Vers une Intelligence Artificielle Autonome et Explicable pour des Environnements Incertains

---

Présenté par

Orhand Romain

Spécialité : Informatique

Thèse présentée en vue de l'obtention du titre de

Docteur de l'Université de Strasbourg

Réalisée dans le Laboratoire de Recherche Icube UMR 7357
avec l'ED 269 Mathématiques, Sciences de l'Information et de l'Ingénieur

Sous la supervision de :
Pr Pierre Collet — PR, Université de Strasbourg
Pr Pierre Parrend — PR, EPITA
Dr Anne Jeannin-Girardon — MCF, Université de Strasbourg

Soutenue le 10/11/2022 devant le jury composé de :

Rapporteurs :
Pr Sylvain Cussat-Blanc — PR, Université de Toulouse 1 Capitole
Pr Cecilia Zanni-Merk — PR, INSA Rouen

Examinateurs :
Pr Pierre De Loor — PR, ENIB
Dr Marc Schoenauer — DR, INRIA Saclay

– Ce n'est pas que je ne pourrais pas te l'expliquer. Ce que je veux dire, c'est qu'à peine l'explication te serait-elle donnée avec des mots que sa signification disparaîtrait.

Haruki Murakami, 1Q84, Livre 2 Juillet - Septembre

Remerciements

Quel long chemin que celui d'un thèse ! En partant de la fin : une soutenance, des rapports, la rédaction d'un manuscrit, des conférences, des articles, des échecs et des succès lors des travaux de recherche, beaucoup de ratés même avant de trouver un début de solution, un sujet à s'approprier, …jusqu'à trouver ce sujet de thèse grâce à un petit e-mail de quelques lignes à peine. Sans compter tout ce qui pu se passer durant ces années comme la Covid-19 ou toutes ces rencontres !

Je tiens alors à d'abord chaleureusement remercier mon jury, Pr Pierre De Loor, Pr Sylvain Cussat-Blanc, Dr Marc Schoenauer et Pr Cecilia Zanni-Merk, qui a accepté d'examiner mon travail, de relire ce manuscrit et pour la patience que vous avez eue à mon égard.

Mes travaux de thèse se sont déroulés sous la direction de Pierre et de Pierre. Je vous remercie sincèrement pour votre confiance et pour l'autonomie que vous m'avez accordée. Ces deux éléments me sont très importants et m'ont permis au travers de cette thèse de pouvoir complètement m'exprimer sous vos yeux avisés et de me pousser toujours plus loin en dehors de mes zones de confort. Il est parfois compliqué de pouvoir trouver les mots justes qui seraient capables de transmettre tout ce que nous pouvons avoir en tête. Il m'est ainsi difficile de savoir comment t'exprimer toute ma gratitude pour tout ce que tu m'as apporté Anne : rigueur, créativité, dépassement de soi, conseils, écoute, moments de partage, … et soutien durant toutes ces années.

Durant ma thèse, je me suis découvert une nouvelle famille. Une famille à laquelle je ne m'attendais pas du tout en arrivant sur Strasbourg et qui est l'une des plus belles surprises qui me soit arrivée. La famille CSTB avec Claudine, Julie, Odile, Arnaud, Kirsley, Audrey, Julio, Yannis, Raymond, Laetitia, Jean-Sébastien, Christian, Quentin, Hiba, Julien, Christelle, Célia, Nathalie, Tam'si, Nicolas, Dorine, Lucille, Arthur, Anna, Rabih, Lalla, Soumaya, Amani et Anne. Je pense qu'il serait possible d'écrire de nombreuses lignes

sur toutes nos aventures, qui en terme de volume dépasseraient largement ce manuscrit ! Merci pour tout ce que vous avez pu m'apporter au quotidien, pour votre gentillesse, pour votre enthousiasme et pour tous ces souvenirs chaleureux et motivants.
Olivier, Luc, Nicolas, Thomas, Ali, je ne vous oublie pas et comment pourrais-je vous oublier ! Un « chef » d'équipe fédérateur, un bon vivant toujours disponible trop adepte de tcl, mes deux compagnons de thèse et un ami où je suis à chaque fois surpris de voir à quel point nos points de vue se ressemblent. Vous avez été tous les 5 une très grande source de motivation, de bienveillance, de rires et de soutien. Pour tout cela et plus encore, merci.

Comme je le disais, cette aventure a pu démarrer grâce à un petit e-mail et grâce à vos soutiens quand je vous avais parlé de mon projet professionnel durant ma scolarité à TELECOM Nancy. Alors merci Olivier Festor d'avoir pensé à moi quand tu as reçu cette offre de thèse à partager et de m'avoir soutenu, merci aussi à Sébastien Da Silva, Marc Tomczak, Vincent Bombardier et Alexandre Parodi. Je vous en suis très reconnaissant.

Encore de nombreuses personnes m'ont accompagné, écouté et soutenu, m'ont permis de me changer les idées quand j'en avais besoin ou m'ont rappelé aux besoins élémentaires des êtres humains. Alors merci les « gendicapés », Marie, Anaïs, Guénolé, Marine, Clémence, Alexis et Loïc. Même si tu as décidé de suivre une voie différente, je tiens sincèrement à te remercier Justine pour tout ce que tu m'a permis d'apprendre. Et pour tous ces éléments et aussi tellement plus, Betty, Julien, Aurélien, merci du fond du cœur.

Enfin, merci également à ma famille qui a toujours été d'un soutien inconditionnel. Moun, Padre, Maïlys, Marine, Alexandre, cette thèse est aussi un peu la vôtre. Et toi, ma Eden, si un jour tu lis ces lignes (tu auras effectivement bien grandi !), j'espère que tu y verras la preuve de toujours essayer et persévérer. Tente ma belle. Car, comme je le voyais très souvent sur le chemin du laboratoire :
« Á force de se planter, un beau jour on devient une fleur ».

Table des matières

Liste des illustrations

Liste des tableaux

Liste des algorithmes

Liste des acronymes

ACS Anticipatory Classifier Systems . . . 52
ALCS Anticipatory Learning Classifier Systems . . . 4
ALP Anticipatory Learning Process . . . 59
BACS Behavioral Anticipatory Classifier System . . . 141
BEACS Behavioral Enhanced Anticipatory Classifier System . . . 199
BSeq Behavioral Sequences . . . 70
EMPA Erreur Moyenne sur la Prédiction de l'Anticipation . . . 132
EPE Enhanced Predictions through Experience . . . 200
GA Genetic Algorithms . . . 38
GABIL Genetic Algorithm Batch-Incremental concept Learner . . . 32
LCS Learning Classifier System . . . 25
PAI Perceptual Aliasing Issue . . . 6
PEP Probability-Enhanced Predictions . . . 94
PEPACS Probability-Enhanced Predictions in Anticipatory Classifier System . . . 173
ZCS Zeroth-level Classifier System . . . 51

Introduction

L'Intelligence Artificielle est aujourd'hui de plus en plus présente dans notre quotidien. Elle est capable de nous recommander des vidéos sur Internet (The Conversation, 2021), d'écrire des articles de presse ou des œuvres de fiction à s'y méprendre avec ceux écrits par un être humain (Midi Libre, 2019), de faire danser des robots (IEEE Spectrum, 2021), de concevoir des antibiotiques (Stokes et al., 2020) ou encore, de participer à la création de contenus vidéoludiques (Ouest France, 2021) et artistiques (Makery, 2016).

L'expression « Intelligence Artificielle », et par extension la discipline associée, a été formulée pour la première fois en 1956 par John McCarthy lors d'un atelier de recherche au *Dartmouth College* (Russell and Norvig, 2003) dont une première définition a été proposée par Marvin Lee Minsky [1]. Bien qu'aucune définition ne soit arrêtée, elle peut aujourd'hui être définie comme la compréhension des fonctions cognitives humaines et les capacités des machines à les reproduire (Villani et al., 2018). Elle emprunte les outils et les théories de nombreuses autres disciplines telles que le traitement de l'information, les mathématiques, les sciences cognitives, la psychologie, la philosophie, ou encore la linguistique.

L'une des disciplines les plus emblématiques de l'intelligence artificielle est l'apprentissage automatique. L'apprentissage automatique désigne la capacité des ordinateurs à apprendre sans être explicitement programmés (Samuel, 1959). Elle représente une classe de méthodes algorithmiques qui établissent des relations à partir de données d'apprentissage afin de réaliser des prédictions ou d'inférer des règles.
L'apprentissage automatique profond est une sous-classe d'algorithmes d'apprentissage automatique dont la particularité est la modélisation des données avec de multiples niveaux d'abstraction (LeCun et al., 2015) : des représen-

1. *« The building of computer programs which perform tasks which are, for the moment, performed in a more satisfactory way by humans because they require high-level mental processes such as perception, learning, memory organization and critical reasoning. »*

tations sont construites en composant différents modules entre eux, eux-mêmes utilisés ensuite pour composer de nouvelles représentations, et ainsi de suite. Les ordinateurs deviennent alors capables d'apprendre de leurs expériences, que ce soit par des retours provenant de leurs interactions avec leur environnement ou par le biais de leurs erreurs.

Ces 10 dernières années, l'apprentissage automatique a connu un essor important grâce au développement de cartes graphiques massivement parallèles et à l'émergence des masses de données. Les capacités d'apprentissage des ordinateurs deviennent de plus en plus importantes, leur permettant de repousser leurs limites et de résoudre par eux-mêmes de nouvelles tâches d'apprentissage. Les réseaux de neurones profonds ont, en particulier, été employés pour leurs capacités de passage à l'échelle et d'extraction des caractéristiques issues de ces grandes masses de données. Ils ont été employés avec succès dans de nombreux domaines, étant même capables de rivaliser avec l'homme sur certaines tâches spécifiques.
Le modèle *AlphaStar* est capable de se confronter aux meilleurs joueurs du jeu de stratégie en temps réel *StarCraft II* (Vinyals et al., 2019) : pour gagner, il faut pouvoir mettre en place des stratégies devenant fructueuses seulement après plusieurs centaines d'actions, alors que de nombreuses actions sont possibles à tout moment.
Le modèle *AlphaFold* est capable de prédire la structure tridimensionnelle des protéines et donc la fonction des protéines, apportant une aide précieuse aux biologistes (Jumper et al., 2020).
Enfin, le modèle *Vision to Phoneme* est capable de convertir les mouvements labiaux en texte avec plus d'exactitude que les spécialistes humains (Shillingford et al., 2018) : *Vision to Phoneme* atteint un taux d'erreur de 40.9% quand les spécialistes atteignent au mieux un taux d'erreur de 86.4% sur le même jeu de données.

Les tâches à résoudre par les algorithmes d'apprentissage automatique sont de plus en plus complexes, d'autant plus que les interactions de ces algorithmes avec leur environnement peuvent être rendues difficiles par la présence, l'absence ou la conjonction de plusieurs facteurs : la perception de leur environnement (*i.e.* leurs données d'entrée) peut être partielle, bruitée ou incomplète ; de nombreuses étapes à réaliser successivement sont nécessaires pour trouver une solution à la tâche, *etc.* Dans le jeu *StarCraft II*, *AlphaStar*

évolue avec une observation partielle de son environnement et doit prendre des décisions en temps réel, quand 10^{26} actions différentes sont possibles à tout moment. Ce jeu ne possède pas de stratégies gagnantes à coup sûr, rendant nécessaires de fortes capacités d'adaptation et de prévision. Ces facteurs sont par ailleurs le propre de nombreux environnements réels dans lesquels les intelligences artificielles doivent évoluer : elles doivent réaliser des prédictions fiables dans le domaine médical même si les données sont particulièrement bruitées et peu nombreuses, ou encore être capables de piloter un véhicule en toute sécurité alors que leur perception peut être obstruée par de nombreux éléments.

Cependant, les modèles d'apprentissage automatique (profonds ou non) ne sont pas seulement à l'origine de *success stories* : des biais présents en amont dans les données d'apprentissage vont se retrouver dans les modèles, pouvant par exemple engendrer des discriminations (Patrice Bertail and Waelbroeck, 2019). Par exemple, les personnes à la peau de couleur noire ont un accès plus restreint que celles à la peau de couleur blanche au système médical des États-Unis (Ledford, 2019). Le modèle *Generative Pre-trained Transformer*, crée par *OpenAI*, produit du texte qui ressemble à s'y méprendre à du texte écrit par l'homme (ZDNet, 2020) : la diffusion de ce modèle a été limitée dans un premier temps parce qu'il pouvait être dangereux de le rendre public et qu'il serve à aggraver des campagnes de désinformation ou d'informations mensongères.

Rendre compréhensible les comportements de ces modèles devient une nécessité quand ceux-ci peuvent avoir de dramatiques conséquences dans le cadre d'applications critiques, telles que celles associées à la santé, à la justice, à la diffusion de l'information ou encore à la navigation autonome (The New York Times, 2018). Pour certaines applications, l'explicabilité des algorithmes (d'apprentissage automatique ou non) est même *« obligatoire »*. En France par exemple, une loi indique que des services de conciliation, de médiation ou d'arbitrage ne peuvent être uniquement fondés sur *« un traitement algorithmique ou automatisé de données à caractère personnel »* : parmi les prérequis, l'explicabilité de l'algorithme *« en détail et sous une forme intelligible »* est obligatoire (legifrance.gouv.fr, 2019).

Quand les algorithmes d'apprentissage automatique sont de plus en plus

utilisés dans notre quotidien, ils doivent être en mesure de s'adapter et de co-évoluer par eux-mêmes avec des environnements complexes, changeants et marqués par l'incertitude (Faraut, 2015). Par exemple, ils peuvent ne disposer que d'informations partielles, potentiellement erronées, pour prendre des décisions et évaluer leur pertinence en vue d'atteindre un but. Une prise de décision peut désigner le choix d'une action à réaliser, une prédiction de classe ou encore une prédiction de valeur. Deux principaux objectifs sont ainsi recherchés afin que ces algorithmes résolvent des tâches de plus en plus complexes et de façon compréhensible dans notre quotidien : l'explicabilité, celle-ci pouvant même être un préalable indispensable avant de permettre leur emploi, et l'autonomie, comme un accroissement de leurs capacités d'apprentissage face à l'incertitude caractéristique du monde réel.

Cette thèse propose d'abord de cadrer le concept d'autonomie et d'explicabilité des modèles d'apprentissage automatique, ainsi que l'incertitude environnementale, comme des vecteurs permettant de guider le développement d'une intelligence artificielle autonome et explicable. Nous montrerons alors que les systèmes d'apprentissage automatique à base de règles, et plus particulièrement les systèmes de classeurs à anticipation (*Anticipatory Learning Classifier Systems (ALCS)*), permettent la mise en place d'une intelligence artificielle autonome et explicable pour des environnements incertains.

Les ALCS sont des modèles d'apprentissage automatique à base de règles que l'on peut considérer comme intrinsèquement explicables : ils génèrent des classeurs (les règles) qui réalisent des approximations locales permettant de résoudre partiellement une tâche et ils font évoluer toute la population de classeurs afin de couvrir l'ensemble du problème pour le résoudre dans son entièreté (Butz, 2015). Ils s'appuient en particulier sur un mécanisme d'anticipation consistant à projeter des représentations dans le futur : ils cherchent à déterminer les effets de leurs décisions dans leur environnement durant leur apprentissage. Par conséquent, leur mécanisme d'anticipation nous permet d'accéder aux prochaines situations qu'ils peuvent envisager pour chaque décision. Il est alors possible de construire les explications de ces systèmes à partir des informations contenues dans chaque classeur (évaluation des décisions, causes et conséquences) et aussi en comparant chaque classeur.

Les systèmes de classeurs à anticipation sont déjà capables, dans une cer-

taine mesure, d'évoluer en autonomie et de manière explicable au sein d'environnements incertains. Il est possible de renforcer leur autonomie en les dotant de moyens supplémentaires pour gérer l'incertitude des environnements, tout en améliorant leurs capacités afin de les rendre plus explicables. Nous avons donc conçu et développé de nouveaux systèmes de classeurs à anticipation capables d'évoluer en autonomie et de manière explicable dans des environnements incertains.

Organisation du manuscrit

Ce manuscrit de thèse est d'abord constitué d'une première partie « **Contexte** » divisée en 3 chapitres :

1. « **Vers des algorithmes d'apprentissage automatique à base de règles** » pose les notions d'autonomie, d'explicabilité et d'incertitude environnementale sur lesquels nous nous appuyons pour motiver l'emploi des systèmes d'apprentissage automatique à base de règles afin de parvenir à une intelligence artificielle autonome et explicable pour des environnements incertains.
2. « **Les systèmes de classeurs** » justifie d'abord l'usage des systèmes de classeurs parmi les principaux systèmes à base de règles qui existent. Il y est ensuite présenté un état des lieux des systèmes de classeurs qui explicite leur fonctionnement et leurs constituants. Ce chapitre motive enfin l'usage des systèmes de classeurs à anticipation à partir d'un paysage des principaux systèmes de classeurs qui existent.
3. « **Les systèmes de classeurs à anticipation** » détaille précisément le fonctionnement et les applications des systèmes de classeurs à anticipation, afin d'introduire les problématiques relatives à l'autonomie et l'explicabilité auxquelles ils sont confrontés dans les environnements incertains. Les approches permettant aux systèmes de classeurs à anticipation de gérer l'incertitude des environnements, à savoir les séquences comportementales et les anticipations améliorées, seront en particulier présentées afin de mettre en avant leurs avantages et leurs limites.

S'ensuit une deuxième partie « **Contributions** » constituée de 5 chapitres :

4. « **Protocole d'évaluation expérimental des systèmes de classeurs à anticipation** » introduit notre proposition de protocole expérimental commun à tous les systèmes de classeurs à anticipation. Il intègre une description des environnements d'apprentissage, des métriques d'évaluation et du protocole d'apprentissage communs à ces systèmes, permettant de garantir une étude la plus équitable possible entre ceux-ci.

