, et pour faciliter l’analyse des expériences, nous nous référons à ces sous-intervalles par *la valeur d’arrêt* de la politique qui lui correspond. Ces politiques ne représentent pas la totalité des politiques déterministes, néanmoins, mettre en place tout autre politique n’appartenant pas à cet ensemble ne semble avoir que peu d’intérêt en pratique. Un exemple de politique qu’on ne considère pas serait de faire baisser le signal (par paliers consécutifs) jusqu’à un seuil donné, et de faire varier ensuite la valeur du signal parmi un ensemble de sous-intervalles proches. Au-delà du fait que ces politiques ne sont pas raisonnables en pratique, il semble toujours possible de proposer une politique ayant une consommation énergétique semblable à la politique instable parmi les politiques considérées. Par exemple, nous pouvons faire baisser le signal pendant un moment pour ensuite s’arrêter à la valeur se trouvant au centre des sous-intervalles où évolue la politique instable.

L’expérience effectuée dans cette sous-section consiste à appliquer de manière répétée toutes les politiques considérées 30 fois pendant 10000 étapes pour différents paramétrages de la simulation. Les paramétrages impliqués dans cette expérience (ainsi que leurs valeurs) sont, le poids associé au confort  $\lambda_{intr}$  (dont les valeurs possibles sont  $\{0, 0.25, 0.75, 0.5, 0.75, 1, 1.5, 2\}$ ), la mémoire de l’usager  $pe$  (les trois types de mémoire de l’usager considérés et qui sont la mémoire à long terme avec  $pe = 0.05$ , la mémoire à moyen terme avec  $pe = 0.35$  et la mémoire à court terme avec  $pe = 0.95$ ), la nombre de sous-intervalles (11 sous-intervalles de  $s_{min} = 0$  à  $s_{max} = 90$  chacun de taille  $\frac{s_{max}-s_{min}}{11}$ ), la durée de maintien des actions (pour laquelle les valeurs considérées sont  $\{1, 5, 7\}$ ). Pour les paramètres de valeurs fixes, nous avons les paramètres initiaux de l’usager qui sont à  $a_0 = 0.1$  pour la sensibilité et  $m_0 35$  pour le seuil de confort de l’usager. Le nombre de sous-intervalles pour la fréquence d’intervention de l’usager est de 8, chacun de taille  $\frac{1}{8}$ . L’expérience a été effectuée pour les valeurs correspondant au produit de toutes les valeurs possibles de ces paramètres. Durant cette expérience, pour chaque paramétrage, nous collectons les valeurs correspondant à l’évolution au cours du temps de l’énergie, du paramètre  $m$  et de la récompense. Pour chaque paramétrage, nous calculons la moyenne de ces valeurs (l’énergie, le paramètre  $m$  et la récompense) pour chaque exécution (10000 étapes), puis nous calculons la moyenne de ces valeurs sur toutes les exécutions (30 exécutions). Le résultat de cette expérience est montrée dans les figures qui suivent (les figures

5.5, 5.6 et 5.7) pour les valeurs  $dur_{ac} = 1$ ,  $dur_{ac} = 5$  et  $dur_{sa} = 7$  respectivement. Dans le reste de cette sous-section, nous tâchons d'analyser ces figures et d'en conclure des tendances utiles relativement à la fonction de récompense ainsi qu'à l'objectif de notre système pour le modèle basé sur le choix des valeurs.

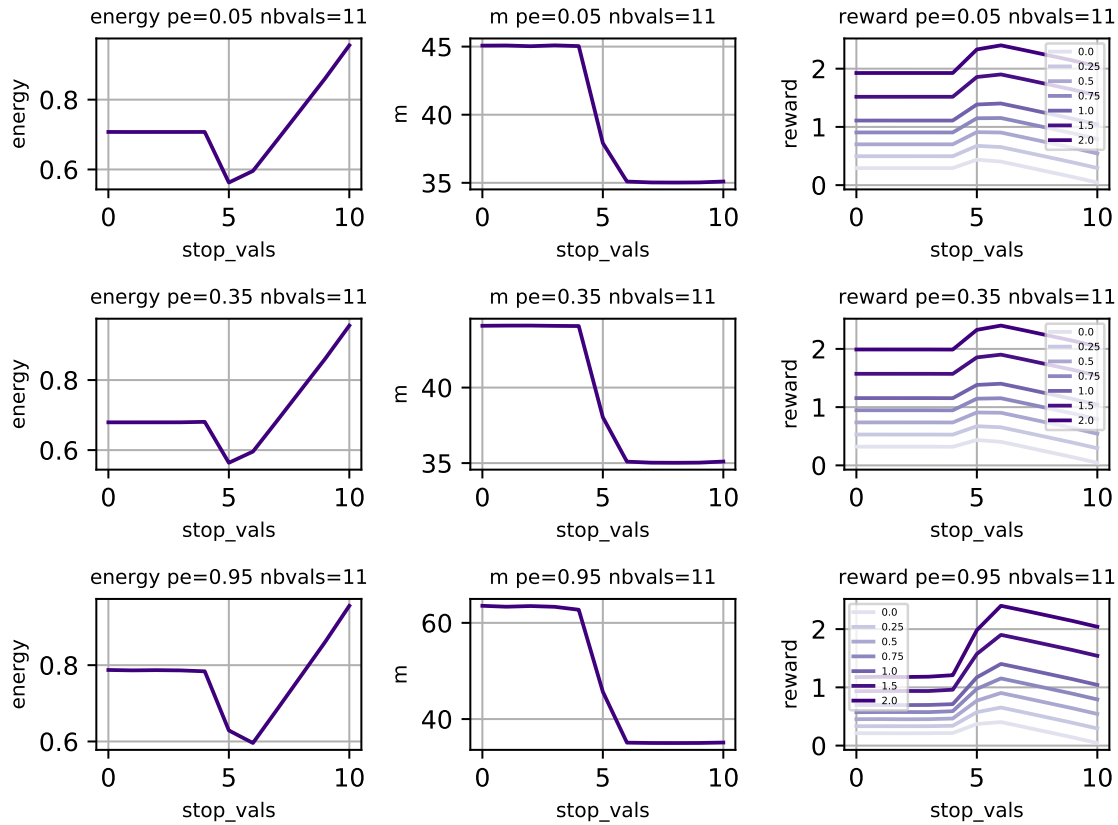


Figure 5.5 – Profil de l'utilisateur (valeur de  $m$ ), de l'énergie et la récompense pour différentes valeurs de  $\lambda_{intr}$  avec  $dur_{sa} = 1$  et  $N_v = 11$ .

Dans la figure 5.5, chaque courbe représente les valeurs moyennes de l'énergie, des valeurs de  $m$  et de la récompense pour les différentes valeurs d'arrêt possibles (axe d'abscisses) avec  $\lambda_{intr}$  fixé (les valeurs de  $\lambda_{intr}$  étant affichées au niveau de la légende). Pour ces courbes, le paramètre de maintien de l'action est fixé à 1 (cela veut dire que le système reprend la main à chaque étape pour décider de la valeur du signal). Dans la figure 5.5, les colonnes représentent l'énergie, le paramètre  $m$  et la récompense (de gauche à droite), les lignes représentent la mémoire de l'utilisateur ( $pe = 0.05$ ,  $pe = 0.35$ ,  $pe = 0.95$  de haut en bas). En observant ces courbes, nous remarquons que l'utilisateur intervient beaucoup pour les valeurs d'arrêt faibles et il intervient peu souvent pour les valeurs d'arrêt fortes. Ceci provient du fait que l'utilisateur est plus sensible quand le signal prend des valeurs faibles. Dans les courbes associées à l'énergie, nous remarquons que pour les valeurs les plus faibles, la valeur de l'énergie est stable. Dans la figure 5.5, cette région où l'énergie reste stable est suivie d'une région dans laquelle la courbe de l'énergie prend une forme creuse. La présence de cette forme creuse de la courbe s'interprète de la même manière que pour le profil énergétique du système basé sur les variations de valeurs présenté dans le chapitre 4. Les valeurs d'arrêt se trouvant à gauche de cette partie mènent le système à consommer plus d'énergie que nécessaire. Pour cette région, l'utilisateur intervient beaucoup trop sur la valeur du signal, causant ainsi le surplus énergétique observé.



Concernant les valeurs d'arrêt se trouvant dans la région à droite de l'axe des abscisses dans la figure 5.5, nous remarquons aussi une surconsommation d'énergie : le système s'arrête sur des valeurs trop fortes induisant cette forte consommation d'énergie. En ce qui concerne la partie de la courbe où l'énergie reste stable, elle l'est, car ces valeurs sont plus faibles que les valeurs pour lesquelles l'utilisateur intervient (ce qui implique que la valeur d'arrêt n'aura plus d'influence sur la consommation d'énergie si celle-ci est assez faible). Remarquons aussi que toutes les courbes induisent un minimum énergétique global unique équilibrant. En nous reposant sur l'analyse précédente, nous concluons que la valeur d'arrêt correspondant à cet optimum forme le meilleur compromis entre énergie et confort. Toutes les remarques et les conclusions faites dans ce paragraphe portent sur les trois types d'utilisateur que nous avons considérés.

Par rapport à la récompense, nous remarquons que sa forme correspond globalement à l'image miroir de la courbe du profil énergétique, et ce, pour tous les types d'utilisateur. Quand  $\lambda_{inter} = 0$ , la récompense constitue l'économie d'énergie achevée par la politique. La récompense avec  $\lambda_{inter} = 0$  devrait théoriquement suffire pour décrire un objectif avec l'optimum énergétique adéquat. Toutefois, nous observons quand même que pour cette paramétrisation de la fonction de récompense, les valeurs d'arrêt trop faibles (qui sont inconfortables pour l'utilisateur) ont une récompense plus élevée que celles qui sont confortables pour l'utilisateur. Néanmoins, nous constatons qu'en augmentant la valeur de  $\lambda_{inter}$  nous pouvons avoir un profil similaire au profil correspondant à  $\lambda_{inter} = 0$  et dans lequel les valeurs d'arrêt trop faibles ont une récompense basse comparativement aux valeurs d'arrêt fortes, et la valeur d'arrêt qui a la récompense maximale est proche de celle dont l'énergie est minimale. Pour ces raisons, nous avons jugé préférable de donner un poids important à la partie confort de la récompense. Il nous reste à indiquer et à justifier quelle valeur de  $\lambda_{inter}$  devra être prise pour le reste de nos expériences. À ce propos, notons que prendre des valeurs trop fortes pour  $\lambda_{inter}$  aura tendance à augmenter les valeurs de la fonction de récompense et donc de l'aplatir sur tout l'intervalle des valeurs d'arrêt.

Pour les figures 5.6 et 5.7 qui correspondent aux valeurs 5 et 7 de la durée de maintien de l'action, toutes les observations et conclusions faites pour le cas  $dur\_ac = 1$  s'appliquent aussi dans ces deux cas. Changer la durée de maintien des actions ne change en rien le profil des courbes sauf pour la fonction de récompense qui semble être très aplatie quand la durée de maintien des actions est longue. Compte tenu de toutes ces observations, nous décidons de prendre la valeur 2 pour le paramètre  $\lambda_{intr}$  car elle équilibre bien entre énergie et confort tout en limitant l'aplatissement de la courbe de la fonction de récompense.

Dans ce qui suit, nous affichons des courbes correspondant à notre expérience, nous permettant d'explorer un peu plus les conséquences du choix  $\lambda_{inter} = 2$  sur le système et l'utilisateur. Dans la figure 5.8 sont affichées les courbes de l'énergie, de la valeur de  $m$  et de la récompense (colonnes) associées aux valeurs d'arrêt (axe des abscisses) pour différentes mémoires de l'utilisateur (colonnes) et pour différentes valeurs du paramètre de la durée de maintien de l'action (légende). Les profils (de l'énergie, de l'état de l'utilisateur et de la récompense) restent globalement les mêmes, bien que nous observions quelques différences semblant dépendre de la longueur de maintien de la durée de l'action ainsi que de l'emplacement des valeurs d'arrêt dans les régions que nous avons considérées auparavant. Nous remarquons, en effet, que pour les valeurs d'arrêt qui sont dans la région où l'énergie est constante, l'énergie consommée est plus grande quand la durée de maintien de l'action

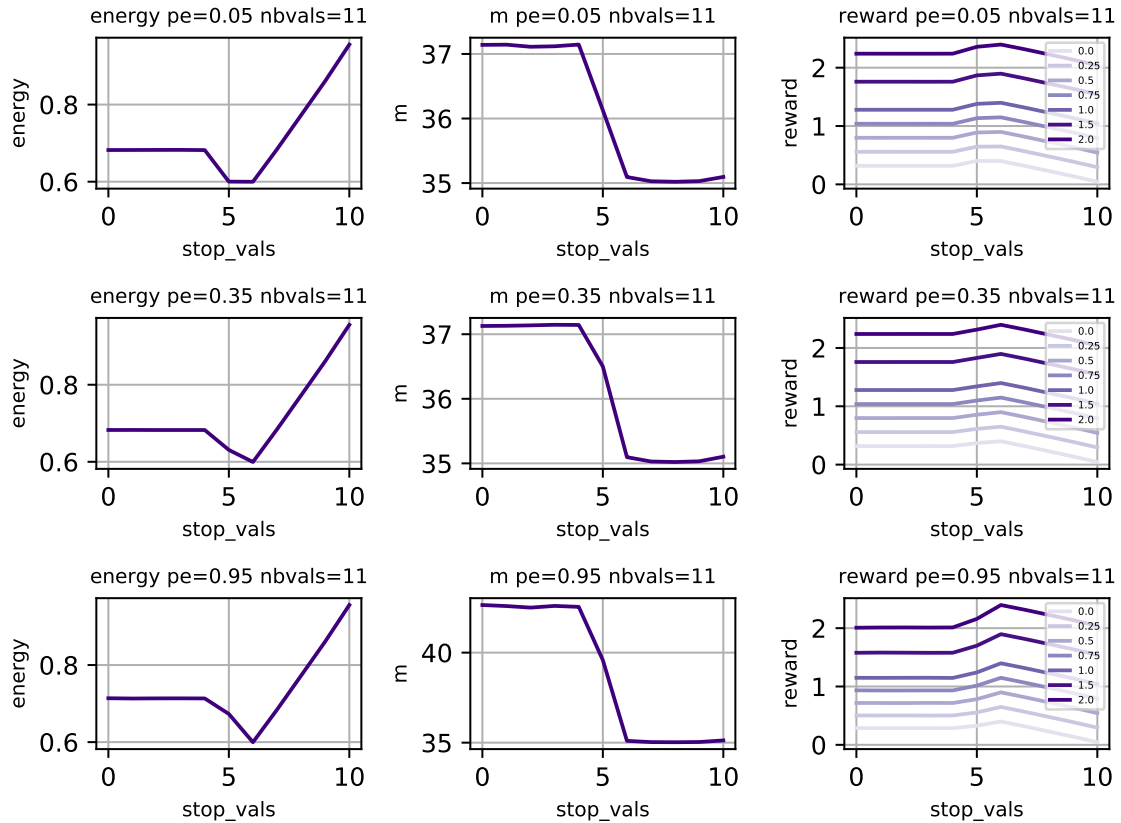


Figure 5.6 – Profil de l'utilisateur (valeur de  $m$ ), de l'énergie et la récompense pour différentes valeurs de  $\lambda_{intr}$  avec  $dur_{sa} = 5$  et  $N_v = 11$ .

est minimale ( $dur_{ac} = 1$ ). Nous remarquons que la tendance s'inverse pour les valeurs d'arrêt qui sont dans la région creuse, à gauche de la valeur d'arrêt la plus économe en énergie (pour une même valeur d'arrêt, l'énergie consommée est faible quand la durée de maintien des actions est faible). Et nous constatons finalement que pour les valeurs d'arrêt qui sont dans la région creuse et à droite de l'optimum énergétique, l'énergie consommée ne change plus en fonction de la durée de maintien des actions. Nous expliquons, dans le paragraphe qui suit, toutes ces observations concernant le lien entre le profil énergétique et la durée de maintien de l'action.

Concernant les valeurs d'arrêt appartenant à la partie constante (dans la figure 5.8), appliquer une faible durée de maintien des actions sur le contrôle du signal fait que le signal arrive plus rapidement aux valeurs faibles du signal. Ceci implique que l'utilisateur sera plus fréquemment dans un état d'inconfort que dans le cas où la durée de maintien des actions est plus grande. Par rapport aux valeurs d'arrêt se trouvant à gauche de l'optimum énergétique dans la région creuse, la consommation énergétique pourrait être plus faible quand la durée de maintien de l'action est de 1, car les durées ne sont pas assez faibles pour susciter un comportement énergétique coûteux de la part de l'utilisateur. D'un autre côté, comme le signal descend plus rapidement quand la durée de maintien des actions est faible, le système consomme moins d'énergie que quand la durée de maintien des actions est grande. Pour les valeurs d'arrêt se trouvant tout à droite, la durée de maintien des actions ne compte plus, car dans cette partie, les valeurs d'arrêt sont fortes au point où l'énergie consommée ne dépend que de la valeur d'arrêt. Elle n'est plus influencée par la durée de maintien de l'action (celle-ci impacte

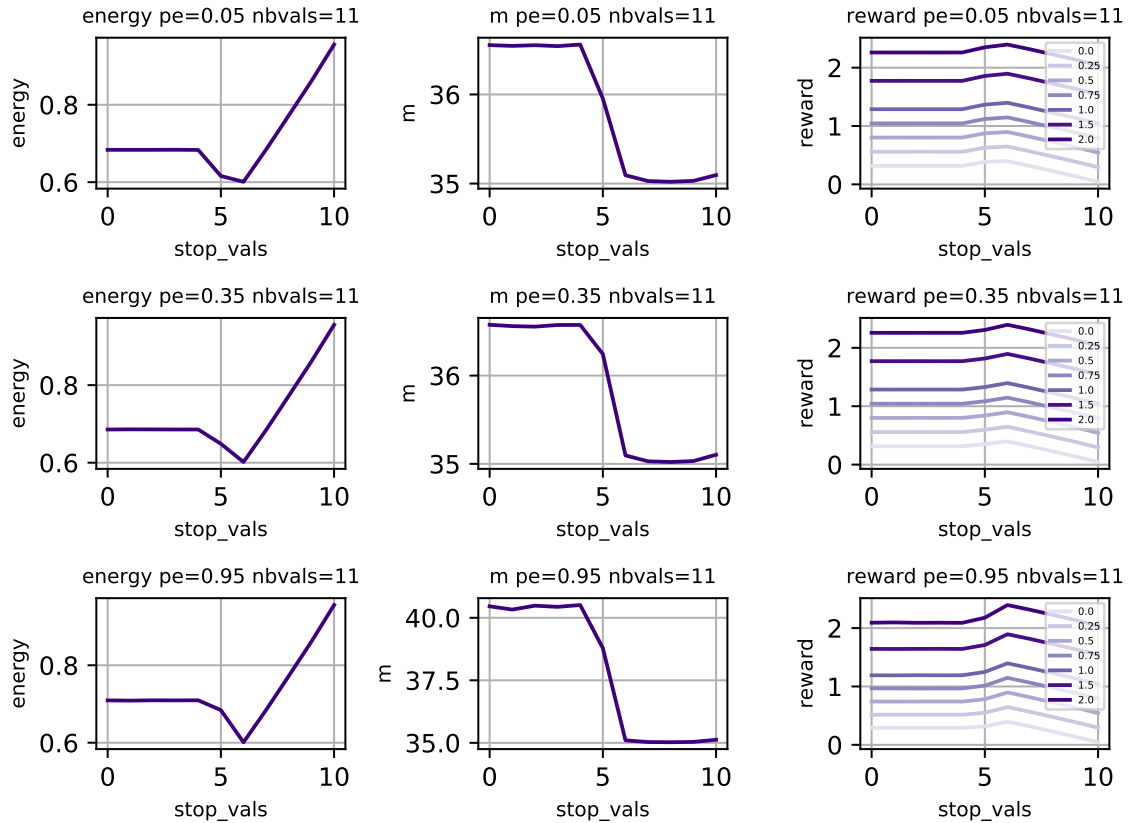


Figure 5.7 – Profil de l'utilisateur (valeur de  $m$ ), de l'énergie et la récompense pour différentes valeurs de  $\lambda_{intr}$  avec  $dur\_sa = 7$  et  $N_v = 11$ .

l'énergie uniquement pendant la baisse du signal). Les courbes sur l'état de l'utilisateur confirment les interprétations faites sur le profil énergétique des valeurs d'arrêt. Nous voyons bien que dans la partie énergétiquement stable, l'utilisateur (la valeur de  $m$ ) est plus sensible quand la durée de maintien des actions est faible et qu'il est moins sensible quand la durée de maintien des actions est grande. Nous notons aussi que, pour les valeurs d'arrêt qui sont à droite de l'axe des abscisses, l'utilisateur est dans un état de repos quelle que soit la durée de maintien des actions. La différence observée en matière de récompense pour les différentes valeurs de la durée de maintien des actions s'explique également par les différences observées dans l'état de l'utilisateur. Toutes les remarques et conclusions abordées ici sont valables pour tous les types d'utilisateur (mémoire à court, moyen et long terme).

Finalement, nous observons, à propos de la récompense, que la valeur  $\lambda_{intr} = 2$  donne le même optimum que pour toutes les valeurs de  $dur\_sa$  et pour toutes les valeurs de mémoire de l'utilisateur et que la valeur d'arrêt correspondant à cet optimum est proche de l'optimum énergétique. En conséquence, ce dernier point confirme la pertinence du choix de  $\lambda_{intr} = 2$  pour le paramétrage de la fonction de récompense.

## 5.2 Évaluation des algorithmes à états

Dans cette partie, nous présentons l'évaluation des algorithmes se basant sur le modèle par valeur. Les algorithmes retenus pour cette évaluation sont les algorithmes de SARSA et de Q-learning présentés dans la section 5.1. Ces algorithmes ont été choisis vu leur popularité et leur simplicité

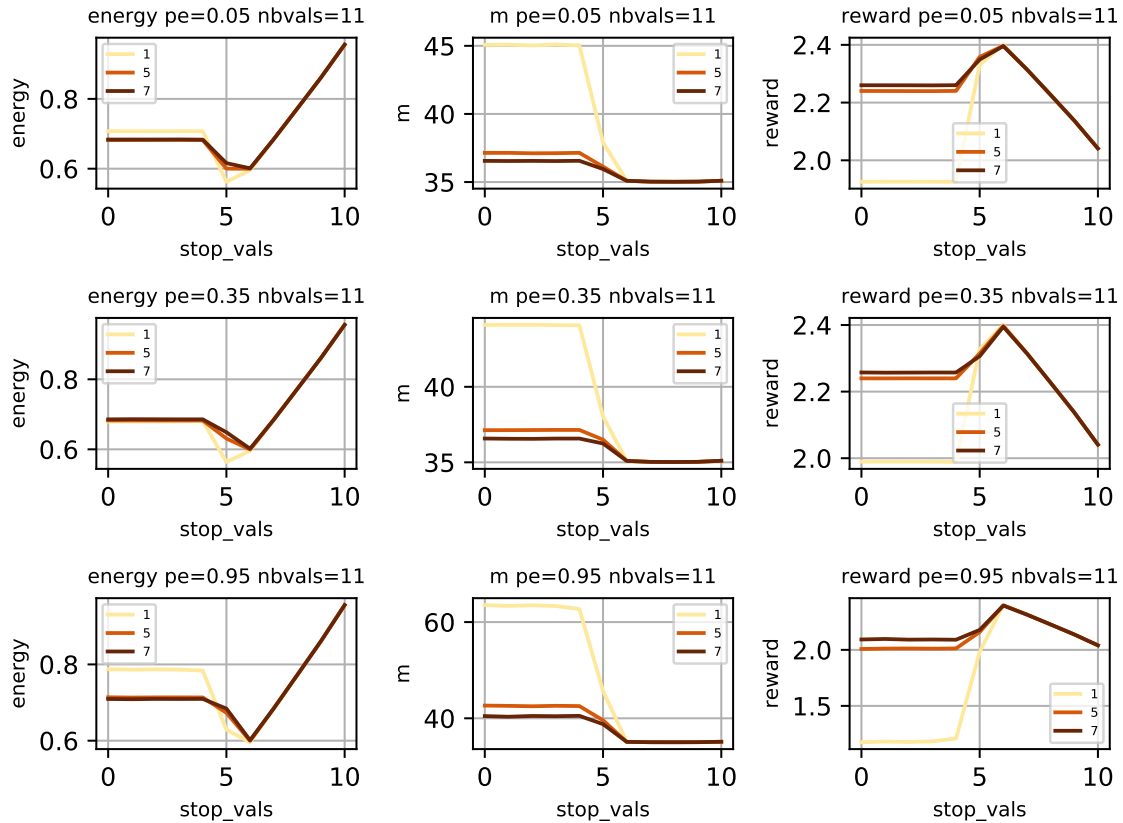


Figure 5.8 – Profile de l'utilisateur (valeur de  $m$ ), de l'énergie et la récompense (les colonnes) pour différentes valeurs de  $dur\_sa$  (1, 5, 7) et pour  $\lambda_{intr} = 2$  avec  $N_v = 11$ .

d'implémentation.

En premier lieu, nous commençons par comparer les algorithmes pour une discrétisation fine de l'ensemble des valeurs du signal. Ensuite, en nous reposant sur cette comparaison, nous prenons le meilleur algorithme et nous étudions avec celui-ci l'effet de la discrétisation (du temps et des valeurs du signal) sur les performances du système. Avant d'entamer ces deux points, nous expliquons dans les paragraphes qui suivent le protocole expérimental que nous avons suivi pour l'évaluation des algorithmes.

L'expérience que nous avons réalisée pour évaluer les performances des algorithmes a été effectuée comme suit. Nous avons fixé les valeurs des différents paramètres de la simulation (ceux concernant l'utilisateur, l'environnement ainsi que le système) à différentes valeurs possibles et pour chacune de ces combinaisons de valeurs, nous avons fait 30 exécutions de la simulation qui durent, chacune,  $5 \times 10^6$  étapes. Les paramètres que nous fixons dans nos simulations sont les suivants :

- Pour les paramètres de l'utilisateur, les valeurs initiales des paramètres  $a$  et  $m$  sont fixées aux valeurs  $a_0 = 0.1$  et  $m_0 = 35$ . Le paramètre de l'effet du présent  $pe$  est pris parmi les valeurs suivantes  $\{0.05, 0.35, 0.95\}$ .
- Concernant les paramètres de l'environnement, l'intervalle des valeurs du signal est compris entre  $s_{min} = 0$  et  $s_{max} = 90$ . Le nombre de sous-intervalles associé à la valeur du signal  $N_v$  est pris parmi l'ensemble  $\{7, 9, 11\}$ . Le nombre de sous-intervalles de la fréquence d'intervention

$N_f$  est fixé à 8 pour toutes les simulations.

- En ce qui concerne les paramètres du système, la durée de maintien des actions  $dur\_sa$  est prise parmi les valeurs suivantes  $\{1, 5, 7\}$ . Le paramètre de l'exploration  $\epsilon$  est choisi parmi les valeurs de l'ensemble  $\{0.01, 0.001\}$  et le taux d'apprentissage est choisi parmi les valeurs  $\{0.01, 0.05\}$ . Dans nos simulations, nous avons considéré les mises à jour **continuelles** et **différées** de la fonction des valeurs estimées. Ces deux cas correspondent à la variable  $upd\_each\_time$  qui vaut 1 pour le cas continu et 0 pour le cas différé.
- Le taux de *discount*  $\gamma$  est fixée à une valeur de 0.95.

Durant chaque exécution, pour chacun des paramétrages possibles de la simulation, nous nous focalisons sur l'évolution de l'énergie, de la valeur du signal, de la récompense et des valeurs du paramètre  $m$ . Puisque la durée des exécutions entrainerait un très grand nombre de données (le nombre d'étapes est de  $5 \times 10^6$ ), le temps est subdivisé en 1000 intervalles de taille 5000. L'évolution de ces paramètres (mentionnés en début de paragraphe) pour une exécution donnée est obtenue en prenant la valeur moyenne de ces paramètres pour chacun des sous-intervalles. Le temps se mesure donc en nombre de sous-intervalles (*bins*) et non pas en étapes.

Afin d'évaluer la vitesse et la qualité de convergence des algorithmes, nous prenons en compte aussi, pour toutes les exécutions de tous les paramétrages de la simulation, l'évolution de la proportion de fois où le système choisit la meilleure stratégie. Nous présentons, dans la prochaine sous-section, les notions relatives à ce point.

### 5.2.1 Métrique pour l'évaluation de la vitesse et de la qualité de convergence

Pour évaluer la vitesse et la qualité de convergence des algorithmes, nous avons besoin de suivre la fréquence à laquelle l'agent prend une valeur proche de l'optimale. Comme nous l'avons dit plus tôt, nous considérons uniquement l'ensemble des politiques *réalistes* consistant à faire baisser le signal continuellement puis à s'arrêter sur un sous-intervalle d'arrêt choisi préalablement. Pour cette raison, et vu que l'énergie est généralement corrélée à la valeur d'arrêt, nous considérons donc l'évolution de la proportion de fois (en nombre *bins*) où le système prend une des valeurs du sous-intervalle pour lequel la récompense est grande. Dans ce calcul, nous prenons en compte le meilleur sous-intervalle, en termes de récompense, ainsi que les sous-intervalles adjacents à celui-ci (le sous-intervalle immédiatement supérieur et le sous-intervalle immédiatement inférieur à celui-ci). Puisque nous calculons déjà l'évolution de la valeur du signal, la proportion de fois où l'agent choisit le meilleur sous-intervalle (ou l'un des sous-intervalles adjacents à celui-ci), est le nombre de *bins* où la valeur du signal appartient au meilleur sous-intervalle (ou l'un des sous-intervalles adjacents) divisé par le temps (mesuré en nombre de *bins*). Ce procédé donne l'évolution au cours du temps de la proportion de fois où l'agent choisit une stratégie dans le voisinage de la meilleure stratégie.

Afin que le lecteur puisse apprécier et interpréter le comportement des algorithmes, nous distinguons dans ce qui suit l'indice du meilleur sous-intervalle (selon la récompense) pour différentes situations. Pour  $N_v = 7$ , le meilleur sous-intervalle est d'indice 4 quelles que soient les valeurs de  $pe$  et de  $dur\_sa$ , ce qui correspond aux valeurs se situant dans  $[51.43, 64.29]$ . Pour  $N_v = 9$ , l'indice

du meilleur sous-intervalle est 5 pour toutes les valeurs de  $pe$  et de  $dur\_sa$ , ce qui correspond aux valeurs [50, 60]. Finalement, pour  $N_v = 11$  l'indice du meilleur sous-intervalle est 6 correspondant à l'intervalle [49.09, 57.27] et ce pour tout  $pe$  et pour tout  $dur\_sa$ . Nous observons naturellement que les trois sous-intervalles se chevauchent.

Par rapport à la métrique que nous choisissons pour étudier le comportement de convergence des algorithmes basés sur le choix de valeur du signal, nous précisons que celle-ci pourrait ne pas refléter correctement la convergence des algorithmes si, par exemple, l'algorithme exhibe un comportement mixte (dans lequel il alterne entre plusieurs stratégies dont l'effet est différent sur l'utilisateur) ou si les stratégies choisies par l'algorithme font intervenir beaucoup l'utilisateur. Néanmoins, nous décidons de garder quand même cette métrique, car premièrement, elle est simple à interpréter, et deuxièmement parce que l'algorithme choisit toujours des valeurs qui sont dans le voisinage de la valeur actuelle. Par ailleurs, nous pourrions toujours nous référer à l'évolution de l'état de l'utilisateur pour détecter des comportements de l'algorithme qui risquent d'avoir des effets erratiques sur ce dernier. Dans ce qui suit, nous présentons, analysons et comparons le comportement des algorithmes à états.

## 5.2.2 Comparaison entre l'algorithme de SARSA et de Q-learning

Dans ce qui suit, nous montrons l'évolution des performances de SARSA et de Q-learning en termes d'énergie, d'état de l'utilisateur et de récompense. Ensuite, nous comparons le comportement de ces deux algorithmes en mettant en relief la qualité et la vitesse de convergence de chaque algorithme.

La figure 5.9 montre le résultat de l'expérience effectuée pour l'algorithme de SARSA. L'évolution moyenne de l'énergie, de  $m$  et de la récompense (les colonnes) est montrée pour les utilisateurs correspondant à  $pe = 0.05$ ,  $pe = 0.35$  et  $pe = 0.95$  (les lignes). Ces résultats correspondent au cas continu où l'on a  $\lambda_{intr} = 2$ ,  $dur\_sa = 1$ ,  $N_v = 11$  et  $N_f = 8$ . Nous remarquons que l'énergie baisse au cours du temps pour atteindre, en moyenne, une valeur légèrement au-dessus de 60% de l'énergie maximale pour  $pe = 0.05$  et  $pe = 0.35$  et une valeur se situant autour de 65% pour  $pe = 0.95$ . Ces performances sont très proches de celles atteintes par la meilleure politique (se trouvant dans l'intervalle [54.54%, 63.63%]). Nous remarquons aussi que la valeur de  $m$  se stabilise sur une valeur très proche de la valeur minimale possible ( $m = 35$ ) et que la récompense augmente de manière continue tout au long de l'apprentissage. En revanche, nous remarquons que le système a des difficultés à gérer le cas  $pe = 0.95$  (dans lequel la mémoire de l'utilisateur est courte). Nous le voyons à travers l'évolution de l'énergie, du paramètre  $m$  ainsi que de la récompense qui sont assez erratiques (comparé aux autres cas) et qui varient d'une exécution à une autre. Nous notons que, pour l'algorithme SARSA, les meilleures valeurs des paramètres sont 0.01 pour la probabilité d'exploration et 0.05 pour le taux d'apprentissage.

Les observations décrites dans le paragraphe précédent montrent que le système arrive à apprendre à contrôler le signal avec une politique qui arrive à équilibrer entre performance énergétique et confort de l'utilisateur. Cette politique apprise par le système est, de surcroît, assez proche de la politique optimale (en termes de récompense). Une explication possible du comportement erratique de l'algorithme observé pour  $pe = 0.95$  est que la mémoire de l'utilisateur a tendance à changer rapidement si l'agent fait abruptement varier le signal entre deux valeurs pour lesquelles l'utilisateur a un

comportement radicalement différent (tout comme pour le cas du système à pente décrit dans le chapitre 4). Un exemple d'un tel scénario serait de faire passer le signal, de manière rapide, d'une valeur très forte à une valeur très faible ou inversement.

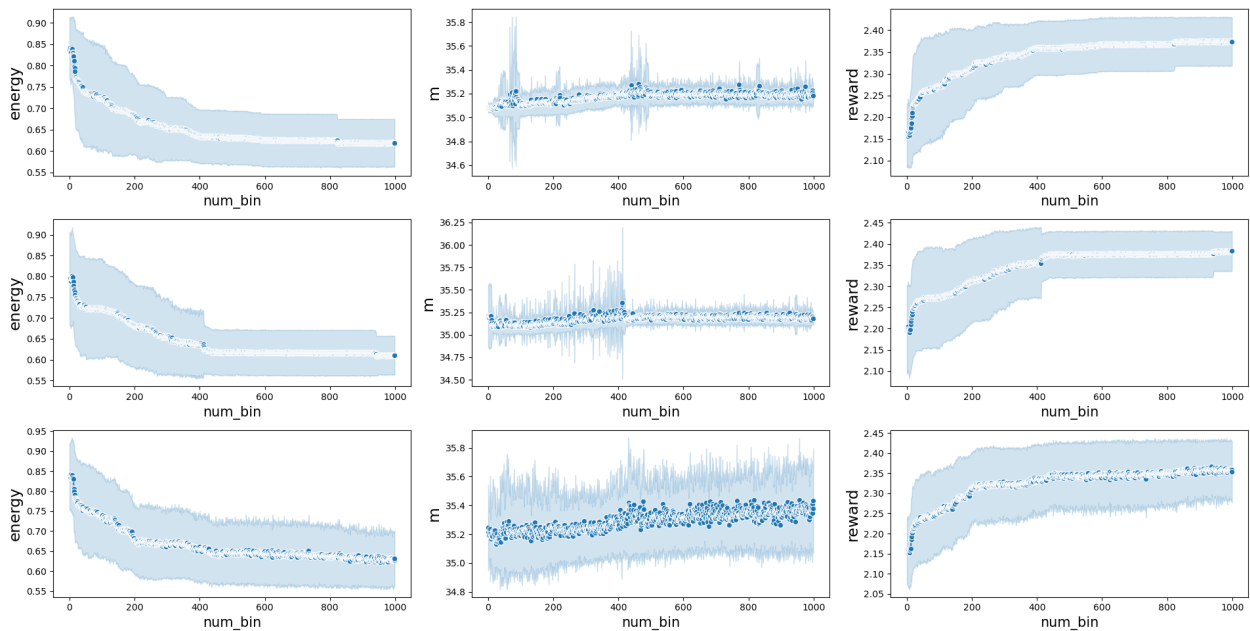


Figure 5.9 – L'évolution de l'énergie (en %), de l'état de l'utilisateur (valeur du paramètre  $m$ ) et de la récompense de l'algorithme de SARSA pour différentes valeurs de  $pe$  (0.05, 0.35 et 0.95 de haut en bas) pour  $\lambda_{intr} = 2$  et avec  $dur_{sa} = 1$ ,  $N_v = 11$ ,  $N_f = 8$  dans le cas continu.

La figure 5.10 montre le résultat de l'expérience effectuée pour l'algorithme Q-learning. La figure 5.10 est structurée de la même manière que la figure 5.9 associée à SARSA et correspond à la même situation que cette dernière. Nous remarquons que l'énergie baisse au cours du temps pour atteindre, en moyenne, une valeur qui ne s'éloigne que de peu de 60% de l'énergie maximale, et ce, pour tous les comportements d'utilisateur considérés. Tout comme pour l'algorithme de SARSA, ces performances sont très proches de celles de la meilleure politique (se trouvant dans l'intervalle [54.54%, 63.63%]). Nous remarquons aussi que l'état de l'utilisateur se stabilise sur une valeur très proche de la valeur minimale possible ( $m = 35$ ) et que la récompense augmente rapidement pendant l'apprentissage pour se stabiliser ensuite. En revanche, tout comme SARSA, nous remarquons également que le système a des difficultés à gérer le cas  $pe = 0.95$  (particulièrement en ce qui concerne la valeur de  $m$ ), même si ces difficultés sont visiblement beaucoup moins présentes dans Q-learning que dans SARSA. Nous notons donc que, contrairement à SARSA, le comportement de l'algorithme de Q-learning montre beaucoup moins de variance entre différentes exécutions de l'algorithme. Pour Q-learning, les meilleures valeurs des paramètres sont aussi 0.01 pour la probabilité d'exploration et 0.05 pour le taux d'apprentissage.

Les conclusions portées sur l'algorithme SARSA tiennent aussi pour l'algorithme Q-learning. Le système arrive à apprendre à contrôler le signal avec une politique correcte qui est très proche de la politique optimale. Et nous faisons l'hypothèse que le comportement erratique observé pour  $pe = 0.95$  dans Q-learning, est expliqué de la même manière que pour SARSA. Nous notons, par rapport à notre problématique, que les algorithmes SARSA et Q-learning convergent malgré la présence

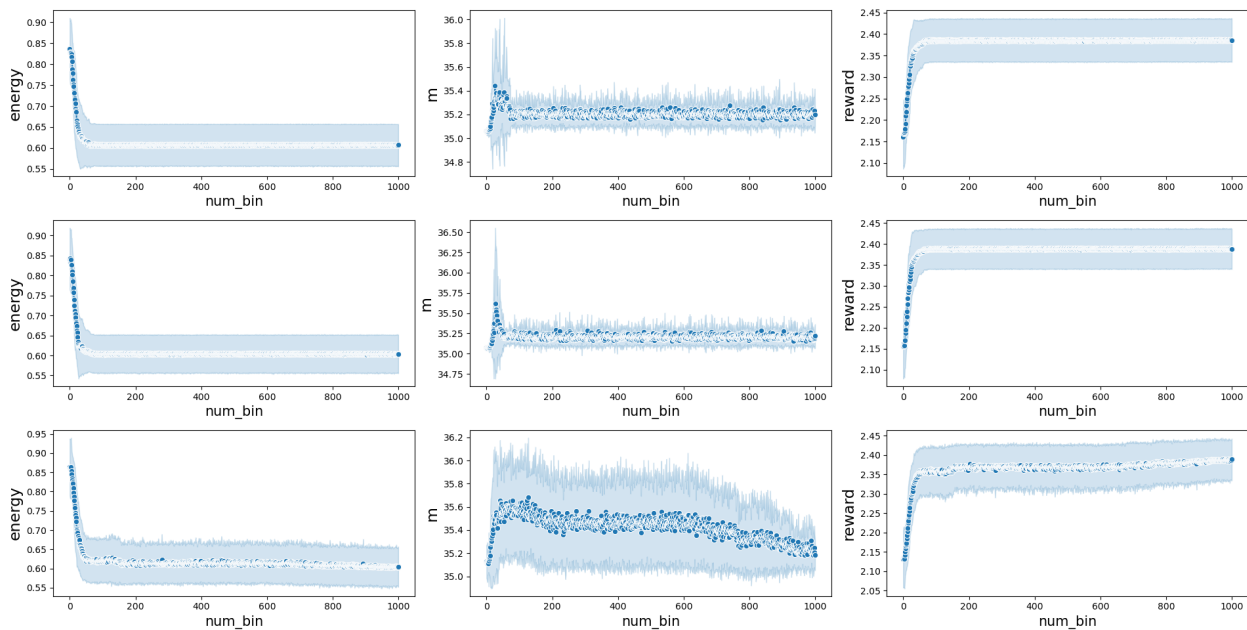


Figure 5.10 – L'évolution de l'énergie, de l'état de l'utilisateur (valeur du paramètre  $m$ ) et de la récompense de l'algorithme de Q-learning pour différentes valeurs de  $p_e$  (0.05, 0.35 et 0.95 de haut en bas) pour  $\lambda_{intr} = 2$  et avec  $dur_{sa} = 1$ ,  $N_v = 11$ ,  $N_f = 8$  dans le cas continu.

de non-stationnarité dans le comportement de l'utilisateur, et malgré le fait que l'environnement n'est pas complètement observable. Dans les paragraphes qui suivent, nous proposons des explications possibles à ce phénomène.

Premièrement, même si l'environnement est non observable, la représentation à état que nous utilisons est bel et bien markovienne (voir l'annexe A). De plus, l'état semble contenir assez d'information sur l'environnement pour que la politique auquel converge l'agent soit la meilleure politique. La fréquence d'intervention de l'utilisateur étant présente dans l'état, il est raisonnable de considérer que l'état prodigué à l'agent soit assez proche du vrai état de l'utilisateur et permet un déroulement correct de l'apprentissage.

Ce qui nous amène à notre deuxième point, et qui est que les approches basées sur le choix de la valeur du signal, tendrait naturellement à choisir des politiques pour lesquelles les valeurs du signal sont plutôt élevées, limitant ainsi le nombre d'états que l'algorithme a besoin d'explorer pour avoir une politique efficace. Cette interprétation des résultats de convergence des algorithmes nous paraît raisonnable vu que la vitesse de baisse du signal est déjà bornée par la taille des sous-intervalles des valeurs (qui a été présélectionnée par nos soins). Par conséquent, l'utilisateur aura tendance à réagir avant que le signal n'atteigne des valeurs qui sont beaucoup trop faibles pour lui. Une conséquence possible de cela est que l'utilisateur devienne peu sensible au contrôle assez rapidement après le début de l'apprentissage. Nous l'observons justement au travers l'évolution du paramètre  $m$  dans la figure 5.9 (SARSA) et dans la figure 5.10 (Q-learning), confirmant ainsi nos interprétations. Le fait que l'utilisateur soit amené vers des états confortables assez rapidement œuvre aussi à réduire le nombre d'états à explorer, parce que la fréquence d'intervention de l'utilisateur (qui fait partie de l'état) baissera nécessairement au fil du temps quand l'utilisateur devient de plus en plus satisfait.



En considérant les deux points précédents, nous pouvons voir pourquoi le décalage qui existe forcément entre l'état réel et l'état estimé de l'utilisateur pourrait être assez faible en réalité. Effectivement, ce décalage est limité par le fait que les états visités, la plupart du temps, sont spécifiquement ceux pour lesquelles l'utilisateur est peu sensible (ils sont donc relativement "similaires"). Bien entendu, si nous voulons que les algorithmes arrivent à contrôler le signal de manière idéale, il est nécessaire d'affiner au maximum la discrétisation au niveau des fréquences d'intervention afin de permettre à l'agent de mieux distinguer les états où l'utilisateur intervient peu. Ceci aura certainement pour effet de rallonger la durée d'apprentissage. En revanche, dans la pratique, nous pouvons nous contenter d'offrir un pilotage énergétique permettant de converger autour de la solution optimale en assurant une qualité de contrôle qui soit acceptable pour l'utilisateur. Il n'est donc peut-être pas si crucial de considérer une discrétisation très fine pour les états pour avoir des performances satisfaisantes en termes d'énergie et de confort de l'utilisateur.

Il est aussi à noter que Q-learning a un comportement beaucoup plus stable que SARSA, et ceci en dépit du fait que Q-learning soit un algorithme *off-policy*. Nous imputons cela aussi au fait que l'utilisateur a tendance à se stabiliser rapidement sur des états où il est confortable.

Il nous reste à analyser la vitesse et la qualité de convergences des algorithmes SARSA et Q-learning pour clore cette sous-section. Pour ce faire, nous affichons l'évolution de ces deux algorithmes dans la figure 5.11 en reprenant les éléments les plus pertinents servant à comparer leurs performances.

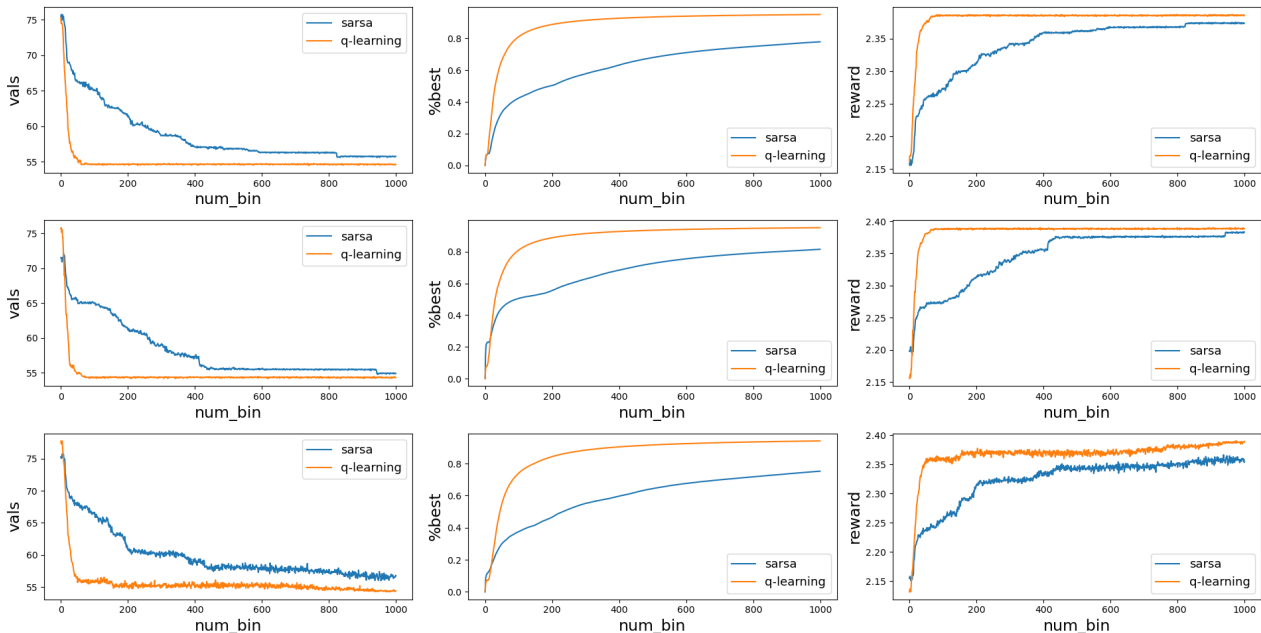


Figure 5.11 – Comparaison entre l'algorithme SARSA et le Q-learning pour différentes valeurs de  $pe$  (0.05, 0.35 et 0.95 de haut en bas).

Dans la figure 5.11, les performances de SARSA sont représentées en bleu et celles de Q-learning sont en orange. Les lignes représentent la mémoire de l'utilisateur ( $pe = 0.05$ ,  $pe = 0.35$  et  $pe = 0.95$  de haut en bas). Les colonnes représentent les éléments sur lesquelles ces algorithmes sont comparés. Les colonnes sont de gauche à droite, l'évolution de la valeur moyenne (dont l'axe des ordonnées est

labellisé par *vals*), l'évolution de la proportion de choix de valeurs d'arrêt se situant dans le voisinage de la meilleure valeur d'arrêt (dont l'axe des ordonnées est labellisé par *%best*) et l'évolution de la récompense.

Les courbes de la figure 5.11 sont obtenues en prenant la moyenne sur les 30 exécutions. Nous remarquons des différences significatives en termes de vitesse de convergence entre l'algorithme SARSA et Q-learning, la convergence de Q-learning semble beaucoup plus rapide et de meilleure qualité que celle de SARSA (pour tous les types d'utilisateur considérés). En effet, la récompense augmente plus rapidement dans Q-learning que dans SARSA, elle atteint aussi des valeurs plus élevées. Il en va de même pour la valeur du signal, celle-ci baisse plus rapidement pour Q-learning (par rapport à SARSA) et atteint de valeurs plus faibles, montrant que l'énergie est gérée plus efficacement avec Q-learning. Nous pouvons aussi le constater à travers l'évolution de la courbe correspondant à *%best* de Q-learning qui est strictement au-dessus de celle de SARSA durant presque la totalité de la durée de l'apprentissage (et ceci pour tous les types d'utilisateur). Ceci vaut aussi pour les courbes associées à l'évolution de la valeur du signal ainsi que la récompense et ceci pour tous les types d'utilisateur. Nous remarquons aussi que la proportion de choix d'une stratégie se trouvant dans le voisinage de la stratégie optimale dépasse les 80% pour tous les types d'utilisateur pour Q-learning, alors que pour SARSA, celle-ci n'avoisine les 80% que pour l'utilisateur associé à  $pe = 0.35$  et se situe entre de 60% et 80% pour les utilisateurs associés à  $pe = 0.05$  et  $pe = 0.95$ . Dans le cas  $pe = 0.05$ , et vers la fin de l'apprentissage, la récompense de SARSA semble légèrement meilleure que celle de Q-learning, ceci vient probablement du fait que SARSA a tendance à converger exactement vers la meilleure stratégie, un peu plus souvent que Q-learning (pour ce type d'utilisateur). Nous pouvons également le constater dans la courbe de l'évolution de la valeur du signal de SARSA qui semble être légèrement au-dessous de celle de Q-learning, montrant que pour l'utilisateur  $pe = 0.05$  Q-learning a tendance à préférer le confort à l'énergie.

Même si l'algorithme Q-learning est généralement connu pour être instable [116], il est souvent plus efficace en pratique que SARSA. Dans le contexte de notre travail, Q-learning converge plus rapidement et mieux que l'algorithme SARSA. L'algorithme Q-learning a un meilleur comportement que SARSA sur tous les points que nous avons examinés. Nous en concluons que, dans le contexte du contrôle énergétique de bâtiment connecté occupé par un utilisateur dont le comportement est non-stationnaire et dépend du comportement passé du système, l'algorithme Q-learning est plus efficace que l'algorithme SARSA. Ainsi, nous sélectionnons l'algorithme Q-learning pour notre étude sur la discrétisation.

## 5.3 Discrétisation

Dans ce qui suit, nous étudions l'effet de la discrétisation sur les performances du système. Nous commençons par étudier les performances du système en mettant en lien la discrétisation de l'espace d'action avec la durée de maintien des actions. Puis nous examinons les performances du système pour différentes valeurs de la durée de maintien de l'action et pour différents types de mise à jour de la politique. Nous nous intéressons principalement à l'évolution dans le temps de la fréquence où l'agent choisit les valeurs d'arrêt se situant dans le voisinage de la meilleure valeur d'arrêt, mais aussi

à la fréquence où l'agent choisit exactement la meilleure valeur d'arrêt. Les résultats sont présentés pour l'algorithme de Q-learning (retenu suite aux conclusions de la sous-section 5.2.2).

### 5.3.1 Discrétisation de l'espace-temps

Dans cette sous-section, nous voulons montrer l'influence de la discrétisation du temps et de la discrétisation des valeurs du signal sur les performances du système, tout en mettant en exergue les relations potentielles pouvant lier les deux. La figure 5.12 montre l'évolution, au cours du temps, de la fréquence de choix de la meilleure action (ou une action dans son voisinage) pour  $dur\_sa \in \{1, 5, 7\}$  (lignes), pour  $pe \in \{0.05, 0.35, 0.95\}$  (colonnes), pour  $N_v \in \{7, 9, 11\}$  et pour le cas différé ( $upd\_each\_time = 1$ ). La figure 5.13 représente l'évolution, au cours du temps, de la fréquence de choix de la meilleure action pour les mêmes paramètres que la figure 5.12 et pour les mêmes valeurs. Ces figures servent à étudier la convergence pure et la convergence mixte de l'algorithme pour différentes discrétisations. Nous commençons par étudier individuellement chacune des figures 5.12 et 5.13 pour faire ressortir les points communs entre la convergence pure et mixte. Et ensuite, nous détaillons et comparons les différences potentielles entre ces deux types de convergence afin de les expliquer.

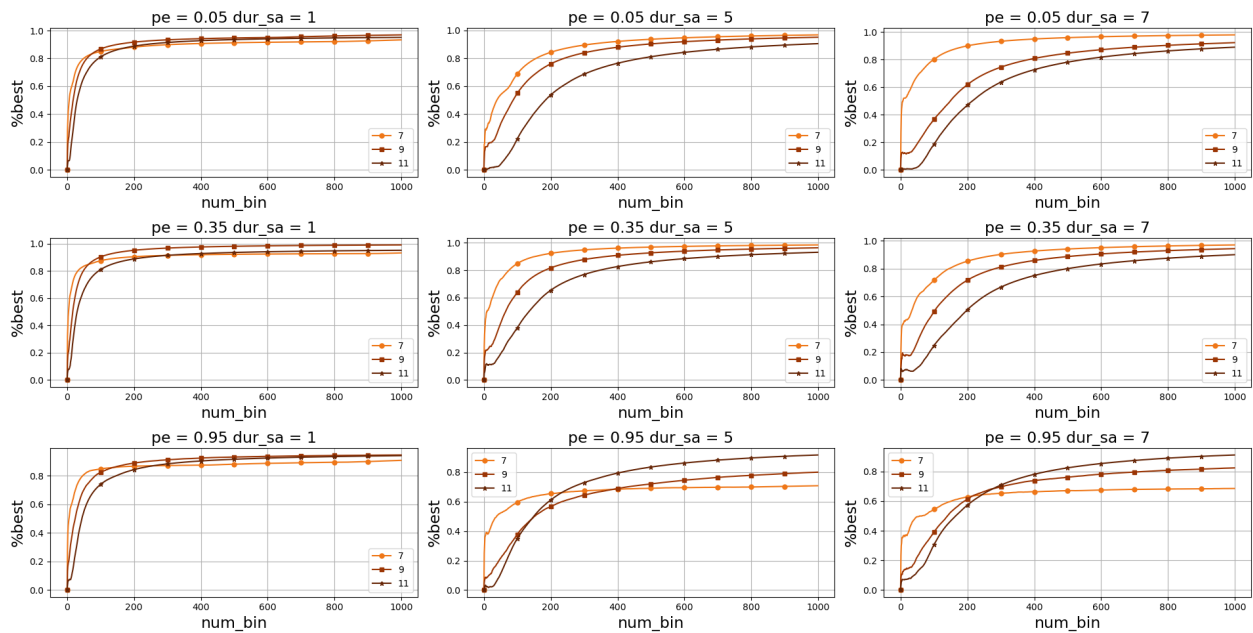


Figure 5.12 – Évolution de la proportion de choix de la meilleure action (ou voisinage) pour le Q-learning pour différentes valeurs de  $pe$  (lignes) et différentes valeurs de  $dur\_sa$  (colonnes).

Détaillons les courbes de la figure 5.12. En premier lieu, nous remarquons que les courbes semblent évoluer de manière assez homogène pour différentes discrétisations du signal quand  $dur\_sa = 1$  et aussi pour les cas où  $pe < 0.95$ . En revanche, nous notons la tendance inverse quand  $pe = 0.95$  pour  $dur\_sa > 1$  : la proportion de choix de la meilleure action (ou une action dans son voisinage) augmente et se stabilise assez rapidement pour le cas  $N_v = 7$  en comparaison aux cas  $N_v = 9$  et  $N_v = 11$ . De plus, la valeur atteinte par cette proportion est plus faible pour  $N_v = 7$  que les valeurs atteintes pour les cas  $N_v = 9$  et  $N_v = 11$ . Nous remarquons aussi que dans tous les cas que nous avons considérés, la discrétisation pour laquelle le système semble converger le plus rapidement est la discrétisation la plus large (correspondant à  $N_v = 7$ ). Nous remarquons aussi que dans les cas

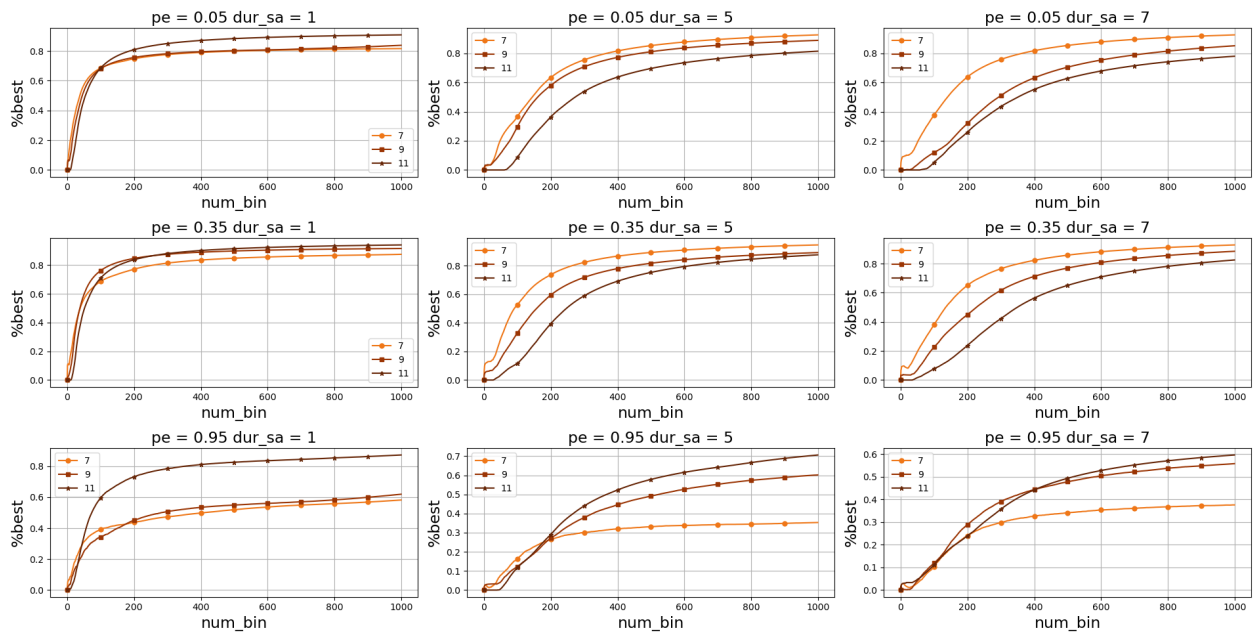


Figure 5.13 – Évolution de la proportion de choix de la meilleure action (seulement) pour le Q-learning pour différentes valeurs de  $pe$  (lignes) et différentes valeurs de  $dur\_sa$  (colonnes).

pour lesquelles  $dur\_sa = 1$  et les cas pour lesquelles  $pe = 0.95$ , la qualité de convergence est meilleure quand la discrétisation est fine ( $N_v = 11$  ou  $N_v = 9$ ) alors que dans les autres cas, la convergence est de meilleure qualité pour la discrétisation la plus large ( $N_v = 7$ ). Finalement, nous remarquons que prendre des grandes valeurs de  $dur\_sa$  ( $dur\_sa > 1$ ) n'améliore ni la qualité de convergence ni la vitesse de convergence.

Dans la figure 5.13, nous notons globalement les mêmes tendances que dans la figure 5.12, à quelques exceptions près. Les courbes sont assez similaires entre elles dans tous les cas montrés dans la figure 5.13, sauf quand  $pe = 0.95$  où nous remarquons que pour les discrétisations éparse, la proportion de choix de la meilleure action arrête d'augmenter au bout d'un temps assez court. Dans les cas pour lesquels  $pe < 0.95$  et  $dur\_sa > 1$ , la discrétisation pour laquelle l'algorithme converge le plus rapidement est la discrétisation la plus large. La qualité de convergence est aussi meilleure pour les discrétisations larges. Dans les paragraphes qui suivent, nous analysons les résultats des figures 5.12 et 5.13 en proposant des interprétations possibles pour ces résultats.

La dissemblance de l'évolution des courbes observée pour  $pe = 0.95$  dans beaucoup des cas montrés dans les figures 5.12 et 5.13 peut être expliquée comme suit. Comme l'utilisateur est très sensible aux valeurs récentes du signal pour  $pe = 0.95$  (mémoire courte), l'utilisateur aura tendance à réagir fortement si la valeur du signal baisse de manière soudaine. Ceci se produira certainement si la discrétisation est assez éparse (le cas  $N_v = 7$ ) et risque d'empêcher l'algorithme d'apprendre de bonnes actions car l'utilisateur réagit beaucoup trop souvent. De ce fait, l'algorithme risque de bloquer sur les mauvaises actions, et ceci se voit à travers les courbes des figures 5.12 et 5.13 qui se stabilisent assez rapidement pour  $pe = 0.95$ . Nous pouvons conclure que la proportion de choix de la meilleure action se stabilise assez rapidement pour les discrétisations éparse. A contrario, l'utilisateur est juste assez réactif pour les discrétisations fines (les cas  $N_v = 9$  et  $N_v = 11$ ) et l'algorithme

peut, de ce fait, apprendre plus rapidement (et converger vers de bonnes actions).

Nous avons noté dans les paragraphes précédents que dans beaucoup de cas, l'algorithme converge au moins aussi rapidement pour les discrétisations éparées que pour les discrétisations fines, et ceci quel que soit le type de convergence considéré. La raison à cela est que l'espace d'état est de plus petite taille pour les discrétisations éparées, l'algorithme a donc besoin de visiter moins d'états pour apprendre à contrôler le signal. Quand la discrétisation est fine, l'algorithme met plus longtemps à converger et nous le remarquons d'autant plus quand  $dur\_sa > 1$  (car pour  $dur\_sa > 1$  le signal reste plus longtemps sur une valeur donnée avant d'être baissé).

Nous avons aussi observé (dans les figures 5.12 et 5.13) que pour les cas dans lesquels  $dur\_sa = 1$  et les cas pour lesquels  $pe = 0.95$ , la qualité de convergence est meilleure quand la discrétisation est fine. Et ceci, car les discrétisations très larges contiennent des sous-intervalles pour lesquelles l'utilisateur réagit de manière très différente (à l'intérieur d'un même sous-intervalle et sur différents sous-intervalles). Ceci veut dire que l'état observé par l'agent ne permet pas de distinguer des situations pour lesquelles l'agent doit agir différemment.

L'idée de prendre de grandes valeurs pour  $dur\_sa$  est de laisser le temps à l'utilisateur d'intervenir avant de baisser la valeur du signal. Cependant, les résultats indiquent que les valeurs élevées de  $dur\_sa$  ralentissent l'apprentissage : le signal met beaucoup trop longtemps à baisser, l'utilisateur n'interagit pas assez souvent avec le système, ne lui offrant ainsi pas assez d'informations pour apprendre. L'algorithme apprend donc moins rapidement et moins bien.