5. « **BACS : Accroître l'autonomie avec des séquences comportementales** » présente et étudie un nouveau système de classeurs à anticipation, *Behavioral Anticipatory Classifier System*. Ce système intègre des séquences comportementales qui consistent en des suites de décisions. Elles permettent aux systèmes de classeurs à anticipation de gérer le problème d'aliasing perceptuel, qui se produit lorsque les systèmes ne peuvent pas différencier des situations qui sont réellement distinctes dans des environnements partiellement observables. Ce système de classeurs à anticipation est ainsi capable d'évoluer en toute autonomie dans de nouveaux environnements.

6. « **PEPACS : Accroître l'autonomie et l'explicabilité avec des anticipations améliorées** » présente et étudie un nouveau système de classeurs à anticipation, *Probability-Enhanced Predictions in Anticipatory Classifier System*. Ce système intègre des anticipations améliorées qui consistent à anticiper explicitement un ensemble de situations quand l'ALCS fait face à différentes formes d'incertitude telles que le *Perceptual Aliasing Issue (PAI)* ou un bruitage des perceptions de l'environnement. L'emploi d'anticipations améliorées permet aux systèmes de classeurs à anticipation d'effectuer leur apprentissage dans de nouveaux environnements, tout en étant capables de construire une représentation complète de ces environnements.

7. « **BEACS : Coupler les séquences de BACS et les anticipations de PEPACS** » introduit *Behavioral Enhanced Anticipatory Classifier System* qui couple pour la première fois les séquences comportementales et les anticipations améliorées. Ce couplage est rendu possible par une refonte des anticipations améliorées permettant à ce système de différencier l'incertitude relative au problème d'aliasing perceptuel des autres formes. Ce nouveau système de classeurs à anticipa-

tion intègre également de nouveaux mécanismes permettant de donner plus d'éléments explicables quant aux représentations environnementales construites, tout en améliorant les capacités d'apprentissage de ces systèmes.

8. Enfin, « **Extraire les connaissances des systèmes de classeurs à anticipation** » dresse un portrait des différentes techniques permettant d'extraire des connaissances des populations de classeurs des ALCS. Ce portrait met en avant le manque d'approches dédiées aux ALCS et justifie le besoin d'un tel algorithme : être capable d'extraire les connaissances des populations permet d'identifier quelles sont les politiques de décisions et les représentations environnementales développées par ces systèmes, et donc de fournir des explications relatives à ces éléments de façon plus détaillée. Nous proposons le premier algorithme de compression des populations de classeurs des systèmes de classeurs à anticipation.

Ce manuscrit de thèse est enfin terminé par un chapitre de conclusions qui dresse un bilan des travaux réalisés et introduits dans le présent document. Nous y décrivons également les différentes perspectives de travail que nous envisageons de poursuivre.

Première partie

Contexte

CHAPITRE 1

Vers des algorithmes d'apprentissage automatique à base de règles

De nombreuses définitions de l'autonomie et de l'explicabilité des modèles d'apprentissage automatique existent. Il est donc nécessaire de d'abord cadrer ces deux notions qui guideront par la suite les choix de conception pour parvenir à une intelligence artificielle possédant ces deux propriétés. De même, l'incertitude propre aux environnements réels peut être décomposée en différentes sources afin de guider à nouveau la mise en place d'une intelligence artificielle adaptée à ces environnements. Ces notions développées permettent d'appuyer plus particulièrement l'utilisation des systèmes d'apprentissage automatique à base de règles parmi les différentes approches existantes pour parvenir à une explicabilité et à une autonomie recherchée dans des environnements incertains.

Sommaire

1.1 Autonomie et explicabilité

1.1.1 Explicabilité

L'explicabilité — du latin *explicare* (déplier) — désigne, pour un modèle d'apprentissage automatique, le fait de fournir des informations pour que son fonctionnement soit clair ou facile à comprendre (Arrieta et al., 2020). Il s'agit de pouvoir transmettre à une tierce personne l'ensemble des éléments ayant conduit aux décisions prises par ces algorithmes (Miller, 2019) : on cherche ainsi à savoir *comment* (la manière) et *pourquoi* (l'origine) telle décision a été prise. L'explicabilité suppose donc de permettre à n'importe quelle personne de « déplier » l'ensemble des éléments ayant mené aux décisions de ces algorithmes et d'accéder à une représentation compréhensible des connaissances apprises, l'ensemble de ces informations étant transmises par ces modèles (Gilpin et al., 2018).

L'explicabilité peut alors prendre la forme d'une interface entre un algorithme d'apprentissage automatique et un être humain : l'algorithme fournit des informations relatives à la prise de décision de façon compréhensible, ces informations sont transmises à l'interface et peuvent alors être effectivement comprises par un être humain. Il existe alors une dépendance entre le public auquel les explications sont destinées et l'intelligibilité de ces explications (Arrieta et al., 2020) : par exemple, on n'explique pas une décision de la même manière à un enfant qu'à un adulte. Quand il est de plus en plus nécessaire d'expliquer les décisions prises par les algorithmes d'apprentissage automatique, il est alors de plus en plus important de prendre en compte cette dépendance au public.

1.1.2 Autonomie

L'autonomie a été définie de différentes manières selon l'interprétation de son étymologie grecque (*αὐτός* — soi-même et *νόμος* — loi) et du contexte dans lequel elle est employée. Elle a été associée à la génération et le maintien d'une identité, à la capacité a être indépendant ou encore, à la faculté de se gouverner soi-même selon ses propres règles auxquelles on se souscrit ou

que l'on créé (Kristen, 2011). La définition de l'autonomie des algorithmes d'apprentissage automatique que nous utiliserons se rapproche ici de la définition de (Boissier et al., 2003) : elle désigne leur capacité de se doter de leurs propres règles selon le contexte, qui est à entendre comme leur environnement d'apprentissage et la tâche à apprendre.

L'autonomie peut donc être directement liée à l'apprentissage automatique, puisqu'il est attendu que ces algorithmes puissent résoudre une tâche sans être programmés explicitement. Ces algorithmes cherchent à identifier quels sont les décisions à prendre ou les éléments propres à une décision, selon le contexte où ils évoluent et à partir des situations décrites par leurs données d'apprentissage. Les règles de ces algorithmes désignent ainsi le résultat de leur apprentissage : elles représentent la mise en relation entre des décisions pour résoudre leur tâche d'apprentissage et des situations pour lesquelles des éléments relatifs aux décisions pourraient être identifiés.

Renforcer l'autonomie de ces algorithmes revient alors à les doter de moyens supplémentaires pour qu'ils puissent construire leurs propres règles afin de résoudre une tâche. En particulier, ils doivent pouvoir réaliser leur apprentissage, tout en étant capables de gérer l'incertitude caractéristique de notre quotidien dans lequel ils sont de plus en plus présents.

1.2 Incertitude environnementale

Pour concevoir un algorithme d'apprentissage automatique autonome et explicable, il est possible de s'inspirer des processus comportementaux des êtres humains dans les environnements incertains. Pour ce faire, trois sources d'incertitude sont distinguables au sein des environnements : l'ambiguïté, la volatilité et la stochasticité (Payzan-LeNestour and Bossaerts, 2011). Une analyse de ces sources permet alors de dégager des pistes permettant la mise en place d'un tel algorithme d'apprentissage automatique.

1.2.1 Ambiguïté

L'ambiguïté désigne le fait que nous pouvons ne pas savoir quelles décisions nous permettent d'atteindre un objectif. L'apprentissage peut agir sur l'incertitude liée à l'ambiguïté, puisque son but est d'identifier ces décisions. (Faraut, 2015) décompose en ce sens notre processus d'apprentissage en un cycle où, à partir d'une situation et d'un objectif donnés, nous nous représentons les prochaines situations et les décisions nous permettant d'atteindre celles-ci, afin d'évaluer si elles sont en accord avec l'objectif affiché et d'adapter notre comportement [1] en conséquence. L'ambiguïté est alors intrinsèquement liée à l'évaluation de ces prochaines situations et des décisions associées, ce qui est également transposable aux algorithmes d'apprentissage automatique.

Des algorithmes d'apprentissage automatique peuvent utiliser des situations annotées avec les décisions qui lui permettent effectivement d'atteindre son objectif. L'évaluation d'une décision par ces algorithmes peut s'appuyer sur ces annotations lors de leur apprentissage ou bien, cette évaluation doit être réalisée selon les moyens propres aux algorithmes, sans éléments extérieurs tels que ces annotations. Notre recherche d'un algorithme d'apprentissage automatique autonome nous guide alors vers des approches où l'algorithme est en mesure de réaliser par lui-même l'évaluation des décisions qu'il prend. Le cadre posé par l'apprentissage par renforcement permet à un algorithme de faire cette évaluation en comparant le résultat attendu d'une décision avec le résultat obtenu suite à cette décision (Sutton et al., 1999), contrairement à l'apprentissage supervisé pour laquelle elle est donnée. Nous nous limiterons alors aux approches s'appuyant sur l'apprentissage par renforcement. La décomposition du processus d'apprentissage décrite ci-dessus (Faraut, 2015) permet de fournir des pistes quant à des éléments d'explicabilité attendus lorsqu'un agent prend une décision spécifique selon une situation et un objectif donnés tels que : les prochaines situations, les décisions permettant de les atteindre ou les différentes évaluations réalisées par un agent.

1. Notre « comportement » désigne l'ensemble des décisions que nous pouvons prendre afin de répondre à un objectif donné.

1.2.2 Stochasticité

La stochasticité est une forme d'incertitude environnementale qui est caractérisée par le non-déterminisme : pour une situation et une décision données, ce qui suit cette décision dans cette situation peut varier. Ce non-déterminisme peut provenir du fait que les retours de l'environnement suite à une décision sont bruités (Stone and Bull, 2005) ou qu'une même décision conduit à différentes situations (Lanzi and Colombetti, 1999). Si un agent est doté de capteurs sensoriels lui permettant d'interagir avec son environnement pour réaliser son apprentissage, il est également sujet à d'autres formes de non-déterminisme qui découlent de ces capteurs. Par exemple, si un agent ne peut différencier les situations environnementales dans lesquelles il se trouve, des situations distinctes seront perçues de manière identique et cela peut prévenir la réalisation de la tâche d'apprentissage : il s'agit du problème d'aliasing perceptif (*Perceptual Aliasing Issue* - PAI) (Kaelbling et al., 1996). Les capteurs sensoriels d'un agent peuvent également être bruités (Butz et al., 2000). Dès lors, différentes formes de non-déterminisme sont possibles et peuvent provenir tant de l'environnement que de la capacité d'un agent à interagir avec son environnement.

(Faraut, 2015) montre que la stochasticité peut être gérée selon deux approches comportementales qui s'appuient sur des annotations ou sur l'expérience. La principale différence entre ces deux approches dépend de la gestion des différentes formes de non-déterminisme : soit un agent doit apprendre de lui-même à les gérer, auquel cas la stochasticité s'accumule avec l'incertitude relative avec l'ambiguïté (approches basées sur l'expérience telles que l'essai-erreur), soit toutes les informations relatives à ces formes sont fournies à un agent (approches basées sur les annotations). L'autonomie présuppose que les agents doivent avoir les capacités de gérer la stochasticité en s'appuyant sur leur expérience. Ils doivent ainsi être en mesure d'identifier et d'évaluer par eux-mêmes les différentes formes de non-déterminisme auxquelles ils sont confrontés lors de leur apprentissage. L'identification de ces formes de non-déterminisme par un agent peut consister en de nouveaux éléments d'explications lors d'une prise de décision, si elle est accessible, en plus de celles avancées lors de l'analyse de l'ambiguïté.

1.2.3 Volatilité

Enfin, la volatilité désigne le caractère changeant d'un environnement : de nouvelles situations jamais perçues peuvent survenir ou encore, les tâches d'apprentissage peuvent être modifiées. À nouveau, la volatilité d'un environnement peut être ou ne pas être signalée à un agent en cours d'apprentissage. Quand les changements dus au caractère volatil d'un environnement ne sont pas signalés, un algorithme d'apprentissage automatique autonome doit alors être capable de les détecter, en plus d'être suffisamment flexible pour adapter son apprentissage à ces changements (Faraut, 2015). Par exemple, la volatilité d'un environnement peut nécessiter à un agent de recommencer un apprentissage. Pour autant, les règles déjà apprises par cet agent peuvent être encore utiles à ce nouvel apprentissage : les règles déjà acquises ne doivent donc pas être perdues et idéalement, doivent être réutilisables. Une approche permettant d'expliquer les changements de comportement dus à la volatilité d'un environnement est d'associer un ensemble de règles à une tâche (Collins and Koechlin, 2012) : un changement de comportement est alors associé à l'usage d'un nouvel ensemble de règles.

1.3 Des algorithmes à expliquer ou intrinsèquement explicables

Cette thèse recherche alors *a minima* un algorithme d'apprentissage automatique par renforcement capable de gérer les différentes formes de non-déterminisme et suffisamment flexible pour s'adapter aux changements de son environnement.

L'apprentissage profond par renforcement fournit des pistes prometteuses de conception d'un tel algorithme, comme illustré par *AlphaStar* où les réseaux de neurones profonds ont été couplés avec, entre autres, de l'apprentissage par renforcement (Vinyals et al., 2019). Cependant, ces modèles sont considérés comme des *boîtes noires* car leur comportement est particulièrement difficile à appréhender (Rudin, 2019). Le comportement appris par AlphaStar a ainsi pu surprendre les personnes expertes du jeu StarCraft, tant les stra-

tégies utilisées étaient aussi complexes que nouvelles (The AlphaStar team, 2019) : comment expliquer les règles apprises par un tel modèle, quand ces règles consistent uniquement en une combinaison non linéaire de millions de paramètres par une composée de fonctions mathématiques ?

Deux approches ont été développées pour expliquer les décisions prises par les algorithmes d'apprentissage automatique. Elles se distinguent selon que (Du et al., 2019) :

- ces algorithmes soient intrinsèquement explicables, c'est-à-dire qu'ils sont capables de fournir par eux-mêmes, et de manière intelligible, les éléments explicatifs sur lesquels ils s'appuient pour prendre leurs décisions ;
- ou au contraire, il soit nécessaire de mettre en place des outils à même de fournir de tels éléments explicatifs à propos des décisions prises par un algorithme après son apprentissage (il s'agit alors d'explicabilité *post-hoc*).

Notre position est alors de nous appuyer sur des algorithmes intrinsèquement explicables, puisqu'ils permettent de fournir des explications du comportement de l'algorithme fidèles à ce dernier (Rudin, 2019). Ce point est d'autant plus important qu'il est difficilement concevable que des modèles de type *boîte noire* puissent être parfaitement expliqués avec des outils mis en place à cet effet en dehors de leur apprentissage. Si de tels outils existaient, ces modèles de type *boîte noire* ne seraient plus nécessaires, puisque ces outils supplanteraient leur usage, auquel cas nous disposerions de modèles intrinsèquement explicables. Autrement, si les éléments explicatifs fournis par ces outils peuvent être erronés, l'explicabilité de ces modèles est alors remise en question, d'autant plus si nous ne pouvons pas savoir si les éléments explicatifs fournis sont cohérents ou non avec le comportement effectif des modèles de type *boîte noire*.

1.4 Algorithmes intrinsèquement explicables

Les principaux algorithmes d'apprentissage automatique intrinsèquement explicables sont d'abord présentés dans les sections suivantes, avant de jus-

tifier le choix des approches à base de règles dans la section 1.4.5 pour concevoir une intelligence artificielle autonome et explicable pour des environnements incertains.

1.4.1 Méthodes des *k* plus proches voisins

Les méthodes des k plus proches voisins s'appuient sur les notions de distance et de similarité entre un nombre prédéfini k d'exemples issus des données d'apprentissage (Cover and Hart, 1967). Quand k vaut 1, la décision associée à la donnée courante dépend uniquement de la décision du voisin le plus proche ; sinon, la décision associée à la donnée courante dépend d'un vote réalisé à partir des k données d'apprentissage les plus proches. Les méthodes des k plus proches voisins ont été utilisées pour des tâches de régression, des tâches de classification et également, pour des problèmes d'apprentissage par renforcement (Martin H et al., 2009).

1.4.2 Arbres de décision et approches à base de règles

Nous incluons les arbres de décision dans les approches à base de règles, car ces deux méthodes s'emploient à explicitement générer des règles caractérisant le comportement d'un agent dans son environnement. Ces règles peuvent prendre la forme de simples règles conditionnelles « Si-Alors » ou être constituées de combinaisons plus complexes pour former les connaissances apprises par ces méthodes. De plus, les problématiques liées à l'explicabilité rencontrées par celles-ci sont similaires : il s'agit de pouvoir contrôler le nombre et la longueur des règles, car des règles trop nombreuses ou trop longues vont être plus difficilement compréhensibles.

1.4.3 Régression linéaire, logistique et modèles additifs généralisés

La régression linéaire est un modèle statistique cherchant à mettre en relation un ensemble de variables et une dernière variable d'intérêt, en présupposant l'existence d'une combinaison linéaire entre celles-ci (Arrieta et al.,

2020). Quand la variable étudiée est caractérisée par des catégories, on utilise des régressions logistiques qui emploient à cet effet des fonctions logistiques telles que des sigmoïdes pour faire cette relation. Autrement, quand la variable étudiée est continue, il s'agit de régressions linéaires utilisant des fonctions affines.

Les régressions linéaires et les régressions logistiques ont été généralisées au sein d'un même cadre : des modèles linéaires généralisés (Nelder and Wedderburn, 1972). Ils permettent de considérer de nouvelles hypothèses sur les distributions d'une variable étudiée et de lier ses observations à un ensemble de variables par le biais d'une fonction lien : les observations de l'ensemble de variables sont ainsi liées à cette fonction, en lieu et place des observations de la variable étudiée.