Nous concluons aussi que maintenir l'action pendant plus d'une étape lors de l'exécution de l'algorithme ne permet pas d'améliorer les performances de l'algorithme. Nous pouvons aussi dire que, globalement, la meilleure discrétisation dépend du type de métrique qui nous intéresse. Si nous nous intéressons à la vitesse de convergence, il est recommandé de prendre une discrétisation large ( $N_v = 7$ ) et si nous nous intéressons à la qualité de convergence, nous pouvons opter pour une discrétisation fine ( $N_v = 11$ ). Quand l'utilisateur a une mémoire relativement grande, nous ne notons pas de dépendance forte entre, d'un côté, la meilleure discrétisation en termes de temps de convergence (respectivement en termes de qualité de convergence) et de l'autre côté, la durée de maintien de l'action. Dans le cas d'un utilisateur ayant une mémoire, augmenter la durée de maintien de l'action ne semble que renforcer des tendances déjà existantes sur les discrétisations quand la durée de maintien de l'action est d'une étape seulement. Notons que l'utilisateur dont la mémoire est à court-terme représente un cas spécial dans le sens où la discrétisation la plus fine donne les meilleurs résultats pour ce cas. Les discrétisations moins fines incitent l'utilisateur à réagir beaucoup trop souvent et ceci entrave l'apprentissage. Comme il ne semble pas y avoir de discrétisation améliorant à la fois la vitesse et la qualité de convergence, il pourrait être recommandé d'utiliser des fonctions d'approximation tel que les réseaux de neurones pour améliorer les résultats. Dans ce qui suit, nous étudions l'effet de la fréquence de mise à jour sur les performances du système. Nous ne voulons faire aucun a priori sur l'utilisateur et sa relation avec les performances du système, nous prenons donc la discrétisation la plus fine ( $N_v = 11$ ) pour cette étude.

### 5.3.2 Discrétisation du temps sur la fréquence de décision

Dans ce qui suit, nous voulons savoir comment la fréquence avec laquelle l'agent prend une décision (la durée de maintien de l'action noté  $dur\_sa$ ) et la fréquence avec laquelle il met à jour sa politique (le cas continu et différé) influencent la convergence de l'algorithme (voir la sous-section 5.1.4 pour plus de détails). La discrétisation prise correspond à  $N_v = 11$ . La figure 5.14 montre l'évolution, au cours du temps, de la fréquence de choix de la meilleure action (ou une action dans son voisinage) pour  $dur\_sa \in \{1, 5, 7\}$  et pour les cas où la mise à jour est continue (haut) et différé (bas) et ce pour tous les types de mémoire de l'utilisateur ( $pe = 0.05$ ,  $pe = 0.35$  and  $pe = 0.95$ ). La figure 5.15 représente l'évolution, au cours du temps, de la fréquence de choix de la meilleure action pour les mêmes paramètres que la figure 5.14. Les résultats de ces simulations sont calculés sur 50 exécutions. Ces figures servent à étudier la convergence pure et la convergence mixte de l'algorithme pour différentes valeurs de la durée de maintien de l'action et pour différents modes de mise à jour de la politique (différé et continu).

En observant les figures 5.14 et 5.15, nous notons des différences significatives dans l'évolution des performances pour le cas où l'utilisateur a une faible mémoire ( $pe = 0.95$ ). En effet, l'algorithme semble mettre plus de temps à converger pour  $pe = 0.95$ , et cela quelle que soit la durée de maintien de l'action, le type de mise à jour considéré ainsi que la métrique de convergence (pure ou mixte). Nous remarquons aussi que l'algorithme converge moins souvent vers la meilleure action pour  $pe = 0.95$ . Cette différence marquée du cas  $pe = 0.95$  provient du fait que l'utilisateur change beaucoup trop souvent. Comme les réactions de l'utilisateur ne dépendent que des valeurs récentes pour  $pe = 0.95$ , les changements des valeurs du signal impactent immédiatement l'utilisateur. Et ces changements incessants de l'utilisateur rendent l'apprentissage plus difficile pour l'agent, ce qui contribue à le ralentir.

D'autres éléments intéressants se dégagent des figures 5.14 et 5.15. Nous observons dans la figure 5.14 que, dans le cas continu, l'agent arrive plus rapidement à trouver la meilleure stratégie (ou une stratégie dans son voisinage) pour les grandes valeurs de  $dur\_sa$  ( $dur\_sa > 1$ ). À l'inverse, nous observons dans la même figure que, dans le cas différé, l'agent peine à trouver la meilleure stratégie (ou une stratégie dans son voisinage) pour les grandes valeurs de  $dur\_sa$ . Les mêmes observations peuvent être relevées pour la convergence pure dans la figure 5.15. Plus important encore, nous constatons que l'évolution de la proportion de choix de la meilleure stratégie (ou une stratégie dans son voisinage) ne dépend pas de la mémoire pour  $dur\_sa = 7$  dans le cas continu (le comportement de l'agent change très peu en fonction de la mémoire de l'utilisateur). Nous observons aussi dans la figure 5.14 une évolution assez rapide de l'apprentissage quand  $dur\_sa = 7$  (où la proportion de choix d'une action proche de l'optimale dépasse 0.8 pour tous les usagers). Nous interprétons ces résultats dans le paragraphe suivant.

Quand la durée de maintien de l'action est grande, l'agent met plus de temps à apprendre la politique optimale, car le signal reste sur des valeurs élevées plus souvent et l'utilisateur donne moins de retours à l'agent. En revanche, en mettant à jour continuellement la politique de l'agent (avant que l'action ne finisse), ce désavantage est contrebalancé par l'information qu'on injecte à l'agent en mettant à jour sa politique. La fréquence de mise à jour étant faible dans le cas différé, l'apprentissage

risque d'être un peu plus lent que dans le cas continu, pour lequel l'apprentissage semble s'adapter plus vite à l'environnement. La mise à jour continue contribue à rendre l'apprentissage très stable indépendamment de la mémoire de l'utilisateur (d'autant plus si nous nous intéressons à la convergence mixte).

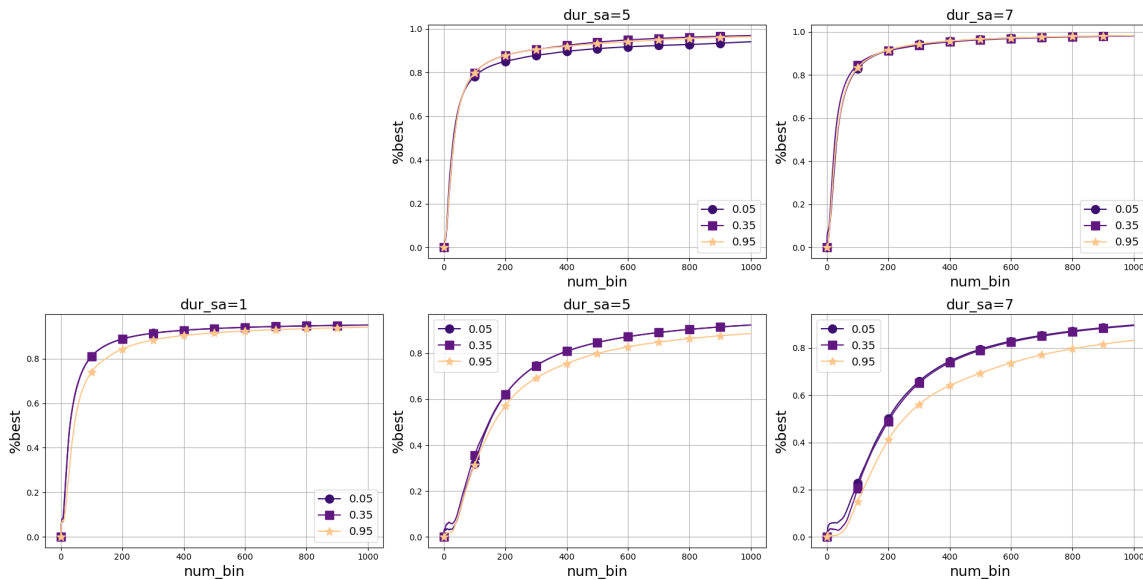


Figure 5.14 – Évolution de la proportion de choix de la meilleure action (ou voisinage) pour le Q-learning dans la mise à jour continue (en haut) et différée (en bas) pour différentes valeurs de  $pe$  (0.05, 0.35 et 0.95).

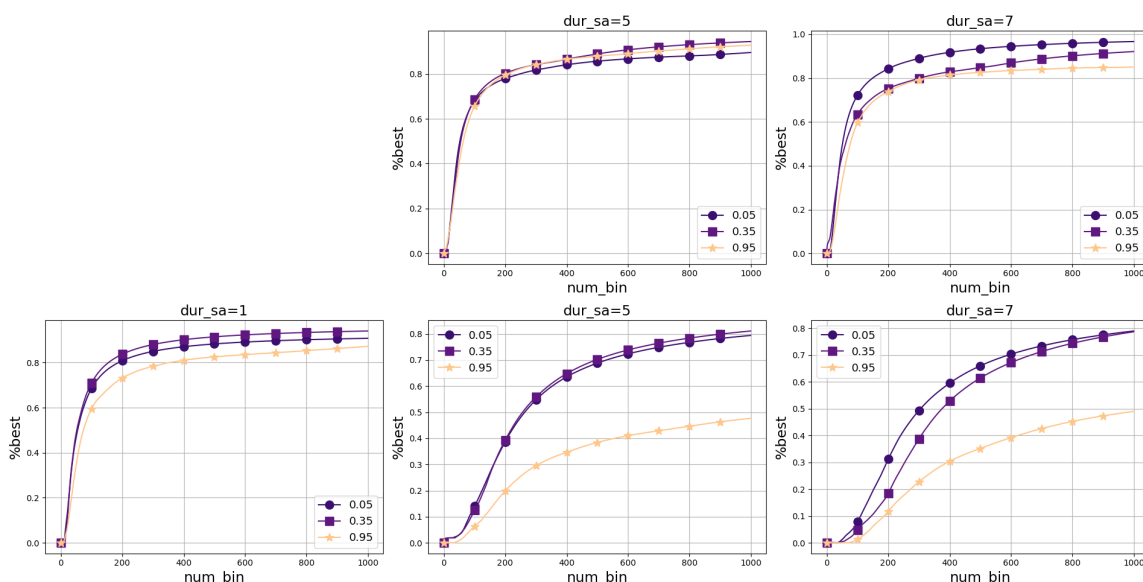


Figure 5.15 – Évolution de la proportion de choix de la meilleure action (convergence pure) pour le Q-learning dans la mise à jour continue (en haut) et différée (en bas) pour différentes valeurs de  $pe$  (0.05, 0.35 et 0.95).

Maintenir l'action pendant un certain temps avant de donner la main au système permet d'améliorer les performances de l'agent en termes de vitesse et de qualité de convergence, et ceci indépendamment de la mémoire de l'utilisateur. Il est possible qu'après avoir exploré les différentes actions

possibles, l'agent finisse par préférer les politiques pour lesquelles le signal est baissé pendant un certain temps puis arrêté sur une certaine valeur. De ce fait, une raison possible pouvant expliquer les résultats concernant le maintien de l'action est que baisser immédiatement la valeur du signal à chaque étape peut conduire l'utilisateur à intervenir plus souvent, car la valeur du signal prendrait des valeurs faibles avant que l'utilisateur n'ait eu le temps d'intervenir. Ceci aurait pour effet d'amener l'utilisateur à être plus sensible, ce qui risque de perturber l'apprentissage de l'agent. Si cela est vérifié, nous pouvons dire que maintenir l'action pendant plusieurs étapes permet d'orienter l'apprentissage par un mécanisme de confirmation de la valeur actuelle pour l'utilisateur. Ce mécanisme permet de limiter la fréquence d'intervention de l'utilisateur et de maintenir l'utilisateur dans un état de confort en empêchant que le signal prenne des valeurs faibles trop rapidement. Et puisque l'utilisateur à faible mémoire ne change qu'en fonction des valeurs récentes du signal, implémenter ce mécanisme de confirmation aurait peu de sens dans ce cas. Et ce, car l'utilisateur ne changera pas si la valeur actuelle est maintenue. De plus, la réactivité de l'utilisateur risque de changer complètement dès que la valeur du signal change. Bien entendu, ces conclusions ne tiennent que pour les valeurs testées de la durée de maintien des actions. En effet, prendre des valeurs trop grandes pour  $dur\_sa$  pourrait aussi gêner l'apprentissage, car le signal prendrait beaucoup trop de temps pour arriver à des valeurs pour lesquelles l'utilisateur donne assez de retours au système. En revanche, les résultats montrent que cet effet est contrebalancé si la politique de l'agent est mise à jour très souvent.

Nous en concluons qu'utiliser la technique consistant à maintenir l'action pour une certaine durée tout en mettant continuellement à jour la politique de l'agent, participe à rendre l'apprentissage plus stable, rendant ainsi le fonctionnement de l'algorithme robuste pour différentes valeurs de la mémoire de l'utilisateur, et impacte positivement la vitesse de convergence. Dans la section suivante, nous comparons les performances du système par choix de valeur du signal et le système par choix de variation du signal.

## 5.4 Comparaison entre le système par choix de variation du signal et le système par choix de valeur du signal

Dans le chapitre 4, nous avons étudié le système par **choix de variation du signal**, nous avons sélectionné les algorithmes les plus performants pour ce modèle ainsi que les paramètres adéquats. Nous avons fait de même pour le système par **choix de valeur du signal** dans le chapitre 5. Dans cette section, nous présentons une comparaison entre le système par choix de variation du signal et le système par choix de valeur du signal sur la base des études faites précédemment. Le système par choix de variation du signal est composé de l'agent des pentes implémenté en algorithme de la poursuite hiérarchique et de l'agent des valeurs d'arrêt implémenté en UCB. Le système par choix de valeur du signal est représenté par un agent Q-learning avec  $N_f = 8$ ,  $\lambda_{intr} = 2$ , et  $dur\_sa = 1$  et où la politique de choix de la valeur du signal est aléatoire. Concernant la discrétisation, nous considérons dans cette comparaison deux cas :  $N_v = 11$  qui est la discrétisation que nous avons sélectionnée dans notre étude du système par choix de valeur du signal et  $N_v = 7$  qui est la discrétisation la plus fine parmi celles que nous avons considérées.

Dans le cadre de notre travail, nous choisissons le système par choix de valeur du signal comme



référence pour mesurer les performances des algorithmes et pouvoir les comparer. Dans ce chapitre, nous avons choisi la fréquence de choix du meilleur sous-intervalle (ou l'un de ses voisins) pour mesurer la rapidité et la qualité de convergence du système par choix de valeur du signal. Nous utilisons cette même métrique pour comparer les performances du système par choix de variation du signal au système par choix de valeur du signal. Dans la sous-section 5.2.1, nous avons dit au sujet de cette métrique qu'elle ne permet pas de rendre compte de la convergence des algorithmes dans le cas où l'utilisateur est amené à avoir un comportement trop erratique. Pour le système basé sur le choix des valeurs, l'utilisateur ne semble pas montrer de tels comportements, dans chaque cas montré et étudié dans ce chapitre, nous avons bien vu que le système ne produit pas d'inconfort chez l'utilisateur. Mais ceci concerne uniquement le système par choix de valeur du signal et pas le système par choix de variation du signal. En effet, si ce dernier est utilisé pour contrôler le signal, nous risquons, bien entendu, de causer une situation d'inconfort chez l'utilisateur (par exemple si le système choisit des pentes dans la région "linéaire" de la figure 4.3). Il est même possible que l'utilisateur ait un comportement instable si le système n'est pas paramétré correctement. De ce fait, en plus de montrer l'évolution de la valeur du signal et de la fréquence de choix du meilleur sous-intervalle (ou un sous-intervalle se trouvant dans son voisinage), nous montrons également l'évolution de l'état de l'utilisateur dans le cadre de notre comparaison entre le système par choix de variation du signal et le système par choix de valeur du signal.

La figure 5.16 montre l'évolution, au cours du temps, des performances du système par choix de variation du signal en comparaison avec le système par choix de la valeur du signal (avec  $N_v = 11$  et  $N_v = 7$ ). Les lignes correspondent aux valeurs 0.05, 0.35 et 0.95 de  $pe$  (de haut en bas respectivement) et les colonnes représentent l'énergie, la valeur du signal et la fréquence de choix du meilleur sous-intervalle (ou l'un de ses voisins) de gauche à droite.

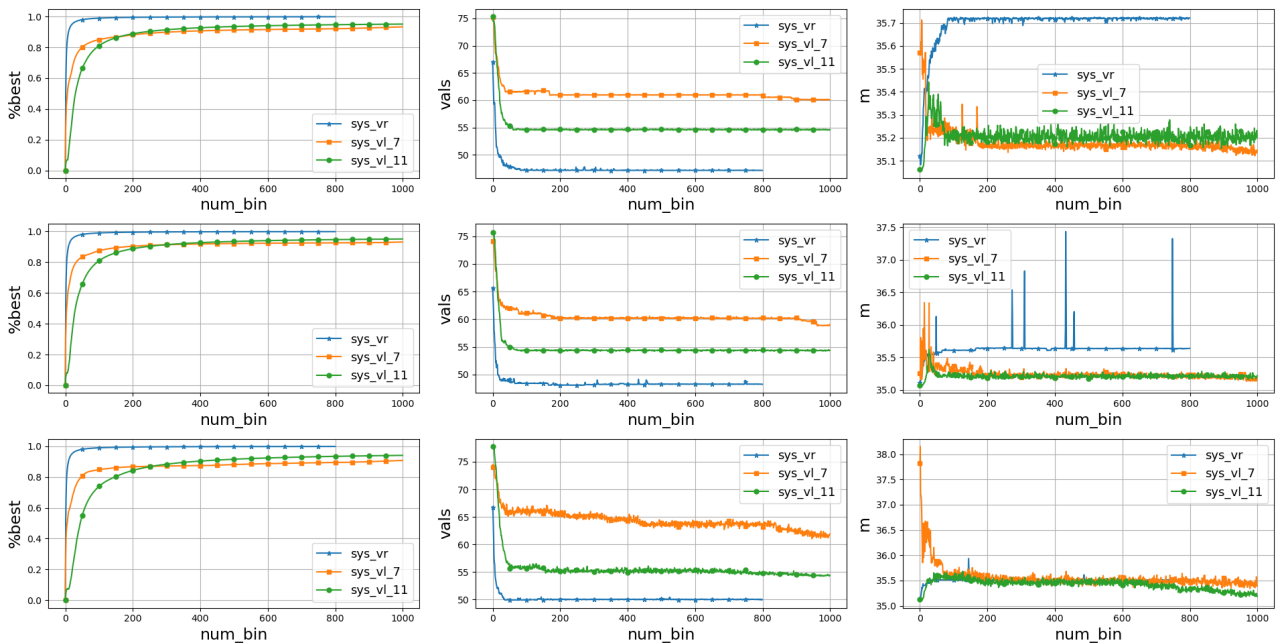


Figure 5.16 – Comparaison des performances du système par choix de variation du signal et du système par choix de valeur du signal pour  $N_v = 7$  et  $N_v = 11$  avec différents types d'utilisateurs.

Nous remarquons, dans la figure 5.16, que la convergence du système par choix de variation

du signal est clairement plus rapide et de meilleure qualité que celle associée au système par choix de valeur du signal. Nous voyons bien que pour tous les types d'utilisateur, la fréquence de choix du meilleur sous-intervalle (ou l'un de ses voisins), atteint presque 100% pour le système par choix de variation du signal dès les 50 premiers bins. En revanche, pour le système par choix de la valeur du signal avec  $N_v = 11$ , la fréquence de choix du meilleur sous-intervalle (ou l'un de ses voisins) n'atteint même pas les 70% durant les 50 premiers bins et elle dépasse à peine 80% pendant cette même durée quand  $N_v = 7$  (et ceci pour tout type d'utilisateur). La rapidité de convergence qu'on observe dans le système par choix de variation de signal peut s'expliquer très facilement par le fait qu'il y a moins d'éléments à "apprendre" pour ce système (comparé au système par choix de valeur du signal dans lequel la combinatoire des états et des actions engendre beaucoup plus d'éléments à tester).

Nous remarquons que les valeurs atteintes vers la fin de l'apprentissage sont différentes entre le système par choix de variation du signal et le système par choix de valeur du signal. Le système par choix de variation du signal semble avoir tendance à prendre des valeurs plus faibles par rapport au système par choix de valeur du signal. En effet, les valeurs choisies par le système par choix de variation de signal vers la fin de l'apprentissage, se situent autour de 50 alors que les valeurs choisies par le système par choix de valeur du signal vont de 55 à 65 (suivant le type d'utilisateur et la discrétisation considérés).

Nous remarquons aussi que l'évolution de l'utilisateur prend différents profils suivant le type de système qui contrôle le signal. Pour le système basé sur le choix des variations du signal, la valeur de  $m$  semble avoir un comportement monotone dans lequel elle a tendance à augmenter très rapidement avant de se stabiliser sur une valeur donnée. Par contre, pour le système basé sur le choix des valeurs du signal, la valeur de  $m$  monte puis descend, pour enfin se stabiliser autour d'une valeur. Et ce profil semble particulièrement saillant pour la discrétisation éparse  $N_v = 7$  : l'utilisateur est relativement inconfortable durant l'apprentissage, car le signal alterne entre des valeurs éloignées entre elles. Mais étant donné que cette discrétisation est très éparse, l'algorithme se stabilise au bout d'un moment sur une valeur qui est assez élevée pour satisfaire l'utilisateur (et qui est probablement plus élevée que nécessaire). Somme toute, nous observons que le système arrive à des compromis différents entre énergie et confort suivant la manière dont *la décision* est modélisée : pour le système par choix de variation du signal, l'énergie aura tendance à être privilégiée (aux dépens du confort), mais pour le système par choix de la valeur du signal, c'est le confort qui sera privilégié (aux dépens de l'énergie).

En conclusion, nous pouvons dire que la manière de décider (et donc d'apprendre) est différente pour les deux systèmes. Dans le système par choix de valeur du signal, le système décide de la valeur suivante et la choisit de sorte qu'elle soit dans le voisinage de la valeur courante. Ceci limite la magnitude des changements qui pourraient affecter l'utilisateur. Dans le système par choix de variation du signal, le système choisit d'emblée la stratégie à suivre, et par conséquent, l'utilisateur peut changer d'états très rapidement, mais le système arrive à apprendre assez rapidement à contrôler le signal dans ce cas, même si l'utilisateur est dans un état moins confortable que pour le premier type de système.

## Conclusion

Dans ce chapitre, nous avons présenté une modélisation à base d'apprentissage par renforcement pour le système par choix de valeur du signal. Nous avons comparé les performances de deux algorithmes classiques de la littérature : SARSA et Q-learning. Nous avons pu constater la supériorité de Q-learning sur tous les points considérés. Nous en concluons que, dans le contexte du contrôle énergétique de bâtiment connecté occupé par un usager dont le comportement est non-stationnaire et dépend du comportement passé du système, l'algorithme Q-learning est plus efficace que l'algorithme SARSA.

Sur la base de cette comparaison, nous avons étudié la discrétisation des valeurs du signal et du temps de décision. Tout comme pour le système par choix de variation du signal, nous notons la nécessité de prendre en compte le compromis entre la finesse de la discrétisation et l'efficacité de convergence du système. Par ailleurs, nous concluons de cette étude qu'il ne semble pas y avoir de discrétisation satisfaisant de manière optimale le critère de vitesse de convergence et de qualité de convergence. Par la suite, nous avons étudié l'effet de la fréquence de mise à jour sur les performances du système. Cette étude nous a amené à conclure que le mécanisme de mise à jour continue de la politique de l'agent conduit à une convergence plus stable et plus rapide de l'algorithme, rendant ainsi son fonctionnement robuste pour différentes valeurs de la mémoire de l'usager. Finalement, nous avons comparé les performances du système par choix de variation du signal et le système par choix de valeur du signal. Cette comparaison a révélé que ce dernier est plus efficace et converge de façon plus rapide que le premier. Notons néanmoins que ces conclusions tiennent uniquement dans le cas que nous avons considéré et doivent être étudiées dans le cas où la luminosité extérieure change en intégrant également l'activité des usagers.



# Conclusions et perspectives

Dans le cadre de cette thèse, nous avons étudié la problématique du contrôle de la lumière dans un bâtiment connecté. Nous vivons dans un monde où l'impact des usagers sur la consommation énergétique, ainsi que les défis et les enjeux autour de l'énergie, des bâtiments, les enjeux climatiques et sociétaux prennent une place de plus en plus importante. Il devient alors nécessaire de rendre la gestion du bâtiment plus intelligente et plus en phase avec ces défis. Cette gestion doit mettre les usagers au centre de ses préoccupations. Toutefois, la majorité des modèles de confort proposés dans la littérature se basent sur une formulation rationnelle du confort des usagers utilisant des éléments descriptifs de l'environnement et des usagers, qui sont souvent indisponibles en situation réelle. En outre, ces modèles ont rapidement montré leur limite face aux investigations approfondies de la communauté scientifique. Celles-ci ont mis en avant les difficultés à prédire de manière fiable le comportement des usagers. Les travaux effectués dans le cadre de cette thèse montrent que l'apprentissage par renforcement semble être une piste intéressante à explorer pour concevoir des méthodes capables de s'adapter au changement de l'environnement physique et aux comportements imprédictibles de l'utilisateur. De telles approches permettraient de réaliser des gains énergétiques importants tout en assurant des conditions confortables aux usagers des bâtiments. Elles seraient capables de s'adapter à des changements de l'environnement et de retrouver un fonctionnement optimal en utilisant la connaissance issue des interactions avec l'utilisateur.

Nous avons au préalable proposé dans le chapitre 3 un modèle simulant les réactions de l'utilisateur afin de pouvoir tester et comparer nos algorithmes. Nous avons considéré l'hypothèse que l'utilisateur peut intervenir librement avec le système, puisque même si le système n'offre pas de possibilité de contrôle, nous savons que l'utilisateur contourne souvent cette limite, en utilisant par exemple des équipements d'appoint. En nous appuyant sur les résultats produits par la communauté scientifique, notamment issus du génie civil, nous avons considéré l'hypothèse que l'utilisateur s'adapte à son environnement et réagit de façon à rétablir son confort. Pour concevoir ce modèle, nous nous sommes appuyés sur une propriété mise en avant par la communauté scientifique : la mémoire sensorielle. Cette propriété signifie que les sensations de l'utilisateur ne dépendent pas uniquement de la valeur courante du signal, mais aussi de la suite des valeurs de signal menant à celle-ci. Cette propriété nous a amené à considérer que la mémoire du passé de l'utilisateur affecte ses réactions futures. L'idée est que l'utilisateur a tendance à intervenir plus souvent quand les valeurs du signal passées lui sont inconfortables. Cela implique que pour une même valeur, l'utilisateur peut réagir différemment suivant ses interactions passées avec le système. Cela crée des difficultés algorithmiques particulières en termes d'apprentissage puisque la conséquence qui en découle est que l'environnement est non stationnaire. Le problème que nous traitons est de savoir s'il est possible de contrôler le signal lumineux du bâtiment de façon à créer des conditions confortables pour l'utilisateur avec le minimum d'énergie

possible. Cet objectif représente un défi, car l'utilisateur peut réagir librement sur la valeur du signal si ce dernier se retrouve dans une situation d'inconfort. En outre, nous utilisons ce modèle de l'utilisateur afin de comparer les approches que nous avons proposées.

Dans les chapitres 4 et 5, nous avons effectué des expériences sur le profil énergétique du système (sur la base de paradigmes utilisés ensuite pour l'apprentissage) en nous reposant sur les hypothèses mentionnées dans le paragraphe précédent sur le comportement de l'utilisateur qui s'alignent sur la littérature du confort adaptatif. Ces expériences nous ont permis de tirer des conclusions très fortes sur le comportement énergétique de l'utilisateur et sa relation avec le système. Elles suggèrent l'existence d'une stratégie optimale en termes énergétique assurant un équilibre entre l'économie d'énergie et le confort de l'utilisateur. Elles indiquent qu'intégrer le confort de l'utilisateur dans le modèle d'apprentissage du système de contrôle est nécessaire afin d'optimiser l'énergie dans le contexte d'un pilotage énergétique réaliste de bâtiment connecté. Nous pouvons retrouver des éléments algorithmiques intéressants, car même si l'idée de trouver un équilibre entre performances énergétiques et confort de l'utilisateur fait l'objet d'un consensus global, beaucoup de personnes (dans le monde industriel comme dans celui de la recherche scientifique) l'entendent dans le sens où il serait nécessaire de faire un *compromis* entre ces deux objectifs, qu'ils sont en opposition. Cela sous-entend qu'il est possible de considérer un choix de pilotage énergétique comme une sélection (humaine, vraisemblablement) parmi des optima de Pareto (en effet, l'optimum de Pareto est un état dans lequel le système n'est pas uniformément améliorable, et où il faut dégrader l'un des objectifs pour améliorer l'autre). Les résultats de nos expériences montrent que cette hypothèse semble infirmée dans le cadre du modèle de l'utilisateur que nous considérons, et ceci d'après les simulations effectuées avec ce modèle dont la conception repose sur des hypothèses validées par l'état de l'art. Nous avons, en effet, constaté expérimentalement l'existence d'un optimum (au sens d'extremum d'un espace convexe). En d'autres termes, cela implique qu'il n'existe pas d'opposition inhérente entre ces deux éléments dans le contexte que nous considérons ici. L'usage de l'apprentissage dans notre problématique met aussi en évidence les efforts de recherche qu'il a fallu fournir afin de proposer une fonction économique devant intégrer le confort pour atteindre l'objectif mentionné précédemment. Cela reflète une particularité de notre problématique qui est la différence entre l'objectif réel de minimiser l'énergie (trouver la stratégie minimisant l'énergie), et l'objectif que nous avons proposé pour l'apprentissage. Et cela s'explique de façon très naturelle dans le contexte du confort adaptatif : si le système choisit des valeurs trop faibles, l'utilisateur aura tendance à rétablir son confort, ce qui provoque une surconsommation énergétique. Ainsi, dans le contexte d'un pilotage énergétique se basant sur l'apprentissage dans lequel l'utilisateur a la possibilité d'interagir librement avec le système, la prise en compte de l'utilisateur par le système est nécessaire pour réduire les coûts liés à l'exploitation du bâtiment.

L'objectif de la thèse étant de concevoir un algorithme qui apprend à faire évoluer le signal de façon satisfaisante pour l'utilisateur, et ce, en consommant le moins d'énergie possible, nous avons étudié deux paradigmes, reflétant deux manières différentes de concevoir le fonctionnement interne du système de contrôle : le paradigme par choix de variation du signal et le paradigme par choix de valeur du signal. La différence entre ces deux paradigmes réside dans le niveau de granularité temporelle de la décision du système. Dans le système par choix de variation du signal, l'idée est de choisir la pente que prend le signal lorsque celui-ci diminue ainsi que la valeur sur laquelle le

signal doit s'arrêter. Le système a un ensemble de trajectoires possibles pour les valeurs du signal au cours du temps et il devra trouver quelle est la trajectoire satisfaisant ses critères de performances. Dans le paradigme par choix de valeur du signal, le système décide de la valeur sur laquelle doit être fixé le signal. La décision du système porte sur l'étape courante, elle est itérative et elle change potentiellement à chaque fois que le système reprend la main. Par ailleurs, la décision est effectuée sur la base d'informations courantes. L'évaluation expérimentale de ces deux paradigmes s'est principalement positionnée sur la vitesse de convergence, la qualité de convergence ainsi que l'évolution de l'énergie et l'évolution de l'état de l'utilisateur. Sur la base de la comparaison expérimentale de différents algorithmes pour ces deux paradigmes, nous avons proposé des mécanismes permettant d'accélérer la convergence de ces algorithmes.

Concernant le système par choix de variation du signal étudié dans le chapitre 4, les conclusions varient suivant le type de système considéré. Pour l'agent apprenant la pente à appliquer au signal, le meilleur algorithme (en termes de vitesse de convergence et de qualité de convergence) semble être celui basé sur la poursuite hiérarchique. Compte tenu du fait que l'état de l'utilisateur change durant la baisse du signal, ce résultat ne semble pas surprenant. Puisque les méthodes du type poursuite se basent sur le principe d'optimisation locale, elles ont tendance à être plus stables. Leurs performances dépendent moins des propriétés de l'environnement. Pour l'agent des valeurs d'arrêt, UCB semble donner de meilleurs résultats. La mémoire étant homogénéisée quand la valeur du signal est fixée, l'environnement s'apparente à un environnement stationnaire pour une valeur fixe du signal. Précisons tout de même que ces résultats sont valables uniquement dans le contexte considéré. En effet, dans le cas où des éléments supplémentaires sont pris en compte (tel que les conditions extérieures et l'activité de l'utilisateur), des simulations doivent être effectuées pour valider ce résultat. En plus de la comparaison entre les différents algorithmes de la littérature, nous avons étudié l'impact du nombre d'actions sur les performances du système. Cette étude a révélé qu'il y a un compromis à trouver entre le nombre d'actions d'un côté et la qualité de convergence ainsi que la vitesse de convergence de l'autre. Il est apparu qu'il est non trivial de trouver ce compromis et de paramétrer le système d'apprentissage. Les premiers résultats obtenus suggèrent le besoin d'utiliser les fonctions d'approximation afin de mitiger ce compromis. Il faudra s'intéresser plus amplement à ces méthodes et comparer leurs performances.

Nous avons proposé dans le chapitre 5 une modélisation à base d'états pour le système par choix de valeur du signal. Même si le système ne connaît pas l'état réel dans lequel se trouve l'utilisateur et son environnement physique, il est néanmoins possible de l'estimer à travers les interactions de l'utilisateur. La valeur actuelle du signal offre aussi une bonne indication sur l'état de l'utilisateur. Sur la base de cette représentation, nous avons comparé les algorithmes les plus classiques de l'apprentissage par renforcement à base d'état : Q-learning et SARSA. Le résultat le plus frappant est la convergence rapide et stable de ces deux algorithmes et ceci quelle que soit la mémoire de l'utilisateur considérée (court ou long terme). Ceci montre bien que les algorithmes d'apprentissage par renforcement à base d'état ont la possibilité d'exploiter l'information (même partielle) afin de guider correctement l'apprentissage de l'agent. L'algorithme Q-learning montre néanmoins de meilleures performances que SARSA et ceci pour toutes les métriques de performances considérées. Nous montrons aussi, par notre étude sur la discrétisation des valeurs du signal, l'existence d'un compromis entre vitesse de convergence et qualité de la solution apprise qu'il est difficile à paramétrer correctement. Là

encore, nous nous attendons à des gains de performances si nous utilisons les méthodes basées sur les fonctions d'approximation.

Suite à une étude approfondie de ces deux paradigmes, nous avons conclu que le paradigme par choix de variation du signal est meilleur selon les hypothèses formulées, et ce, malgré la simplicité apparente de cette approche. L'une des raisons pouvant expliquer ce résultat est la combinatoire des actions qui est plus simple à gérer dans ce paradigme. Ceci montre le potentiel des méthodes d'apprentissage par renforcement pour le contrôle lumineux et ouvre de nouvelles perspectives sur son application dans un contexte réaliste tenant compte d'éléments divers sur l'utilisateur et son environnement physique. Cependant, l'un des défis qu'il faut prendre en compte est la vitesse de convergence de ces algorithmes. Il paraît nécessaire de trouver des façons de mieux paramétrer ces algorithmes via l'inclusion de connaissances a priori au niveau du système. Pour ce faire, des données d'exploitation doivent être disponibles afin de générer la politique de départ à partir de ces données. En cas d'indisponibilité de ces données, un environnement de simulation doit être créé à partir d'un modèle d'utilisateur simple de la littérature. L'initialisation de la politique dépendra bien entendu du type d'algorithme utilisé.

Dans la suite de cette thèse, nous proposons des algorithmes mieux adaptés à nos hypothèses de contrôle afin d'améliorer les performances. En effet, nous avons utilisé les algorithmes prévus généralement pour les environnements stationnaires. Si nous laissons à l'agent la possibilité d'exploiter les informations collectées sur les réactions de l'utilisateur, nous pouvons espérer un gain plus important des performances de ces algorithmes. Par exemple, l'agent peut mémoriser la fréquence d'intervention et l'énergie associée à chaque stratégie et s'appuyer sur ces valeurs afin de tenter de délimiter les régions intéressantes.

Un des prolongements évidents de ces travaux est une implémentation en conditions réelles de nos algorithmes, afin d'assurer empiriquement la pertinence de notre démarche et renforcer nos contributions scientifiques. Nous envisageons actuellement une telle implémentation dans le bâtiment connecté de CESI. Toutefois, l'implémentation d'un tel système fait face à des difficultés que nous pouvons isoler sur la base de notre travail actuel. Tout d'abord, les conditions météorologiques doivent être favorables à un usage raisonnable de lumière. En effet, si l'utilisateur ne ressent pas assez le besoin d'utiliser la lumière artificielle, les conclusions de l'expérience peuvent être biaisées. Néanmoins, la difficulté la plus remarquable est constitutive des algorithmes d'apprentissage par renforcement : le fait que l'algorithme ait besoin d'explorer continuellement pour pouvoir s'adapter. En effet, l'agent risque de choisir de mauvaises stratégies, poussant l'utilisateur à interagir beaucoup trop fréquemment avec le bâtiment. Dans le cadre de nos expériences, l'agent retrouve éventuellement un comportement adéquat en essayant les autres actions. En revanche, dans la réalité, un tel comportement risque de constituer un problème non négligeable de validation des résultats de notre système et nous mettrait aussi face à des difficultés éthiques liées au bien-être des utilisateurs. En effet, suivant les conclusions du confort adaptatif, l'utilisateur pourrait complètement contourner le système de contrôle, comme nous l'avons mentionné à plusieurs reprises. Ceci serait assez détrimentaire à la consommation d'énergie. Une façon naïve d'assurer une implémentation correcte serait simplement de paramétrer l'algorithme de façon à privilégier le confort de l'utilisateur à l'économie d'énergie dans la récompense, à prendre des taux d'exploration très faible ou rapidement décroissant et choisir un



taux d'apprentissage assez élevé pour garantir une convergence rapide. Un algorithme de la sorte pourrait donner l'illusion de fonctionner correctement pendant un certain temps, si les conditions environnementales ne changent pas trop au cours du temps. Il est certain aussi que le comportement d'un tel algorithme ne sera pas stable en raison du manque d'exploration de l'agent. De plus, l'économie d'énergie réalisée risque d'être extrêmement faible par rapport à celle atteinte par la stratégie optimale. Au-delà de la question sur l'exploration, il y a aussi la question du temps de convergence qui est généralement assez prohibitif dans les algorithmes d'apprentissage par renforcement. Et ceci sans compter le fait que l'algorithme devra aussi intégrer dans son fonctionnement divers éléments sur l'utilisateur ainsi que son environnement physique tel que : le nombre d'utilisateurs, l'activité des utilisateurs, la luminosité extérieure, l'état des stores et possiblement des prédictions météorologiques et des prédictions sur l'occupation du bâtiment. L'intégration de tous ces éléments introduit de la complexité dont la gestion nécessitera l'usage de méthodes basées sur les fonctions d'approximation.

À ce sujet, l'usage de fonction d'approximation telle que l'apprentissage profond nécessite des quantités non négligeables de données ainsi que l'infrastructure matérielle et logicielle pouvant les prendre en charge. L'apprentissage en ligne n'étant pas envisageable et compte tenu du caractère imprévisible des réactions de l'utilisateur, nous soutenons l'idée que toute implémentation basée sur l'apprentissage profond doit reposer (au moins en partie) sur un modèle de l'utilisateur construit en simulation. Pour concevoir un tel modèle et entraîner nos algorithmes sur ce modèle, des données d'exploitation devant contenir les réactions de l'utilisateur doivent être collectées. En utilisant l'apprentissage supervisé sur ces données, nous pourrions produire un modèle raisonnable prédisant le confort des utilisateurs et sur lequel entraîner l'agent en simulation. Par la suite, il sera possible d'appliquer ensuite le *transfert learning* ([49]) afin d'entraîner l'agent en situation réelle, l'idée étant de figer certains paramètres du modèle et laisser l'algorithme s'entraîner seulement avec les autres paramètres. Ainsi, une vision se basant sur le confort personnalisé sera nécessaire pour produire un environnement de simulation servant à entraîner l'agent pour le contrôle lumineux. Cette vision de l'apprentissage permettra à terme de réaliser les objectifs du confort personnalisé.

Rappelons que le confort personnalisé représente une vision générale du confort adaptatif. Elle repose sur l'idée que les conditions de confort doivent être assurées non pas à travers un modèle général, mais à travers de modèles qui seraient ajustés pour des utilisateurs spécifiques. Poussée à bout, l'idée du confort personnalisé devra permettre de produire des modèles capables de s'adapter aux besoins de n'importe quel individu en s'appuyant uniquement sur les données de comportements des utilisateurs. Ainsi, même en présence de comportements particuliers des utilisateurs (différentes activités au sein d'un même bureau, différentes périodes de congés), le système saura toujours s'adapter et même anticiper ces comportements en vue d'assurer un contrôle lumineux adapté aux préférences des uns et des autres. Face à de telles possibilités, le confort personnalisé pose aussi des défis par rapport à la confidentialité de données devant être adressés de manière approfondie.

Par ailleurs, il paraît pertinent d'étendre nos travaux au contrôle thermique, compte tenu de l'impact énergétique considérable de cette application. Or, contrairement aux phénomènes lumineux, les phénomènes thermiques font intervenir des latences et des non-linéarités qui rendent difficile toute prédiction sur l'évolution de l'environnement, et sur son impact sur le comportement de l'utilisateur ainsi que sur ses sensations. Au-delà de ce point, le calcul même de l'énergie consommée durant un certain

laps de temps et en fonction des actions de pilotage serait difficile à réaliser. Ainsi, en plus d'un modèle complexe de l'usager, un modèle thermodynamique et un modèle de consommation énergétique complexe devront être pris en compte dans l'environnement de simulation. Ces modèles devront être implémentés dans un environnement logiciel offrant la possibilité d'effectuer des simulations à la demande faisant intervenir tous les composants du bâtiment et permettant de connaître avec exactitude l'évolution thermique du bâtiment, sa consommation énergétique ainsi que son empreinte carbone, et ce, pour différents scénarios. Même s'il n'existe pas encore de logiciel ouvert et simple d'utilisation offrant ces services, les ressources matérielles et logicielles des ordinateurs actuels ainsi que l'expertise dont nous disposons à l'issue de nos travaux laissent suggérer que le développement d'un tel système est envisageable. Il nécessitera cependant des modèles physiques particulièrement détaillés et sophistiqués, qui devront collaborer (dans une approche de cosimulation), ce qui fait de cet objectif un sujet de recherche en génie civil à part entière.

Finalement, compte tenu du caractère récent des thèmes relatifs au bâtiment intelligent et de la complexité des principes sous-jacents à ces thèmes, nous notons que l'exploitation efficace de bâtiment intelligent nécessite une expertise scientifique considérable. Cette expertise devra se recouper sur plusieurs volets. Elle devra se concentrer sur les aspects physiques de la gestion du bâtiment, mais aussi sur la collecte et l'organisation des données, la mise en place et le paramétrage des algorithmes de contrôle ainsi que le suivi des performances de ces algorithmes et les interventions devant être effectuées dans le cas où des problèmes surviendraient durant cette phase d'exploitation. Compte tenu du caractère récent des sujets liés au bâtiment intelligent, et l'expertise pour gérer ce type de bâtiments étant très pluridisciplinaire, il n'existe pas encore de filières de formation ni de programmes académiques capables de développer le niveau d'expertise requis pour une telle gestion d'un bâtiment intelligent. L'existence de telles formations permettrait de créer une réelle différence sur l'implémentation à grande échelle de bâtiment intelligent et représente alors un enjeu aussi important que l'étude des principes algorithmiques sous-jacents à celle-ci.

# Annexe A

## Méthodes classiques d'apprentissage par renforcement

L'apprentissage par renforcement regroupe une famille de méthodes d'apprentissage permettant à un agent donné de résoudre des problèmes de contrôle dans son environnement. L'agent peut effectuer plusieurs actions, et il poursuit un objectif donné sous la forme d'une fonction à maximiser. Il doit apprendre à amener son environnement dans une situation qui maximise son objectif. Dans l'apprentissage par renforcement, l'agent n'apprend pas seulement à prédire l'évolution de l'environnement, mais plutôt à agir dans cet environnement de sorte à maximiser son objectif. Il va effectuer cet apprentissage sur la base de ses expériences passées, et plus il accumule de l'expérience, plus il s'améliore dans la poursuite de son objectif.

Il existe plusieurs types d'apprentissage par renforcement : l'apprentissage par renforcement basé sur les états dans lequel l'environnement change à chaque fois que l'agent exécute une action (changement que l'apprentissage prend en compte) et l'apprentissage par renforcement sans état dans lequel l'environnement ne change pas.

Dans ce qui suit, nous présentons une vue d'ensemble sur les différents types de méthodes d'apprentissage par renforcement ainsi que leurs propriétés. Nous commençons par introduire les concepts liés à l'apprentissage par renforcement sans états, puis nous présentons l'apprentissage par renforcement basé sur les états.

### A.1 Apprentissage par renforcement sans états

Il est possible d'exprimer le problème d'apprentissage par renforcement sans état comme un problème de bandit manchot [9]. Dans ce problème, on considère un agent qui a en face de lui  $n$  bandits manchots. À chaque fois que l'agent joue un bandit, un gain est retourné par ce dernier. L'objectif de l'agent est de maximiser le gain cumulé apporté par ces bras (dans la pratique, ceci revient à identifier et à jouer le bras ayant le plus grand gain). L'agent fait face à une difficulté particulière qui est que le gain associé à chaque bras n'est pas déterministe. De plus, les caractéristiques statistiques de ces gains sont complètement inconnues de l'agent. La seule manière qu'a l'agent de connaître le gain d'un bras est de faire plusieurs essais sur ce bras afin d'avoir une estimation assez précise sur le gain de ce bras. Construire des estimations fiables du gain de chaque bras permet à l'agent

de trouver comment choisir les bras de façon à maximiser son gain. Précisons que le terme *action* est utilisé pour se référer au choix d'un bras et que les termes *récompense* et *valeur* peuvent être utilisés de manière indifférenciée pour désigner le gain.

Nous pouvons déjà remarquer que les caractéristiques du problème de bandit manchot servent à modéliser des situations dans lesquelles l'agent a très peu de connaissances a priori. Les problèmes qui pourraient être exclus de ce cadre sont, par exemple, les problèmes qui sont complètement déterminés par des règles simples (problème de recherche), ou les problèmes dont la formulation intègre des éléments stochastiques pouvant facilement être modélisés à l'aide de distributions de probabilité. Dans les deux cas, un modèle peut être exploité afin de déduire l'action qui optimise le mieux l'objectif de l'agent (effectivement, il existe des alternatives à l'apprentissage par renforcement [92]). Dans le problème du bandit, la machine à sous est complètement inconnue de l'agent et seuls des essais (amenant forcément à des échecs plus ou moins récurrents) pourraient trancher sur la manière de choisir les bras. C'est la caractéristique principale et différenciante des algorithmes d'apprentissage par renforcement (par rapport aux autres types d'algorithmes d'apprentissage tel que l'apprentissage supervisé).

Il existe une multitude d'algorithmes d'apprentissage par renforcement sans états. Ces algorithmes viennent de directions de recherches assez différentes. Dans cette sous-section, nous allons passer en revue les différentes approches proposées dans la littérature. Les différences principales que nous pouvons souligner se recoupent sur deux volets : la représentation de la **politique de l'agent** ainsi que la manière d'apporter des réponses au dilemme **exploration vs exploitation** caractérisant l'apprentissage par renforcement. La politique de l'agent regroupe l'algorithme ainsi que les structures de données utilisées par l'agent dans le processus la sélection des actions. L'agent devra apprendre à équilibrer entre la tendance à préférer les actions ayant apporté le gain le plus conséquent (l'exploitation) et la tendance à vouloir essayer les autres actions dans l'espoir que celles-ci soient meilleures en termes de gain (l'exploration). La mise à jour de la politique de l'agent devra prendre en compte ce dilemme.

Dans l'apprentissage par renforcement, une des approches possibles que nous distinguons est celles où l'agent apprend à choisir la meilleure action en se basant sur les estimations empiriques des récompenses des actions. Dans ces approches, les estimations des récompenses des actions permettent de représenter la politique de l'agent. Ceci signifie que ces estimations servent elles-mêmes à choisir l'action qui devra être appliquée par l'agent. Mais il n'est pas possible de faire de l'**exploration** en utilisant uniquement les estimations des récompenses des actions de l'agent. Si l'on se repose uniquement sur ces estimations (qui sont calculées à partir d'essais sur un nombre fini d'étapes), l'agent pourrait finir par choisir une mauvaise action. En effet, si une action n'a pas été appliquée suffisamment de fois, l'estimation de sa récompense risque d'être incertaine, ce qui pourrait empêcher l'agent de discerner entre la meilleure action et les autres actions.

La question qui s'ensuit est de savoir comment créer de l'exploration à partir seulement des estimations de l'agent. Il y a là aussi plusieurs manières pour répondre à cette question, reflétant des façons différentes de voir l'apprentissage par renforcement. La manière la plus simple pour y parvenir est d'introduire de l'aléatoire à partir des estimations des actions de l'agent. L'agent, au

lieu de se limiter au choix de l'action qui a été la plus prometteuse par le passé, va essayer de ponctuellement d'autres actions afin de voir si celles-ci sont vraiment moins bonnes que la meilleure action actuelle. Il y a globalement deux manières simples d'introduire de l'aléatoire dans le choix des actions,  $\epsilon$ -greedy [116], et l'exploration de Boltzmann [116]. L'approche  $\epsilon$ -greedy choisit avec une faible probabilité notée  $\epsilon$  une action aléatoire parmi les actions disponibles, mais la plupart du temps (avec une probabilité  $1 - \epsilon$ ), c'est l'action correspondant à la meilleure estimation qui sera choisie. Dans l'exploration de Boltzmann, chaque action est associée à un poids. L'agent choisit l'action à prendre par tirage aléatoire sur un vecteur stochastique calculé à partir de ces poids et des estimations des récompenses des actions. Le défaut principal de ces deux méthodes est que le *regret* (qui est l'écart entre la récompense optimale et la récompense perçue par l'agent) s'accroît linéairement en fonction du temps [9].

Un autre paradigme communément adopté en apprentissage par renforcement se caractérise par ce qu'on appelle des mécanismes d'*optimisme face à l'incertitude* [9]. L'idée est de jouer plus souvent les actions pour lesquelles l'estimation de la récompense est incertaine, afin de donner à l'agent la possibilité d'explorer ces actions. Plus l'agent est incertain sur la valeur réelle de la récompense d'une action, plus grande est la chance que cette action soit bel et bien l'action optimale, et plus il devient important d'explorer cette action. Il existe deux manières d'implémenter le principe d'optimisme face à l'incertitude : une dans laquelle la certitude est modélisée explicitement dans l'agent, et une autre où elle ne l'est pas. Dans chacune des deux manières, il y a aussi plusieurs approches à considérer.

Nous avons d'abord les méthodes n'utilisant pas d'*à priori* explicite sur les valeurs des actions comme l'algorithme UCB (Upper Confidence Bound) [9] et d'autres méthodes qui en utilisent comme UCB Bayésien [6]. L'idée de l'UCB (Upper Confidence Bound) est d'associer à chaque action un bonus traduisant la certitude sur l'estimation de sa récompense. Plus l'estimation de la récompense d'une action est incertaine (à savoir, moins elle a été prise), plus la valeur de ce bonus est grande. Ce sont ces nouvelles valeurs (estimations des récompenses plus bonus) qui seront utilisées pour choisir l'action courante (car elles représentent des "bornes supérieures" sur les valeurs réelles des récompenses des actions). L'algorithme de l'UCB se base des théorèmes généraux de la théorie des probabilités.

D'autres méthodes se reposent sur des *à priori* concernant les valeurs possibles des actions comme l'UCB Bayésien [6]. D'autres encore, n'implémentent pas explicitement le principe d'*optimisme face à l'incertitude*, mais choisissent l'action courante en fonction de la probabilité que l'action soit la meilleure, conditionnellement sur les connaissances de l'agent, c'est le *matching de probabilité* [120]. Finalement, on distingue aussi les méthodes qui modélisent explicitement les informations accumulées par l'agent durant l'apprentissage. Les informations de l'agent sont vues comme l'état d'un processus de décision markovien où le but est de maximiser la récompense de l'agent en se basant sur ses connaissances actuelles sur son environnement. Ces méthodes sont appelées méthodes par *espace d'états d'information* [48].

Nous distinguons aussi les algorithmes venant de la littérature de la théorie des automates [92], on peut citer LRI (Linear Reward Inaction) [119] et LRP (Linear Reward Penalty) [119]. Ces algorithmes sont parmi les premiers algorithmes d'apprentissage par renforcement et sont apparus

dans les années soixante. Dans ces algorithmes, la politique est toujours représentée explicitement à l'aide d'un vecteur stochastique qui associe à chaque action la probabilité de choix de l'action (contrairement aux approches basées sur la valeur où les probabilités de choix sont construites à partir des estimations des récompenses de l'agent). Le choix de l'action se fait par tirage aléatoire sur le vecteur stochastique. À chaque fois que l'agent applique une action et reçoit une récompense, l'agent met à jour le vecteur stochastique de façon à améliorer sa politique. Le principe est le même pour tous les algorithmes, seule change l'action à mettre à jour ainsi que la manière de la mettre à jour. L'avantage de cette formulation de l'agent est qu'il n'est pas nécessaire d'implémenter un mécanisme incitant l'agent à diversifier le choix des actions, car les actions sont choisies de manière aléatoire. La question d'équilibrer entre exploration et exploitation se réduit seulement à la question de savoir comment mettre à jour le vecteur de probabilité ainsi que la fréquence des mises à jour, car l'exploration est déjà intégrée dans la manière de représenter la politique.

## A.2 Apprentissage par renforcement basé sur les états

Dans ce qui suit, nous traitons les algorithmes d'apprentissage par renforcement basés sur les états. Comme pour la formulation de l'apprentissage par renforcement sans états, l'agent dispose d'un ensemble d'actions et l'objectif est le même, choisir les actions maximisant la récompense reçue de l'environnement. La différence principale avec la formulation précédente est que l'agent évolue dans un environnement dynamique. À chaque fois que l'agent choisit une action et l'applique dans l'environnement, l'environnement change, réagit en fonction de l'action et retourne *une récompense immédiate*. La récompense immédiate reflète dans quelle mesure l'action a été bénéfique pour l'agent. Dans ce cas, l'agent devra apprendre à maximiser la récompense à long terme (c'est-à-dire, le cumul des récompenses, ce qui n'est pas la même chose que d'optimiser la récompense immédiate). Nous pouvons dire globalement que la différence principale entre l'apprentissage par renforcement sans états et avec états, se résume dans le fait que les actions de l'agent affectent les récompenses que l'agent reçoit dans le futur. Au niveau formel, la différence entre les deux types d'apprentissage par renforcement est que la distribution de la récompense change à chaque fois qu'une action est appliquée, et que ce changement dépend des actions choisies par l'agent.

Dans l'apprentissage par renforcement basé sur les états, l'agent perçoit (partiellement ou entièrement) la situation dans laquelle se trouve l'environnement. Et l'état va contenir les informations dont a besoin l'agent pour prendre ses décisions. Nous précisons dans les paragraphes suivants, les implications principales qu'amènent les différences distinguant cette variante de l'apprentissage par renforcement de celle où l'on n'a pas d'états. Déjà, nous notons que l'agent ne choisit plus une seule action, mais que celui-ci devra choisir une succession d'actions (les actions optimales) lui permettant d'atteindre son objectif de maximisation de gain. L'action à choisir à chaque étape dépend naturellement de la situation dans laquelle se trouve l'environnement. Par exemple, un robot aspirateur (explorant l'environnement qu'il doit nettoyer) réagira différemment s'il y a un obstacle dans son passage. Le robot se déplacera de sorte à l'éviter alors qu'il ira tout droit s'il n'y a rien qui ne le gêne dans son chemin.

En plus du dilemme *exploration-exploitation* de la formulation sans états de l'apprentissage par renforcement (dilemme venant de l'incertitude sur l'effet des actions sur l'environnement), il y a certains éléments à prendre en compte dans l'apprentissage par renforcement à base d'états qui ne sont pas présents dans la première formulation. En effet, comme pour la première formulation, l'agent devra choisir les bonnes actions et il ne sait pas lesquelles le sont à moins de les avoir essayés, mais la principale difficulté de l'apprentissage par renforcement à base d'états est que les choix présents ont une influence sur les états futurs.

Ainsi, même si une action ne semblant pas retourner une forte récompense est choisie à l'instant présent, celle-ci pourrait mener à des récompenses considérables dans le futur. L'agent peut choisir de sacrifier la récompense à court terme pour avoir une grande récompense à long terme (nous pouvons voir cela comme une sorte d'investissement que l'agent devra apprendre à faire). Il y a aussi des applications pour lesquelles les récompenses ne trouvent des valeurs non nulles que vers la fin (nous pouvons mentionner les jeux de plateaux tel que le jeu d'échec ainsi que le jeu de go, où le résultat de la partie n'est connu que quand celle-ci se termine). Du point de vue de l'agent, nous disons dans ce cas qu'il y a *des délais* dans la réception de la récompense. Cette difficulté se recoupe avec *l'exploration-exploitation* et l'amplifie : afin de connaître quelles sont les bonnes actions, l'agent devra essayer différentes actions dans différents états et isoler celles qui le mènent vers les bons états et qui lui assurent à terme une récompense élevée. Il peut y avoir des états pour lesquelles le choix de l'action n'a que peu d'importance, et d'autres états qui mènent à des récompenses favorables uniquement quand une action spécifique est choisie. D'autres états seraient déterminants pour atteindre l'objectif de l'agent et dans lesquels il faut d'être prudent par rapport au choix de l'action à prendre. Tout cela doit être pris en compte par l'agent.

Un troisième élément nécessitant d'être mentionné concernant l'apprentissage par renforcement à base d'états et qui est la conséquence des deux premières, c'est ce qu'on appelle *le problème d'affectation de crédit* [116]. L'idée du *problème d'affectation de crédit* [116] est de déterminer, parmi les actions choisies par l'agent, l'action où la séquence d'actions qui sont les plus déterminantes dans l'obtention des récompenses. Par exemple, isoler les coups les plus décisifs dans une partie d'échec. Comme nous l'avons dit auparavant, l'apprentissage par renforcement prend son intérêt dans les modalités où l'agent ne connaît rien de son environnement. L'agent ne connaît donc pas quels états sont les plus importants et de ce fait, il ignore aussi quelles actions seraient cruciales dans l'obtention des récompenses.

Dans l'apprentissage par renforcement, le principe général consiste à ce que l'agent apprenne à prendre les bonnes actions *pour différentes situations dans lesquelles pourrait se trouver l'environnement*. Si la dimension nécessaire pour décrire ces situations (qu'on appelle généralement *état*) est grande, l'algorithme pourra mettre extrêmement longtemps à apprendre. Et ceci nous mène au quatrième élément à mentionner distinguant l'apprentissage par renforcement par états de l'apprentissage par renforcement sans états : *le fléau de dimensionnalité* (curse of dimensionality [116]). Le fléau de dimensionnalité désigne les difficultés qui surviennent quand l'agent traite des données (sur son expérience) de dimension très élevée. En effet, le nombre d'états dans lequel peut se trouver l'environnement augmente de manière exponentielle en fonction du nombre de valeurs caractérisant l'état. Mais la raison principale du fléau de dimensionnalité vient de la complexité des

problèmes rencontrés. Dans la pratique, cette complexité fait que l'agent aura besoin d'énormément d'informations pour apprendre. Ces informations doivent être présentes en quantité suffisante, mais doivent aussi être assez variées pour refléter les différents aspects du problème traité par l'agent. D'un autre côté, l'agent aura besoin de ressources considérables en temps de calcul et en mémoire afin de pouvoir exploiter ces données. Pour déjouer le *fléau de dimensionnalité*, l'idée est de pouvoir généraliser les éléments appris sur un nombre limité de situations, à des situations jamais vues par l'algorithme. L'apprentissage supervisé est utilisé à cette fin (problème de généralisation [88]). Pour ce faire, l'agent maintient un modèle paramétré (dont le nombre de paramètres est inférieur au nombre d'états). Quand l'agent interagit avec son environnement, ce modèle apprend à ajuster ses paramètres en fonction des données observées de son expérience avec l'environnement. Nous disons dans ce cas qu'on utilise *une fonction d'approximation* pour représenter la politique. Mais il est généralement difficile d'utiliser les fonctions d'approximation dans le contexte de l'apprentissage par renforcement : puisque les données sont *corrélées dans le temps* et ne viennent donc pas de manière i.i.d (indépendantes et identiquement distribuées), l'agent aura tendance à ajuster ses poids de façon à mémoriser les expériences les plus récentes aux dépens des expériences passées.

Les éléments mentionnés précédemment caractérisant l'apprentissage par renforcement sans états : le dilemme exploration-exploitation, le dilemme récompense précoce vs récompenses tardives, le problème d'affectation de crédit, le fléau de dimensionnalité ainsi que la dépendance temporelle des données doivent être adressés de manière équilibrée et efficace par l'agent afin que celui-ci arrive à trouver les actions optimales. Ces difficultés sont d'autant plus complexes que l'agent devra être capable d'y faire face en apprenant à optimiser son objectif à partir de son expérience avec l'environnement.

Il existe plusieurs types d'algorithmes permettant d'apporter une réponse à ces problèmes. Tout comme dans les algorithmes sans états, la représentation de la politique peut varier. Il y a des algorithmes qui représentent la politique directement, généralement l'apprentissage se fait par descente de gradient pour ces algorithmes [105]. Il y a aussi les algorithmes qui maintiennent des estimations pour les valeurs des actions (ou des états). Ces algorithmes ne représentent pas la politique explicitement, mais celle-ci s'y trouve tout de même de manière implicite [86]. Dans ces algorithmes, l'agent utilise ses estimations afin de choisir l'action à prendre en se reposant, en outre, sur une stratégie d'exploration. Il existe aussi des algorithmes qui combinent l'utilisation des estimations des valeurs des actions et une représentation explicite de la politique [72]. À l'inverse des algorithmes sans états, il y a aussi des algorithmes qui connaissent (ou apprennent) un modèle de la dynamique de l'environnement [111]. En se reposant sur un modèle de l'environnement, l'agent peut choisir l'action en considérant les évolutions possibles que peut prendre l'environnement. L'agent peut aussi générer des expériences simulées à partir de son modèle afin de renforcer l'apprentissage.

## A.2.1 Interface entre l'agent et environnement

Comme nous l'avons mentionné, l'apprentissage par renforcement consiste pour l'agent à apprendre par interaction avec l'environnement les actions à prendre afin d'atteindre son objectif. L'agent a à sa disposition un ensemble d'actions et il doit choisir à chaque étape de l'apprentissage



l'action qu'il doit prendre pour atteindre son objectif qui est de maximiser la récompense. L'environnement change à chaque fois que l'agent applique *l'action* qu'il aura choisi, puis il retourne à l'agent *la récompense immédiate* ainsi que son prochain *état* (ou *une observation* liée à son état). Dans cette sous-section, nous décrivons tous ces éléments en détail et formalisons les concepts cités précédemment : *l'action*, *l'observation*, *la récompense immédiate*.

Dans ce qui suit, nous formalisons les notions dont nous avons parlé jusqu'ici de manière intuitive. À chaque étape  $t$  de l'apprentissage, l'action exécutée par l'agent est dénotée  $A_t$ . La récompense immédiate est définie par un scalaire  $R_t$  qui représente le bénéfice dû à la dernière action choisie par l'agent. Dans la formulation la plus générale de l'apprentissage par renforcement, la récompense est considérée comme stochastique, elle est représentée par une variable aléatoire qui dépend du temps (même si dans la pratique la récompense est souvent déterministe). La récompense indique la performance de l'agent à l'étape  $t$ . La récompense est une valeur scalaire qui dépend de l'environnement et de l'action. L'objectif de l'agent étant de maximiser la récompense, pas uniquement dans l'instant présent, mais également dans le futur, l'agent devra apprendre à maximiser le cumul de la récompense (présente et future). Comme il a été dit précédemment, l'agent pourra donc sacrifier la récompense à court terme pour avoir une forte récompense à long terme. L'hypothèse sous-jacente à l'apprentissage par renforcement est que, dans chaque problème impliquant la réalisation d'une tâche par un agent, quelque soit l'objectif de l'agent, nous pouvons toujours formuler cet objectif en termes de maximisation du cumul de l'espérance de la récompense. L'agent devra donc apprendre à sélectionner les actions lui permettant d'atteindre son objectif.