Les modèles additifs généralisés vont plus loin en généralisant les modèles linéaires généralisés avec un modèle additif (Hastie and Tibshirani, 2017). Ces modèles permettent de représenter des liens non linéaires dans un ensemble de variables. Ils remplacent l'approximation linéaire faite à partir de l'ensemble des variables par une somme de fonctions lisses [2] associées à chacune de ces variables.

1.4.4 Modèles bayésiens

Les modèles bayésiens s'appuient sur l'inférence bayésienne, une méthode statistique où les probabilités de diverses causes pour un événement sont calculées à partir de l'observation de cet événement (Charniak, 1991).

Les modèles bayésiens permettent en particulier de fournir des outils pour gérer l'incertitude des environnements (Ghavamzadeh et al., 2015), en particulier le caractère volatil des environnements où de nouvelles situations peuvent survenir. Ces outils sont entre autres capables de capturer l'état des connaissances d'un agent pour une situation donnée. Ils peuvent alors orienter la prise de décision d'un agent selon qu'il lui serait plus profitable d'acquérir de nouvelles connaissances, ou d'exploiter les connaissances déjà acquises afin de résoudre sa tâche d'apprentissage.

2. Les fonctions lisses sont des fonctions dérivables et continues à l'infini.

1.4.5 Vers des approches à base de règles

Notre choix s'est dirigé vers des algorithmes d'apprentissage automatiques à base de règles. Utiliser une définition de l'autonomie comme la capacité à se doter de ses propres règles prend d'autant plus son sens si l'explicabilité est aussi mise en œuvre par une exposition des règles aboutissant à des décisions. Explicabilité et autonomie utilisent alors une seule et même représentation prenant la forme des règles issues de ces approches.

L'explicabilité de ces modèles est également renforcée par le fait que ces modèles trouvent leurs inspirations dans le comportement des êtres humains. Leur explicabilité est ainsi reconnue dans de nombreux domaines où ils ont été employés, que ce soit pour directement résoudre une tâche d'apprentissage, ou pour expliquer le comportement d'autres modèles d'apprentissage automatique (Arrieta et al., 2020).

De plus, nous indiquions que l'explicabilité passe par un transfert d'information entre un algorithme d'apprentissage automatique et un public. Pour que ce transfert soit réalisé de manière satisfaisante, une explication doit s'adapter aux besoins et aux connaissances des destinataires (Burkart and Huber, 2021). La proximité des approches à base de règles avec le comportement des êtres humains facilite alors ce transfert d'information par rapport aux autres méthodes mentionnées (Arrieta et al., 2020).

En particulier, (Miller, 2019) montre que de « bonnes » explications doivent en effet dépasser le cadre des statistiques, sans pour autant les mettre de côté, car elles apportent tout de même une information (de l'ordre d'un lien associatif entre deux éléments). Il montre que les *causes* sont plus pertinentes que les statistiques, quand il s'agit d'expliquer pourquoi un événement se produit ou mieux, pourquoi un événement a eu lieu à la place d'un autre. Les approches à base de règles sont capables de mettre en avant ces *causes via* les structures conditionnelles des règles et aussi, *via* des comparaisons entre les différentes règles.

Toutes ces autres approches ne sont pas à écarter pour autant : elles peuvent se révéler être complémentaires et pallier mutuellement leurs lacunes. En ce sens, le cadre des algorithmes d'apprentissage automatique à base de règles

rend possible l'emploi de plusieurs approches intrinsèquement explicables au sein d'un même système. Par exemple, les représentations utilisées dans les structures conditionnelles peuvent s'appuyer sur des arbres de décision (Bernadó et al., 2001) et l'évaluation des règles peut s'appuyer sur une approche bayésienne (Aliprandi et al., 2006), sur une méthode des k plus proches voisins (Ahluwalia et al., 1999), ou sur des techniques de régression linéaire (Wilson, 1999).

Notre position est alors de nous appuyer sur des algorithmes d'apprentissage automatique par renforcement à base de règles : ces approches doivent ainsi être capables de gérer les différentes sources d'incertitude des environnements *réels* et leurs règles, de s'adapter et de co-évoluer avec l' environnement.
Le chapitre suivant discute plus en détail les approches existantes à base de règles, pour concevoir à une intelligence artificielle autonome et explicable pour des environnements incertains.

CHAPITRE 2

Les systèmes de classeurs

Ce chapitre motive l'emploi des systèmes de classeurs en comparant les principales approches à base de règles qui existent, à partir des systèmes experts. Les systèmes de classeurs sont ensuite présentés, afin de justifier le choix d'une famille particulière de systèmes de classeurs : les systèmes de classeurs à anticipation.

Sommaire

2.1 Systèmes à base de règles

Les systèmes à base de règles sont des systèmes qui sont capables de manipuler des connaissances sous la forme de règles conditionnelles, traditionnellement de la forme « Si-Alors ». Ils regroupent un très large éventail de modèles issus d'approches et d'inspirations très différentes les unes des autres. Les premiers systèmes à base de règles ayant été historiquement développés sont les systèmes experts Feigenbaum (1981).

2.1.1 Systèmes experts

Les systèmes experts tentent de résoudre des problèmes de manière analogue au raisonnement humain à partir de connaissances d'experts (Jackson,

1986). Pour ce faire, les systèmes experts sont principalement composés des éléments suivants (DeTore, 1989) :

- une base de faits qui intègre des hypothèses de départ pour un problème à résoudre et les éléments déduits par le système.
- une base de règles qui contient l'ensemble des connaissances des experts sous la forme de règles mettant en relation des faits associés au problème à résoudre ;
- un moteur d'inférence qui met en œuvre des raisonnements déductifs logiques pour produire de nouveaux faits à partir de la base de connaissance, jusqu'à parvenir à une solution pour le problème à résoudre. Par exemple, l'une des règles d'inférence utilisée dans ces moteurs est basée sur le syllogisme : si le fait $F1$ est vrai et si une règle stipule que le fait $F1$ implique le fait $F2$, alors le fait $F2$ est vrai ;
- une interface utilisateur qui permet à un utilisateur de fournir les hypothèses de départ et un problème à résoudre, et aux experts de manipuler les règles de la base de connaissance. Elle peut aussi être utilisée pour demander à l'utilisateur de fournir de nouveaux faits si le système ne parvient pas à une conclusion.

Les systèmes experts ont été utilisés dans des domaines comme le diagnostic médical ou encore la conception d'objets sous contraintes (Hayes-Roth et al., 1983). Deux exemples notables de systèmes experts sont *DENDRAL* et *MYCIN*. *DENDRAL* est le premier système expert appliqué à un problème de raisonnement scientifique qui consistait à identifier les constituants chimiques d'un matériau à partir de spectrométrie de masse et de résonance magnétique nucléaire (Lindsay et al., 1993). Le système expert *MYCIN* a été développé en 1973 afin de diagnostiquer les maladies du sang et de prescrire les médicaments adéquats (Buchanan and Shortliffe, 1984).

Cependant, les systèmes experts ont rapidement connu des limitations relatives à la gestion des connaissances (Coats, 1988). Bien que des outils participant à l'acquisition des connaissances aient été développés, la base de règles doit être maintenue manuellement : les systèmes experts ne sont donc pas autonomes. L'ajout de règles n'entrant pas en contradiction avec celles existantes est d'autant plus difficile que la base est grande et que le problème défini est complexe. Quand la taille de la base des connaissances et le nombre

de règles impliquées dans les raisonnements augmentent, la compréhensibilité des raisonnements effectués par ces systèmes devient alors de plus en plus délicate.

Pour permettre aux systèmes experts de gérer l'incertitude issue des faits ou des connaissances, des moteurs d'inférence fondés sur la logique floue ont été développés (Tavana and Hajipour, 2019). La logique floue permet de modéliser des faits ayant des valeurs de vérité réelles (comprises entre 0 et 1), au lieu de valeurs booléennes, et de décrire les règles d'inférence associées à de tels faits (Zadeh, 1988).

Cependant, l'introduction de ces nouveaux moteurs d'inférence ne permet pas aux systèmes experts de pouvoir s'adapter et co-évoluer avec leur environnement. Les bases de règles exploitées par ces systèmes restent élaborées par des experts et fixes, dans le sens où même si ces bases peuvent être complétées par inférence des règles préétablies, elles ne créent pas de règles à partir de nouveaux faits.

De nouveaux systèmes à base de règles s'appuyant sur l'apprentissage automatique ont alors été conçus afin de les rendre plus autonomes : ils deviennent capables de se doter de leurs propres règles à partir d'interactions avec des données d'apprentissage, et de faire évoluer ces règles.

2.1.2 Apprentissage automatique à base de règles

L'apprentissage automatique à base de règles désigne l'ensemble des systèmes cherchant explicitement à identifier des ensembles de règles permettant la résolution d'une tâche d'apprentissage. Ces systèmes peuvent être séparés en plusieurs catégories différenciables par leurs buts ou par leurs approches conceptuelles. Les principales catégories sont associées aux arbres de décision, à l'apprentissage de règles d'association ou encore aux systèmes de classeurs (Learning Classifier System (LCS)).

Arbres de décision

Les arbres de décision sont des algorithmes d'apprentissage très populaires

dans le domaine de la fouille de données (Wu et al., 2008). Ils sont également utilisés afin de produire des politiques de décision explicables dans le domaine de l'apprentissage par renforcement (Bastani et al., 2018).

Ils réalisent un partitionnement des données d'apprentissage sous une structure d'arbre (Rakotomalala, 2005) : ce partitionnement est réalisé en déterminant quelle variable permet de discriminer le plus efficacement les données d'apprentissage, quand celles-ci sont décrites par un ensemble de variables. La variable la plus discriminante est calculée pour chaque ensemble de données relatif à un nœud de l'arbre, afin de séparer l'ensemble en plusieurs sous-groupes. Ce calcul est alors répété récursivement pour chacun des sous-groupes, et prend fin lorsqu'un critère d'arrêt de cette récursion est rempli (par exemple, l'arbre a atteint une certaine profondeur). Les règles apprises par un arbre de décision sont alors accessibles en parcourant l'arbre, tel qu'illustré par la figure 2.1.

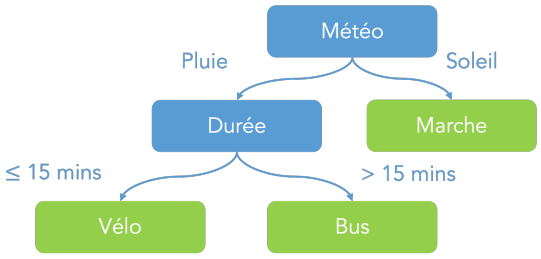


FIGURE 2.1 – Exemple d'un arbre de décision indiquant le moyen de transport choisi selon la météo et la durée du trajet.
Par exemple, une règle donnée par cet arbre de décision est : si la météo est pluvieuse et que la durée du trajet est de moins d'un quart d'heure, alors le moyen de transport préféré est le vélo.

La construction des règles par les arbres de décision a ensuite été complexifiée avec par exemple l'introduction de transitions plus complexes entre les nœuds des arbres (Oliver et al., 1992). Des méthodes ensemblistes permettant de faire apprendre conjointement plusieurs arbres de décision ont été aussi développées pour que ces modèles puissent résoudre des tâches plus complexes, au prix de leur explicabilité (Breiman, 2001).

Apprentissage des règles d'association

L'apprentissage des règles d'association désigne une approche d'extraction de connaissances permettant d'identifier les principales associations et cor-

rélations entre des données issues de grands ensembles (Subasi, 2020). Les données sont alors représentées par un ensemble de règles d'association faisant apparaître les attributs des données qui sont fréquemment associés entre eux selon une mesure d'intérêt des règles (Capozzoli et al., 2016).

Pour déterminer quelles sont les règles d'association d'intérêt d'un ensemble de données, cette approche s'appuie par exemple sur le support et la confiance des règles. Le support permet de mesurer si les règles calculées s'appliquent à un nombre élevé de données. La confiance permet de déterminer si les règles sont effectivement vérifiées sur les données auxquelles elles sont applicables (Witten et al., 2017). D'autres mesures d'intérêt des règles sont par ailleurs possibles et détaillées dans (Hahsler, 2015).

Systèmes de classeurs

Les systèmes de classeurs (LCS) ont été conçus à partir des difficultés qu'un système peut avoir à réaliser une tâche d'apprentissage dans des environnements complexes et changeants continuellement (Holland et al., 1999), à l'image des environnements du monde réel. Les LCS ont alors été proposés comme des modèles de processus inductif trouvant leur inspiration dans les systèmes cognitifs de l'homme (Holland et al., 1989). Ils intègrent dans un même cadre des idées issues de l'intelligence artificielle, de l'apprentissage automatique, de la psychologie cognitive, de l'économie, de l'évolution et de la conception informatique, dans le but de proposer un cadre permettant de réaliser cet apprentissage (Holland et al., 1999).

Les LCS sont des algorithmes d'apprentissage automatique à base de règles qui combinent des mécaniques de découverte de nouvelles règles et des mécaniques de rétribution de ces règles pour construire une population de règles adaptée à leurs données d'apprentissage et à la tâche d'apprentissage (Sigaud and Wilson, 2007). Les règles des LCS réalisent des approximations locales du problème auquel ils sont confrontés, et font évoluer l'ensemble des règles de manière à former des approximations précises et efficacement réparties sur le problème à résoudre (Butz, 2015).

Les LCS peuvent alors décrire de manière flexible et efficace des problèmes complexes et variés tels que la modélisation du comportement, l'approxima-

tion de fonctions, la classification, la prédiction et l'extraction de connaissance (Urbanowicz and Moore, 2009).

2.1.3 Le choix des systèmes de classeurs

La direction prise dans cette thèse est de s'appuyer sur des modèles d'apprentissage automatique par renforcement à base de règles, afin de concevoir un algorithme d'apprentissage automatique autonome et explicable pour des environnements incertains. Nous proposons alors de nous appuyer sur les systèmes de classeurs.

Les systèmes de classeurs par rapport aux arbres de décision

Les systèmes de classeurs ont été préférés aux arbres de décision puisque (Charbuty and Abdulazeez, 2021) montrent que les arbres de décision ont pour principales limites la robustesse, la hiérarchie des règles imposée par la structure en arbre et la gestion de la hauteur des arbres. Ces limites impactent directement la capacité des arbres de décision à gérer l'incertitude des environnements et à fournir des éléments explicatifs.

Les LCS sont par conception robustes aux changements environnementaux : ils sont ainsi plus adaptés à gérer la volatilité des données d'apprentissage (Butz, 2001). Un changement des données d'apprentissage peut nécessiter de recalculer l'ensemble de l'arbre de décision, alors que les LCS peuvent s'appuyer directement sur leur propre mécanique pour créer, si besoin est, de nouveaux classeurs adaptés à ces données.

Enfin, l'explicabilité des arbres de décision et leur capacité d'apprentissage dépendent conjointement de la structure en arbre. Des arbres profonds peuvent améliorer leurs capacités (avec un risque de surapprentissage (Rakotomalala, 2005)), mais la profondeur complexifie dans le même temps les règles apprises. Des mécanismes d'élagage permettent de réduire la profondeur des arbres (Mingers, 1989) tout en limitant l'impact sur la résolution de la tâche d'apprentissage. Dans une certaine mesure, les mécaniques de découverte de règles et l'absence de structure entre les règles permet aux LCS de ne pas faire de tel compromis : les LCS sont capables de trouver les règles les plus

générales couvrant chacune une partie du problème à résoudre. Néanmoins, il peut être nécessaire de faire attention à ce que les LCS ne construisent pas un nombre trop élevé de règles, sous peine que leur explicabilité soit impactée (Tan et al., 2013).

Les systèmes de classeurs par rapport à l'apprentissage des règles d'association

Les systèmes de classeurs ont ensuite été préférés à l'apprentissage des règles d'association pour leur polyvalence ainsi que leur autonomie, puisqu'ils ont la capacité à s'adapter aux données d'apprentissage ou à leur environnement.

L'apprentissage des règles d'association est destiné aux problèmes d'extraction de connaissance, quand les LCS peuvent être employés pour ces mêmes problèmes et aussi pour des problèmes d'apprentissage par renforcement, de régression, ou de classification, à titre d'exemples. Or, nous avons justifié, dans la section 1.2, notre choix d'un algorithme d'apprentissage par renforcement afin qu'un modèle d'apprentissage automatique puisse gérer l'ambiguïté relative aux données d'apprentissage.

Par ailleurs, l'apprentissage des règles d'association vise à construire des règles représentant les principales caractéristiques des données d'apprentissage : ces règles ne font donc pas apparaître des corrélations pouvant être faiblement représentées dans les données, d'autant plus si ces corrélations sont sujettes à de l'incertitude. Au contraire de cette approche, les LCS sont capables de modéliser ces corrélations faiblement représentées ou sujettes à l'incertitude (Urbanowicz and Moore, 2015). En outre, un changement des données d'apprentissage peut nécessiter de recalculer l'ensemble des règles d'associations, quand les LCS peuvent à nouveau s'appuyer sur leur mécanique de découverte de nouvelles règles pour adapter leurs populations de classeurs à ces nouvelles données. L'apprentissage des règles d'association n'est donc pas à même de gérer la volatilité des environnements incertains.

La section 2.2 présente alors plus en détail les principes et les différents constituants des LCS.

2.2 Principes et composants des systèmes de classeurs

2.2.1 Structure générale d'un classeur

Chaque classeur est d'abord composé d'une ou plusieurs règles. Une règle est composée *a minima* d'une structure conditionnelle *Si-Alors* avec une *Condition* et un *Conséquent*, de paramètres indiquant son adéquation aux situations décrites par la structure conditionnelle et enfin, de paramètres internes permettant de faire évoluer le classeur au sein de la population et en accord avec les données en entrée du système (Holland and Reitman, 1978). La structure conditionnelle des classeurs peut également être étendue : une mémoire des entrées perçues par le système peut être ajoutée aux règles (Lanzi, 1998), les règles peuvent être chaînées entre elles (Tomlinson and Bull, 2001), ou les effets du *Conséquent* peuvent être décrits (Stolzmann, 1999).

Les systèmes de classeurs font évoluer leurs classeurs selon deux principales approches, illustrées par la figure 2.2 : l'approche *Michigan* ou l'approche *Pittsburgh* (De Jong et al., 1993).

L'approche *Michigan* fait correspondre à chaque classeur une unique règle. L'ensemble de la population de classeurs désigne alors la solution apprise par le système pour un problème donné. Chaque classeur propose une solution locale à la tâche à apprendre par le système et les classeurs sont mis en compétition entre eux afin de recouvrir le plus efficacement possible l'espace de recherche du problème à résoudre. Ainsi, un système de classeurs sous l'approche *Michigan* recherche une combinaison adéquate de solutions locales pour résoudre un problème dans son ensemble.

L'approche *Pittsburgh* permet de représenter dans un unique classeur un ensemble de règles désignant une potentielle solution à un problème donné. L'ensemble de la population de classeurs désigne alors différentes solutions apprises par le système pour un problème donné : celles-ci sont mises en compétition afin de répondre le plus efficacement possible au problème donné.

Cependant, les systèmes de classeurs de type *Pittsburgh* nécessitent des res-

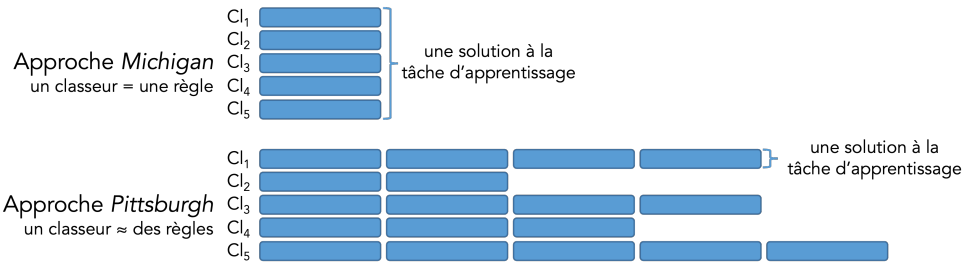


FIGURE 2.2 – Représentation de l'approche *Michigan* et de l'approche *Pittsburgh* dans les systèmes de classeurs.
La population de classeurs sous l'approche Michigan *représente la solution apprise, quand la population sous l'approche* Pittsburgh *représente plusieurs solutions.*

sources calculatoires importantes et souffrent d'une grande complexité algorithmique, sans pour autant surpasser les performances de ceux de type *Michigan* (Bishop et al., 2022) : évaluer l'adéquation d'un seul classeur peut impliquer l'utilisation d'un ensemble de données d'apprentissage volumineux à concevoir en amont d'un apprentissage. Les systèmes de classeurs de type *Pittsburgh* sont peu adaptés au caractère changeant des environnements incertains, puisqu'une modification des données d'apprentissage peut nécessiter de relancer *a minima* l'évaluation de tous les classeurs d'une population et au pire, de complètement relancer leur apprentissage. Nous nous limiterons alors aux systèmes de classeurs de type *Michigan*, ceux-ci pouvant prendre en compte la volatilité des environnements (Urbanowicz and Moore, 2009). Par la suite, nous désignerons de manière interchangeable les classeurs et les règles puisque nous ne traiterons que de systèmes de classeurs de type *Michigan*.

2.2.2 Interfacer le système de classeurs avec les données d'apprentissage

Afin que ces systèmes puissent construire leurs classeurs, il est nécessaire de préciser que les données d'apprentissage sont représentées dans la structure conditionnelle des classeurs : cela revient à spécifier l'encodage qui permet à ces systèmes de décrire les données dans leur structure conditionnelle *Si-Alors*. Ces données peuvent être décrites par un ensemble d'attributs discrets dont le nombre de valeurs possibles est limité, ou continus quand toutes les valeurs sont possibles pour un intervalle donné.

La structure conditionnelle d'un classeur est descriptible entre autres grâce à des conjonctions logiques des attributs des données (Wilson, 1995), de la logique du premier ordre (Mellor, 2005), de la logique floue (Casillas et al., 2007), des graphes (Tufts, 1995 ; Landau et al., 2005 ; Dam et al., 2007), des fonctions (Lanzi and Loiacono, 2007) ou des combinaisons de ces descriptions (Bull and O'Hara, 2002).

Les systèmes de classeurs réalisent leur apprentissage grâce à leurs interactions avec les données d'apprentissage : ces systèmes s'appuient alors sur ces représentations, tout autant qu'ils sont contraints par celles-ci (Lanzi and Riolo, 2003). L'explicabilité des LCS est ainsi directement liée aux représentations manipulées : des règles dont la *Condition* est décrite par des réseaux de neurones sont plus difficilement explicables par rapport à des règles dont la *Condition* est décrite par des conjonctions logiques.

Dans la suite, nous nous concentrerons plus particulièrement sur les représentations les plus couramment employées dans les systèmes de classeurs (Urbanowicz and Moore, 2009 ; Pätzel et al., 2021). La *Condition* d'un classeur est décrite par la conjonction logique de ses attributs, quand le *Conséquent* est composé d'un ou plusieurs attributs relatifs à la tâche à apprendre telle que des actions ou des classes. Il reste à définir comme chacun de ces attributs peut alors être décrit selon qu'il est discret ou continu.

Représentation discrète d'attributs

Dans le cadre d'un attribut discret, les principales représentations utilisées sont la représentation binaire (voire ternaire), la représentation symbolique ou la représentation issue du système *Genetic Algorithm Batch-Incremental concept Learner (GABIL)* (DeJong and Spears, 1990).

La représentation binaire décrit chaque attribut par 0 ou 1, quand la représentation symbolique décrit chaque valeur possible d'un attribut par un unique symbole. Un symbole générique, souvent dénoté par $\#$, peut aussi être manipulé dans les représentations binaires (devenant ainsi ternaires) et symboliques afin d'indiquer que n'importe quel symbole peut être associé à l'attribut représenté par $\#$ (*i.e.* l'attribut a donc été généralisé).
Par exemple, les jours de la semaine peuvent être décrits symboliquement

sous la forme d'un attribut dont la valeur appartiendrait à l'ensemble {*Lu*, *Ma*, *Me*, *Je*, *Ve*, *Sa*, *Di*}, quand la représentation binaire permettrait de décrire si les températures sont positives (bit à 1) ou négatives (bit à 0). Le symbole générique # correspondrait à tous les jours de la semaine et indiquerait que les températures peuvent être positives ou négatives.

Enfin, *GABIL* décrit chaque attribut sous la forme d'une disjonction de l'ensemble des valeurs possibles de l'attribut. Cette disjonction est codée par une chaîne de bits de longueur fixe où chaque bit correspond à une valeur possible de l'attribut. *GABIL* s'appuie ainsi sur une représentation binaire pour représenter des éléments plus complexes.
Par exemple, les jours de la semaine sont représentés avec GABIL par une chaîne de 7 bits et il est possible de modéliser le week-end (*i.e.* samedi ou dimanche) avec la chaîne 0000011, en considérant que la représentation démarre avec un lundi.

Représentation continue d'attributs

Dans le cadre d'un attribut continu, il est d'abord nécessaire de savoir si l'espace associé à cet attribut est infini. Dans le cas où celui-ci est infini, une troncature peut avoir lieu afin de limiter l'exploration du système à des plages de valeurs définies par la troncature. En outre, un encodage des attributs peut aussi avoir lieu, indépendamment du caractère infini de leur espace, afin de faciliter la manipulation des représentations continues de ces attributs.

Dès lors, il est possible d'opérer une discrétisation afin de se ramener au cas d'un attribut discret (Kozlowski and Unold, 2019), ou de représenter des ensembles de valeurs continues sous la forme d'intervalles (Wilson, 1999, 2000 ; Stone and Bull, 2003 ; Dam et al., 2005), d'enveloppes convexes (Lanzi and Wilson, 2006) ou d'ellipsoïdes (Butz, 2005). La figure 2.3 donne ainsi quelques exemples de représentations continues pour un attribut appartenant à l'ensemble des réels.

Le choix de la troncature et des représentations à employer pour des attributs continus n'est pas sans conséquence et reste un problème ouvert (Stone and Bull, 2003) : celles-ci vont transformer l'espace de recherche associé au problème à résoudre et vont également impacter tant la manière d'explorer

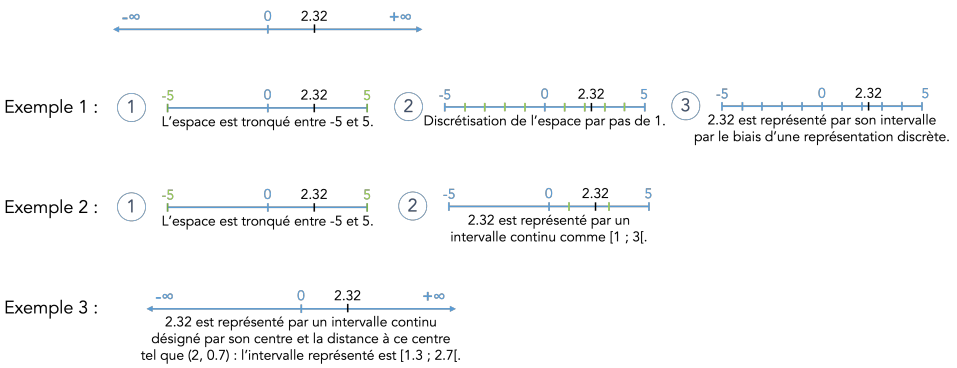


FIGURE 2.3 – Exemples de représentations continues employées par les systèmes de classeurs pour un attribut appartenant à l'ensemble des réels.
L'exemple 1 *est inspiré de (Kozlowski and Unold, 2019), l'exemple* 2 *de (Wilson, 2000) et enfin , l'exemple* 3 *de (Wilson, 1999).*

cet espace que l'évolution et la répartition des classeurs sur cet espace. Elles doivent donc être définies avec soin afin de favoriser la création de classeurs adéquats au problème, sous peine que les systèmes de classeurs ne puissent pas résoudre la tâche à apprendre.

Représentation mixte d'attributs

Selon le problème étudié, il est possible que les systèmes de classeurs doivent s'interfacer avec des données pouvant être à la fois discrètes et continues. Une combinaison des représentations présentées précédemment est alors possible, comme le propose l'*Attribute List Knowledge Representation* (Bacardit and Krasnogor, 2009) ou la *Discrete-Continuous Attribute List Knowledge Representation* (Urbanowicz and Moore, 2015).

L'*Attribute List Knowledge Representation*, tout comme la *Discrete-Continuous Attribute List Knowledge Representation*, proposent de stocker dans la *Condition* d'un classeur les attributs qui auront été spécifiés par le système, autrement dit ceux qui sont d'importance dans l'usage de la règle. Ces représentations permettent de faciliter la lecture d'une règle (comme illustrée par la figure 2.4) ainsi que de réduire le temps d'exécution des systèmes de classeurs : pour savoir si les classeurs correspondent aux données d'entrée, il est plus rapide de considérer uniquement les attributs spécifiés au lieu de prendre en compte tous les attributs.

Dans le cas d'un attribut discret, l'*Attribute List Knowledge Representation* utilise la représentation issue de *GABIL*, quand *Discrete-Continuous Attribute List Knowledge Representation* utilise une représentation symbolique. Dans le cas d'un attribut continu, toutes deux emploient l'*Ordered Bounded Representation* de (Wilson, 2000), illustrée par l'exemple 2 de la figure 2.3. L'*Ordered Bounded Representation* décrit directement des intervalles de valeurs réelles par les bornes de ces intervalles, de façon ordonnée.

Représentation symbolique de la météo pour tous les jours de l'année

	[Pluie,	Soleil,	#,	#,	…,	Neige,	#,	#	#]
Jour	*1er janvier*	*2 janvier*	*3 janvier*	*4 janvier*	…	*28 décembre*	*29 décembre*	*30 décembre*	*31 décembre*

Discrete-Continuous Attribute List Knowledge Representation équivalente

	[Pluie,	Soleil,	Neige]
Référence à l'attribut	[1,	2,	362]
Jour	*1er janvier,*	*2 janvier,*	*28 décembre*

FIGURE 2.4 – Exemple de représentation symbolique de la météo pour tous les jours de l'année et son équivalent avec *Discrete-Continuous Attribute List Knowledge Representation.*
Dans cet exemple, la représentation de la météo n'est précisée que pour trois jours de l'année, puisque des symboles génériques sont utilisés pour les 362 jours restants. Discrete-Continuous Attribute List Knowledge Representation permet une représentation plus condensée et lisible de cette information, en stockant la position des attributs spécifiés.

Le choix d'une représentation des attributs dépend des données d'apprentissage et aussi des capacités des systèmes de classeurs : toutes ces représentations peuvent ne pas être directement utilisables par des systèmes de classeurs, nécessitant *de facto* des modifications. Bien que les représentations mixtes offrent plus de souplesse, ce dernier point est particulièrement vrai pour celles-ci, parce qu'elles impliquent qu'un système de classeurs puisse faire évoluer des représentations discrètes et continues en même temps.

2.2.3 De l'ensemble de la population de classeurs au sous-ensemble d'apprentissage

Lors de leur apprentissage, les systèmes de classeurs font évoluer des classeurs qui correspondent à la donnée d'apprentissage courante. Ils constituent alors, à partir de leur population de classeurs, notée $[P]$, un sous-ensemble

de classeurs dont la *Condition* correspond à la donnée d'apprentissage, noté $[M]$, comme illustré par la figure 2.5. Ce passage de $[P]$ à $[M]$ désigne **le processus d'appariement** : ce processus est réalisable grâce à l'interfaçage réalisé en amont entre les données d'apprentissage et leur représentation employée par les classeurs.

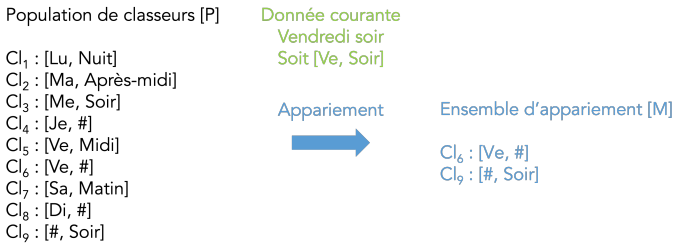


FIGURE 2.5 – Illustration du processus d'appariement permettant de passer de la population de classeurs $[P]$ au sous-ensemble de classeurs $[M]$ correspondant à la donnée d'apprentissage courante.
Dans cet exemple, une représentation symbolique est utilisée pour représenter les jours de la semaine ({Lu, Ma, Me, Je, Ve, Sa, Di}) et pour représenter le moment du jour ({Matin, Midi, Après-midi, Soir, Nuit}). L'ensemble d'appariement $[M]$ est formé de tous les classeurs correspondant au vendredi soir. Seule la condition des classeurs est illustrée. # est un symbole générique pouvant être associé à n'importe quel symbole correspondant à l'attribut.

Le sous-ensemble d'appariement $[M]$, correspondant à l'entrée courante du système, peut alors à nouveau être réduit en d'autres sous-ensembles de classeurs selon le type d'apprentissage réalisé par le système. Dans le cadre d'un apprentissage supervisé, le sous-ensemble de classeurs $[M]$ est séparé en deux sous-ensembles disjoints $[C]$ et $[I]$, selon que le *Conséquent* de ces classeurs correspond à l'étiquette de la donnée d'entrée, ou non. Dans le cadre d'un apprentissage par renforcement, les actions à réaliser afin de répondre au problème posé ne sont pas connues à l'avance. Le choix de cette action par les systèmes de classeurs conduit le système à réduire $[M]$ au sous-ensemble d'action de classeurs $[A]$: $[A]$ correspond à l'ensemble des classeurs de $[M]$ dont le *Conséquent* correspond à l'action choisie par le système.

Le choix de la politique d'action dans un système de classeurs détermine sa capacité à explorer l'espace de recherche permettant de construire de nouveaux classeurs et sa capacité à exploiter les récompenses reçues au cours de l'apprentissage. Différentes politiques d'action ont été employées dans les systèmes de classeurs telles que :

— La *roulette wheel selection*. Elle permet de choisir de manière stochastique une action selon son adéquation [1] avec l'entrée courante par rapport aux autres actions : plus l'action est adéquate, plus elle a de chance d'être choisie.
— La *sélection par tournoi*. Elle met en compétition les classeurs afin de retourner l'action a priori la plus adéquate avec l'entrée courante.
— L'*ϵ-greedy policy*. Elle propose un compromis entre exploration et exploitation, où l'action est aléatoirement choisie selon une probabilité ϵ, sinon l'action est choisie selon son adéquation avec l'entrée courante.

Les algorithmes d'apprentissage des systèmes de classeurs interviennent alors principalement sur ces ensembles de classeurs $[A]$ ou $[C]$, $[I]$, pour construire de nouveaux classeurs et pour mettre à jour les classeurs selon leur adéquation avec les données d'apprentissage.

2.2.4 Processus de recouvrement

Le **processus de recouvrement** fabrique un nouveau classeur dont la structure conditionnelle correspond à la donnée d'apprentissage courante. Ce processus permet de garantir qu'au moins un classeur de la population correspond à chaque entrée.

FIGURE 2.6 – Illustration du processus de recouvrement fabriquant de nouveaux classeurs correspondant à la donnée d'apprentissage courante.
Dans cet exemple, une représentation symbolique est utilisée pour représenter les jours de la semaine ({Lu, Ma, Me, Je, Ve, Sa, Di}) et pour représenter le moment du jour ({Matin, Midi, Après-midi, Soir, Nuit}). Seule la condition des classeurs est illustrée. # est un symbole générique pouvant être associé à n'importe quel symbole correspondant à l'attribut.