Comme nous l'avons mentionné auparavant, *l'état* décrit la situation dans laquelle se trouve l'environnement de l'agent. Et comme nous le verrons plus tard, dans l'apprentissage par renforcement, la notion d'état fait souvent intervenir des propriétés qui en simplifient la formulation. À cet égard, nous parlons souvent d'*états Markovien*. Mais avant de décrire la notion d'état, nous devons d'abord introduire certaines notions sur lesquelles nous nous reposerons afin de définir l'état et qui sont *l'observation* et *l'historique*. L'observation à l'étape  $t$  notée  $O_t$  correspond à ce que perçoit l'agent de l'environnement à cette étape. L'historique, quant à lui, est la séquence de toutes les observations, actions et récompenses depuis le tout début de l'interaction de l'agent avec son environnement. Par exemple, pour une voiture autonome, cela peut être l'image à l'instant courant des caméras avant et arrière de la voiture. L'évolution de l'apprentissage dépend de l'historique et l'état représente l'information nécessaire à l'agent lui permettant de déterminer cette évolution (dans le cadre de la tâche que nous voulons que l'agent apprenne à effectuer). D'un point de vue formel, l'état est décrit en fonction de l'historique  $S_t = f(H_t)$ . En général, l'état peut être n'importe quelle fonction faisant intervenir l'historique des observations et des actions de l'agent. Mais, si nous voyons l'environnement comme une sorte de simulateur, ce simulateur aurait un état  $S_t^e$  qui servirait à générer la prochaine observation, c'est l'état réel de l'environnement. Quant à l'agent, il aurait à sa disposition un autre état  $S_t^a$  qu'il calcule à partir de l'historique et qui contient l'information accessible à l'agent utilisée pour choisir ses actions. Notons que cette distinction entre état de l'environnement et état de l'agent est faite à titre indicatif seulement afin de clarifier ce à quoi pourrait correspondre cette notion d'état ainsi que sa dépendance à l'historique des observations et des actions. Dans les faits, seul l'état de l'agent nous intéresse, car nous ne connaissons jamais l'état réel de l'environnement. Précisons finalement que tous les éléments cités précédemment ayant trait à l'environnement sont

stochastiques. Ceci concerne la récompense  $R_t$ , l'observation  $O_t$  et l'état  $S_t$ . L'apprentissage par renforcement à base d'états est montré dans la figure A.1<sup>1</sup>.

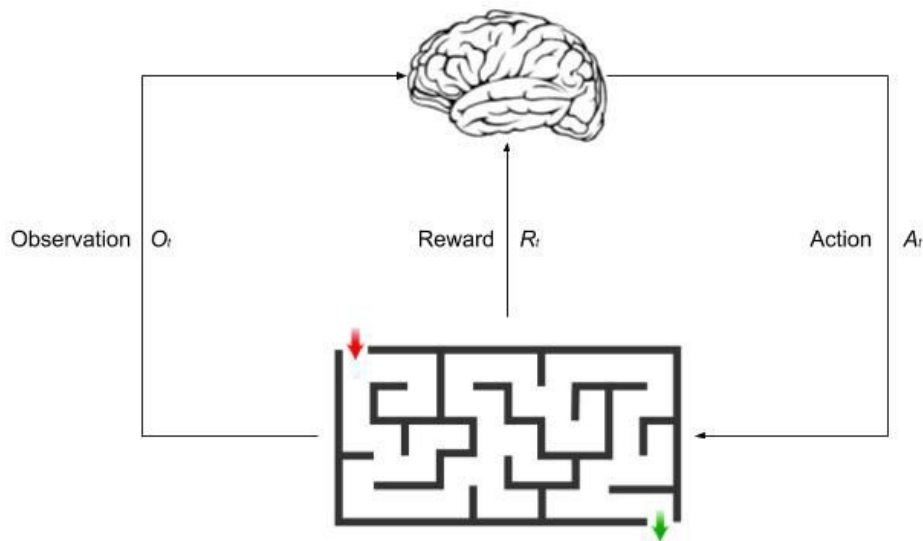


Figure A.1 – Schéma représentant l'apprentissage par renforcement à base d'états. Dans cet exemple, nous considérons un robot qui veut se déplacer dans un labyrinthe pour en sortir. L'observation  $O_t$  correspond aux valeurs retournées par les capteurs de l'agent à cet instant. Et l'état de l'agent  $S_t^a$  peut être pris comme étant le vecteur composé des valeurs des  $n$  dernières observations de l'agent, ou alors une reconstruction d'une carte de ce labyrinthe à partir de ces observations. Alors que l'état réel  $S_t^e$  de l'environnement est le labyrinthe : c'est la position de toutes les particules composant ce labyrinthe.

En revanche, même si nous ne connaissons pas l'état réel de l'environnement, il convient de poser des définitions permettant de nous informer si la représentation à état de l'agent contient assez d'information utile pour permettre l'apprentissage. À ce sujet, un état est dit *markovien* si l'état futur est indépendant des états passés compte tenu de l'état présent. Formellement, la propriété de Markov est exprimée comme suit :

**Definition A.2.1.** Un état  $S_t$  est Markovien ssi :

$$P[S_{t+1} | S_t] = P[S_{t+1} | S_1, \dots, S_t]$$

Cette définition implique que l'agent pourra toujours prendre des décisions pertinentes et informées lui permettant d'accomplir son objectif en se basant uniquement sur les informations qui lui sont disponibles actuellement. L'agent n'a pas besoin de l'historique complet pour guider le processus de sélection des actions. Dans l'exemple de la voiture autonome, nous pourrions par exemple mettre dans l'état les 10 dernières images des caméras avant et arrière. Si nous voulons créer un agent conversationnel permettant d'interagir efficacement avec les visiteurs d'un site internet, l'état pourrait être constitué des dernières phrases écrites par le visiteur du site. Dans tous les cas, le mieux est que l'état contienne assez d'information pour que l'apprentissage par renforcement soit théoriquement possible. Si l'état markovien est connu par l'agent, l'environnement est dit *complètement*

1. Source : [towardsdatascience.com/reinforcement-learning-an-introduction-to-the-concepts-applications-and-code-ced6fbfd882d](https://towardsdatascience.com/reinforcement-learning-an-introduction-to-the-concepts-applications-and-code-ced6fbfd882d)

observable [116]. Mais il existe des formulations de l'apprentissage par renforcement où l'état n'est pas connu par l'agent, dans ce cas, nous disons que l'environnement est *partiellement observable* [116]. Des exemples d'environnement partiellement observables incluent des jeux tels que le poker. Dans le cadre de cette thèse, nous nous intéressons essentiellement aux environnements complètement observables, sauf mention contraire.

À chaque état, l'agent a le choix entre plusieurs actions possibles, ce choix est fait à l'aide de *la politique de l'agent*. La politique notée  $\pi$  permet de déterminer l'action que l'agent va prendre, elle décrit le comportement de l'agent dans son environnement. La politique peut être déterministe ou stochastique. Dans le cas où la politique est déterministe, la politique est une fonction qui retourne, pour chaque état, l'action que choisit l'agent dans cet état. Dans ce cas,  $\pi(s) = a$  signifie que l'agent prend l'action  $a$  à l'état  $s$ . Si la politique est stochastique, la politique est une distribution de probabilité de sélection de l'action conditionnée par l'état. Nous pouvons bien entendu représenter une politique déterministe avec une politique stochastique, et donc nous considérons que les actions  $A_t$  sont aussi stochastiques. Dans ce cas,  $\pi(a | s) = \mathbb{P}[A_t = a | S_t = s]$  représente la probabilité de prendre l'action  $a$  dans l'état  $s$ . La politique peut aussi être paramétrée par un paramètre qu'on note  $\theta$ , dans ce cas la politique sera notée  $\pi_\theta$ . Par exemple, le paramètre  $\theta$  pourrait être représenté par un vecteur servant au calcul de l'espérance et de la variance de la distribution  $\pi_\theta$  conditionnée sur l'état. Après avoir formalisé l'interaction entre l'agent et son environnement ainsi que formaliser l'interface les séparant, nous décrivons dans ce qui suit la manière de formaliser l'environnement ainsi que la manière dont il change.

## A.2.2 L'environnement : processus de décision Markovien

Comme nous l'avons dit auparavant, l'état de l'environnement représente la situation dans laquelle il se trouve. L'état change à chaque fois qu'une action est appliquée à l'environnement et ce changement est décrit par une probabilité conditionnée sur l'état courant et l'action courante. L'environnement est formellement décrit par un processus de décision Markovien. Un processus de décision Markovien est une chaîne de Markov auquel on rajoute la fonction de récompense ainsi que les actions. Les définitions de ces notions [13] sont présentées dans ce qui suit :

### Chaîne de Markov

Une chaîne de Markov est un couple  $M = \{\mathcal{S}, P\}$  où :

- $\mathcal{S}$  est l'espace d'états discret ou continu.
- $P$  est la fonction de transition qui est une matrice stochastique contenant les probabilités de passer d'un état à l'autre. La probabilité de transition de l'état  $s$  vers l'état  $s'$  à l'instant  $t$  est notée  $P_{ss'} = \mathbb{P}[S_{t+1} = s' | S_t = s]$ .

### Processus de décision Markovien

Le processus de décision Markovien contient tous les éléments d'une chaîne de Markov auquel on rajoute un ensemble d'actions ainsi que la fonction de récompense. Le processus de décision Markovien est un 4-uplet  $M = \{\mathcal{S}, \mathcal{A}, P, R\}$  :

- $\mathcal{S}$  est l'espace d'état qui peut être discret ou continu.
- $\mathcal{A}$  est l'ensemble des actions qui peut être discret ou continu aussi.
- $P$  est la fonction de transition. Contrairement aux probabilités de transition dans les chaînes de Markov, celle-ci dépend de l'action en plus de l'état courant. La probabilité de transition de l'état  $s$  vers l'état  $s'$  quand une action  $a$  est appliquée s'écrit comme suit  $P_{ss'}^a = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$ .
- $R$  est la fonction de récompense. Elle retourne, pour chaque couple d'état-action  $(s, a) \in \mathcal{S} \times \mathcal{A}$ , l'espérance de la récompense immédiate obtenue après application de l'action  $a$  à l'état  $s$ . Elle est définie formellement par  $R : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$  avec  $R_a^s = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$ .

Il y a des liens intéressants entre les processus de décision Markoviens et les chaînes de Markov. Nous pouvons voir que si nous fixons la politique de l'agent, les probabilités de sélection des actions ne changent pas au cours du temps, le processus de décision markovien induit donc une chaîne de Markov. Pour cette nouvelle chaîne de Markov, la probabilité notée  $P'$  de passer de l'état  $s$  vers l'état  $s'$  est

$$P_{ss'} = \mathbb{P}[S_{t+1} = s' \mid S_t = s] = \sum_{a \in \mathcal{A}} \pi(a \mid s) \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a] \quad (\text{A.1})$$

Nous pouvons aussi remplacer le processus de décision markovien par une chaîne de Markov en considérant que les états de la nouvelle chaîne de Markov sont les couples d'états-actions  $(s, a) \in \mathcal{S} \times \mathcal{A}$ . La probabilité  $P''$  de transitionner de l'état  $(s, a)$  vers l'état  $(s', a')$  s'écrit alors

$$P''_{(s,a)(s',a')} = \mathbb{P}[(S_{t+1}, A_{t+1}) = (s', a') \mid (S_t, A_t) = (s, a)] = \pi(a' \mid s') \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a] \quad (\text{A.2})$$

Dans le cas où l'état n'est pas observable, nous intégrons les observations dans le processus de décision markovien, ce qui donne le processus de décision markovien partiellement observable dont la définition est donnée comme suit :

### Processus de décision Markovien partiellement observable

Le processus de décision Markovien partiellement observable est composé de tous les éléments d'un processus de décision Markovien auquel on ajoute l'ensemble des observations ainsi que les probabilités d'émission. Les probabilités d'émission sont les probabilités d'avoir les observations sachant l'état. Dans les environnements modélisés par les processus de décision Markovien partiellement observables (POMDPs), l'agent n'a pas accès à l'état, mais seulement à l'observation. Le processus de décision markovien partiellement observable est un 4-uplet  $M = \{\mathcal{S}, \mathcal{A}, \mathcal{O}, P, \mathcal{Z}, R\}$  :

- $\mathcal{S}$  est l'espace d'état qui peut être discret ou continu.
- $\mathcal{A}$  est l'ensemble des actions qui peut être discret ou continu.
- $\mathcal{O}$  est l'ensemble d'observation. Les observations de l'agent ne sont pas Markoviennes.
- $P$  est la fonction de transition, elle est définie exactement comme pour les processus de Markov. On a donc  $P_{ss'}^a = \mathbb{P}[S_{s+1} = s' \mid S_t = s, A_t = a]$ .
- $\mathcal{Z}$  est une matrice contenant les probabilités d'émission. À chaque instant  $t$ , la probabilité d'émission de l'observation  $o$  est la probabilité d'observer  $o$  sachant que l'état est  $s$  et que l'action est  $a$ , elle est notée comme suit  $\mathcal{Z}_{s'o}^a = \mathbb{P}[O_{s+1} = o \mid S_t = s', A_t = a]$ .

- $R$  est la fonction de récompense,  $R_a^s = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$  représente l'espérance de la récompense quand l'agent applique une action donnée dans un état donné.

### A.2.3 Tâches et objectif en apprentissage par renforcement

Nous avons posé les éléments décrivant l'interaction entre l'agent et l'environnement, nous avons aussi exposé les outils qui servent à décrire l'environnement ainsi que les changements opérant dans celui-ci. Dans cette partie, nous spécifions l'objectif de l'apprentissage par renforcement en commençant par poser les notions dont nous avons besoin. La formulation de l'objectif de l'apprentissage par renforcement dépend du type de tâche traité.

À cet égard, on peut distinguer deux types de tâches. Nous avons, d'abord, les tâches qui ont une durée finie avec un début et une fin qui se nomment *tâches à horizon fini* ou *tâches épisodiques*, et pour lesquelles la durée de l'apprentissage est limitée à un horizon  $T$  qui est le nombre d'étapes de la tâche. Notons que  $T$  est une variable aléatoire qui varie potentiellement d'une exécution à l'autre. Une instance de l'exécution d'une tâche épisodique est appelée *épisode*. La suite d'états, d'actions et de récompenses (les transitions) composant l'épisode d'une tâche épisodique se nomme *trajectoire*. Formellement, dans les tâches épisodiques, l'environnement atteint un état terminal en un temps fini. Les tâches épisodiques incluent par exemple un robot réparant les parties d'une pièce cassée ou bien le jeu d'échec. Il y a aussi des tâches pour lesquelles nous ne pouvons pas spécifier de fin. Ces tâches sont exécutées de manière continue par l'agent. Ce type de tâches est appelée *tâche à horizon infini*, l'agent agissant de manière perpétuelle dans son environnement. Pour les tâches à horizon infini, il n'y a naturellement ni début ni fin. Nous voyons que dans ce cas, l'objectif peut diverger vers l'infini s'il est pris comme étant naïvement la somme de toutes les récompenses obtenues durant l'apprentissage.

Il existe heureusement plusieurs manières de formuler l'objectif d'apprentissage par renforcement qui fonctionnent tout autant pour les tâches à horizon fini que pour les tâches à horizon infini. L'une des formulations permettant d'éviter la divergence de l'objectif est de prendre l'espérance du cumul de récompense divisé par le nombre d'étapes. Cette formulation est appelée *l'espérance du cumul de récompense par pas de temps*. Pour une politique fixée, si la chaîne de Markov induite par les paires d'état-actions est *ergodique* [13] (toutes les paires d'état-actions sont atteignables à partir de n'importe quelle autre paire d'état-actions) et non-périodique, il existe une distribution stationnaire sur les paires d'état-actions de la chaîne de Markov [13]. Ceci veut dire que la probabilité d'atteindre une paire d'état-action ne change plus après un certain temps. Nous pourrions alors évaluer les performances de la politique sur cette distribution. Dans ce cas, les performances de la politique sont asymptotiquement équivalentes à la récompense immédiate obtenue dans chaque paire d'état-action pondérée par la probabilité d'atteindre cette paire d'état-action. Nous détaillerons ce point vers la fin de cette sous-section.

Nous pouvons aussi partir de l'idée que nous ne nous soucions pas des récompenses trop lointaines dans le futur. Dans ce cas, la récompense de chaque étape est multipliée par un facteur diminuant exponentiellement en fonction du temps. Notons ce facteur  $\gamma$ , et considérons les récompenses obtenues à partir de l'instant  $t$ . Nous voulons évaluer les performances de l'agent. La récompense

obtenue pour chaque instant  $t + k$ ,  $R_{t+k}$  vaudra  $\gamma^k$  fois moins et sera donc remplacée par  $\gamma^k R_{t+k}$  dans le cumul des récompenses. Ceci donne

$$R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \gamma^4 R_{t+5} + \dots$$

Les deux objectifs présentés précédemment ainsi que tous les éléments servant à évaluer les performances et l'efficacité des algorithmes d'apprentissage par renforcement sont présentées dans ce qui suit.

**Definition A.2.2.** *Le retour est la somme totale des récompenses (possiblement diminuée par un facteur  $\gamma$  à chaque étape) défini comme suit,*

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

La constante  $\gamma$  s'appelle le *facteur de discount*. L'intérêt du facteur de discount est de paramétrer l'objectif de façon à exprimer dans quelle mesure nous nous intéressons aux récompenses présentes (par rapport aux récompenses futures). Nous pouvons aussi retrouver le premier objectif si nous posons  $\gamma = 1$ . Si on choisit  $\gamma = 0$ , l'agent ne considère plus que la récompense immédiate. L'usage de  $\gamma$  est mathématiquement pratique, il permet d'éviter la divergence de l'objectif ainsi que de faciliter les preuves de convergence de certains algorithmes. La formulation de l'objectif avec le facteur de discount nous permet de prendre en compte la certitude quant aux récompenses à venir. Par exemple, nous pouvons attribuer une valeur faible à ce facteur si nous sommes moins sûrs des récompenses qu'on aura dans le futur. Une autre manière d'interpréter le paramètre  $\gamma$  est de dire qu'à chaque instant la tâche a une probabilité de  $1 - \gamma$  de prendre fin (l'environnement a une probabilité de  $1 - \gamma$  de passer de l'état courant vers l'état terminal). Si nous partons de ce principe, nous obtenons, à partir de la formulation originale du processus de Markov, un nouveau processus de Markov équivalent au premier (les politiques ont la même utilité). Seules les probabilités de transitions et la fonction de récompenses changent. Dans le nouveau processus de Markov, la probabilité de passer de l'état  $s$  à l'état  $s'$  suite à l'application d'une action  $a$  est de  $\tilde{P}_{ss'}^a = \gamma P_{ss'}^a$ . L'objectif dans le deuxième processus de Markov est le même que dans le premier, mais sans discount.

Le retour permet de définir d'autres notions essentielles aux algorithmes d'apprentissage par renforcement. Nous avons, en premier lieu, *la fonction de valeur des états* qui représente, pour chaque état, à quel point il est bon pour l'agent d'être dans cet état si la politique  $\pi$  est suivie par l'agent. Nous avons aussi *la fonction de valeurs des actions* (aussi nommée *Q-valeurs*) qui détermine l'impact à long terme d'une action donnée dans un état donné, si une certaine politique  $\pi$  est appliquée par l'agent. Les deux définitions qui suivent posent de manière formelle ces deux concepts.

**Definition A.2.3.** *La fonction de valeur des états notée  $v_{\pi}(s)$  est l'espérance du retour obtenu partant d'un état  $s$  si l'agent suit la politique  $\pi$  ensuite,*

$$v_{\pi}(s) = \mathbb{E}_{\pi}[G_t \mid S_t = s]$$

**Definition A.2.4.** *La fonction de valeurs des actions notée  $q_{\pi}(s, a)$  est l'espérance du retour, si l'environnement est dans l'état  $s$ , l'agent applique l'action  $a$  puis la politique  $\pi$  pour les étapes suivantes,*

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a]$$

Comme nous l'avons mentionnée auparavant, pour les tâches épisodiques (à horizon fini de valeur  $T$ ) l'objectif peut être formulé simplement comme étant l'espérance du retour divisé par le nombre d'étapes (la moyenne du retour par pas de temps). La définition suivante formalise ce point.

**Definition A.2.5.** *L'espérance du cumul de récompense par pas de temps pour une politique  $\pi$  est*

$$J_\pi = \frac{\sum_{t=1}^T \mathbb{E}[R_t]}{T} = \frac{\mathbb{E}[G_1]}{T} = \frac{v_\pi(s_0)}{T}$$

ou  $\gamma = 1$  et  $s_0$  est l'état initial.

Pour les tâches à horizon infini, il est aussi possible de formuler un objectif partant de l'idée de la moyenne du retour par pas de temps. Nous avons dit auparavant que pour une politique  $\pi$  de l'agent, le processus de Markov induit une chaîne de Markov sur les paires d'état-actions. Pour une politique  $\pi$  fixée, si cette nouvelle chaîne de Markov est ergodique et non-périodique, alors l'algorithme converge vers une distribution stationnaire sur les paires d'état-actions [13]. En notant cette distribution stationnaire  $P_\pi$ , l'objectif précédent est équivalent à l'espérance de la récompense immédiate suivant cette distribution  $P_\pi$ , comme le montre les formules suivantes :

$$J_\pi = \lim_{T \rightarrow \infty} \frac{\sum_{t=1}^T \mathbb{E}[R_t]}{T} = \mathbb{E}_{(s,a) \sim P_\pi(s,a)}[R_a^s]$$

## A.2.4 Politique optimale

Comme nous l'avons dit auparavant, le but des algorithmes d'apprentissage par renforcement est d'accomplir la tâche spécifiée par la maximisation de la récompense à long terme. Globalement, ceci veut dire qu'il faut trouver la politique optimale parmi toutes les politiques possibles. Nous présentons, dans cette sous-section, les éléments permettant de définir la politique optimale. Nous pouvons définir, à partir des politiques réalisables dans le processus de décision markovien, un ordre partiel de ces politiques. Nous pouvons dire par exemple qu'une politique  $\pi$  est au moins aussi bonne qu'une politique  $\pi'$  si et seulement si  $\pi$  a une valeur au moins aussi grande que  $\pi'$  pour tous les états. Cette propriété est formalisée comme suit :

$$\pi \geq \pi' \iff v_\pi(s) \geq v_{\pi'}(s) \quad \forall s \in S$$

Suivant cette définition, les politiques optimales sont celles atteignant la plus grande valeur pour tous les états. Dans les environnements markoviens, il existe toujours une politique déterministe optimale. L'objectif de l'agent est de trouver cette politique optimale. Les politiques optimales ont la même fonction de valeurs des états ainsi que la même fonction de valeurs des actions. Nous avons les définitions suivantes concernant les performances de la politique optimale.

**Definition A.2.6.** *La fonction de valeurs optimale notée  $v_*(s)$  est la fonction de valeur qui pour chaque état associé la valeur maximale réalisable depuis cet état parmi toutes les politiques possibles.*

$$v_*(s) = \max_{\pi} v_\pi(s)$$

**Definition A.2.7.** La fonction optimale des valeurs des actions notée  $q_*(s, a)$  est la fonction de valeur des actions qui associe à chaque paire d'état-action la valeur maximale possible réalisable suite à l'application de l'action dans l'état

$$q_*(s, a) = \max_{\pi} q_{\pi}(s, a)$$

## Les équations de Bellman

Les fonctions de valeur (des états et des actions) ainsi que les retours sont liés les uns aux autres et satisfont des relations récursives fondamentales dans la théorie des algorithmes d'apprentissage par renforcement. Ces équations reflètent des conditions de cohérence dans les fonctions de valeur permettant de mettre en place des algorithmes itératifs d'apprentissage par renforcement pouvant converger vers un point fixe. L'application répétée de ces relations renforce ainsi la cohérence de ces équations et mène l'algorithme progressivement vers la solution optimale. Ces équations sont *les équations de Bellman* [116]. Nous présentons dans cette partie ces équations qui caractérisent les fonctions de valeur. Les algorithmes se basant sur ces équations seront présentés dans la section 2.2.

Les équations de Bellman ont toutes la même forme globale, elles sont obtenues simplement en décomposant les formules des valeurs en récompense immédiate suivie de la valeur de l'élément qui suit (état, paire d'état d'action). L'équation suivante montre l'équation de Bellman pour les valeurs d'états. Elle exprime l'idée que la valeur d'un état  $s$  pour une politique  $\pi$  est l'espérance de la récompense immédiate reçue suite à l'application de l'action courante, additionnée avec la valeur de l'état suivant multiplié par le facteur de discount  $\gamma$ .

$$v_{\pi}(s) = \mathbb{E}_{\pi}[R_{t+1} + \gamma v_{\pi}(S_{t+1}) \mid S_t = s] \quad (\text{A.3})$$

L'équation de Bellman pour les valeurs des actions est écrite de manière similaire à la précédente, c'est l'espérance de la récompense achevée par l'action  $a$  additionnée à la q-valeur de l'action suivante à l'état suivant.

$$q_{\pi}(s, a) = \mathbb{E}_{\pi}[R_{t+1} + \gamma q_{\pi}(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a] \quad (\text{A.4})$$

Dans les deux équations, l'espérance fait intervenir les récompenses reçues qui dépendent de manière probabiliste des actions ainsi que des états traversés par le processus de décision markovien. Les sources de l'aléatoire viennent donc de la politique (l'action choisie) ainsi que l'environnement. Dans le détail, ces équations donnent, pour les fonctions de valeur

$$v_{\pi}(s) = \sum_{a \in A} \pi(a \mid s) (R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_{\pi}(s')) \quad (\text{A.5})$$

et pour les fonctions de valeur des actions, nous aurons,

$$q_{\pi}(s, a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \sum_{a' \in A} \pi(a' \mid s') q_{\pi}(s', a') \quad (\text{A.6})$$

Il est aussi possible d'écrire les valeurs des états en fonction des valeurs des actions, c'est la somme des probabilités des actions fois la valeur atteinte par ces actions à l'état courant,

$$v_{\pi}(s) = \sum_{a \in A} \pi(a \mid s) q_{\pi}(s, a) \quad (\text{A.7})$$



Nous pouvons aussi exprimer les valeurs des actions en fonction des valeurs des états. La valeur d'une action  $a$  est la somme de l'espérance de la récompense immédiate associée à l'action  $a$  plus la somme des valeurs des états suivants pondérées par la probabilité de transitionner vers ces états partant de l'état courant  $s$ .

$$q_{\pi}(s, a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_{\pi}(s') \quad (\text{A.8})$$

Toutes les équations présentées ici sont les *équations d'espérance de Bellman* [116]. Les équations de Bellman s'appliquent aussi pour les valeurs des états et les valeurs des actions associées à la politique optimale. Mais vu qu'elles correspondent aux meilleures valeurs possibles, elles peuvent être exprimées sans références à aucune politique en particulier. Elles se basent sur *le principe d'optimalité* de la programmation dynamique stipulant que la valeur d'une solution optimale peut être écrite sous forme de solutions partielles d'un problème de plus petite taille. Elles sont décrites dans les équations suivantes,

$$v_*(s) = \max_{a \in A} \sum_{s', r} p(s', r | s, a) [r + \gamma v_*(s')] \quad (\text{A.9})$$

Pour les valeurs optimales des actions l'équation s'écrit comme suit,

$$q_*(s, a) = \sum_{s', r} p(s', r | s, a) [r + \gamma \max_{a'} q_*(s', a')] \quad (\text{A.10})$$

Ces équations s'appellent les *équations d'optimalité de Bellman* [116]. Si nous disposons des valeurs optimales des états, nous pourrions très facilement trouver la politique optimale. Pour chaque état, la politique optimale correspond simplement à l'action ayant la plus grande valeur. Pour la trouver, tous les états doivent être parcourus pour calculer, pour chaque action, la valeur obtenue si cette action est prise et que la politique optimale est suivie. Pour un état donné, l'action ayant la plus grande valeur correspond à la meilleure action (choisie par la politique optimale). Nous pouvons voir à partir des formules précédentes que nous aurons besoin du modèle pour trouver la meilleure politique à partir seulement des valeurs optimales des états. Nous pouvons nous affranchir de la nécessité d'exploiter le modèle si nous avons les valeurs optimales des actions. En effet, la valeur de chaque action serait déjà calculée et il suffit de sélectionner pour chaque état l'action ayant la plus grande q-valeur. Avoir les q-valeurs permet d'éviter de faire une recherche avec le modèle de l'environnement. Le désavantage d'utiliser les q-valeurs est que celles-ci nécessitent plus d'espace de stockage que les valeurs des états. Cette tension entre représentation de la politique à l'aide des fonctions de valeurs des états et les fonctions de valeur des actions se rencontre assez souvent en apprentissage par renforcement.

## A.2.5 Architecture et types d'algorithmes d'apprentissage par renforcement

Les algorithmes d'apprentissage par renforcement ont généralement la même structure. Ils sont constitués de trois blocs essentiels intervenant de manière itérative dans l'apprentissage. Nous avons premièrement la partie de l'algorithme qui *collecte les données*, cette partie mémorise l'expérience (simulée ou réelle) de l'agent (et qui va servir à l'apprentissage). Deuxièmement, il y a la partie

dont le rôle est *d'estimer* l'effet de la politique de l'agent sur l'environnement (à travers une fonction de valeurs ou un modèle de l'environnement). Elle repose donc sur les données récoltées dans le premier bloc. Et finalement, la dernière partie de l'algorithme vise à *améliorer* la politique de l'agent. L'idée est de modifier légèrement la politique actuelle pour obtenir une nouvelle politique qui soit localement meilleure que la première. Ces trois étapes sont appliquées itérativement (et pas nécessairement de manière purement séquentielle) dans tous les algorithmes d'apprentissage par renforcement jusqu'à atteindre la meilleure politique. Tous les algorithmes que nous présentons ont ces trois composantes, ils se différencient seulement par la manière dont ces composantes sont implémentées.

Les trois blocs, montrés dans la figure A.1<sup>2</sup>, composant tout algorithme d'apprentissage par renforcement, font ressortir deux problèmes que l'algorithme devra traiter correctement. Le premier consiste à prédire la récompense apportée par une politique prédéfinie  $\pi$ , qu'on appelle *problème de prédiction*. Ce problème consiste globalement à estimer la politique  $\pi$  et évaluer son impact pour l'agent. Le second, appelé *problème de contrôle*, consiste à trouver la meilleure politique à partir des estimations actuelles de l'agent. Les algorithmes pour résoudre ces deux problèmes sont similaires, dans le sens où si l'on sait déjà estimer l'effet d'une politique, il est possible d'obtenir un algorithme d'apprentissage par renforcement en intégrant à cela un mécanisme permettant d'améliorer la politique.

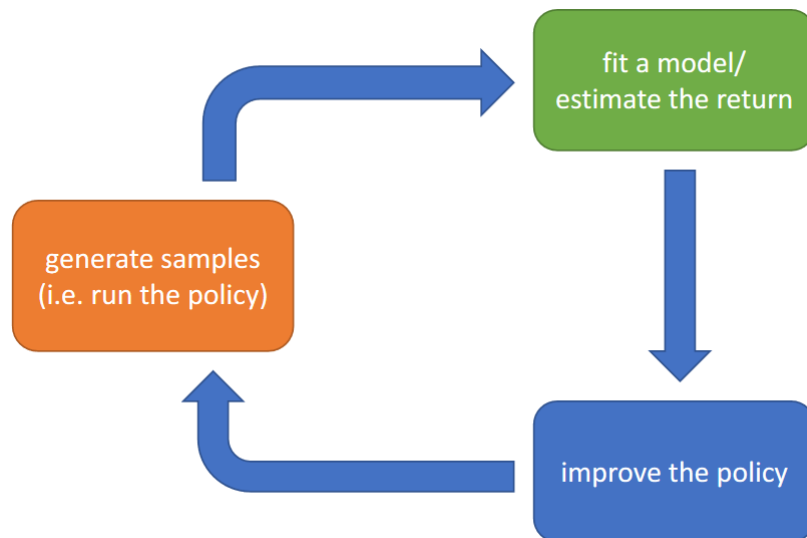


Figure A.2 – Architecture des algorithmes d'apprentissage par renforcement à base d'états.