Les systèmes de classeurs créent leurs classeurs par recouvrement lors de ces occasions :

1. L'adéquation est à définir par l'utilisateur à partir des attributs des classeurs selon l'importance qui peut leur être donnée.

- lorsqu'aucun classeur ne correspond à la donnée d'apprentissage courante ($[M] = \varnothing$) ;
- lorsqu'aucun classeur correspondant à la donnée d'apprentissage courante ne prédit correctement l'étiquette de cette donnée ($[C] = \varnothing$), dans un contexte d'apprentissage supervisé ;
- lorsqu'aucun classeur correspondant à la donnée d'apprentissage courante n'est associé à une action réalisable par le système ($[A] = \varnothing$), dans un contexte d'apprentissage par renforcement.

Ce processus de recouvrement peut être guidé ou stochastique : les attributs de la *Condition* du classeur peuvent être explicitement spécifiés selon les différences entre des données d'entrée reçues successivement (Stolzmann, 1999), ou bien une probabilité déclenchant leur spécification peut être utilisée (Wilson, 1995). La figure 2.6 donne trois exemples de classeurs qui pourraient être créés lors du processus de recouvrement.

Il est possible que des classeurs supplémentaires soient créés en d'autres occasions par le biais de mécanismes propres à certains systèmes de classeurs. Par exemple, (Butz and Wilson, 2000) génèrent de nouveaux classeurs selon le nombre de *Conséquents* différents présents dans $[M]$ afin que l'usage de chaque *Conséquent* soit éprouvé par le système de classeurs pour chaque donnée d'apprentissage. (Butz and Stolzmann, 2001) construisent de nouveaux classeurs quand le système détecte que la structure conditionnelle de ces classeurs peut être spécialisée afin de renforcer leur adéquation aux données d'entrée.

2.2.5 Processus de découverte de nouvelles règles

Les processus de découverte de nouvelles règles désignent le **processus de recouvrement** qui génère de nouvelles règles à partir des données d'entrée et qui a été présenté dans la section précédente. Ils désignent également les processus qui créent de nouvelles règles à partir de celles existantes : les **algorithmes génétiques** (Genetic Algorithms (GA)) sont principalement utilisés en ce sens dans une grande majorité de systèmes de classeurs (Urbanowicz and Moore, 2009).

Les GA sont inspirés de la théorie de la sélection naturelle (Holland et al., 1992 ; Goldberg and Holland, 1988) : des individus d'une population sont manipulés par le biais de processus de sélection et d'opérations génétiques telles que le croisement ou la mutation, afin de découvrir les individus les plus adaptés à un environnement.

Les classeurs sont générés au fil de l'eau par les algorithmes génétiques, un par un, pour que les GA interagissent plus efficacement avec l'ensemble des composants des systèmes de classeurs (Sigaud and Wilson, 2007) : il s'agit alors de *Steady State GA*.

Par ailleurs, certains LCS s'appuient sur des mécaniques différentes des algorithmes génétiques afin de guider plus explicitement la découverte de règles. Ils peuvent directement comparer les données d'entrée entre elles avec les *Conditions* des classeurs pour en générer de nouveaux (Stolzmann, 1999) : les GA ont alors pour but de généraliser les classeurs existants (*i.e.* d'étendre le nombre de situations où le classeur est adapté) (Butz and Stolzmann, 2001). Des mécanismes de suivi des attributs peuvent aussi être mis en place afin d'orienter l'évolution des structures conditionnelles des classeurs (Urbanowicz et al., 2012a), donnant par exemple une indication sur les attributs nécessaires dans les *Conditions* pour que les classeurs soient adaptés aux données d'entrée.

La structure conditionnelle d'un classeur peut être modifiée par les processus de découverte de nouvelles règles. Un changement dans la structure conditionnelle d'un classeur implique une modification de l'ensemble des situations décrites par la structure conditionnelle de ce classeur. Il est alors possible que ce changement rende le classeur inadéquat aux données d'apprentissage. Si ce changement est effectué directement sur un tel classeur, la règle apprise par le système pourrait être perdue et par conséquent, ralentir l'apprentissage ou bien prévenir le système de résoudre la tâche à apprendre. Une copie du classeur est donc réalisée en amont d'un tel changement, où sa structure conditionnelle (et potentiellement ses paramètres internes) est dupliquée dans un nouveau classeur.

La figure 2.7 présente un exemple d'application des processus de découverte de nouvelles règles sur un classeur, ce dernier étant dupliqué et sa structure conditionnelle modifiée. Il sera ensuite mis à jour par le système de classeurs

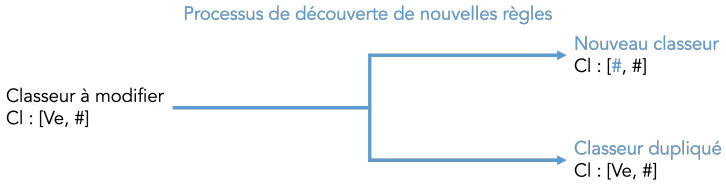


FIGURE 2.7 – Modification de la structure conditionnelle d'un classeur suite au processus de découverte de nouvelles règles.
 # est un symbole générique pouvant être associé à n'importe quel symbole correspondant à l'attribut.

afin d'évaluer son adaptation à la tâche d'apprentissage.

2.2.6 Mise à jour des classeurs et rétribution

La mise à jour des paramètres internes des classeurs intervient après la création, si nécessaire, de nouveaux classeurs. Elle est décomposable en deux parties où :

- Les valeurs d'adéquation (la *fitness*) des classeurs à la tâche à apprendre sont actualisées par un **processus de rétribution** (Buche et al., 2006). Il s'agit alors de renforcer les classeurs les plus adéquats à une tâche. Cette étape est aussi appelée *credit assignment* ou *reinforcement component* (Urbanowicz and Moore, 2009).
- D'autres paramètres internes tels que les marques temporelles des classeurs sont actualisées. Ces autres paramètres sont associés à des mécanismes propres à certains LCS tels que des mécanismes de découverte de nouvelles règles ou de gestion de la population de classeurs.

La mise à jour de la *fitness* des classeurs est une étape importante puisque cette valeur peut être réutilisée pour guider la création de classeurs de plus en plus adéquats à la tâche, pour guider la sélection d'un *Conséquent* ou pour supprimer des classeurs de la population.

Les processus de rétribution peuvent être séparés en deux grandes catégories selon que la *fitness* des classeurs est basée sur la *force* ou la *précision* (Kovacs, 1999).

Une *fitness* basée sur la force lie directement l'adéquation du classeur à la tâche par la prédiction d'une récompense liée à son usage. Les systèmes

de classeurs fondés sur la *force* cherchent à conserver et faire évoluer les classeurs qui ont les plus grandes récompenses. Les principaux processus de rétribution par la *force* sont inspirés de l'algorithme du *Bucket Brigade* (Holland, 1985) dont l'idée est de renforcer, selon leur participation, tous les classeurs qui sont intervenus dans l'usage d'un *Conséquent* (Buche et al., 2006).

Une *fitness* basée sur la *précision* découple la prédiction d'une récompense de l'exactitude de cette prédiction au cours de l'apprentissage. Les approches fondées sur la *précision* tendent alors à conserver et faire évoluer des classeurs dont la prédiction de la récompense converge durant l'apprentissage, indépendamment de la valeur de celle-ci. Un classeur dont la prédiction a convergé implique que celui-ci s'est adapté aux données d'apprentissage. Au contraire, un classeur dont la prédiction ne converge pas suppose que celui-ci peut ne pas être suffisamment spécifique à certaines données d'entrée ou bien trop général.
Les principaux processus de rétribution par la *précision* sont inspirés de l'algorithme du *Q-Learning* (Watkins and Dayan, 1992) pour prédire la récompense d'un classeur, dont l'équivalence avec le *Bucket Brigade* est montrée par (Dorigo and Bersini, 1994). Le calcul de l'exactitude de la récompense est majoritairement réalisé à partir d'une erreur sur la prédiction (Wilson, 1995), ou en comparant le nombre de classifications correctes à la correspondance d'un classeur avec une donnée d'entrée (Bernadó-Mansilla and Garrell-Guiu, 2003).

En complément des processus de rétribution dont de nombreuses variantes existent (Urbanowicz and Moore, 2009), d'autres paramètres propres aux classeurs sont également mis à jour tels que l'expérience, la numérosité ou différentes empreintes temporelles. Ces paramètres interviennent dans la gestion des différents ensembles de classeurs ($[P]$, $[M]$, $[A]$ ou $[C]$, $[I]$).
L'expérience d'un classeur, souvent notée *exp*, désigne le nombre de fois où ce classeur a pu appartenir au sous-ensemble d'appariement $[M]$ ou au sous-ensemble d'action $[A]$.
Un classeur peut exister en différents exemplaires au sein d'une population de classeurs. Chaque exemplaire est alors appelé *micro-classeur* quand l'ensemble des exemplaires est groupé dans un *macro-classeur*. La numérosité, souvent notée *num*, désigne alors le nombre de *micro-classeurs* représentés

par un (*macro*-) classeur.
Enfin, les empreintes temporelles d'un classeur désignent les instants où le classeur appartenait à différents sous-ensembles de classeurs ou a été utilisé dans des mécanismes de découverte de nouvelles règles. Ces empreintes permettent de contrôler l'activation de ces mécanismes.

2.2.7 Suppression de classeurs et subsomption

Les classeurs peuvent être supprimés de la population lorsque :

- ils sont inadéquats à la tâche à résoudre ou aux données d'entrée ;
- la taille limite de la population de classeurs est atteinte (ou celle d'un sous-ensemble de la population) ;
- ils sont subsumés par d'autres classeurs, c'est-à-dire qu'il existe une relation d'inclusion entre deux classeurs.

L'adéquation d'un classeur à une tâche ou aux données d'entrée peut être définie par une métrique bornée comme dans (Stolzmann, 1999). Un seuil empirique, défini par l'utilisateur en amont de l'apprentissage, est alors associé à cette métrique en dessous duquel un classeur est automatiquement supprimé par le système.

Afin de limiter l'expansion de la population de classeurs, il est possible de définir une taille limite sur l'ensemble de la population de classeurs ou sur un sous-ensemble de la population de classeurs tels que $[A]$. Lorsque ces limites sont dépassées, des classeurs appartenant aux ensembles correspondants sont supprimés selon une *roulette wheel selection* basée sur la *fitness des classeurs* (Butz and Wilson, 2000) ou selon une approche où les paramètres internes d'un classeur ont une probabilité d'être directement comparés avec un classeur sélectionné au hasard (Butz and Stolzmann, 2001).

Un **processus de subsomption** est également intégré à des systèmes de classeurs (Butz and Wilson, 2000 ; Butz and Stolzmann, 2001) et appelé avant d'insérer un classeur à une population. Par exemple, si des classeurs nouvellement créés possèdent une structure conditionnelle identique à celle de classeurs déjà présents au sein d'une population, ces nouveaux classeurs sont alors supprimés et les paramètres internes de ceux déjà présents sont

modifiés. Les processus de subsomption suppriment également les classeurs dont les situations décrites par leurs structures conditionnelles (*Condition* et *Conséquent*) sont incluses dans celles issues de classeurs adaptés aux données d'apprentissage. La figure 2.8 illustre ces situations où les processus de subsomption interviennent pour supprimer des classeurs des populations des LCS.

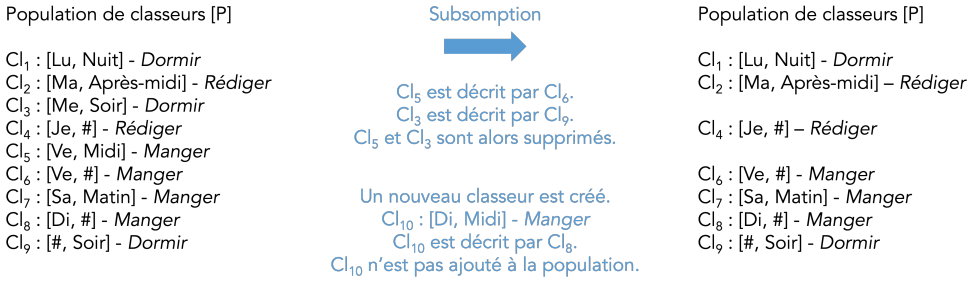


FIGURE 2.8 – Illustration du processus de subsomption au sein d'une population de classeurs.
On considère dans cet exemple que tous les classeurs de la population sont adaptés aux données d'apprentissage.

2.2.8 Récapitulatif

La figure 2.9 synthétise l'ensemble des composants présents dans les systèmes de classeurs, ainsi que leur fonctionnement. Les classeurs doivent dans un premier temps être interfacés avec les données afin de permettre l'apprentissage. Lors de l'apprentissage, les classeurs sont ensuite gérés par différents processus de découverte de règles pour créer de nouveaux classeurs, par des processus de rétribution et de mise à jour pour réaliser effectivement l'apprentissage et enfin, par des processus de suppression et de subsomption pour retirer les classeurs inadéquats à la résolution de la tâche de la population. Ces différents processus peuvent s'appliquer sur :

- l'ensemble de la population de classeurs ($[P]$) ;
- la sous-population de classeurs dont la *Condition* correspond à la donnée d'entrée (ensemble d'appariement – $[M]$) ;
- la sous-population de classeurs dont la *Condition* correspond à la donnée d'entrée et selon l'étiquette de celle-ci ($[I]$ et $[C]$) ou à l'action élicitée par le système ($[A]$).

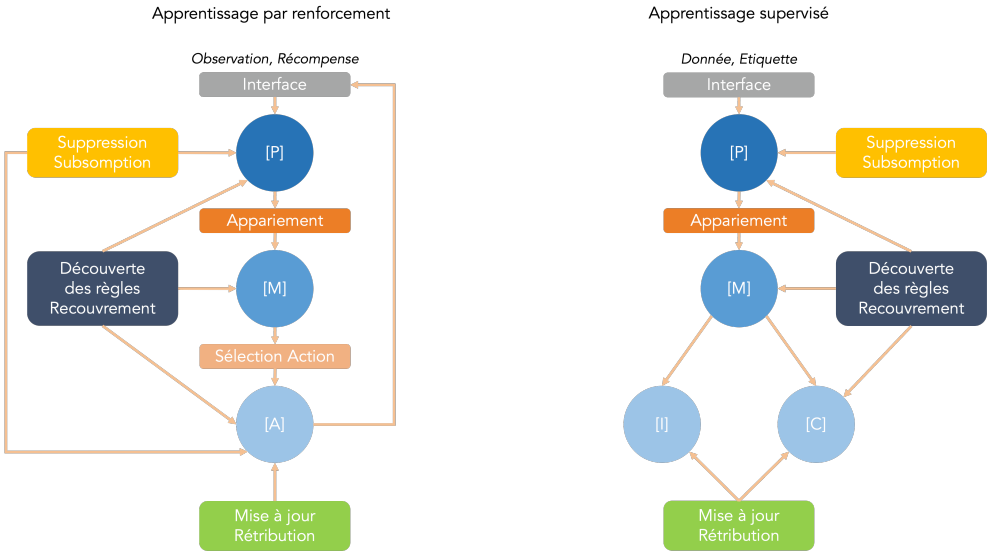


FIGURE 2.9 – Synthèse des principaux composants des systèmes de classeurs sous apprentissage par renforcement et sous apprentissage supervisé.

2.2.9 Exploitation des classeurs

L'apprentissage des systèmes de classeurs a pour but principal de développer une population de classeurs représentant une solution à un problème complète, compacte, avec le moins de recouvrement possible et correcte (Urbanowicz and Moore, 2009).

Parvenir à une telle population de classeurs dès la fin de l'apprentissage n'est pas nécessairement garanti par le système. De multiples pressions sont exercées sur ces systèmes et interagissent entre elles afin de faire converger les populations vers ce but (Butz and Pelikan, 2001). Un équilibre entre celles-ci est alors recherché : par exemple, une spécialisation trop forte des classeurs par les processus de découverte de règles peut améliorer la résolution de la tâche à apprendre, avec le risque de surapprendre et d'accroître la taille de la population de classeurs.

À la fin de l'apprentissage, la population de classeurs obtenue peut être manipulée afin de la faire tendre vers la solution idéale (complète, correcte et compacte) par le biais d'outils d'analyse de la population, avant d'être exploitée en vue de résoudre la tâche apprise. Les outils d'analyse des populations de

classeurs cherchent à visualiser ou à extraire les connaissances acquises au sein de celles-ci. Ils permettent également d'obtenir des indicateurs permettant de suivre l'évolution de l'apprentissage et potentiellement de le guider.

Visualisation des populations de classeurs

Différents outils de visualisation ont été développés pour les systèmes de classeurs permettant de suivre les structures conditionnelles, les valeurs de *fitness* ou les prédictions des classeurs. Par exemple, la figure 2.10 illustre un outil permettant de suivre les structures conditionnelles des classeurs. (Urbanowicz et al., 2012b) et (Zhang et al., 2021) proposent des cartes de chaleur des attributs des structures conditionnelles des classeurs pour identifier des motifs issus des attributs en lien avec les prédictions des classeurs, ainsi que des réseaux de co-occurences entre les attributs des structures conditionnelles des classeurs. (Liu et al., 2021) proposent des outils de visualisation mettant en avant des motifs de structures conditionnelles propres à un *Conséquent*, des motifs de structures conditionnelles propres à l'ensemble du problème à résoudre ou encore l'importance des attributs pour un *Conséquent*.

Extraction de connaissance des populations de classeurs

Différentes techniques pour extraire les connaissances des populations de classeurs existent telles que :

- la compression (Tan et al., 2013) : cette technique cherche à supprimer de la population des classeurs qui se recouvrent, des classeurs de faible *fitness* et des classeurs dont les *Conséquents* se contredisent ;
- la condensation (Kovacs, 1998), où le système de classeurs est mis en œuvre sans les processus de découverte de règles, afin de laisser le système supprimer les classeurs les moins adéquats au problème ;
- la construction de nouvelles règles (Kharbat et al., 2007) : des classeurs similaires sont regroupés afin de construire un nouveau classeur représentant les caractéristiques du groupe.

Ces techniques pour extraire les connaissances des populations de classeurs se reposent sur les données d'apprentissage (*« data-driven »*) ou bien, direc-

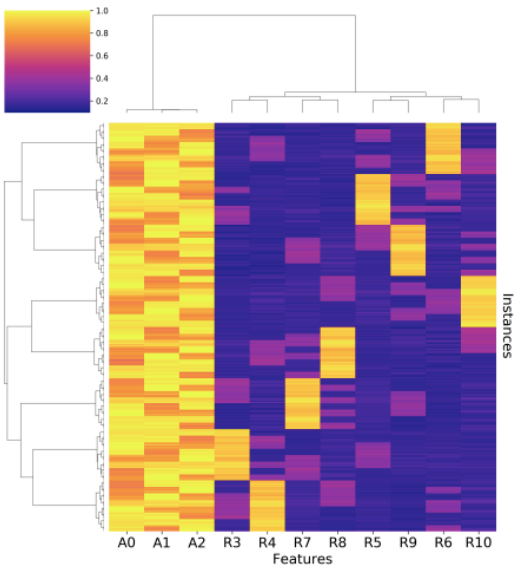


FIGURE 2.10 – Exemple de visualisation d'une population de classeurs à l'issue de son apprentissage sur le problème du multiplexeur à 11 bits.
La figure est tirée de (Zhang et al., 2021). Dans le multiplexeur à 11 bits, trois bits d'adressage (A0, A1 et A2) pointent vers un bit de registre dont la valeur correspondra à la sortie (R3, R4, ..., R10). Un système d'apprentissage automatique doit alors apprendre à corréler ces bits d'adressage vers les bits de registre. Cette illustration met en avant huit principaux regroupements de classeurs, ceux-ci ayant été réalisés à partir d'un score calculé pour chaque attribut reflétant son importance dans la résolution du multiplexeur.

tement sur les classeurs (« *rule-driven* ») sans tenir compte des données d'apprentissage (Urbanowicz et al., 2012b). En particulier, la condensation se repose principalement sur les données d'apprentissage, quand la compression et la construction de nouvelles règles peuvent s'appuyer tant sur les données que les règles.

Résoudre une tâche d'apprentissage

Pour résoudre le problème appris, les systèmes de classeurs doivent identifier dans leur population les classeurs les plus adaptés à une donnée d'entrée. Les *Conditions* de ces classeurs doivent d'abord correspondre à cette donnée : l'ensemble d'appariement $[M]$ est ainsi construit.

Cependant, les classeurs de cet ensemble d'appariement $[M]$ n'ont pas nécessairement tous le même *Conséquent*. Différentes stratégies sont possibles pour choisir le *Conséquent* qui serait le plus adapté à la donnée d'entrée :

en sélectionnant le classeur avec la plus grande *fitness* (Butz and Stolzmann, 2001) (potentiellement pondérée par la numérosité des classeurs (Urbanowicz and Moore, 2015)), en utilisant la *roulette wheel selection* (Wilson, 1994), ou en employant des schémas de vote inspirés de l'apprentissage ensembliste [2] (Gao et al., 2007).

Nous reviendrons en particulier dans le chapitre 8 sur l'extraction de connaissances des populations de classeurs, car nous proposons d'employer une technique de compression dédiée à une famille de systèmes de classeurs afin de faciliter la compréhensibilité des solutions construites.

2.3 Systèmes de classeurs actuels

La recherche dans le domaine des systèmes de classeurs se concentre autour des thématiques suivantes (Pätzel et al., 2020, 2021) :

- la conception de nouveaux LCS ;
- l'amélioration de systèmes de classeurs existants ;
- la théorie formelle ;
- les applications des LCS.

Dans cette section, nous présentons quelques travaux récents dans chacune de ces thématiques (Pätzel et al., 2020, 2021).

2.3.1 Conception de nouveaux systèmes de classeurs

De nouveaux LCS s'appuient sur les principes présentés dans la section 2.2 pour proposer de nouvelles approches permettant aux LCS de résoudre des problèmes qui ne leur étaient pas accessibles, ou permettant de mettre en avant certaines propriétés des LCS.

(Heider et al., 2022) proposent ainsi un nouveau système de classeurs dont le but est de faciliter l'explicabilité des solutions mises en œuvre par les LCS en

2. L'apprentissage ensembliste désigne le fait d'utiliser plusieurs modèles d'apprentissage automatique afin de résoudre un même problème (Zhou, 2021). Les systèmes de classeurs peuvent être abordés comme des méthodes ensemblistes où chaque classeur désigne un modèle entraîné. Les techniques employées avec les méthodes ensemblistes sont adaptables aux LCS (Liu et al., 2018).

leur permettant un meilleur contrôle de leur population de classeurs. Pour ce faire, ce système de classeurs utilise en particulier de nouvelles mécaniques de découverte de règles permettant d'identifier quels classeurs vont composer une solution à la tâche d'apprentissage.

(Liu, 2021) propose un nouveau LCS dont le but est de pallier une surgénéralisation des classeurs observée dans d'autres LCS : des classeurs optimaux et des classeurs trop généraux sont introduits et supprimés de manière répétée, impactant par exemple le maintien des performances de prédiction de ce système. (Liu, 2021) développe alors ce nouveau LCS où les processus de découverte de nouvelles règles s'appuient sur la subsomption et un mécanisme complémentaire, l'absomption. Le but de l'absomption est de spécifier les classeurs surgénéraux à partir des entrées pour lesquels le *Conséquent* de ces classeurs est erroné.

(Siddique et al., 2021) s'inspirent de la capacité des vertébrés à réemployer des connaissances acquises sur des tâches simples vers des tâches plus complexes pour concevoir un nouveau système de classeurs. Cette capacité serait attribuée à la latéralisation et à la modularité du cerveau, qu'ils adaptent alors au cadre posé par les LCS. Leurs travaux proposent alors de nouvelles représentations pour manipuler les connaissances des classeurs, de nouveaux processus de découverte de règles en particulier pour gérer l'incertitude inhérente à des problèmes de décision.

2.3.2 Amélioration des systèmes de classeurs

L'amélioration des LCS peut prendre différentes formes : les systèmes de classeurs peuvent être couplés avec d'autres approches ; des extensions peuvent être conçues dans le but d'améliorer leur capacité d'apprentissage ou l'exploitation des populations de classeurs.

Une approche avec laquelle les systèmes de classeurs sont couplés concerne les réseaux de neurones. (Irfan et al., 2022) et (Tadokoro et al., 2021) couplent un réseau de neurones avec un système de classeurs afin de résoudre des tâches classification d'images. Le réseau de neurones a pour but d'extraire les caractéristiques des images, qui sont ensuite encodées pour être manipu-

lées par un système de classeurs. (Shiraishi et al., 2021) suivent cette même approche en cherchant à mettre l'accent sur l'explicabilité du système résultant, *i.e.* en ne focalisant pas uniquement sur les performances liées à la classification. (Irfan et al., 2022), (Tadokoro et al., 2021) et (Shiraishi et al., 2021) proposent ainsi de nouvelles méthodes pour interfacer les systèmes de classeurs avec des données à haute dimensionnalité.
(Preen and Bull, 2021) proposent d'utiliser deux réseaux de neurones profonds pour modéliser la *Condition* et le *Conséquent* des classeurs pour des tâches de classification d'images. Le nombre de neurones et la connectivité des neurones de ces réseaux peuvent également évoluer au cours de l'apprentissage grâce à l'emploi d'algorithmes évolutionnaires. (Preen et al., 2021) emploient cette même idée pour permettre à un système de classeurs de décomposer de manière adaptative l'espace associé aux données d'entrée en une collection de petits auto-encodeurs.
(Chen et al., 2021) emploient également des LCS avec des réseaux de neurones dans le cadre de jeux où un agent doit développer une stratégie gagnante contre un concurrent. La stratégie du concurrent est modélisée sur le tas, par le biais d'un réseau de neurones, et les décisions associées servent à établir la stratégie de l'agent construite par le système de classeurs.

(Kozłowski and Unold, 2021) proposent une nouvelle mécanique de rétribution pour un système de classeurs conçu pour répondre à des problèmes de décision à une ou plusieurs étapes. Cette nouvelle mécanique s'appuie sur la maximisation de la moyenne des récompenses successives, au lieu de maximiser la récompense totale.

(Wagner and Stein, 2021) intègrent le mécanisme d'absomption introduit par (Liu, 2021) (présenté dans le paragraphe précédent) dans un système de classeurs pour lui permettre de spécialiser les classeurs surgénéraux, sachant que ce système ne disposait pas de méthodes dédiées à cet effet.

2.3.3 Théorie formelle des systèmes de classeurs

La théorie formelle inclut l'étude des paramètres d'apprentissage des LCS, ou encore l'étude des propriétés des LCS en lien avec leur tâche d'apprentissage.

(Hansmeier and Platzner, 2021) comparent différentes stratégies d'exploration et d'exploitation des connaissances acquises par un système de classeurs pour résoudre des tâches d'apprentissage par renforcement. Ils étudient différentes stratégies d'exploration et d'exploitation et montrent qu'aucune des stratégies étudiées n'est supérieure aux autres.

(Nakamura et al., 2021) étudient les processus de découverte de nouvelles règles d'un système de classeurs, et plus particulièrement l'évolution de la généralité des classeurs. Ils démontrent alors que la généralité des classeurs peut converger vers sa valeur optimale sous certaines conditions, en plus de fournir une ligne directrice aidant à la configuration de ces processus.

2.3.4 Application des systèmes de classeurs

Les domaines d'application des systèmes de classeurs sont variés allant par exemple de l'extraction de connaissances à l'analyse de la satisfaction d'une clientèle, en passant par l'Internet des Objets ou encore la détection de tentatives d'hameçonnage (Pätzel et al., 2020, 2021). Récemment, (Büttner and Von Mammen, 2021) emploient un système de classeurs pour lui apprendre à jouer au jeu du serpent. Le but de ce jeu est pour un agent de faire grandir un serpent en mangeant le plus de pommes, sans que celui-ci ne se heurte sur un obstacle, quel qu'il soit. (Zhang et al., 2021) introduisent un ensemble de techniques et d'outils permettant de visualiser et d'interpréter les populations de classeurs associées à des tâches de classification dans le domaine biomédical. Le but est ainsi de faciliter l'extraction des connaissances contenues dans les populations de classeurs. (Rosenbauer et al., 2021) utilisent des systèmes de classeurs dans l'automatisation des étapes de tests lors du développement de logiciels. Plus précisément, ces systèmes interviennent afin de sélectionner un sous-ensemble de tests à réaliser, quand il n'est pas possible de tous les exécuter à cause de ressources limitées. (Chi et al., 2021) emploient un système de classeurs afin de détecter en temps réel une addiction à Internet sur la base de symptômes psychologiques et de troubles respiratoires.

2.4 Le choix de l'anticipation

Une multitude de systèmes de classeurs existent : à la date de leur bibliographie, (Urbanowicz and Moore, 2009) relevaient alors plus de 80 systèmes de classeurs différents. Cependant, parmi tous ces systèmes et ceux qui ont été conçus depuis, trois systèmes de classeurs sont à la base de nombreux autres systèmes (Butz, 2015). Il convient alors de choisir quelle famille de systèmes de classeurs serait la plus adaptée à nos travaux visant à doter une intelligence artificielle d'une plus grande autonomie et d'une meilleure explicabilité. Ces trois principales familles sont d'abord introduites dans les sections suivantes pour motiver le choix des systèmes de classeurs à anticipation dans la section 2.4.3.

2.4.1 Des systèmes de classeurs issus de ZCS et XCS

Quand (Holland and Reitman, 1978) posaient les fondations de ce que serait un système de classeurs, une étape importante a été franchie lorsque (Wilson, 1994) introduit le système de classeurs *Zeroth-level Classifier System (ZCS)* puis l'année suivante, lorsque (Wilson, 1995) proposait le système de classeurs XCS [3] (Sigaud and Wilson, 2007 ; Urbanowicz and Moore, 2009 ; Butz, 2015). Ces deux systèmes de classeurs sont le résultat d'une simplification du cadre des LCS posé par (Holland and Reitman, 1978) et de l'intégration de mécanismes adaptés de l'algorithme du *Q-Learning* (Watkins and Dayan, 1992), permettant de démontrer la viabilité des idées initialement introduites dans les LCS (Urbanowicz and Moore, 2009).

ZCS et XCS sont ainsi deux LCS à partir desquels d'autres systèmes ont été développés, formant chacun une famille de systèmes de classeurs. Ces deux familles se distinguent principalement par :

- des valeurs d'adéquation (la *fitness*) calculées sur la *force* des récompenses pour ZCS, alors qu'elles sont basées sur la *précision* pour XCS ;
- des mécaniques de découverte de nouvelles règles s'appliquant sur

3. XCS désigne le nom de ce système de classeurs (Wilson, 1995). L'acronyme XCS peut être vu comme récursif, puisque le nom originel du système est *XCS Classifier System* (Pätzel et al., 2021). Bien qu'inexacte, l'appellation *eXtended Classifier System* peut être donnée dans la littérature à ce système de classeurs XCS.

l'ensemble de la population pour ZCS, quand elles sont appliquées sur une population de niche (*i.e.* un sous-ensemble précis de classeurs) pour XCS ;
— de nouvelles politiques de sélection d'actions tirant parti des nouvelles valeurs d'adéquation des classeurs d'XCS.

Ces changements introduits dans XCS avaient pour but de répondre à des problèmes présents dans ZCS associés à l'adéquation des classeurs. L'usage de la *force* pour calculer la *fitness* des classeurs peut prévenir le développement de classeurs adéquats à des situations environnementales à faibles récompenses ou à faibles occurrences : des classeurs ayant une *force* élevée peuvent ainsi être favorisés dans ces situations pour intégrer les différents processus des LCS, quand bien même ils ne seraient pas les plus adaptés (Wilson, 1995).
Par ailleurs, une *force* élevée indique une prédiction d'une récompense importante, sans que celle-ci ne soit garantie : s'appuyer sur la *précision* revient alors à préférer les classeurs pour lesquels les récompenses prédites seront celles effectivement obtenues (Wilson, 1995).

2.4.2 Une troisième famille de systèmes de classeurs

La troisième principale famille de LCS est apparue peu après la conception de ZCS et de XCS, lorsque (Stolzmann et al., 1998) ont proposé leur système *Anticipatory Classifier Systems (ACS)* : il s'agit des systèmes de classeurs à anticipation (*Anticipatory Learning Classifier System* – ALCS). Les ALCS s'appuient sur le cadre des LCS et plus particulièrement celui posé par XCS, partageant ainsi des éléments de conception tels que l'application des mécaniques de découverte de nouvelles règles sur des niches. Cependant, ils diffèrent principalement des familles issues de ZCS et XCS par la structure des classeurs qu'ils manipulent (Butz, 2015).

La structure des classeurs des ALCS est composée d'une *Condition*, d'un *Conséquent* et d'une *Anticipation*, qui représente les conséquences anticipées du *Conséquent* pour l'ensemble de situations décrit par la *Condition*. L'*Anticipation* permet ainsi aux ALCS de se doter d'une représentation de l'environnement dans lequel ils réalisent leur apprentissage, aussi appelé

modèle de transitions dans la littérature sur l'apprentissage par renforcement (Sigaud and Wilson, 2007). Cette représentation est obtenue grâce aux processus de découverte de nouvelles règles qui s'appuient sur le cadre du contrôle comportemental anticipatif (Hoffmann, 1993).

Le contrôle comportemental anticipatif décrit l'acquisition et l'usage de connaissances, en s'appuyant sur l'idée que le comportement issu de ces connaissances est orienté pour répondre à un objectif. Les actions, ou les décisions, sont alors réalisées afin de répondre à cet objectif, d'autant plus quand leurs effets anticipés correspondent à ce qui est observé. Ce cadre, schématisé dans la figure 2.11, décrit comment cet apprentissage s'opère :

1. une action est choisie afin d'atteindre un objectif, où les effets anticipés de cette action vont dans le sens de cet objectif ;
2. les effets attendus suite à cette action sont comparés avec les effets réels, afin de faire correspondre l'action à ses effets ;
3. l'action choisie et ses effets anticipés sont contextualisés selon les situations où l'action choisie est réalisée, permettant ainsi de conditionner l'action choisie et ses effets attendus.

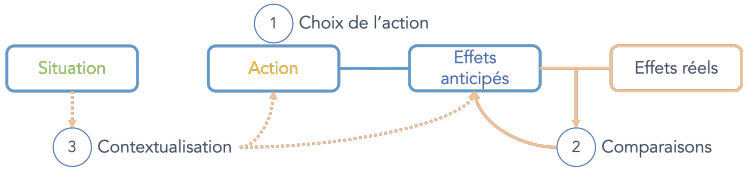


FIGURE 2.11 – Synthèse de l'apprentissage décrit par le contrôle comportemental anticipatif.

Concrètement, les ALCS construisent leurs classeurs en comparant dans un premier temps des perceptions successives de leur environnement, tel qu'illustré par la figure 2.12. Ces comparaisons leur permettent alors de déterminer quels sont les effets des actions (*i.e.* les anticipations) qu'ils peuvent réaliser dans leur environnement et quelles sont les conditions où ces anticipations construites correspondent aux observations, même en l'absence de renforcement ou de récompenses. Les ALCS sont ainsi capables d'apprentissage latent (Stolzmann et al., 1998), puisque cette représentation n'est utilisée que dans un second temps, pour doter les ALCS d'un comportement propice à la résolution de la tâche d'apprentissage.