Comme nous l'avons dit auparavant, il existe plusieurs types d'algorithmes dans l'apprentissage par renforcement. Nous avons tout d'abord, les méthodes qui représentent explicitement la politique de l'agent par une distribution de probabilité paramétrée. Ces méthodes sont appelées les méthodes *basées sur la politique*. Parmi ces méthodes, nous notons les méthodes par *policy gradient* [105]. Le principe général de ces méthodes est d'exécuter une politique puis se baser sur ses performances afin d'estimer son gradient et améliorer la politique. Il existe par ailleurs les méthodes *basées sur les valeurs* [115]. Ces méthodes estiment directement les valeurs des actions et l'utilisent pour choisir

2. Source : <https://rail.eecs.berkeley.edu/deeprlcourse/static/slides/lec-4.pdf>

l'action à prendre ainsi que pour améliorer la politique. Notons aussi les méthodes présentant une combinaison des deux approches : l'utilisation d'une représentation explicite de la politique ainsi que des fonctions de valeur. Les méthodes de type *acteur-critique* [72] suivent ce schéma. Les méthodes de type *acteur-critique* sont analogues aux méthodes par *policy gradient*, la différence principale entre les deux types d'approches est que dans le cas de l'acteur-critique, l'agent utilise les fonctions de valeur (des états et/ou des actions) afin d'améliorer sa politique. Les approches d'apprentissage par renforcement peuvent aussi être distinguées selon que l'agent intègre *un modèle de l'environnement* dans l'apprentissage ou non [111]. Afin de faciliter l'apprentissage, certains algorithmes modélisent (ou apprennent) comment l'environnement change en réaction aux actions et ceci pour différentes configurations dans lesquelles pourraient se trouver l'environnement. Les points abordés dans ce qui a précédé sont à prendre en compte dans le choix de l'algorithme. Chaque type d'algorithme a ses avantages et ses inconvénients, nous présentons dans ce qui suit tous les éléments intervenant dans la sélection de l'algorithme.

## Compromis concernant les différents algorithmes d'apprentissage par renforcement

Comme nous l'avons vu antérieurement, il existe énormément de façon de concevoir les algorithmes d'apprentissage par renforcement, les points à considérer concernent la façon dont on implémente les trois blocs essentiels de ces algorithmes (la collecte des données, l'estimation de la politique et l'amélioration de la politique). Tous ces points rendent le choix d'un algorithme complexe, et ce choix doit se faire sur la base de compromis posés par la diversité des algorithmes existants.

Parmi tous les compromis à considérer, nous avons *l'efficacité d'échantillonnage* (anglicisme de *sample efficiency*) qui est la quantité de données requise pour que l'algorithme puisse obtenir une politique assez efficace. Les algorithmes peuvent varier en termes d'efficacité d'échantillonnage et ce point est généralement en opposition avec deux autres éléments qui sont *la stabilité et la facilité d'utilisation* (nous détaillons ce point plus loin). Il y a énormément d'éléments qu'il faut prendre en compte dans les algorithmes d'apprentissage par renforcement, nous pouvons citer le dilemme *exploration-exploitation*, l'utilisation de fonction d'approximation, la mise à jour de la politique, la collecte des données. Tous ces éléments introduisent des hyperparamètres supplémentaires. Les algorithmes d'apprentissage par renforcement varient en termes de propriété de convergence et ceci rend certains algorithmes plus faciles à configurer que d'autres. Aussi, ces algorithmes ne sont pas forcément prévus pour le même contexte (les hypothèses sous-jacentes à ces algorithmes peuvent être très différentes). Nous pouvons noter des différences au niveau de l'environnement qui peut être déterministes ou stochastiques, l'espace d'actions (et l'espace des états) qui peut être continu ou discret, ou bien la tâche (qui peut être à horizon fini ou infini). Et comme précisé précédemment, la représentation de la politique est un élément clé dans les algorithmes d'apprentissage par renforcement.

Dans certaines situations, il est plus simple de représenter la politique directement par une distribution de probabilité sur les actions à prendre (comme pour le problème du *walking robot* [117]). Dans d'autres situations, il est à l'inverse plus simple de représenter la politique à l'aide des valeurs des actions (comme pour le problème du sac à dos). La question d'intégrer un modèle de l'environ-

nement à l'apprentissage intervient aussi dans le choix des algorithmes, la question étant de savoir s'il est plus simple d'apprendre un modèle qui sera utilisé par l'agent (comme pour le jeu de go [111]) ou alors d'apprendre la politique directement par interaction avec l'environnement (les jeux Atari [86]). Il existe toujours des compromis à prendre en compte en fonction des situations, il n'y a pas d'algorithme parfait, et opter pour certaines caractéristiques désirables de l'algorithme impose forcément en sacrifier d'autres. Dans ce qui suit, nous allons détaillons comment tous les éléments mentionnés précédemment interviennent dans la sélection de l'algorithme.

En ce qui concerne *l'efficacité d'échantillonnage*, il est généralement préférable que l'agent soit en mesure d'apprendre à exécuter la tâche à partir de données passées (ou idéalement de données qui ne sont pas générées par l'agent lui-même, par exemple des données simulées ou des données produites par des démonstrateurs humains ou par un autre agent). Si un algorithme est capable d'apprendre (à évaluer sa politique et à l'améliorer) sans avoir besoin de générer des données d'interaction avec *sa propre politique*, l'algorithme est dit *off-policy* [116]. La capacité des algorithmes *off-policy* à apprendre à partir d'échantillons de données (d'expérience) de *diverses* sources, est ce qui les rend efficaces en termes de nombre d'échantillons nécessaire pour apprendre. Un algorithme est dit *on-policy*, si, au contraire, l'agent utilisant cet algorithme est capable d'apprendre, à partir seulement de données générées par *sa* politique actuelle. Dans les algorithmes *on-policy* [116], dès que la politique change, ne serait-ce que légèrement, de nouvelles données doivent être produites sur la politique (en l'appliquant sur l'environnement) afin de pouvoir l'améliorer encore. Il est dit que *la politique comportementale* est la même que la politique apprise pour les algorithmes *on-policy* (et le contraire vaut pour les algorithmes *off-policy*). Les algorithmes *off-policy* prennent leur intérêt principalement quand l'agent n'a pas la possibilité d'interagir librement avec son environnement. Dans ce cadre, il est dit que l'apprentissage s'effectue *hors-ligne*. Si, au contraire, l'agent interagit librement avec l'environnement, nous disons que l'apprentissage s'effectue *en-ligne* [116].

Les méthodes d'apprentissage par renforcement utilisant un modèle de l'environnement pour l'apprentissage sont les plus efficaces en termes d'échantillons. Leurs performances dépendent moins de l'origine des données étant donné que celles-ci disposent déjà d'un modèle qui est censé être capable de prédire les changements potentiels de l'environnement en réponse aux actions appliquées. À l'inverse, les méthodes par *policy gradient* [105] (ou plus généralement les méthodes basées sur le principe de l'optimisation locale) sont complètement *on-policy* car elles ont besoin d'estimer les performances de la politique actuelle avant de pouvoir continuer de l'améliorer. Notons que le temps de calcul d'un algorithme ne se résume pas à son *efficacité d'échantillonnage*. Par exemple, si l'on dispose d'un modèle de l'environnement, exploiter ce modèle pour rechercher la meilleure politique nécessite beaucoup trop de ressources et pourrait prendre beaucoup plus de temps qu'apprendre la politique directement par interaction avec l'environnement. Par ailleurs, dans le cas où il est possible de simuler l'interaction avec l'environnement, il pourrait être plus efficace d'utiliser des algorithmes *on-policy* car si l'on a la possibilité de générer beaucoup d'échantillons en peu de temps, nous pourrions ne plus avoir besoin de l'expérience passé de l'agent.

Le terme *stabilité des algorithmes* réfère généralement aux propriétés de convergence des algorithmes. En outre, dans l'apprentissage par renforcement, il n'y a pas énormément de preuves sur la convergence des algorithmes et une multitude de questions se posent les concernant. L'algorithme

converge-t-il toujours ? S'il converge, converge-t-il vers une solution correspondant à un optimum global ou local ? De quel objectif ? À quel point les performances de cette solution sont proches de la meilleure solution ? La solution à laquelle converge l'algorithme est-elle loin ou proche de la solution optimale ? Si l'algorithme ne converge pas, alterne-t-il continuellement entre différentes solutions ? À quel point ces solutions sont loin de la meilleure solution ? Les réponses à ces questions varient en fonction du type d'algorithmes. Les méthodes basées sur la descente de gradient (REINFORCE [133], acteur-critique [72]) convergent en général, mais elles convergent vers un optimum local et elles sont assez lentes. Les algorithmes d'apprentissage par renforcement n'utilisent pas toujours la descente de gradient, beaucoup d'algorithmes se basent surtout sur des composantes dans lesquelles sont effectuées des opérations de points fixes (Q-learning [130], SARSA [116]). En revanche, les méthodes se basant sur des opérations de point fixe sont connues pour avoir de mauvaises propriétés de convergence en dehors du cas *tabulaire* et *markovien* (c'est ce qui est appelée la *deadly triad*) [116]. Si les fonctions d'approximation sont utilisées (pour prédire la récompense des actions), la garantie de convergence caractérisant le cas *tabulaire* est perdue. Généralement, les algorithmes utilisant un modèle convergent de manière stable si l'algorithme a un bon modèle de l'environnement, mais il n'y a pas de garanties qu'avoir un meilleur modèle permettra d'obtenir une meilleure politique. En plus du fait qu'il faut avoir les ressources pour exploiter le modèle, la qualité de la politique apprise dépend de la précision du modèle, et optimiser la précision du modèle n'est pas nécessairement préférable au fait d'optimiser directement et uniquement la récompense.

Finalement, les méthodes par fonctions de valeur requièrent en général que l'environnement soit entièrement observable (l'état est markovien). Si l'environnement n'est pas complètement observable, il faudra inclure de la mémoire dans l'état. Les méthodes classiques par *policy gradient* [105] sont naturellement formulées pour les tâches épisodiques. Certains algorithmes basés sur les modèles [111] ne considèrent aussi que les tâches épisodiques. Il existe aussi des méthodes qui font l'hypothèse que l'espace d'action est continu (contrôle optimal [125]). Dans ce qui suit, nous présentons les algorithmes par programmation dynamique. Ces approches représentent la base de beaucoup d'algorithmes d'apprentissage basé sur les valeurs et sont détaillés ici pour cette raison.

## A.2.6 Méthodes par valeurs : La programmation dynamique

Dans l'apprentissage par renforcement, l'agent ne connaît, en général, rien de l'environnement, et même si l'agent apprend un modèle de l'environnement, il aura forcément besoin d'interagir avec l'environnement afin d'apprendre à mieux choisir les actions à prendre. Néanmoins, il est possible de formuler les algorithmes basés sur les valeurs en partant de formulations dans lesquelles l'agent connaît le modèle de l'environnement.

En supposant que le modèle soit connu par l'agent, que la fonction de valeurs des états (ou des actions) est représentée sous forme tabulaire (et non pas sous forme de fonction d'approximation) et que l'environnement est markovien, il est possible pour l'agent d'apprendre à évaluer une politique (ou de trouver la politique optimale) en utilisant des algorithmes de *programmation dynamique*. Pour cette raison, nous commençons par exposer les algorithmes de programmation dynamique dont s'inspire beaucoup d'algorithmes d'apprentissage par renforcement. Partant de ces algorithmes et en relaxant les hypothèses sur les connaissances préalables que nous devons avoir sur l'environnement,

nous pouvons obtenir les algorithmes d'apprentissage par renforcement basés sur les états.

Le principe général de la programmation dynamique est d'utiliser les estimations actuelles de l'agent afin d'en produire de meilleures à l'itération suivante, et ceci en exploitant le modèle de l'environnement. Les *équations de Bellman* (présentés dans la sous-sous-section A.2.4) sont les équations récursives liant la valeur de l'état (resp. la paire d'état-action) actuelle aux valeurs correspondant aux états suivants (resp. la paire d'état-action suivante). La programmation dynamique consiste à itérer continuellement sur ces équations. Ce processus est nommé *l'itération de la politique* (ou *policy iteration* en anglais) quand il s'agit de trouver la meilleure politique (résoudre le problème de contrôle). Quand il s'agit d'évaluer les performances d'une politique fixée, le terme utilisé est *l'évaluation de la politique*. Comme l'évaluation est un sous-problème du contrôle, nous nous concentrons dans les paragraphes qui suivent uniquement sur ce dernier.

Ainsi, la programmation dynamique (ainsi que toutes les méthodes basées sur les valeurs) suivent le principe général des algorithmes d'apprentissage par renforcement : un processus collecte les données, un autre évalue la politique et un troisième améliore cette politique. Les trois s'enchaînent continuellement pour assurer l'apprentissage (voir la sous-section A.2.5). L'algorithme débute avec une politique déterministe. Pour chaque état, la valeur de chaque action est évaluée par itération sur *l'équation de Bellman*. Ensuite, la politique est mise à jour en prenant, pour chaque état, l'action dont l'estimation est maximale (la nouvelle politique consiste à prendre, pour chaque état, l'action ayant la plus grande estimation selon la politique précédente). De ce fait, la nouvelle politique est au moins aussi bonne que l'ancienne (le lecteur pourra se référer au *théorème d'amélioration de la politique* dans le livre [116] pour se convaincre de ce point). Concernant l'algorithme de *l'itération de la politique*, il est possible, soit de stocker les estimations des valeurs des états, soit d'utiliser les estimations des valeurs des actions. Puisque ces deux concepts sont liés par les équations présentées dans la sous-sous-section A.2.4, la programmation dynamique peut être appliquée dans les deux cas.

L'algorithme de *l'itération de la politique* maintient une table des estimations des valeurs des états et évalue les états au début de chaque itération, puis l'améliore en calculant pour chaque état l'action maximisant les estimations des valeurs des actions. L'évaluation se fait en utilisant la formule récursive des estimations des valeurs (*équations de Bellman*), en itérant plusieurs fois sur celle-ci jusqu'à convergence. Cela permet d'obtenir une meilleure estimation des valeurs des états en partant des estimations actuelles. L'amélioration se fait en prenant simplement, pour chaque état, l'action correspondant aux meilleures estimations. Nous disons alors que la nouvelle politique est la *politique gloutonne* (*greedy policy*) relativement aux estimations actuelles de la fonction des valeurs des états.

D'autres variantes de la programmation dynamique peuvent être formulées. Puisque le calcul de l'amélioration porte sur les valeurs des actions, et que le but est d'obtenir des politiques ayant des valeurs plus grandes à chaque itération, nous pouvons supprimer toute mention de la politique en appliquant directement l'opérateur de maximisation juste après avoir évalué les actions. Cette variante de la programmation dynamique s'appelle *l'itération de la valeur* (car les valeurs des actions sont améliorées directement à chaque itération). L'itération de la valeur peut aussi être formulée avec les estimations de la fonction de valeur des actions. Dans ce cas, les valeurs des paires d'états-action sont stockées dans une table qui sera mise à jour par l'algorithme.

Dans l'itération de la valeur, l'évaluation et l'amélioration de la politique sont exécutées sur une seule étape. L'idée est qu'il n'est pas nécessaire de finir le processus d'évaluation avant d'entamer celui de l'amélioration (une seule étape d'évaluation suffit pour que l'algorithme converge vers la meilleure politique au bout de l'apprentissage). Mais les déroulés des algorithmes de l'itération de la politique et de l'itération de la valeur sont très similaires, seule change la formulation de l'algorithme ainsi que la représentation de la politique. Aussi, l'itération de la valeur exécute seulement une étape d'évaluation. La différence entre les deux algorithmes est plus facilement identifiable lorsque l'on utilise des fonctions d'approximation sur les valeurs des actions. L'itération de la valeur permet d'estimer les valeurs des actions à partir d'essais (en permettant de calculer le maximum), alors que l'itération de la politique ne permet pas de le faire. Néanmoins, ces deux algorithmes forment la base de tous les algorithmes d'apprentissage par renforcement basés sur les valeurs. Nous pouvons aussi voir l'itération de la valeur comme un algorithme qui estime directement la valeur de la politique optimale.

Même si les deux algorithmes que nous considérons sur la programmation dynamique -l'itération de la politique et l'itération de la valeur- diffèrent sur certains points, ils reposent néanmoins sur le même principe. Effectivement, tous deux évaluent la politique puis l'améliorent itérativement. Nous pouvons d'ailleurs mieux appréhender les méthodes de programmation dynamique (et même les méthodes d'apprentissage par renforcement en général) sous l'angle d'un procédé itératif global impliquant l'évaluation et l'amélioration et qui est appelée *itération de la politique généralisée* (IGP) [116]. L'*itération de la politique généralisée* est l'idée générale de deux processus interactifs impliquant la politique actuelle de l'agent et la fonction de valeur censée l'approcher (comme montré dans la figure A.3). L'un des processus prend la politique en entrée et met à jour la fonction de valeur de façon à ce qu'elle soit de plus en plus proche de la véritable fonction de valeur de cette politique. L'autre processus prend la fonction de valeur en entrée et effectue une forme d'amélioration de la politique. Bien que chaque processus modifie la base de l'autre, dans l'ensemble, ces deux processus coopèrent pour trouver une solution commune : une politique et une fonction de valeur qui sont inchangées par l'un ou l'autre processus et qui, par conséquent, sont optimales. Les processus d'évaluation et d'amélioration de l'itération de la politique généralisée peuvent donc être considérés comme étant à la fois concurrents et coopérants. Ils sont en concurrence dans le sens où ils changent les estimations de la fonction de valeur dans des sens opposés. L'amélioration de la politique produit une nouvelle politique qui est gloutonne par rapport à la fonction de valeur. Ceci signifie que les estimations actuelles de la fonction de valeur sont différentes de celles de cette nouvelle politique. Mais, en affinant les estimations de la nouvelle politique, le processus d'évaluation fait que cette politique n'est généralement plus la politique gloutonne relativement aux estimations des valeurs des états. Cependant, à long terme, ces deux processus interagissent pour trouver une seule solution commune : la fonction de valeur optimale et une politique optimale. En somme, tant que les deux processus continuent d'agir sur tous les états, le résultat final est généralement le même - la convergence vers une fonction de valeur optimale et une politique optimale.

Les algorithmes d'*itération de la politique* et d'*itération de la valeur* sont des instances de l'itération de la politique généralisée. Différentes manières d'arranger et d'ordonner ces processus aboutissent à différents algorithmes (de programmation dynamique ou d'apprentissage par renfor-

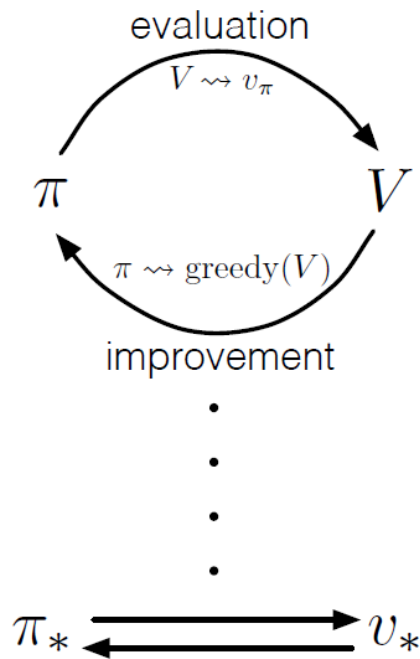


Figure A.3 – L’itération de la politique généralisée [116].

ement). Une représentation schématique de l’itération de la politique est présentée dans la figure A.4, dans laquelle, chaque point de l’espace représente globalement une politique et une fonction de valeur. Il y a deux lignes qui nous intéressent parmi ces points. La ligne en haut représente l’ensemble des politiques avec leurs fonctions de valeurs et la ligne en bas représente les politiques gloutonnes relativement à ces mêmes fonctions de valeurs (ne correspondant pas forcément aux vraies fonctions de valeurs de politique gloutonnes). L’algorithme de l’itération de la politique part d’une fonction de valeur et d’une politique arbitraires qui ne sont pas nécessairement en cohérence l’une avec l’autre. L’évaluation conduit, à chaque étape, à ce que la fonction de valeur corresponde entièrement à la fonction de valeur réelle de la politique (les flèches bleues montantes). L’étape d’amélioration change la politique actuelle de sorte qu’elle soit gloutonne relativement à la fonction de valeur (les flèches bleues descendantes). L’algorithme s’arrête quand les deux processus se stabilisent, ce qui se produit quand l’agent trouve la politique optimale. Pour l’itération de la valeur, l’évaluation est faite seulement sur une seule étape, elle fait des pas plus faibles vers la flèche de haut.

Même si l’itération de la politique généralisée nous aide à comprendre, à concevoir et à situer les algorithmes d’apprentissage par renforcement ainsi que la programmation dynamique dans un seul cadre de pensée, il ne faut pas en voir une garantie de convergence pour ces algorithmes. Concernant les algorithmes de programmation dynamique, les algorithmes d’*itération de la politique* et d’*itération de la valeur* convergent vers la politique optimale, si le *modèle* de l’environnement est disponible, la politique est représentée sous forme *tabulaire* et l’environnement est *complètement observable* [116]. Mais ces hypothèses ne sont pas toujours vérifiées en pratique, particulièrement l’hypothèse sur la disponibilité du modèle. Même si un modèle de la dynamique de l’environnement est disponible, sa précision peut avoir une grande influence sur les performances des algorithmes de programmation dynamique. En effet, sur de longs horizons, les erreurs du modèle (même si elles sont très faibles au départ) se cumulent et faussent les prédictions. Somme toute, un agent utilisant l’apprentissage par renforcement pour agir dans son environnement est censés être capable



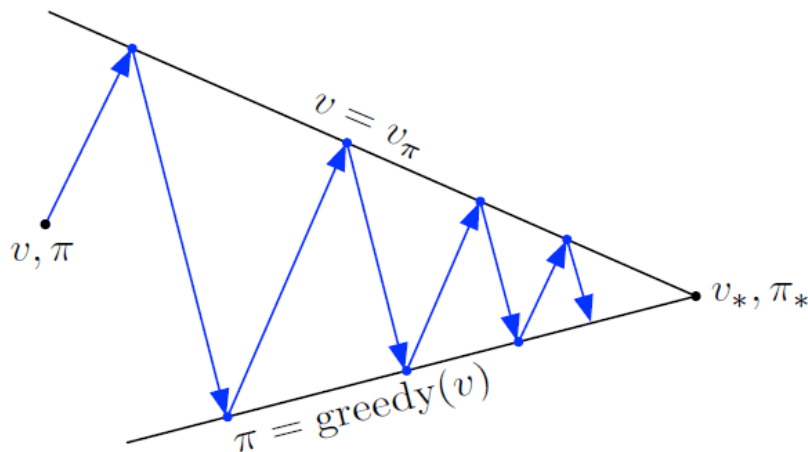


Figure A.4 – Schéma de l'itération de la politique sous forme d'itération de la politique généralisée [116].

d'apprendre un comportement optimal sans connaissances préalables sur l'environnement (et donc sans forcément avoir de modèle explicite sur la dynamique de l'environnement). Heureusement, il est tout à fait possible d'adapter les algorithmes de la programmation dynamique au contexte de l'apprentissage par renforcement en modifiant ces algorithmes de façon qu'ils puissent, au travers de statistiques issues de l'environnement, apprendre par interaction avec l'environnement. Dans ce qui suit, les algorithmes d'apprentissage par renforcement seront présentés en commençant par le *problème de prédiction* puis le *problème de contrôle*.

## A.2.7 Problème de prédiction : Monte-Carlo et Différence Temporelle

Si l'on veut évaluer les performances d'une politique (c'est-à-dire la valeur de cette politique pour chaque état), la manière la plus simple de procéder est d'utiliser la méthode de Monte-Carlo (sur plusieurs essais de cette politique dans l'environnement). Dans le contexte de l'apprentissage par renforcement, générer des échantillons pour calculer une statistique signifie que l'agent exécute la politique dans son environnement, interagit avec celui-ci et rassemble les données reflétant cette interaction pour calculer cette statistique. Vu que la valeur d'une politique dans un état est l'espérance du cumul diminué de la récompense partant de cet état, nous pouvons l'estimer si nous exécutons plusieurs fois la politique en partant de cet état, et que nous prenons la moyenne des retours pour chacune de ces exécutions. Si nous faisons ceci pour chaque état de la chaîne de Markov induite, nous pourrions ainsi produire des statistiques permettant d'évaluer les performances de la politique dans l'environnement. Pour que le calcul de ces moyennes soit possible, la tâche doit naturellement être épisodique (à horizon fini).

Une dernière précision qui concerne le calcul des moyennes est que dans l'apprentissage par renforcement, le calcul des moyennes se fait de manière incrémentale afin de faciliter la formulation des algorithmes (et aussi pour des raisons de stabilité numérique). La dérivation suivante montre le calcul de la moyenne. Nous notons par  $V_k(S_t)$  l'estimation de la valeur d'une politique donnée dans un état  $S_t$  donnée sur  $k$  exécutions de cette politique, et par  $G^k$  le retour obtenu partant de cet état à la  $k$ 'ième exécution.

$$\begin{aligned}
V_{k+1}(S_t) &= \frac{1}{k+1}G^{k+1} + \frac{1}{k+1} \sum_{i=1}^k G^i \\
&= \frac{1}{k+1}G^{k+1} + \frac{k}{k+1} \frac{1}{k} \sum_{i=1}^k G^i \\
&= \frac{1}{k+1}G^{k+1} + \frac{k}{k+1} \tilde{v}_k \\
&= V_k(S_t) + \frac{1}{k+1}(G^{k+1} - V_k(S_t))
\end{aligned}$$

Si l'environnement change (il est non-stationnaire), nous devons privilégier les valeurs les plus récentes du retour. Dans ce cas, nous utilisons un pas constant pour le calcul, ce qui donne

$$V(S_t) \leftarrow V(S_t) + \alpha(G^{k+1} - V(S_t)) \quad (\text{A.11})$$

et ceci correspond à calculer la moyenne mobile exponentielle des retours associés aux exécutions de la politique, ce qui donne lieu à l'équation suivante,

$$V(S_t) = \alpha \sum_{i=1}^k (1 - \alpha)^{k-i} G^i$$

L'équation A.11 est centrale dans la mise à jour des estimations dans l'algorithme de Monte-Carlo. En général, elle prend la forme suivante :

$$\text{nouvelle\_valeur} \leftarrow \text{valeur\_courante} + \text{pas}(\text{cible} - \text{valeur\_courante}) \quad (\text{A.12})$$

Ceci signifie qu'on veut changer la valeur courante de sorte qu'elle s'approche un peu plus de la cible. La cible étant une valeur bruitée de ce qu'on veut estimer, appliquer cette mise à jour sur toutes les cibles du jeu de données permet d'avoir une estimation de plus en plus précise de cette valeur. La différence à droite de l'équation A.12 agit comme un correcteur pour la valeur courante. Comme nous le verrons plusieurs fois au cours de cet état de l'art, la façon dont nous calculons la cible dans cette équation générique donne lieu à différents algorithmes d'apprentissage par renforcement. L'équation A.12 forme le cœur de la partie évaluation dans les méthodes où la politique est représentée *implicitement* à l'aide de la *fonction de valeurs* dans le cas *tabulaire*.

La méthode de Monte-Carlo pour l'estimation d'une politique  $\pi$  est présentée dans l'algorithme 10. L'algorithme génère plusieurs exécutions de la politique  $\pi$  et pour chacun de ces épisodes, les estimations des états visités durant ces épisodes sont mises à jour. La boucle externe génère les épisodes (les données de l'algorithme) et la boucle interne met à jour l'estimation de chaque état visité tel qu'expliqué précédemment.

Dans l'algorithme de Monte-Carlo la cible est le retour de la politique et son espérance vaut exactement la valeur qu'on veut estimer (la valeur de l'état). Les estimations de Monte-Carlo n'ont pas de *biais*. Cela implique que la précision de ces estimations dépend seulement du nombre d'exécutions effectuées avec la politique. Pour cette raison, l'algorithme de Monte-Carlo *converge* même quand l'hypothèse de Markov n'est pas vérifiée [113]. Mais le défaut principal de cette méthode est que les estimations qui lui sont associées ont une grande *variance*. Ceci signifie que les estimations produites par l'algorithme varient beaucoup d'une exécution à une autre. La raison en est que le retour est une somme composée de beaucoup de termes prenant diverses valeurs possibles. Nous devons donc faire énormément d'essais avant d'espérer avoir de bonnes estimations pour la politique. De plus, l'algorithme de Monte-Carlo n'apprend rien pendant l'exécution de la tâche (il apprend uniquement vers la fin des épisodes).

---

**Algorithme 10** L'algorithme de Monte-Carlo pour l'évaluation des états pour une politique  $\pi$ .

---

**Inputs :**  $V(s)$  - les valeurs des états initialisées arbitrairement.

$\pi$  - La politique à évaluer  $\pi$  où  $\pi(s)$  est l'action  $a \in A$  choisie à l'état  $s$ .

$critere\_arret$  - Critère d'arrêt de l'algorithme.

$\alpha$  - Le taux d'apprentissage (le pas de l'évaluation).

```

1: while  $critere\_arret = False$  do
2:   Générer un épisode en utilisant la politique  $\pi : S_0, A_0, R_1, S_1, A_1, R_2, \dots, S_{T-1}, A_{T-1}, R_T$ 
3:    $G \leftarrow 0$ 
4:   for  $t$  allant de  $T - 1$  à  $0$  do
5:      $G \leftarrow \gamma G + R_{t+1}$ 
6:      $V(S_t) \leftarrow V(S_t) + \alpha(G - V(S_t))$  ▷ Evaluation
7:   end for
8: end while

```

---

Les méthodes utilisant le principe de *différence temporelle* (expliquée plus loin) permettent de répondre aux défauts de Monte-Carlo. Pour ce faire, ces approches utilisent les estimations dont l'agent dispose déjà afin d'améliorer la précision des estimations actuelles (s'inspirant ainsi de la programmation dynamique). Dans ce cas, il est dit que les méthodes par différence temporelle sont des méthodes qui font du *bootstrapping*. En effet, revenons vers les équations récursives que nous avons présenté auparavant (les équations de Bellman) :

$$v_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s]$$

et supposons que l'agent a appliqué une action  $A_t$  à l'instant  $t$  dans l'état  $S_t$ , que l'état suivant est  $S_{t+1}$  et que la récompense reçue est  $R_{t+1}$ . Dans ce cas, il est possible d'obtenir une nouvelle estimation de l'état  $S_t$ , en remplaçant dans l'équation précédente l'espérance de la récompense par la récompense observée  $R_{t+1}$  et en remplaçant aussi l'espérance de la valeur de l'état suivant  $S_{t+1}$  par la valeur estimée de cet état  $V(S_{t+1})$ . La nouvelle estimation qui est exprimée ici est donc  $R_{t+1} + \gamma V(S_{t+1})$ . En remplaçant, dans l'équation générique A.12, la cible par cette nouvelle estimation, nous obtenons la règle mise à jour suivante.

$$V(S_t) \leftarrow V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)) \tag{A.13}$$

La différence  $R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$  est nommée *la différence temporelle*. Elle est utilisée afin de corriger l'estimation actuelle de l'agent à l'état courant. La différence temporelle est la différence entre deux estimations de la même quantité, l'une se situant à une étape ultérieure par rapport à l'autre. En minimisant la différence temporelle, l'agent apprend à avoir des prédictions non seulement cohérentes entre elles, mais aussi cohérente avec les observations de l'environnement. Contrairement à la cible de Monte-Carlo (voir l'équation A.12) l'avantage d'utiliser la cible  $R_{t+1} + \gamma V(S_{t+1})$  dans l'estimation de la valeur des états est qu'elle a une faible variance comparée à la première (malgré la présence de biais dans cette dernière). La variance de cet estimateur est faible, car la variabilité dans les données ne dépend que de la récompense suivante (et pas de tous les termes présents dans le retour). Quant à la présence de biais dans l'estimateur, elle vient du fait que les estimations des

états ont besoins d'être initialisées, et que souvent les estimations initiales sont éloignées des vraies valeurs des états. Mais ce biais est censé diminuer au fur et à mesure que l'agent interagit avec l'environnement et recueille des données sur la qualité des états. Un autre avantage d'utiliser cette cible dans l'apprentissage est qu'elle permet d'avoir des estimations à la volée, sans avoir besoin d'attendre que l'épisode se termine pour en avoir. Globalement, la différence temporelle permet de rendre l'évaluation plus rapide que par rapport à celle de Monte-Carlo. Le désavantage principal des méthodes par différence temporelle est que leur fonctionnement est beaucoup trop dépendant de la véracité de l'hypothèse de Markov dans l'environnement. En effet, par définition de la différence temporelle, nous voyons bien que la cible servant à corriger l'estimation d'un état donné dépend de l'estimation de l'état suivant. Ainsi, pour que le biais associé à la différence temporelle se réduise avec le temps, l'évolution de l'environnement devrait dépendre uniquement du présent [116].