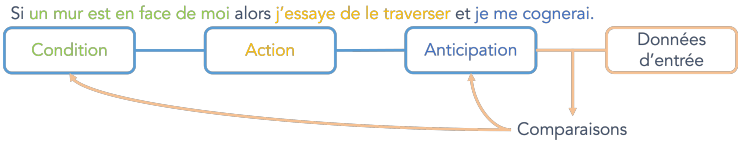


FIGURE 2.12 – Illustration de la structure et de la construction d'un classeur d'un ALCS.
Un agent ne peut pas traverser un mur et se heurte donc à celui-ci. Les perceptions reçues successivement par l'ALCS, avant et après la réalisation de l'action, vont lui permettre de se doter d'un classeur caractéristique de cette transition environnementale. Seule la structure du classeur d'un ALCS est ici présentée : les autres informations contenues dans ces classeurs sont présentées dans le chapitre suivant, section 3.1.

Les systèmes de classeurs à anticipation sont caractérisés par l'intégration des principes du contrôle comportemental anticipatif au sein des différents processus qui interviennent dans le cadre des LCS. L'idée seule d'anticiper les effets des actions n'est donc pas suffisante pour rattacher un LCS aux ALCS. Néanmoins, d'autres systèmes de classeurs ont pu tenter d'apprendre à anticiper les effets de leurs actions :

- (Sigaud and Wilson, 2007) relevaient l'absence de résultats suffisamment convaincants pour doter XCS de capacités d'anticipation explicites. Par la suite, (Siddique et al., 2021) proposent un LCS s'appuyant sur XCS qui construit un modèle de transitions d'environnements incertains pour résoudre leurs tâches d'apprentissage. À cette fin, (Siddique et al., 2021) s'inspirent de la capacité du cerveau des vertébrés à réemployer des connaissances.
- (Zatuchna, 2004) propose un LCS ne s'inscrivant dans aucune des trois principales familles et dont le but est aussi de construire un tel modèle de transitions : il s'appuie sur les processus de perception et d'association présents chez les êtres humains et les animaux décrits par (Lorenz, 1935). Un agent apprend à former des « images » indécomposables et uniques à partir d'informations environnementales reçues successivement et non modifiées, que cet agent agisse ou non dans son environnement.

2.4.3 Choix des systèmes de classeurs à anticipation

Le cadre des LCS permet de concevoir des systèmes d'apprentissage automatique explicables pouvant effectuer leur apprentissage dans des environnements incertains en autonomie. Le positionnement des systèmes de classeurs à anticipation est particulièrement intéressant pour plusieurs raisons.

La présence d'une structure dédiée à l'*Anticipation* dans les classeurs des ALCS permet de fournir une information supplémentaire relative à l'usage des classeurs par rapport aux autres systèmes. Plus précisément, elle permet de lier les effets d'une action dans un environnement aux situations environnementales qui ont déclenché l'action. Ces liens de cause à effet sont permis par les principes du contrôle comportemental anticipatif qui guident la création de ces classeurs en mettant l'accent sur les changements perçus par les systèmes. Les explications issues des ALCS sont ainsi plus complètes, puisque le comportement des ALCS devient prédictible (Gilpin et al., 2018).

Ensuite, contrairement aux ALCS, les systèmes de classeurs tels que ceux basés sur ZCS ou XCS s'appuient sur des processus stochastiques pour réaliser leur apprentissage : ils cherchent à trouver les corrélations leur permettant de résoudre leur tâche d'apprentissage. Quand des liens de cause à effet sont recherchés dans le domaine de l'apprentissage automatique explicable (Molnar et al., 2020), les ALCS sont donc en mesure de fournir plus d'éléments explicatifs par rapport à ces autres LCS.

L'anticipation permet aux ALCS de se doter d'une représentation de l'environnement : l'*Anticipation* d'un classeur peut être chaînée à des *Conditions* d'autres classeurs. Cette représentation, en plus de fournir un modèle de transition, s'appuie sur des mécaniques de généralisation (contrairement à (Zatuchna, 2004) et (Siddique et al., 2021)). Ces mécaniques de généralisation présentent deux avantages :

- ils permettent une description des situations environnementales plus compacte, en identifiant les caractéristiques propres aux situations conditionnelles. La généralisation permet ainsi d'obtenir des éléments explicatifs sur *comment* les différentes situations environnementales peuvent être distingués par les ALCS ;
- ils facilitent l'emploi de connaissances déjà acquises par les ALCS à de

nouvelles situations. Autrement dit, ils renforcent la capacité de ces systèmes à prendre en charge la volatilité des environnements incertains.

Enfin, les systèmes de classeurs à anticipation découplent l'apprentissage d'un ensemble de règles qui s'appliquent dans un environnement, de leur usage pour réaliser un objectif. L'apprentissage latent des ALCS leur alors confère plus d'adaptabilité si la tâche d'apprentissage vient à changer : il ne serait pas nécessaire de reconstruire l'ensemble des classeurs, puisqu'il suffirait d'adapter seulement leur usage à la nouvelle tâche.

Les systèmes de classeurs à anticipation sont l'objet du prochain chapitre, où ils sont présentés plus en détail. Nous serons alors en mesure de discuter de leurs limites quand ces derniers doivent réaliser leur apprentissage au sein d'environnements incertains.

CHAPITRE 3

Les systèmes de classeurs à anticipation

Le premier système de classeurs à anticipation, ACS, est d'abord introduit, afin de présenter tous les autres ALCS qui ont suivi, et plus particulièrement ACS 2, ainsi que leurs applications. Des limites relatives à l'autonomie et l'explicabilité des ALCS quand ils doivent évoluer dans des environnements incertains sont mises en évidence.

Sommaire

3.1 ACS : le premier système de classeurs à anticipation

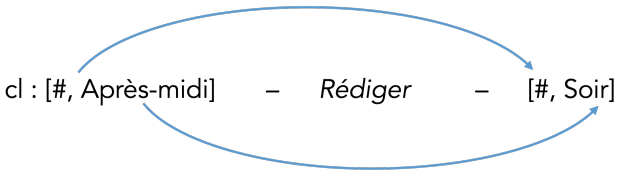
L'ACS de (Stolzmann, 1999) pose le cadre des systèmes de classeurs à anticipation, avec l'idée qu'en anticipant les conséquences des actions réalisées par ces systèmes, ils peuvent adapter leur comportement à l'évolution de leur environnement de façon plus fine et plus rapide.

3.1.1 Structure des classeurs

ACS construit des classeurs dont la structure est composée d'une *Condition*, d'un *Conséquent* prenant le plus souvent la forme d'une action et d'une *Anticipation* des effets du *Conséquent* si la *Condition* est respectée. Cette structure est notée sous la forme d'un tuple $\{C - A - E\}$ [1] où C désigne la *Condition*, A le *Conséquent* et E l'*Anticipation*.

Les *Conditions* et *Anticipations* d'ACS sont décrites par une conjonction logique des attributs perceptifs, eux-mêmes décrits par une représentation symbolique. Un symbole générique # est utilisable dans les *Conditions* et *Anticipations* des classeurs (*cf.* figure 3.1) :

- un attribut perceptif dont le symbole est # dans une *Condition* signifie que n'importe quelle valeur symbolique possible pour cet attribut peut y être associée ;
- un attribut perceptif dont le symbole est # dans une *Anticipation* signifie que l'ACS ne prédit pas de changement perceptif suite à la réalisation de l'action décrite par le *Conséquent* du classeur.



L'action *Rédiger* entraîne un unique changement :
- peu importe le jour de la semaine, il ne change pas ;
- le moment passe de l'*Après-midi* au *Soir*.

FIGURE 3.1 – Représentation de la structure des classeurs d'ACS au travers de deux exemples.
Les perceptions, Conditions *et* Anticipations *sont composées d'un jour de la semaine ({Lu, Ma, Me, Je, Ve, Sa, Di}) et d'un moment ({Matin, Midi, Après-midi, Soir, Nuit}). Les* Conséquents *désignent une action parmi {Manger, Rédiger, Dormir}. Le classeur cl indique que rédiger dans l'après-midi amène à percevoir le soir du même jour. Cette structure est commune à tous les ACS et les ALCS.*

Les classeurs d'ACS sont composés de trois attributs supplémentaires :

1. Provient de l'anglais, {*Condition - Action - Effect*}. La traduction peut paraître surprenante, mais elle a pour origine le contrôle comportemental anticipatif qui décrit les *effets E* des *actions A* dans les situations décrites par les *conditions C*. Les systèmes de classeurs à anticipation ont ensuite été utilisés pour résoudre de nouvelles tâches d'apprentissage, sans que cette notation soit modifiée. Le A de ce tuple peut désigner une action, une décision ou une classification, d'où l'utilisation du mot *Conséquent* en français.

— une marque M (*mark*), qui désigne une situation environnementale où le classeur a échoué à anticiper la situation suivant l'action décrite par son *Conséquent* ;
— une mesure q de la qualité (*quality*) de l'*Anticipation* du classeur ;
— une prédiction r de la récompense (*reward*) obtenue par le système suite à la réalisation de l'action décrite par son *Conséquent*.

Les marques M et les qualités q des anticipations, ainsi que les *Conditions* et les *Anticipations*, sont mises à jour au travers d'un processus d'apprentissage par anticipation (*Anticipatory Learning Process (ALP)*), qui est dérivé des principes du contrôle comportemental anticipatif décrit précédemment. Les prédictions r des récompenses des classeurs sont mises à jour grâce à un processus de rétribution par renforcement. L'ALP construit des classeurs qui décrivent correctement des transitions environnementales, alors que le processus de rétribution adapte ces classeurs à la résolution d'une tâche d'apprentissage.

3.1.2 Cycle d'apprentissage d'ACS

Le système ACS dispose initialement d'une population de classeurs ayant un statut particulier :
— leur *Condition* est uniquement constituée de symboles # ;
— il y a autant de classeurs que d'actions réalisables par le système ;
— leur *Anticipation* est aussi uniquement constituée de symboles # ;
— cet ensemble de classeurs ne peut être supprimé de la population, assurant ainsi l'existence d'au moins un classeur pour chaque situation environnementale perçue par ACS.

Autrement dit, avant de débuter son apprentissage, ACS prédit que toutes ses actions n'entraînent aucun changement environnemental, pour toutes les situations environnementales possibles.

ACS réalise son apprentissage en interagissant avec l'environnement dans lequel il évolue, ce qui lui permet d'adapter cette population de classeurs à l'environnement par le biais de son processus d'apprentissage par anticipation et par le biais de son processus de rétribution. Pour ce faire, ACS s'appuie sur un processus d'appariement qui fait correspondre les *Conditions* des

classeurs aux observations de l'environnement. Son cycle complet d'apprentissage est initialisé avec un classeur choisi au hasard parmi la population initiale et avec une première observation de l'environnement. Cette observation est gardée en mémoire dans une *liste de messages*, dont le rôle est de stocker toutes les observations nécessaires au déroulement de l'apprentissage. L'action décrite dans le classeur aléatoirement choisi est réalisée, puis ACS itère le cycle suivant (*cf.* figure 3.2) :

1. ACS observe une nouvelle situation et reçoit des récompenses de l'environnement. Cette observation est ajoutée dans la *liste des messages*. Les auteurs ne précisent pas si des observations sont retirées de la *liste des messages* ou combien d'observations peuvent être contenues dans cette liste. Si l'objectif d'apprentissage est atteint, les processus d'apprentissage par anticipation et par renforcement sont appliqués une dernière fois sur le classeur sélectionné et le cycle s'arrête.
2. Le processus d'appariement est appliqué sur la population de classeurs $[P]$, permettant de former le sous-ensemble d'appariement $[M]$ où la *Condition* des classeurs correspond à la nouvelle observation.
3. L'ALP est appliqué sur le classeur qui a été précédemment sélectionné par le système (*cf.* section 3.1.3). Un classeur nouvellement généré par l'ALP peut être ajouté à $[P]$ et à $[M]$.
4. Le processus de rétribution par renforcement est appliqué sur le classeur précédemment sélectionné (*cf.* section 3.1.4).
5. Un classeur *cl* est sélectionné soit aléatoirement, soit à partir des classeurs de $[M]$ par la *roulette wheel selection* (équation 3.1) où la probabilité de choisir un classeur *cl* est le rapport de sa *fitness*, qui est le produit entre q_{cl} et r_{cl}, sur la somme des *fitness* des classeurs de $[M]$:
$$P_{\text{cl}} = \frac{q_{\text{cl}}.r_{\text{cl}}}{\sum_{c \in [M]} q_c.r_c} \tag{3.1}$$
6. ACS interagit avec son environnement en réalisant l'action décrite dans le classeur sélectionné.

Bien qu'ACS a été conçu pour résoudre des tâches d'apprentissage par renforcement dans des environnements multiétapes, il peut résoudre des tâches de classification sans modifier son cycle d'apprentissage. Pour ce faire, une *causalité perceptive* est introduite dans les observations d'ACS. Elle prend la forme d'un symbole ayant trois valeurs possibles : 0 si la classification n'a

pas encore eu lieu ; 1 si la classification faite est erronée ; 2 si la classification faite est correcte.

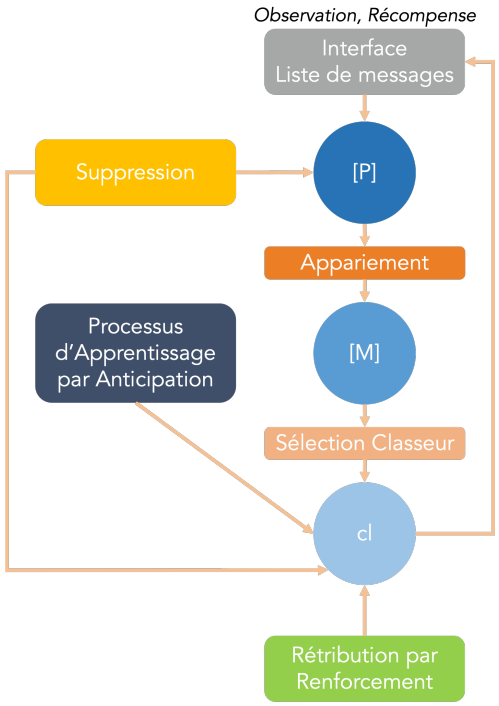


FIGURE 3.2 – Cycle d'apprentissage d'*Anticipatory Classifier System*.
Les principaux composants d'ACS et leurs interactions sont décrits dans cette illustration qui s'appuie la présentation des systèmes de classeurs de la section 2.2 et la synthèse proposée par la figure 2.9.

3.1.3 Processus d'apprentissage par anticipation

L'ALP est le processus de découverte de règles dont le rôle est de créer, mettre à jour et supprimer les classeurs, afin d'apprendre les interactions entre un environnement et ACS.

Il s'appuie d'abord sur l'opérateur *passthrough* qui construit une séquence d'attributs perceptifs de longueur L à partir d'une séquence Seq_2, celle-ci étant spécifiée par une séquence Seq_1 quand un attribut perceptif de Seq_2 correspond au symbole générique # (algorithme 3.1.1) :

- Seq_1 est une *Condition* ou une *Anticipation* d'un classeur, ou une perception de l'environnement ;
- Seq_2 est une *Condition* ou une *Anticipation* d'un classeur ;

— L désigne le nombre d'attributs perceptifs dont sont composées les perceptions, les *Conditions* et les *Anticipations*.

Cet opérateur est utilisé pour savoir si un classeur anticipe correctement ou non des changements perceptifs à partir des observations de la *liste des messages* : un classeur *cl* anticipe bien des changements si la situation courante décrite par P_t est identique au résultat de *passthrough* entre la perception précédente P_{t-1} et son *Anticipation* E_{cl} (*passthrough*(P_{t-1}, E_{cl}, L) == P_t).

Algorithme 3.1.1 Opérateur *passthrough*

```
1 : function PASSTHROUGH(Seq1, Seq2, L)
2 :     Initialize result of length L
3 :     for i ← 0 to L − 1 do
4 :         if Seq2[i] == # then
5 :             result[i] ← Seq1[i]
6 :         else
7 :             result[i] ← Seq2[i]
8 :         end if
9 :     end for
10 :    return result
```

L'ALP s'appuie ensuite sur un pas d'apprentissage β_{alp} pour mettre à jour les qualités des classeurs selon qu'un classeur parvienne ou échoue à anticiper les changements environnementaux. L'équation 3.2 décrit les formules de mise à jour. La qualité q des classeurs est bornée entre 0 et 1 et initialisée à 0.5 : un classeur ayant une qualité de 1 anticipe parfaitement les changements perceptifs ; au contraire, un classeur ayant une qualité nulle échoue complètement à les anticiper.

$$q_{\text{cl}} \leftarrow \begin{cases} (1-\beta_{\text{alp}})q_{\text{cl}} & \textit{si} \text{ l'anticipation est incorrecte} \\ (1-\beta_{\text{alp}})q_{\text{cl}} + \beta_{\text{alp}} & \textit{sinon} \end{cases} \tag{3.2}$$

L'ALP s'appuie enfin sur deux seuils paramétrables θ_{i} et θ_{r} à partir desquels un classeur est respectivement considéré comme inadapté et fiable, avec $\theta_{\text{i}} < \theta_{\text{r}}$. Un classeur inadapté est un classeur dont la qualité q est strictement inférieure à θ_{i} : il est alors supprimé de la population d'ACS par l'ALP. Un classeur fiable est un classeur dont q est strictement supérieure à θ_{r}.