Nous présentons dans ce qui suit le premier algorithme qui se base sur le principe de différence temporelle. L'algorithme de  $TD(0)$  (Temporal Difference) pour l'évaluation consiste globalement à effectuer la mise à jour de l'équation A.13 à chaque fois qu'une transition est effectuée (choix et application d'une action suivie de la réception de la récompense et du changement d'état). L'algorithme 11 présente l'algorithme tabulaire de  $TD(0)$  pour l'estimation d'une politique  $\pi$ .

---

**Algorithme 11** L'algorithme de  $TD(0)$  pour l'évaluation des états d'une politique  $\pi$ .

---

**Inputs :**  $V(s)$  - les valeurs des états initialisées arbitrairement sauf pour l'état final  $V(F)$ .

$\pi$  - La politique à évaluer  $\pi$  où  $\pi(s)$  est l'action  $a \in A$  choisie à l'état  $s$ .

$N$  - le nombre d'épisodes.

$\alpha$  - Le taux d'apprentissage (le pas de l'évaluation).

```

1: for  $i$  allant de 1 à  $N$  do
2:   Initialiser l'état courant  $S$ 
3:   while  $S \neq F$  do
4:      $A \leftarrow \pi(S)$ 
5:     Pendre l'action  $A$ , observer  $R, S'$  ▷ Transition de l'environnement
6:      $V(S) \leftarrow V(S) + \alpha(R + \gamma V(S') - V(S))$  ▷ Evaluation
7:      $S \leftarrow S'$ 
8:   end while
9: end for

```

---

Tout comme dans l'algorithme de Monte-Carlo, il y a deux boucles imbriquées, la première itère sur les épisodes de l'algorithme, et la deuxième exécute la politique pour chaque épisode et effectue l'évaluation (comme stipulé auparavant) sur chacune des transitions occurring durant l'épisode. L'algorithme est formulé pour le cas épisodique, mais vu que la mise à jour des estimations se fait à chaque étape, il est possible de formuler cet algorithme même pour le cas où la tâche est continue.

En résumé, nous pouvons dire que si les estimations basées sur le *bootstrapping* sont utilisées comme cible d'apprentissage pour l'évaluation, la variance de la cible sera assez faible, mais elle ne sera pas sans biais. Inversement, si nous utilisons les estimations de Monte-Carlo, nous n'aurons pas de biais, mais la variance deviendra assez forte. La question qui se pose naturellement dans cette

situation est de savoir s'il est possible de combiner ces deux méthodes pour en obtenir une nouvelle méthode possédant les avantages des deux méthodes, mais sans leurs désavantages respectifs ? En d'autres termes, existe-t-il un moyen de paramétrer le compromis entre biais et variance dans le contexte de l'évaluation de politique en apprentissage par renforcement (tout comme pour l'apprentissage supervisé) ? La réponse est qu'il est possible de le faire si nous nous autorisons à faire du bootstrapping sur les estimations des états sur plusieurs étapes futur. Au lieu de prendre seulement la première récompense immédiate dans l'estimation basée sur le bootstrapping, le principe ici est de considérer les  $n$  prochaines récompenses dans la cible et d'estimer le reste du retour avec les estimations de l'agent. L'agent choisit alors, pour plusieurs étapes consécutives, les actions à prendre et les appliquent sur l'environnement, et il ne met à jour la valeur de l'état dans lequel il était, qu'après avoir choisi et exécuté ces actions et que l'environnement ait transitionné autant de fois. L'équation A.14 montre la forme de la cible pour cette approche. Si nous notons par  $n$  le nombre de fois où l'on applique successivement la politique  $\pi$  avant de la mettre à jour, alors *le retour en  $n$  étapes* ( *$n$ -step return* en anglais) noté  $G_{t:t+n}$  est l'estimation dont l'expression est :

$$G_{t:t+n} = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{n-1} R_{t+n} + \gamma^n V(S_{t+n}) \quad (\text{A.14})$$

En rajoutant des termes à l'estimation basée sur le bootstrapping (augmenter le  $n$  dans les retours en  $n$  étapes), nous introduisons plus de termes (et donc plus de sources d'aléatoire) dans l'expression de l'estimation, ce qui augmente la variance de cette estimation. En contrepartie, ceci aura pour effet de réduire le biais, car l'expression dépendra plus d'échantillons réels et un poids exponentiellement plus faible sera associé à l'estimation des états futurs (notons que le poids associé aux estimations de l'agent est de  $\gamma^n$  dans l'équation A.14). Nous pouvons ainsi décider quel sera le niveau permettant d'équilibrer le mieux entre le biais et la variance en choisissant soigneusement la valeur de  $n$  (la cible étant différente suivant que l'on privilégie le biais ou la variance des estimations). Les méthodes se basant sur les retours en  $n$  étapes sont nommées  *$n$ -step TD*, l'équation gérant l'évaluation dans ces méthodes est présentée dans ce qui suit,

$$V(S_t) \leftarrow V(S_t) + \alpha(G_{t:t+n} - V(S_t)) \quad (\text{A.15})$$

Dans les méthodes  *$n$ -step TD*, l'agent met à jour, à chaque étape, l'estimation de l'état visitée  $n$  étapes dans le passé (comme montré dans la figure A.5). Cela nécessite de stocker les récompenses appropriées, et de faire les opérations de mise à jour des estimations en décalage de  $n$  étapes. Mais hors mis ces deux points, l'algorithme d'évaluation de la politique pour les méthodes  *$n$ -step TD* est très similaire à l'algorithme d'évaluation de TD(0) (1-step TD). Le choix la valeur que doit prendre  $n$  peut s'avérer complexe si l'on ne connaît pas la durée des épisodes, surtout si celles-ci peuvent varier d'une exécution à l'autre ou si la tâche à accomplir est à horizon infini. Heureusement, il n'est pas nécessaire de spécifier une valeur unique de  $n$  au-delà de laquelle on utilise seulement les estimations de l'agent. L'alternative à cela est de prendre une moyenne pondérée de tous les retours en  $n$ -étapes (les  $G_{t:t+n}$ ).

Concernant le type de pondération le plus adéquat à choisir, notons que le plus gros problème dans l'estimation des états est la variance des estimations. N'oublions pas que l'apprentissage par renforcement à base d'états est formulé dans le cadre des problèmes de décision séquentiel et vu la combinatoire exponentielle des possibilités qui en résultent, nous pouvons voir pourquoi la variance aurait la plus grande responsabilité dans la précision des estimations. Il est donc préférable



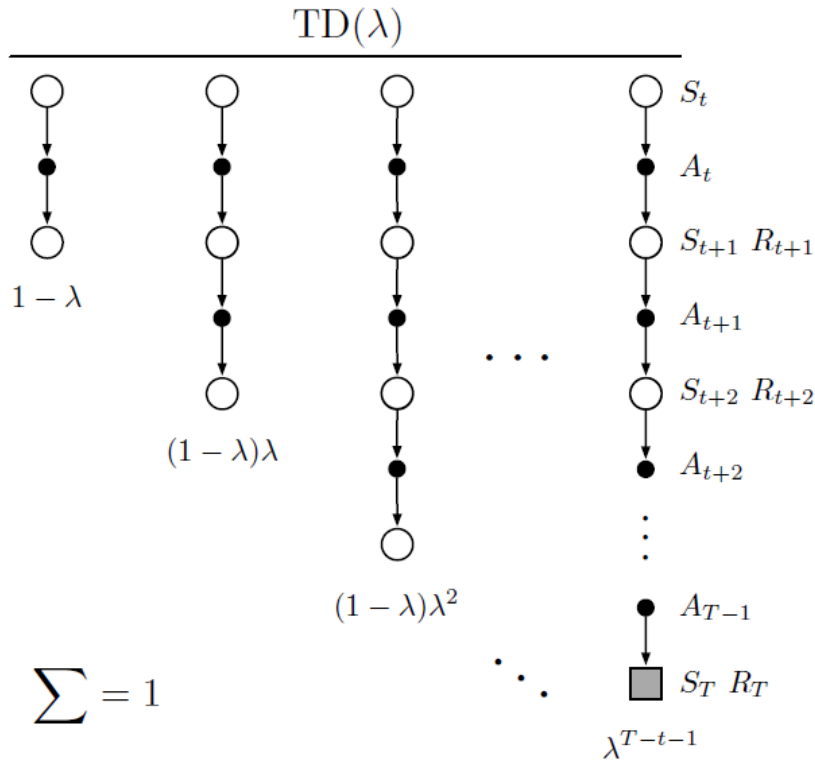


Figure A.6 – Schéma représentant le calcul de la cible dans TD( $\lambda$ ) [116].

fait, nous aurons énormément d'éléments à mémoriser et à calculer avec TD( $\lambda$ ). Mais les difficultés liées à TD( $\lambda$ ) viennent d'une propriété qui n'est pas propre à cet algorithme. Dans TD( $\lambda$ ) (tout comme dans  $n$ -step TD et dans Monte-Carlo), afin de pouvoir mettre à jour l'état visité actuellement, l'agent a besoin d'éléments futur sur l'effet et les performances de sa politique. L'agent ne met à jour l'estimation d'un état que bien après que celui-ci eut été visitée. De telles formulations, basées sur l'anticipation à partir de l'état mis à jour, sont appelées *vues prospectives* (*forward view*). Afin de simplifier l'implémentation de TD( $\lambda$ ), une formulation alternative de cet algorithme, utilisant uniquement les informations disponibles actuellement (ainsi qu'un mécanisme de mémoire à court terme) a été proposée [112]. Dans cette version, l'algorithme maintient un vecteur nommé *trace d'éligibilité* qui contient pour chaque état la fréquence selon laquelle cet état a été visitée dans le passé récent de l'agent. Ce vecteur agit comme une sorte de résumé de l'historique récent de l'agent. Et de ce fait, il permet de quantifier pour chaque état l'importance de son rôle dans l'obtention de la récompense. Ce vecteur agit comme une réponse directe au problème d'*affectation de crédit* en associant une valeur numérique au crédit de chaque état.

Dans TD( $\lambda$ ) avec *trace d'éligibilité*, à chaque étape, on incrémente dans la trace d'éligibilité la composante associée à l'état courant et on décrémente les composantes de tous les autres états exponentiellement avec le temps (en multipliant leurs traces respectives avec  $\lambda$ ). La mise à jour des estimations porte sur tous les états de l'environnement. Pour chaque état, la trace est alors combinée avec la différence temporelle actuelle et l'estimation actuelle de l'état afin de produire des estimations à peu près équivalentes à celles de TD( $\lambda$ ) sans traces d'éligibilité. Ainsi, chaque état se verra être mis à jour en fonction du crédit qui lui est associé par la trace d'éligibilité. Si on note par  $E_t(s)$  la trace associée à l'état  $s$  à l'instant  $t$ , les équations de mise à jour de la trace d'éligibilité

sont décrites dans ce qui suit,

$$E_t(s) = \begin{cases} 0 & \text{si } t=0 \\ \gamma\lambda E_{t-1}(s) + \mathbb{1}(S_t = s) & \text{sinon} \end{cases}$$

Si on note  $\delta_t$  la différence temporelle associée à l'instant  $t$ , alors la mise à jour des estimations de l'algorithme de TD( $\lambda$ ) avec trace d'éligibilité prend la forme suivante,

$$\begin{aligned} \delta_t &= R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \\ V(S_{t+1}) &\leftarrow V(S_t) + \alpha \delta_t E_t(S_t) \end{aligned}$$

## A.2.8 Problème de contrôle : SARSA et Q-learning

Dans cette partie, nous présentons les algorithmes de contrôle les plus utilisés dans l'apprentissage par renforcement tabulaire à base d'états. Le problème de prédiction étant une sous-partie du problème de contrôle, les algorithmes de contrôle sont assez similaires aux algorithmes de prédiction. Nous retenons, à ce propos, deux algorithmes (se basant sur le principe de différence temporelle) qui sont *SARSA*[116] et *Q-learning*[130]. Ces algorithmes sont présentés dans le chapitre 2. Dans ce qui suit, nous passons en revue certaines différences de représentations entre les algorithmes de *prédiction* et les algorithmes de *contrôle* (voir la sous-section A.2.6).

Tout d'abord, dans le problème de contrôle, connaître l'effet des actions (pouvoir les estimer) permet de choisir les bonnes actions et ainsi améliorer petit à petit la politique de l'agent. Dans le cas où l'agent dispose d'un modèle de l'environnement, l'amélioration se fait essentiellement par choix glouton sur la base des estimations des actions. Mais quand l'algorithme ne dispose pas d'un modèle de l'environnement, il n'est plus fiable de se reposer sur les estimations des actions pour faire évoluer la politique. La cause en est que les estimations de l'agent sont construites à partir de tirages aléatoires ou en utilisant le *bootstrapping* et elles sont donc sujettes à des erreurs. En utilisant des estimations imprécises comme base pour améliorer la politique, l'agent pourrait surestimer/sous-estimer la qualité des actions et risque de finir avec une politique moins bonne que sa politique de départ. En l'absence de modèle, l'agent ne peut pas se restreindre à choisir la meilleure action (contrairement à la programmation dynamique), il devra de temps en temps choisir des actions estimées moins bonnes afin d'obtenir les connaissances nécessaires et compenser l'absence de modèle.

Bien entendu, l'exploration nécessite que le choix des actions soit fait de façon stochastique. Toutes les méthodes présentées dans l'apprentissage par renforcement sans états peuvent servir à implémenter des stratégies d'exploration dans l'apprentissage par renforcement à base d'état. La méthode la plus choisie pour cela est  $\epsilon$ -Greedy [9] vu sa simplicité : dans chaque état, l'agent choisit, la plupart du temps, l'action ayant l'estimation maximale, et de temps en temps une action aléatoire parmi les autres actions.

Diversifier le choix des actions nécessite de pouvoir représenter les estimations des actions (et non pas des états seulement). Ainsi, afin de représenter explicitement les performances des actions, la politique est représentée non pas avec les valeurs des états, mais avec les valeurs des paires d'états-actions. L'agent maintient une table dans laquelle chaque entrée correspond à la valeur d'une action



dans un état donné. Pour chaque état, l'agent mémorise l'action ayant la meilleure valeur. Si l'agent décide qu'il veut exploiter (au sens du compromis exploration/exploitation), il pourra choisir cette action, sinon il pourra en choisir une autre.

## A.2.9 Les méthodes d'approximation

Nous savons que les algorithmes de SARSA et de Q-learning ainsi que leurs variantes basées sur la généralisation du principe de différence temporelle (SARSA( $\lambda$ ) [106] et Q( $\lambda$ ) [129]) convergent pour le cas tabulaire, quand l'environnement est Markovien et complètement observable. Mais en ce qui concerne la condition que la politique soit *tabulaire*, celle-ci peut s'avérer problématique dans beaucoup de problèmes vu le nombre d'états que peut prendre l'environnement dans la réalité. Prenons un exemple pour illustrer ce point. Imaginons que l'agent soit un robot équipé d'une caméra renvoyant une image RGB de taille  $128 \times 128$  pixels. Comme l'état est représenté à l'aide des images perçues par l'agent, si l'on doit représenter la politique sous forme de table, nous aurons besoin d'une entrée pour chacun des  $255^{128 \cdot 128 \cdot 3}$  états de l'environnement (et encore beaucoup plus si on veut représenter les q-valeurs). Dans ce cas, la table des valeurs (qui est l'élément principal sur lequel s'appuie l'agent pour agir dans l'environnement) serait impossible à représenter et à exploiter. C'est ce que l'on appelle *le fléau de dimensionnalité* (ou *curse of dimensionality* en anglais). À défaut de représenter les estimations des valeurs des états de manière exhaustive dans une table, nous pourrions les approcher par *une fonction d'approximation*. Pour une politique  $\pi$  et un état  $s$ , cette fonction d'approximation a pour entrée *les features* (ou des *attributs*) de l'état  $s$  noté  $\phi(s)$  et retourne une estimation approchée des performances de la politique  $\pi$  dans cet état. La *fonction d'approximation* dépend d'un vecteur de paramètres noté  $w$ . Modifier la valeur de ce paramètre équivaut à changer l'association entre les états et leurs estimations. En notant la valeur de la fonction d'approximation à l'état  $s$  par  $\hat{v}(s, w)$ , le but est de trouver un paramètre  $w$  de sorte que  $\hat{v}(s, w) \approx v_\pi(s)$ . L'agent exécute la politique  $\pi$  et utilise les données récoltées avec celle-ci pour ajuster le paramètre  $w$  de sorte à obtenir une fonction dont les prédictions sont plus proches des valeurs réelles des états. De cette façon, l'agent obtient une fonction capable de généraliser la relation liant l'état et sa valeur. Fondamentalement, l'agent apprend à prédire la valeur des états en tentant de résoudre un problème de *généralisation* à l'aide d'un algorithme de *régression*. Effectivement, le paramètre  $w$  peut représenter les paramètres d'un réseau de neurones, d'une régression linéaire ou de tout autre modèle de régression. Typiquement, le nombre de poids (la dimensionnalité  $d$  de  $w$ ) est bien inférieur au nombre d'états ( $d \ll |\mathcal{S}|$ ), et la modification d'un poids change la valeur estimée de nombreux états. Par conséquent, lorsqu'un seul état est mis à jour, le changement se généralise à partir de cet état pour affecter les valeurs de nombreux autres états. Une telle généralisation rend l'apprentissage potentiellement plus puissant, mais aussi potentiellement plus difficile à gérer et à comprendre.

Dans le but d'apprendre à estimer les valeurs des états, l'agent dispose d'un jeu de données sur son expérience avec l'environnement. Ce jeu de données est constitué des *features* (les attributs) des états visités accompagnés de leurs cibles qui sont des estimations associées à ces états. Les cibles —que l'agent doit apprendre à approcher— sont construites à partir des récompenses réelles retournées par l'environnement. Comme nous l'avons mentionné auparavant, les méthodes tabulaires améliorent l'estimation de l'agent en changeant les estimations actuelles de sorte qu'elles soient plus

en cohérence avec les récompenses reçues de l'environnement (comme nous l'avons montré dans l'équation A.12). Les méthodes par fonction d'approximation suivent le même principe, à la différence près que c'est le vecteur de poids  $\mathbf{w}$  qui est changé de façon à améliorer les prédictions (au lieu des estimations elles-mêmes). Tout comme dans n'importe quel problème de prédiction, la qualité des prédictions de l'agent est mesurée à l'aide d'une *fonction de perte* qui est généralement la MSE (erreur quadratique moyenne) entre les prédictions de l'agent et les cibles. L'agent doit optimiser cette fonction de perte afin d'avoir de bonnes prédictions (généralement en utilisant l'algorithme de *la descente de gradient stochastique*). Formellement, l'évaluation d'une politique  $\pi$  nécessite de trouver un paramètre  $\mathbf{w}$  minimisant l'erreur quadratique moyenne entre la valeur approximée  $\hat{v}(s, \mathbf{w})$  et la vraie valeur  $v_\pi(s)$ . En notant la distribution stationnaire sur les états de la politique  $\pi$  par  $P_\pi$ , l'erreur devant être minimisée notée  $J(\mathbf{w})$  est définie comme suit

$$J(\mathbf{w}) = \sum_{s \in \mathcal{S}} [P_\pi(s)(v_\pi(s) - \hat{v}(s, \mathbf{w}))^2] \quad (\text{A.18})$$

Il est possible de trouver un optimum local minimisant cette erreur en appliquant la descente de gradient. Après différenciation de l'objectif (en  $\mathbf{w}$ ), la mise à jour de la descente de gradient est décrite pour l'étape  $t$  comme suit,

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \alpha(v_\pi(S_t) - \hat{v}(S_t, \mathbf{w}_t)) \nabla \hat{v}(S_t, \mathbf{w}_t) \quad (\text{A.19})$$

Bien entendu, nous n'avons pas accès aux vraies valeurs des états, les cibles sont des approximations aléatoires de  $v_\pi(s)$  et elles pourront être estimées comme dans les algorithmes d'évaluation prévus pour le cas tabulaire. Par exemple, les cibles peuvent être prises comme étant le retour de Monte-Carlo, ce qui donne lieu à la mise à jour suivante.

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \alpha(G_t - \hat{v}(S_t, \mathbf{w}_t)) \nabla \hat{v}(S_t, \mathbf{w}_t) \quad (\text{A.20})$$

Sur ce point, nous pouvons utiliser toutes les estimations présentées auparavant dans les algorithmes de prédiction du cas tabulaire. Il est possible d'utiliser une approximation basée sur le bootstrapping, ou alors combiner les estimations de Monte-Carlo et les estimations basées sur le bootstrapping comme dans TD( $\lambda$ ), nous aurons ainsi des algorithmes différents pour chaque cible.

Concernant les propriétés de convergence de ces algorithmes, il est connu que les algorithmes de prédiction basés sur les méthodes d'approximation convergent bien quand les estimations de Monte-Carlo sont utilisées [116] (même dans le cas non-linéaire). À cet égard, il existe des résultats de convergence pour différents algorithmes *on-policy* [116] dans le cas linéaire, les algorithmes *off-policy* étant plus difficile à gérer même dans le cas linéaire [116]. Pour le problème de contrôle, beaucoup de garanties de convergence sont perdues pour différents algorithmes [116]. Globalement, la présence de relations non-linéaire entre l'état et sa valeur, l'utilisation de bootstrapping ou l'utilisation d'algorithmes *off-policy* introduisent de l'instabilité dans le comportement des algorithmes jusqu'à même provoquer la divergence dans certains cas. Mais ces éléments sont souvent essentiels pour avoir de bonnes performances, le bootstrapping permet d'exploiter les connaissances déjà acquises par l'algorithme afin de mieux estimer la politique. Les algorithmes *off-policy* sont très utiles, car ils permettent de réutiliser et d'exploiter les données collectées par le passé afin d'améliorer l'apprentissage, et les modèles non-linéaires (tels que les réseaux de neurones) capturent des relations complexes et peuvent être très pertinents quand il s'agit de traiter certaines problématiques

concrètes. L'algorithme du Deep Q-Learning [87], et un algorithme basé sur le Q-learning classique dans lequel l'estimation des valeurs des actions est représentée avec un réseau de neurones profond. Un élément caractéristique du Deep Q-learning (DQN) est qu'il rassemble les trois éléments décrits précédemment qui peuvent être sources d'instabilité (c'est ce qu'on appelle la *deadly triade* en apprentissage par renforcement). Dans le but d'éliminer l'instabilité présente dans le DQN, deux éléments supplémentaires ont été intégrées à l'algorithme : l'*experience replay* et le *target network*.

L'objectif de l'*experience replay* est d'éliminer les problématiques liées aux corrélations de données observées dans les problèmes d'apprentissage par renforcement. En effet, lorsque l'agent exécute sa politique, les états qu'il perçoit ainsi que les récompenses qu'il reçoit sont corrélées dans le temps. Ce qu'observe l'agent à un instant donnée influence son processus de prise de décision et peut l'amener à visiter des états qu'il n'aurait pas visités s'il avait pris d'autres actions, ce qui pourrait biaiser son évaluation de la politique. Rappelons que les méthodes d'évaluation sont basées sur des algorithmes d'apprentissage supervisé, et que le fonctionnement correct de ces algorithmes nécessite que les données collectées par l'agent soient i.i.d. (indépendants et identiquement distribués). Si les données sont corrélées dans le temps, ceci pousse l'algorithme d'apprentissage à ajuster les poids de la fonction d'approximation de sorte que ceux-ci ne mémorisent que les données corrélées récemment observées et oublie les tendances apprises des données passées. Le principe de l'*experience replay* est de réajuster les paramètres de la fonction de valeur en utilisant les transitions *passées* de l'algorithme (on ne se limite plus seulement aux transitions les plus récentes), ce qui va permettre de rompre la corrélation des données dans l'apprentissage. Dans l'*experience replay*, les données (c'est-à-dire les transitions de l'environnement dues aux actions appliquées par l'agent), sont collectées et stockées dans un *buffer*. À chaque étape, l'algorithme tire aléatoirement un *batch* (un ensemble) de transitions de ce *buffer* et effectue l'apprentissage uniquement sur ce *batch* (l'estimation de la politique). Le *buffer* est rempli périodiquement avec les nouvelles expériences pour permettre à l'algorithme de poursuivre l'apprentissage (le rechargement des données étant souvent implémenté en FIFO). Dans certaines approches d'*experience replay* le tirage des transitions à mettre dans le *batch* se fait de manière complètement uniforme [87], d'autres priorisent les transitions les plus récentes, ou encore les transitions amenant à corriger le plus les estimations actuelles de l'agent [109].

L'implémentation naïve du DQN présente un autre problème, le fait que la fonction d'approximation (le réseau de neurones) est entraînée à approcher des cibles qui sont produites avec cette même fonction d'approximation. De ce fait, dès que le réseau met à jour ses paramètres, les prochaines cibles qui seront calculées seront assez différentes de celles qui auraient été calculées avec les poids précédents du réseau. C'est comme si le réseau de neurones voulait approcher une cible mouvante dont la mouvance dépend des paramètres de ce réseau. L'idée du *target network* est de générer les cibles avec un réseau différent du réseau utilisé actuellement pour la prédiction. Pour rendre l'apprentissage possible, le *target network* est choisi comme étant les poids associés au réseau (destiné à la prédiction) pour une itération passée de l'algorithme. Le *target network* est généralement mis à jour périodiquement (avec des périodes assez grandes), et certaines méthodes génèrent le *target network* en calculant la moyenne mobile exponentielle de tous les réseaux passés.

## A.2.10 Les méthodes basées sur les politiques : policy-gradient et acteur-critique

Comme nous l'avons dit précédemment, les méthodes basées sur la politique représentent la politique directement (contrairement aux méthodes basées sur les valeurs). La politique prend la forme d'une distribution sur les actions, conditionnée sur l'état, et qui dépend d'un paramètre  $\theta$ . Le paramètre  $\theta$  peut être aussi bien l'espérance/variance d'une distribution donnée, que les poids d'un réseau de neurones. Dans le cas où la politique est représentée par un réseau de neurones, l'entrée est constituée des *features* (caractéristiques) de l'état et la sortie correspond aux probabilités de sélection des actions.

Comme tout algorithme d'apprentissage par renforcement, les méthodes basées sur la politique ont les composantes de collecte de données, d'évaluation de la politique et d'amélioration de celle-ci. Dans la version la plus classique des algorithmes basés sur la politique, l'évaluation de la politique se fait directement par exécution de celle-ci et calcul du retour observé lors de l'exécution. Ensuite, l'agent infère, à partir des performances observées de sa politique, quelle serait la meilleure manière de l'améliorer. L'amélioration se base généralement sur une descente de gradient (approches qu'on appelle *policy gradient*). En effet, l'agent tente d'estimer le gradient de la politique à partir des données collectées lors de l'exécution de la politique. Le gradient permet alors de renseigner comment doivent être modifiés les paramètres de la politique afin d'améliorer localement les performances de la politique (via une approximation linéaire locale des performances). L'équation de mise à jour de la politique est la suivante,

$$\theta \leftarrow \theta + \alpha \left( \sum_{k=1}^T \gamma^{k-1} R_k \right) \left( \sum_{k'=0}^{T-1} \nabla_{\theta} \ln(A_{k'} | S_{k'}, \theta) \right) \quad (\text{A.21})$$

Cette équation correspond à l'algorithme REINFORCE [134], qui est l'algorithme le plus basique des méthodes basées sur la politique. Dans cette équation, le gradient des performances de l'agent dépend, d'un côté, du cumul de récompense et de l'autre, du gradient du logarithme de la politique. L'idée des méthodes par gradient de politique dans l'apprentissage par renforcement est très similaire à celle de l'apprentissage supervisé. Si nous considérons que les données viennent d'un humain, et que l'agent doit apprendre la politique avec laquelle cet humain sélectionne les actions, il est possible d'utiliser l'apprentissage supervisé dans le contexte d'un problème de classification afin d'apprendre à imiter l'humain. Dans ce cas, si on utilise l'entropie croisée comme fonction de perte, son gradient s'écrit comme suit,

$$\theta \leftarrow \theta + \alpha \left( \sum_{k'=0}^{T-1} \nabla_{\theta} \ln(A_{k'} | S_{k'}, \theta) \right)$$

Le gradient des méthodes par *policy gradient* est donc similaire à celui de l'*entropie croisée* [55] avec pour différence que ce dernier est corrigé par le cumul de la récompense. Dans la classification, les probabilités des actions observées sont incrémentées, car on sait que celles-ci sont les bonnes actions (elles viennent du démonstrateur), alors que dans l'apprentissage par renforcement, ce sont les récompenses obtenues qui déterminent les bonnes actions. C'est pour cette raison que la partie des logarithmes des probabilités est multipliée par la récompense dans l'équation A.21.

Puisque les méthodes basées sur la politique utilisent le calcul du gradient (ou sur le principe d'approximation locale), l'agent aura besoin que les données utilisées dans l'apprentissage soient générées par la politique actuelle. Les algorithmes basés sur la politique sont donc naturellement *on-policy*. Ceci ne pose pas de problème si nous disposons d'un simulateur capable de générer rapidement d'énormes quantités de données, mais dans la pratique ça n'est souvent pas le cas. Contrairement aux méthodes basées sur les valeurs, les méthodes basées sur la politique ont de bonnes garanties de convergence. Étant donné que ces méthodes sont basées sur la descente de gradient stochastique, elles convergent toujours vers un optimum local. Mais n'oublions pas que la politique de l'agent est stochastique et donc que le retour peut changer d'une exécution à une autre de la politique. L'estimation du gradient peut varier grandement suivant la trajectoire suivie par l'agent. En effet, les méthodes basées sur la politique sont connues pour avoir une grande variance (ce qui nuit aux performances, car on est contraint d'utiliser de faibles pas d'apprentissage). Le problème est que l'on ne peut pas corriger la variance en faisant intervenir les données passées de l'algorithme, car la politique change sans cesse, comme nous l'avons dit, les algorithmes par *policy gradient* sont naturellement *on-policy*.

Toutes les améliorations proposées pour les méthodes à base de politique servent à corriger ce problème de variance ou à rendre l'agent *off-policy*. Dans un premier temps, nous pouvons utiliser le principe de causalité : dans la mise à jour de la politique, et pour chaque action, la somme des récompenses porte uniquement sur les récompenses retournées *après* l'action. Dans l'équation A.21, après distribution des sommes des récompenses dans la somme des logarithmes de probabilité, on supprime donc les termes qui précèdent le terme de l'action courante, dans chacun des termes de la somme. Ceci donne  $\sum_{k'=0}^{T-1} \nabla_{\theta} \ln(A_{k'} | S_{k'}, \theta) \sum_{k=t+1}^T \gamma^{k-1} R_k$ .

Une des manières permettant de stabiliser le calcul du gradient dans les méthodes à base de politique consiste à retrancher une quantité nommée *baseline* [116] des récompenses. On retranche la *baseline* pour faire en sorte que les bonnes actions aient des récompenses positives et les mauvaises actions aient des récompenses négatives. La *baseline* ne change pas l'espérance du gradient, elle ne rajoute donc aucun biais. La *baseline* peut être la récompense moyenne. Dans ce cas-là, les bonnes actions sont les actions qui mènent l'agent à obtenir plus de récompense qu'habituellement et les mauvaises actions sont l'inverse. Nous obtenons alors, ce qui est appelé l'*avantage* de l'action par rapport au retour moyen.

Une manière de contourner le problème de la variance est de rendre l'algorithme *off-policy*. De ce fait, l'algorithme pourra effectivement apprendre partant de données de sources diverses. Pour cela, le principe de l'*importance sampling* [116] est utilisée. Dans l'*importance sampling*, on considère deux distributions, la première qu'on s'autorise d'échantillonner et le deuxième qui nous est inconnue et avec laquelle nous voulons estimer l'espérance de notre fonction. L'*importance sampling* permet justement d'estimer l'espérance d'une fonction donnée suivant une certaine distribution, à partir de tirage issue d'une autre distribution. L'idée est d'estimer cette espérance à partir d'un échantillon de la distribution dont on a accès et puis de corriger chaque terme dans cette estimation, de sorte à surestimer les valeurs qui apparaissent plus fréquemment avec la distribution qui nous intéresse qu'avec celle dont on a accès (et à sous-estimer les valeurs qui apparaissent moins fréquemment dans la distribution qui nous intéresse qu'avec celle qu'on utilise). L'*importance sampling* est une

technique qui est difficile à mettre en pratique, les estimations par l'*importance sampling* ont tendance à varier énormément si la politique qu'on veut apprendre est trop différente de celle pour laquelle les données sont disponibles, aggravant ainsi le problème même que nous voulons résoudre.

La version la plus basique de *policy gradient* utilise le retour de Monte-Carlo pour évaluer la politique. L'estimateur de Monte-Carlo est sans biais, mais nous pouvons obtenir de meilleures estimations si nous utilisons des fonctions d'approximation entraînées à approcher la fonction de valeur. Ceci nous permet d'avoir de meilleures estimations même pour les états jamais visités. De ce fait, remplacer les retours par la fonction d'approximation permet d'estimer le gradient sans avoir besoin d'attendre la fin de l'épisode, mais aussi de rendre la descente de gradient plus stable. Précisons que ce réseau ne sert pas à choisir les actions à prendre, mais uniquement à orienter l'apprentissage de l'agent. Cette approche se nomme d'ailleurs *acteur-critique* [72]. Nous avons d'un côté un module dont le rôle est de choisir les actions que l'agent prend dans l'environnement (l'*acteur*), et de l'autre un module dont le rôle est d'informer l'acteur comment modifier ses paramètres de façon à pouvoir choisir de meilleures actions. La mise à jour de l'*acteur-critique* est présentée dans ce qui suit,

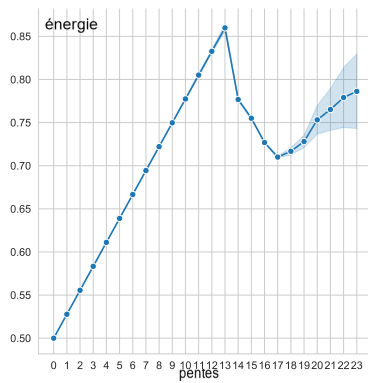
$$\theta \leftarrow \theta + \alpha \sum_{k'=0}^{T-1} \nabla_{\theta} \ln(A_{k'} | S_{k'}, \theta) (R_{k+1} + \gamma V_w^{\pi}(S_{k+1}) - V_w^{\pi}(S_k)) \quad (\text{A.22})$$

Dans cette version de l'acteur-critique, l'estimation utilise le principe de *bootstrapping* sur l'approximation de la fonction de valeur et la *baseline* utilisée est l'estimation approximée à l'état courant. Concernant le calcul des cibles servant à entraîner la fonction d'approximation, toutes les méthodes et optimisations dont le but est d'estimer les fonctions de valeur peuvent s'appliquer ici. L'acteur-critique constitue un bon compromis entre biais et variance, du biais est introduit dans les estimations, mais celles-ci seront à termes de meilleure qualité, car elles dépendent de connaissances partielles de l'environnement déjà acquises par l'agent. Par contre, le désavantage des méthodes par acteur-critique est que le nombre de paramètres utilisés augmente (nous avons deux fonctions d'approximation). Il faudra aussi correctement paramétrer les deux fonctions afin que l'apprentissage sur l'une des fonctions n'altère pas l'apprentissage sur l'autre fonction (par exemple si l'un des fonctions est mise à jour beaucoup plus rapidement que l'autre). Finalement, les algorithmes à base de politique sont assez analogues dans le cas sans états et avec état. L'algorithme REINFORCE correspond globalement à LRI et l'acteur-critique correspond à l'algorithme de la poursuite. L'algorithme de l'acteur-critique et de la poursuite tentent tous deux de modifier localement la politique en se basant sur les performances estimées des actions de l'agent. L'algorithme LRI peut être vue comme un cas particulier de REINFORCE dans lequel la politique est représenté par une loi multinomiale.

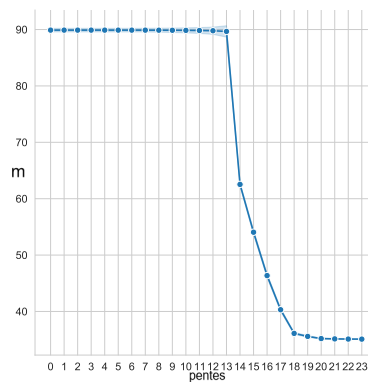
# Annexe B

## Profil du système par choix de variation du signal

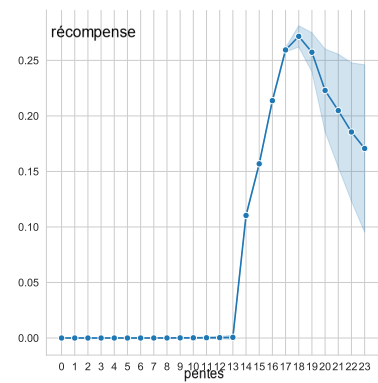
### B.1 L'agent des pentes



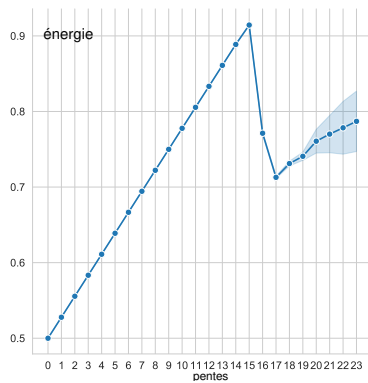
(a) Énergie -  $pe = 0.05$



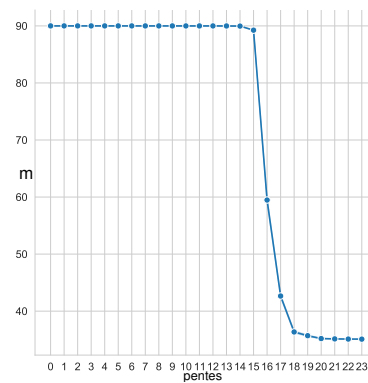
(b) Valeurs de  $m$  -  $pe = 0.05$



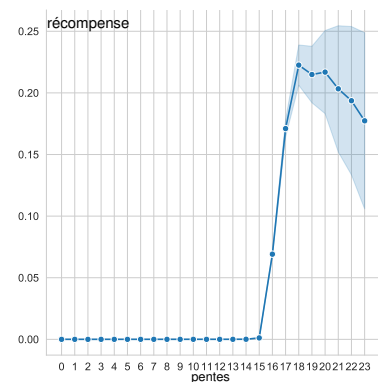
(c) Récompenses -  $pe = 0.05$



(d) Énergie -  $pe = 0.95$



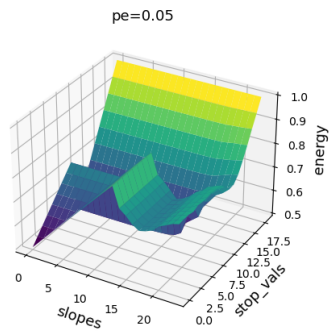
(e) Valeurs de  $m$  -  $pe = 0.95$



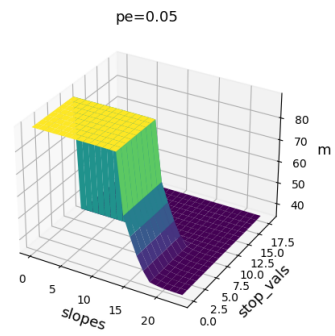
(f) Récompenses -  $pe = 0.95$

Figure B.1 – Performances des pentes en termes d'énergie, de confort de l'utilisateur ainsi que de récompense pour  $pe = 0.05$  et  $pe = 0.95$

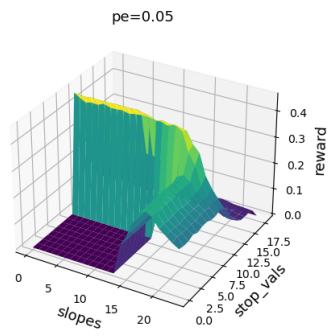
### B.2 Le système complet



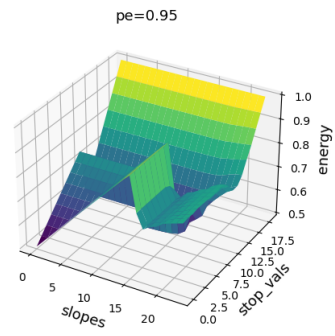
(a) Énergie -  $pe = 0.05$



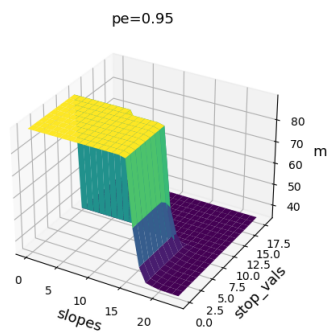
(b) Valeurs de  $m$  -  $pe = 0.05$



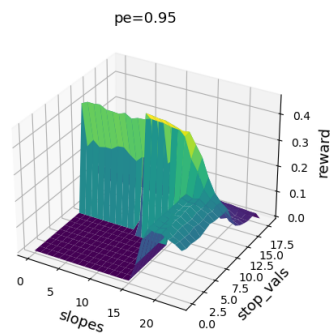
(c) Récompenses -  $pe = 0.05$



(d) Énergie -  $pe = 0.95$



(e) Valeurs de  $m$  -  $pe = 0.95$



(f) Récompenses -  $pe = 0.95$

Figure B.2 – Performances des stratégies du système par variation de signal en termes d'énergie, de confort de l'utilisateur ainsi que de récompense pour  $pe = 0.05$  et  $pe = 0.95$



# Bibliographie

- [1] Gracia AD, Fernández C, Castell A, Mateu C, and Cabeza LF. Control of a pcm ventilated facade using reinforcement learning techniques. *Energy and Buildings*, 2015.
- [2] International Energy Agency. Tracking clean energy progress 2021. Technical report, International Energy Agency, 2021.
- [3] Smart Building Alliance. Ready2services le label du batiment connecte et communicant. Technical report, Smart Building Alliance, 2022.
- [4] Marilyn Andersen and al. An intuitive daylighting performance analysis and optimization approach. *Building Research and Information*, 2008.
- [5] Tuomas Haarnoja and Aurick Zhou and Pieter Abbeel and Sergey Levine. Soft actor-critic : Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *Proceedings of the 35th International Conference on Machine Learning*, 2018.
- [6] Emilie Kaufmann and Olivier Cappe and Aurelien Garivier. On bayesian upper confidence bounds for bandit problems. *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, 2012.
- [7] Association Suisse Des Electriciens (ASE). Eclairage interieur par la lumière du jour. 1989.
- [8] ASHRAE. *Handbook—Fundamentals Chapter 9*. 2009.
- [9] Peter Auer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 2002.
- [10] Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. Gambling in a rigged casino : The adversarial multi-armed bandit problem. *Proceedings of IEEE 36th Annual Foundations of Computer Science*, 1995.
- [11] C. Benton. Lockheed 157 monitoring project phase ii : The lighting control system. *Lawrence Berkeley National Laboratory*, 1989.
- [12] R.R. Bitmead, M. Gevers, and V. Wertz. Adaptive optimal control : the thinking man's gpc. *Prentice-Hall*, 1990.
- [13] Pierre Bremaud. *Markov Chains : Gibbs Fields, Monte Carlo Simulation, and Queues*. Springer, 2008.
- [14] Alex Buckman, Martin Mayfield, and Stephen Beck. What is a smart building? *Smart and Sustainable Built Environment*, 2014.
- [15] Salvatore Carlucci, Francesco Causone, Francesco De RosaLorenzo, and Lorenzo Pagliano. A review of indices for assessing visual comfort with a view to their use in optimization processes to support building integrated design. *Renewable and Sustainable Energy*, 2015, 2015.

- [16] S. Carlucci<sup>1</sup>, and L. Bai, and R. de Dear, and L. Yang. Review of adaptive thermal comfort models in built environmental regulatory documents. *Building and Environment*, 2018.
- [17] M.E. Fountain C.C. Benton, F.S. Bauman. A field measurement system for the study of thermal comfort. *ASHRAE Trans.*, 1990.
- [18] P. Chauvel, J.B. Collins, R. Dogniaux, and J. Longmore. Glare from windows : current views of the problem. *Lighting*, 1982.
- [19] Yujiao Chen, Leslie K. Norford, Holly W. Samuelson, and Ali Malkawi. Optimal control of hvac and window systems for natural ventilation through reinforcement learning. 2018.
- [20] Zhijin Cheng, Qianchuan Zhao, Fulin Wang, Yi Jiang, Li Xia, and Jinlei Ding. Satisfaction based q-learning for integrated lighting and blind control. *Energy and Buildings*, 2016, 2016.
- [21] Citepa. Inventory of atmospheric pollutant and greenhouse gas emissions in france -ecten format. Technical report, Citepa, 2019.
- [22] I. Cowling, S. Coyne, and G. Bradley. Light in brisbane office buildings : A survey. *Research Report, Centre for Medical and Health Physics*, 1990.
- [23] Ernst D, Geurts P, and Wehemkel L. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 2005.
- [24] R. de Dear. Thermal comfort in practice. *Indoor Air*, 2004.
- [25] R. de Dear. Revisiting an old hypothesis of human thermal perception : alliesthesia. *Build. Res. Inf.* 39, 2011.
- [26] de Dear Richard and Brager G.S. Developing an adaptive model of thermal comfort and preference. *AHRAE Transactions 1998*, 1998.
- [27] Ministère de la Transition écologique et de la Cohésion des territoires. Énergie dans les bâtiments. Technical report, Ministère de la Transition écologique et de la Cohésion des territoires, 2021.
- [28] Ministère de la Transition écologique et de la Cohésion des territoires. Loi relative à la transition énergétique pour la croissance verte (tepcv). Technical report, Ministère de la Transition écologique et de la Cohésion des territoires, 2022.
- [29] David L. DiLaura and al. *The lighting handbook*. Illuminating Engineering Society of North America, 2000.
- [30] Richard O. Duda and al. *Pattern Classification (2nd ed.)*. New York : John Wiley Sons, 2001.
- [31] Kahkonen E. Draught, radiant temperature asymmetry and air temperature – a comparison between measured and estimated thermal parameters. *Indoor Air*, 1, 1991.
- [32] Kahkonen E and Ilmarinen R. Assessing indoor thermal climate – a critical discussion. *Indoor Air*, 1, 1989.
- [33] Bradley Efron. Bootstrap methods : Another look at the jackknife. *Ann. Statist*, 1979.
- [34] H.D. Einhorn. Discomfort glare : a formula to bridge differences. *Lighting Research and Technology*, 1979.
- [35] Joao M.M. Gomesa Eusébio Z.E.Conceição, Nuno H. Antãoa, and M Manuela J.R. Lúcio. Application of a developed adaptive model in the evaluation of thermal comfort in ventilated kindergarten occupied spaces. *Processes 2011*, 2011.

- [36] Ruelens F, Claessens BJ, Quaiyum S, Schutter BD, Babuška R, and Belmans R. Reinforcement learning applied to an electric water heater : From theory to practice. *IEEE Transactions on Smart Grid*, 2016.
- [37] P. O. Fanger. Thermal comfort : analysis and applications in environmental engineering. *Danish Technical Press*, 1970.
- [38] Asma Ahmad Farhan and al. Predicting individual thermal comfort using machine learning algorithms. *IEEE International Conference on Automation Science and Engineering (CASE)*, 2015.
- [39] A. Fawzi, M. Balog, and A. Huang et al. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature* 610, 2019.
- [40] Michael Humphreys Fergus Nicol. Adaptive thermal comfort and sustainable thermal standards for buildings. *Energy and Buildings*, 2002.
- [41] Pedro M. Ferreira and al. Neural network pmv estimation for model-based predictive control of hvac systems. *IEEE World Congress on Computational Intelligence*, 2012.
- [42] R. A. Fisher and Edward John Russell. On the mathematical foundations of theoretical statistics. *Philosophical Transactions of the Royal Society*, 1922.
- [43] Global Alliance for Buildings and Construction. 2021 global status report for buildings and construction. Technical report, Global Alliance for Buildings and Construction, 2021.
- [44] M.E. Fountain, G.S. Brager, and R.J. Dear. A field measurement system for the study of thermal comfort. *Expectations of indoor climate control, Energy Build*, 1996.
- [45] Christoph F.Reinhart. Lightswitch-2002 : a model for manual and automated control of electric lighting and blinds. *Solar Energy*, 77 :15–28, 2004.
- [46] M. Frontczak and P. Wargocki. Literature survey on how different factors influence human comfort in indoor environments. *Build. Environ.* 46, 2011.
- [47] Vishal Garg and N.K.Bansal. Smart occupancy sensors to reduce energy consumption. *Energy and Buildings*, 32 :91–87, 2000.
- [48] J. C. Gittins. Bootstrap methods : Another look at the jackknife. *Journal of the Royal Statistical Society*, 1979.
- [49] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016. <http://www.deeplearningbook.org>.
- [50] BIM Interoperability Expert Group. Bim interoperability expert group report. Technical report, BIM Interoperability Expert Group, 2020.
- [51] Costanzo GT, Iacovella S, Ruelens F, Leurs T, and Claessens BJ. Experimental analysis of data-driven control for a building heating system. *Sust. Energy. Grids*, 2016.
- [52] S.K. Guth. A method for the evaluation of discomfort glare. *Illuminating Engineering*, 1963.
- [53] Moradi H, Saffar-Avval M, and Bakhtiari-Nejad F. Nonlinear multivariable control and performance analysis of an air-handling unit. *Energy and Buildings*, 2012.
- [54] Mengjie Han, Xingxing Zhang, Liguu Xu, Ross May, Song Pan, and Jinshun Wu. A review of reinforcement learning methodologies on control systems for building energy. *Borlänge : Högskolan Dalarna*, 2018., p.26, 2018.

- [55] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning : Data Mining, Inference, and Prediction*. Springer Series in Statistics, 2009.
- [56] J.L.M. Hensen. Literature review on thermal comfort in transient conditions. *Building and Environment*, 1990.
- [57] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 1963.
- [58] W. Huang and H.N. Lam. Using genetic algorithms to optimize controller parameters for hvac systems. *Energy and Buildings*, 1997.
- [59] M.A. Humphreys. The clothing and thermal comfort of secondary school children in summertime. *The Building Services Engineer*, 1973.
- [60] Michael Humphreys and Fergus Nicol. Understanding the adaptive approach to thermal comfort. *ASHRAE Transactions*, 1998.
- [61] IES. Approved method : les spatial daylight autonomy (sda) and annual sunlight exposure (ase). *Illuminating Engineering Society*, 2012.
- [62] A. Iggo. Cutaneous thermoreceptors in primates and sub-primates. *J Physiol.*, 1969.
- [63] T. Iwata and M. Tokura. Examination of the limitations of predicted glare sensation vote (pgsv) as a glare index for a large source : towards a comprehensive development of discomfort glare evaluation. *Lighting Research and Technology*, 1998.
- [64] Ahn J, Cho S, and Chung DH. Analysis of energy and control efficiencies of fuzzy logic and artificial neural network technologies in the heating energy supply system responding to the changes of user demands. *Applied energy*, 2017.
- [65] Bai J and Zhang X. A new adaptive pi controller and its application in hvac systems. *Energy Conv Mana*, 2007.
- [66] Liang J and Du R. Thermal comfort control based on neural network for hvac application. *Proc of IEEE Conf on Contr Appl*, 2005.
- [67] Judith D. Jennings, Francis M. Rubinstein, Dennis DiBartolomeo, and Steven L. Blanc. Comparison of control options in private offices in an advanced lighting controls testbed. *Journal of Illuminating Engineering Society*, 29 :39–60, 2000.
- [68] Dalamagkidis K, Kolokotsa D, Kalaitzakis K, and Stavrakakis GS. Reinforcement learning for energy conservation and comfort in buildings. 2007.
- [69] Rajaraman Kanagasabai and P. S. Sastry. Finite time analysis of the pursuit algorithm for learning automata. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 1996.
- [70] S. Kanaja. Response to 'lighting quality : the unanswered questions' by peter boyce. *In : Proceedings of the First CIE Symposium on Lighting Quality*, 1998.
- [71] Z. Kanyu and Z. Jun. A particle swarm optimization approach for optimal design of pid controller for temperature control in hvac. In *2011 Third International Conference on Measuring Technology and Mechatronics Automation(ICMTMA)*, volume 01, pages 230–233, 01 2011.

- [72] Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in Neural Information Processing Systems 12*, 1999.
- [73] A. Kusiak, G. Xu, and F. Tang. Optimization of hvac control system strategy using two-objective genetic algorithm. *Energy*, 2011.
- [74] T. L. Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Applied Mathematics*, 1985.
- [75] S. Lakshminarayanan. Learning algorithms theory and applications. 1981.
- [76] Seungjae Lee and al. Inference of thermal preference profiles for personalized thermal environments with actual building occupants. *Building and Environment*, 2019.
- [77] A.S. Linney. Maximum luminances and luminance ratios and their impact on users' discomfort glare perception and productivity in daylight offices. Victoria University of Wellington. : Wellington (NZ), 2008.
- [78] Siliang Lu, Erica Cochran Hameen, and Azizan Aziz. Dynamic hvac operations with real-time vision-based occupant recognition system. *ASHRAE Winter Conference*, 2018.
- [79] M. Luckiesh and S.K. Guth. Brightness in visual field at borderline between comfort and discomfort (bcd). *Illuminating Engineering 44*, 1949.
- [80] Anderson M, Buehner M, Young P, Hottle D, Anderson C, Tu J, and al. Mimo robust control for hvac systems. *IEEE Transactions on Control Systems Tech*, 2008.
- [81] Y. Ma and et al. Model predictive control for the operation of building cooling systems. *IEEE Transactions on Control Systems Technology*, 2012.
- [82] J. Mardaljevic. Rethinking daylighting and compliance. *SLL/CIBSE International Lighting Conference. 2013*, 2013.
- [83] D.A. McIntyre. Chapter 13 design requirements for a comfortable environment. *Studies in Environmental Science*, 1981.
- [84] Fountain M.E., Brager G.S., and de Dear R.J. Expectations of indoor climate control. *Energy and Building*, 1996.
- [85] B. Meerbeek, T. van Druenen, M. Aarts, and E. van Loenen. Impact of blinds usage on energy consumption : automatic versus manual control. *European Conference on Ambient Intelligence*, 2014.
- [86] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. 2013. cite arxiv :1312.5602Comment : NIPS Deep Learning Workshop 2013.
- [87] 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, Dharmashan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540) :529–533, February 2015.
- [88] M. Mohri, A. Rostamizadeh, and A. Talwaker. *Foundations of Machine learning*. Boston : MIT Press, 2 edition, 2018.

- [89] A. Nabil and J. Mardaljevic. Useful daylight illuminances : A replacement for daylight factors. *Energy and Buildings*, 2006.
- [90] Zoltan Nagy, Fah Yik Yong, Mario Frei, and Arno Schlueter. Occupant centered lighting control for comfort and energy efficient building operation. *Energy and Buildings*, 94 :100–108, 2015.
- [91] Zoltán Nagy and al. Illuminating adaptive comfort : Dynamic lighting for the active occupant. *Conference : Proceedings of 8th Windsor Conference : Counting the Cost of Comfort in a changing World*, 2014.
- [92] Kumpati S. Narendra and Mandayam A.L. Thathachar. *Learning Automata : An Introduction*. Prentice Hall, 1989.
- [93] N. Nassif, S. Kajl, and R. Sabourin. Optimization of hvac control system strategy using two-objective genetic algorithm. *HVACR Research*, 2005.
- [94] Bill Von Neida, Dorene Maniccia, and Allan Tweed. An analysis of the energy and cost savings potential of occupancy sensors for commercial lighting systems. *Journal of Illuminating Engineering Society*, 30 :111–125, 2001.
- [95] Rahul Nellikkath and Spyros Chatzivasileiadis. Physics-informed neural networks for ac optimal power flow. *Electric Power Systems Research*, 2022.
- [96] V. Novák, I. Perfilieva, and J. Močkoř. *Mathematical principles of fuzzy logic*. Springer, 1999.
- [97] International Commission on Illumination (CIE). Discomfort glare in the interior working environment. *CIE Publication 55*, 1983.
- [98] W.K.E. Osterhaus. Discomfort glare from daylight in computer offices : how much do we really know? In : *Proceedings of LUX Europa 2001, 9th European Lighting Conference*, 2001.
- [99] W.K.E. Osterhaus and I.L. Bailey. Large area glare sources and their effect on discomfort and visual performance at computer workstations. *Proceedings of the IEE Industry Applications Society Annual Meeting*, 1992.
- [100] S. Tanishima P. Waide. Light's labour's lost : Policies for energy efficient lighting. *OECD/IEA*, 2006.
- [101] June Young Park, Thomas Dougherty, Hagen Fritz, and Zoltan Nagy. *LightLearn : An adaptive and occupant centered controller for lighting based on reinforcement learning*. *Building and Environment*, 2019.
- [102] P. Petherbridge and R.G. Hopkinson. Discomfort glare and the lighting of buildings. *Transactions of the Illuminating Engineering Society*, 1950.
- [103] W. Pohl and M. Werner. Lighting, comfort and energy and the evaluation of the recommendations of en 15251. *Intelligent Energy Europe*, 2010.
- [104] L.S. Pontryagin. *Mathematical Theory of Optimal Processes*. CRC Press, 1987.
- [105] Sutton RS, McAllester DA, Singh SP, and Mansour Y. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems (NIPS)*, 1999.
- [106] G. A. Rummery. *Problem Solving with Reinforcement Learning*. PhD thesis, Cambridge University, 1995.

- [107] Liu S and Henze GP. Experimental analysis of simulated reinforcement learning control for active and passive building thermal storage inventory part 1. *Theoretical foundation*, 2006.
- [108] Liu S and Henze GP. Experimental analysis of simulated reinforcement learning control for active and passive building thermal storage inventory part 2. *Theoretical foundation*, 2006.
- [109] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. 2016.
- [110] F. Sicurella, G. Evola, and E. Wurtz. A statistical approach for the evaluation of thermal and visual comfort in free-running buildings. *Energy and Buildings*, 2012.
- [111] D. Silver, A. Huang, C. Maddison, and al. Mastering the game of go with deep neural networks and tree search. *Nature* 529, 2016.
- [112] S. P. Singh and R. S. Sutton. Reinforcement learning with replacing eligibility traces. machine learning. *Machine Learning*, 1996.
- [113] Satinder P. Singh and Richard S. Sutton. Reinforcement learning with replacing eligibility traces. *Machine Learning* 22, pages 123–158, 1996.
- [114] Nandan Sudarsanam and Balaraman Ravindran. Linear bandit algorithms using the bootstrap. 2016.
- [115] Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning* 3, 1988.
- [116] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998.
- [117] R. Tedrake, T. W. Zhang, and H. S. Seung. Stochastic policy gradient reinforcement learning on a simple 3d biped. *International Conference on Intelligent Robots and Systems (IROS)*, 2004.
- [118] Jean Terrien and François Desvignes. *La Photométrie*. Presses universitaires de France edition, 1972.
- [119] M A L Thathachar. Stochastic automata and learning systems. *Sadhana, Vol 15, 1990*, 1990.
- [120] William R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 1933.
- [121] William R. Thompson. On the theory of apportionment. *American Journal of Mathematics*, 1935.
- [122] J.N. Tsitsiklis and B. Van Roy. An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, pages 674–690, 1997.
- [123] Joost van Hoof, and Mitja Matej, and Jan LM Hensen. Thermal comfort : research and practice. *Frontiers in Bioscience*, 2010.
- [124] Pierre-François Verhulst. *Deuxième mémoire sur la loi d'accroissement de la population*. Mémoires de l'Académie Royale des Sciences, des Lettres et des Beaux-Arts de Belgique, 1847.
- [125] Richard Vinter. *Optimal Control*. Springer, 2010.
- [126] O. Vinyals, I. Babuschkin, W.M. Czarnecki, and al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature* 575, 2019.

- [127] J.W.T. Walsh. The early years of illuminating engineering in great britain. *Transactions of the Illuminating Engineering Society*, 1951.
- [128] Zhe Wan, Richard de Dear, Maohui Luoa, Borong Linc, Yingdong Hea, Ali Ghahramania, and Yingxin Zhuc. Individual difference in thermal comfort : A literature review. *Building and Environnement 138 (2018)*, 2018.
- [129] Christopher Watkins and John Cornish Hellaby. *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, UK, May 1989.
- [130] C.J.C.H. Watkins and Peter Dayan. Q-learning. *Machine Learning 8*, 1992.
- [131] T. Wei, Q. Zhu, and M. Maasoumy. Co-scheduling of hvac control, ev charging and battery usage for building energy efficiency. *ICCAD*, 2014.
- [132] Tianshu Wei, Yanzhi Wang, Qi Zhu, and al. Deep reinforcement learning for building hvac control. *IEEE*, 2017.
- [133] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Machine Learning*, pages 229–256, 1992.
- [134] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning 8*, page 229–256, 1992.
- [135] Zhibin Wu and al. Using an ensemble machine learning methodology-bagging to predict occupants' thermal comfort in buildings. *Energy and Buildings*, 2018.
- [136] Xu X, He H, and Hu D. Efficient reinforcement learning using recursive least-squares methods. *Journal of Artificial Research*, 2002.
- [137] Lei Yang, Nagy Zoltán, Philippe Goffin, and Arno Schlueter. Reinforcement learning for optimal control of low exergy buildings. *Applied Energy*, 2015.
- [138] Anis Yazidi, Xuan Zhang, Lei Jiao, and B John Oommen. The hierarchical continuous pursuit learning automation : a novel scheme for environments with large numbers of actions. *IEEE transactions on neural networks and learning systems*, 31(2) :512–526, 2019.
- [139] Wang YG, Shi ZG, and Cai WJ. Pid autotuner and its application in hvac systems. *IEEE American Control Conf*, 2001.
- [140] L. Yu and al. Multi-agent deep reinforcement learning for hvac control in commercial buildings. *IEEE Transactions on Smart Grid*, 12(1) :407–419, 2021.
- [141] Yu Z and Dexter A. Online tuning of a supervisory fuzzy controller for low-energy building system using reinforcement learning. *Contr Engi Prac*, 2010.
- [142] Yu Z, Huang G, Haghghat F, Li H, and Zhang G. Control strategies for integration of thermal energy storage into buildings : State-of-the-art review, 2015.
- [143] H. Zhang, E. Arens, C. Huizenga, and T. Han. Thermal sensation and comfort models for non-uniform and transient environments : Part i : Local sensation of individual body parts. *Build. Environ.* 45, 2010.
- [144] H. Zhang, E. Arens, C. Huizenga, and T. Han. Thermal sensation and comfort models for non-uniform and transient environments : Part ii : Local comfort of individual body parts. *Build. Environ.* 45, 2010.



- [145] H. Zhang, E. Arens, C. Huizenga, and T. Han. Thermal sensation and comfort models for non-uniform and transient environments : Part iii : Whole-body sensation and comfort. *Build. Environ.* 45, 2010.
- [146] Zhiang Zhang, Adrian Chong, Yuqi Pan, Chenlu Zhang, and Khee Poh Lam. Whole building energy model for hvac optimal control : A practical framework based on deep reinforcement learning. *Energy and Buildings*, 2019.