La représentation environnementale du système est l'ensemble des classeurs fiables. ACS a la capacité de réaliser un apprentissage *latent* qui consiste

à se doter d'une représentation complète de son environnement sans qu'il ait besoin de récompenses pour établir des liens entre différentes situations environnementales (Stolzmann et al., 1998). Les liens entre la résolution d'une tâche d'apprentissage et les interactions environnementales apprises dans les classeurs d'ACS sont appris par le processus de rétribution décrit dans la section 3.1.4.

Partant de l'idée que des changements environnementaux sont attendus par ACS suite à son interaction avec l'environnement, l'ALP d'ACS fait évoluer sa population de classeurs au travers de quatre cas :

1. *Useless case*, qui fait décroître la qualité des classeurs quand aucun changement n'est perçu, c'est-à-dire que deux observations successives sont identiques. Les classeurs dont les actions n'ont pas de conséquences sur leur environnement sont défavorisés : ACS ne peut alors maintenir de tels classeurs dans sa population, même si leur anticipation est correcte. La figure 3.3 illustre le *useless case* où un classeur *cl* correspond à la perception P_1 et anticipe l'absence de changement puisque son *Anticipation* est uniquement composée de #. Même si son *Anticipation* est correcte, sa qualité est diminuée puisque les perceptions P_1 et P_2 sont identiques.

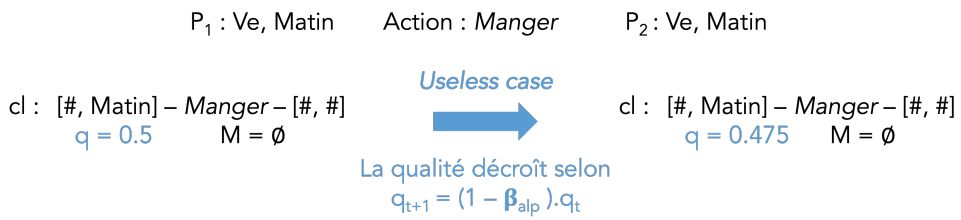


FIGURE 3.3 – *Useless case* de l'ALP d'ACS.

2. *Expected case*, qui fait croître la qualité des classeurs anticipant correctement la prochaine situation à partir de la perception courante. La figure 3.4 illustre l'*expected case* où un classeur *cl* correspond à la perception P_1 et anticipe bien le changement sur le moment du jour : son *Anticipation* indique que le *Matin* devient le *Midi*, ce qui est corrélé par la perception P_2 reçue après avoir réalisé l'action *Rédiger*. Notons que seule la qualité du classeur est augmentée, puisque la marque d'un classeur ne concerne que les cas où un classeur échoue à anticiper les changements perceptifs.

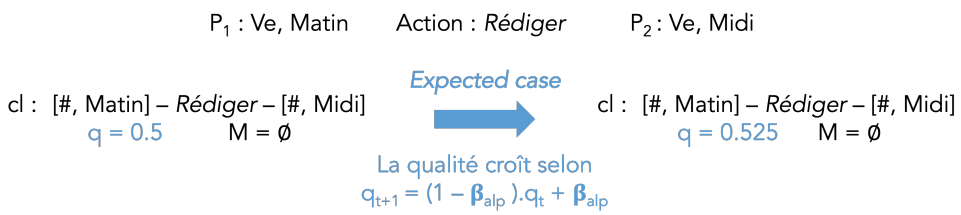


FIGURE 3.4 – *Expected case* de l'ALP d'ACS.

Les deux autres cas correspondent aux classeurs qui échouent à correctement anticiper la prochaine situation environnementale. Leur qualité est diminuée et leur marque est mise à jour avec la situation à l'origine de l'échec de leur anticipation. Si la qualité d'un classeur entrant dans ces cas passe sous le seuil θ_i, il est supprimé de la population. Ces deux dernières situations se différencient selon qu'il est possible ou non de construire un classeur capable de correctement anticiper la prochaine situation environnementale :

3. *Correctable case*, qui permet la construction d'un nouveau classeur si l'*Anticipation* d'un classeur ne décrit pas de changements perceptifs antagonistes avec la transition environnementale courante. Il n'y a pas de changements perceptifs antagonistes si tous les attributs perceptifs spécifiés d'une *Anticipation* sont identiques à la perception suivant la réalisation de l'action. Autrement dit, il est possible de spécifier les attributs inchangés d'une *Anticipation* (symbolisés par #) pour qu'ils correspondent à la transition courante. Quand un attribut perceptif d'une *Anticipation* est spécifié, l'attribut correspondant dans la *Condition* est spécifié si sa valeur est #, c'est-à-dire s'il peut prendre toutes les valeurs possibles. La figure 3.5 illustre le *correctable case* où un classeur *cl* correspond à la perception P_1 et n'anticipe pas le changement sur le moment du jour : son *Anticipation* indique une absence de changement avec # alors que le moment devient le *Midi*. La séquence construite par l'opérateur *passthrough* diffère ainsi de la perception P_2 reçue après avoir réalisé l'action *Rédiger*. La qualité du classeur *cl* est diminuée et sa marque est mise à jour avec P_1. Un nouveau classeur *cl*′ est construit par copie de *cl* où le moment du jour est spécifié dans sa *Condition* et dans son *Anticipation*.

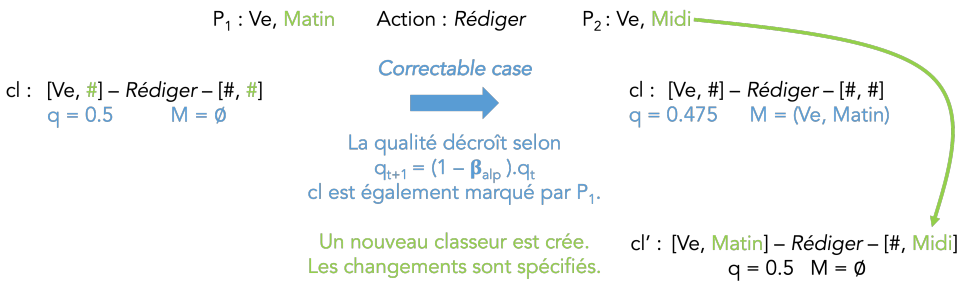


FIGURE 3.5 – *Correctable case* de l'ALP d'ACS.

4. *Not correctable case* qui ne permet pas la construction d'un nouveau classeur. Au contraire du *Correctable case*, l'*Anticipation* des classeurs entrant dans ce cas décrit des changements qui sont différents de la transition environnementale courante et qui peuvent correspondre à d'autres transitions environnementales. La figure 3.6 illustre le *not correctable case* où un classeur *cl* correspond à la perception P_1 et anticipe un changement erroné sur le moment du jour : son *Anticipation* indique le *Soir* alors que le moment devient le *Midi*. La séquence construite par l'opérateur *passthrough* diffère ainsi de la perception P_2 reçue après avoir réalisé l'action *Rédiger*. La qualité du classeur *cl* est alors diminuée et sa marque est mise à jour avec P_1. Cependant, aucun nouveau classeur n'est construit puisque le moment du jour est déjà spécifié dans *cl*.

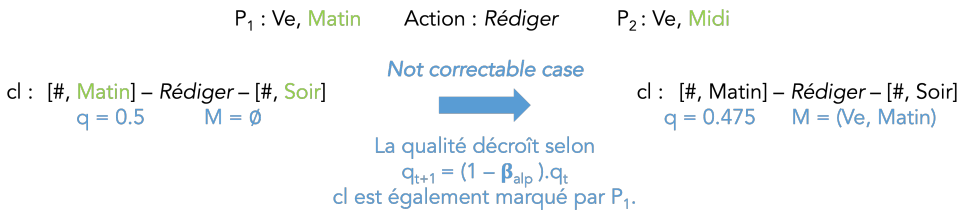


FIGURE 3.6 – *Not correctable case* de l'ALP d'ACS.

L'ALP d'ACS créé de nouveaux classeurs *via* le Correctable case. Ces nouveaux classeurs sont ajoutés à la population seulement s'ils n'existent pas déjà dans la population, sinon leur qualité est augmentée comme si leur anticipation était correcte.

3.1.4 Processus d'apprentissage par rétribution

Le processus de rétribution par renforcement vient s'ajouter à l'ALP pour renforcer ou affaiblir les classeurs avec des récompenses issues de l'environnement. Le rôle de ce processus est d'adapter la population de classeurs (qui contient une représentation latente de l'environnement) à la résolution de tâches particulières.

Le processus utilisé dans ACS s'appuie sur le *Bucket Brigade Algorithm* de (Holland, 1985) : si une récompense fournie par l'environnement est non nulle (positive ou négative), elle est utilisée pour mettre à jour la récompense du classeur ; sinon, la récompense du prochain classeur sélectionné par ACS est rétropropagée. La récompense r des classeurs est mise à jour selon l'équation 3.3, où β_{rl} désigne le pas d'apprentissage, ρ_t la récompense reçue au temps t par le système et r_{cl}, la récompense du prochain classeur sélectionné par ACS (il s'agit donc d'un classeur dont la condition correspond à la situation perçue au temps $t+1$). L'adéquation d'un classeur à sa tâche d'apprentissage, sa *fitness*, est calculée comme le produit de sa qualité par sa prédiction de la récompense.

$$r \leftarrow \begin{cases} (1-\beta_{\mathrm{rl}})r + \beta_{\mathrm{rl}}.\rho_t & si\ \rho \neq 0 \\ (1-\beta_{\mathrm{rl}})r + \beta_{\mathrm{rl}}.r_{\mathrm{cl}} & sinon \end{cases} \tag{3.3}$$

3.1.5 Spécification des éléments inchangés

La spécification des éléments inchangés est une première extension à ACS introduite par (Stolzmann, 1999), afin de spécifier des attributs perceptifs qui ne changent pas (suite à la réalisation d'une action) dans les *Conditions* et les *Anticipations* des classeurs. Elle intervient quand des classeurs parviennent à correctement anticiper les changements environnementaux et que ceux-ci, en d'autres circonstances, ont également échoué à anticiper ces changements : la spécification des éléments inchangés a donc pour but de prévenir l'usage de ces classeurs dans les situations inadaptées à leur *Anticipation*.

Ce mécanisme est intégré dans l'*expected case* du processus d'apprentissage

par anticipation et concerne les classeurs qui ont été marqués. Il génère à partir de ces classeurs de nouveaux classeurs dont la *Condition* et l'*Anticipation* ont été spécifiées à partir d'une différence entre une perception de l'environnement et la marque de ces classeurs : si un attribut d'une marque n'est pas identique à l'attribut relatif dans une perception, les attributs associés dans la *Condition* et l'*Anticipation* sont précisés à partir de la perception de telle sorte que le nouveau classeur ne corresponde plus à la situation marquée.

L'*expected case* peut se séparer en deux sous-cas suite à l'intégration de la spécification des éléments inchangés :

- si un classeur entrant dans l'*expected case* ne possède pas de marque ou que sa marque ne présente pas de différences avec l'observation courante, sa qualité est augmentée ;
- sinon, la qualité de ce classeur est inchangée et l'une des différences, choisie aléatoirement, est utilisée pour générer un nouveau classeur.

La spécification des éléments inchangés permet ainsi de renforcer les capacités d'ACS à contextualiser les classeurs aux situations environnementales observées. La figure 3.7 reprend la figure de l'*expected case* de l'ALP pour illustrer une application de la spécification des éléments inchangés, où un classeur *cl'* est généré à partir de la marque du classeur *cl* et de la perception P_1.

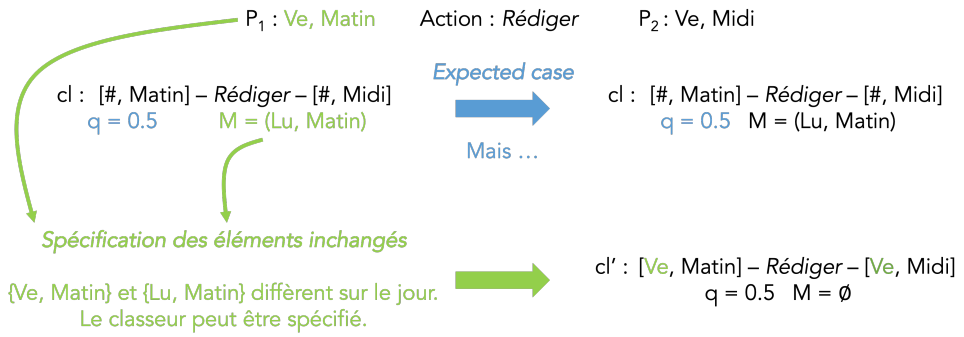


FIGURE 3.7 – Mécanisme de spécification des éléments inchangés intervenant lors de l'*expected case* du processus d'apprentissage par anticipation d'ACS. *Si d'autres différences sont présentes, seule l'une d'entre elles est aléatoirement choisie pour être utilisée. La spécification est réalisée différence après différence, afin d'éviter une surspécification des classeurs qui serait inutile pour les adapter à leur environnement.*

Après avoir introduit cette première extension à ACS, (Stolzmann, 1999) géné-

ralise les marques des classeurs, afin que celles-ci puissent inclure toutes les situations environnementales pour lesquelles l'*Anticipation* d'un classeur était incorrecte. Les éléments d'une marque ne consistent plus en un attribut perceptif, mais en un ensemble d'attributs perceptifs. La figure 3.8 illustre un exemple où la marque d'un classeur comporte plusieurs situations environnementales.

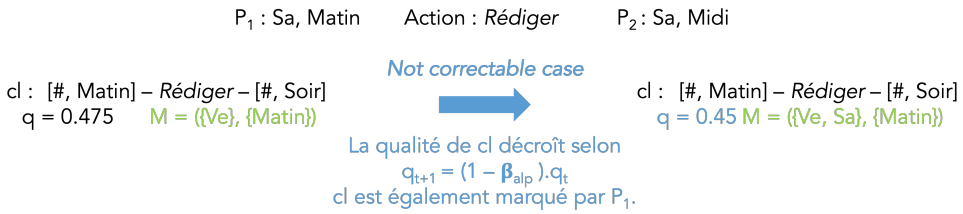


FIGURE 3.8 – Généralisation des marques des classeurs d'ACS.
Le classeur cl entre dans le not correctable case *de l'ALP puisqu'il n'anticipe pas le changement sur le moment du jour. La marque de cl est complétée avec les attributs de* P_1*, en particulier sur le jour.*

Puisque la spécification des éléments inchangés s'appuie sur les marques maintenant généralisées des classeurs, (Stolzmann, 1999) redéfinit ce qu'est une différence entre un attribut d'une perception et l'ensemble d'attributs correspondant de la marque généralisée : il est nécessaire que l'attribut de la perception ne soit pas inclus dans l'ensemble associé de la marque, à l'image de la figure 3.9.

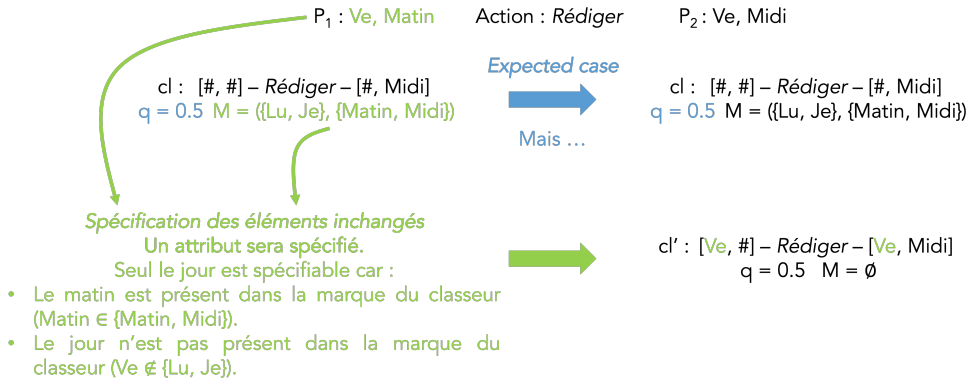


FIGURE 3.9 – Spécification des éléments inchangés avec les marques généralisées.
La marque généralisée de cl ne comporte qu'une seule différence avec la perception P_1 *sur le jour. Un classeur cl' peut être construit par la spécification des éléments inchangés à partir de cl en précisant le jour dans sa* Condition *et son* Anticipation.

3.1.6 Séquences comportementales

(Stolzmann, 1999) introduit ensuite une seconde extension permettant à ACS de réaliser sa tâche d'apprentissage dans des environnements où le système est confronté au problème d'aliasing perceptif. Le problème d'aliasing perceptif est une forme d'incertitude où un système ne peut différencier deux situations environnementales uniquement à partir de sa perception de ces situations. Nous revenons plus en détail sur le problème d'aliasing perceptif dans la section 4.1.3 du chapitre 4.

Les séquences comportementales ont pour but d'éviter qu'ACS ne se retrouve dans une situation où il ne peut pas apprendre quelle est l'action à réaliser pour résoudre sa tâche d'apprentissage. ACS construit en ce sens de nouveaux classeurs dont le *Conséquent* consiste en une chaîne d'actions devant leur permettre de traverser ces situations particulières. Son extension est alors composée de trois parties où :

1. ACS devient capable de détecter les perceptions liées au PAI et pouvant correspondre à plusieurs situations environnementales ;
2. une séquence comportementale est introduite dans le *Conséquent* d'un nouveau classeur, celui-ci étant conditionné à la détection précédente ;
3. un mécanisme de contrôle des séquences limite l'usage de chaînes d'actions qui n'entraînent pas de changements perceptifs.

Détection du problème d'aliasing perceptif

Avec le PAI, si deux situations environnementales distinctes sont perçues avec une même observation, l'emploi d'un classeur s'appariant à cette observation peut entraîner différents changements perceptifs antagonistes. Or, les classeurs d'ACS ne sont pas capables de décrire différents changements perceptifs dans leur *Anticipation*. Le problème d'aliasing perceptif implique alors que ces classeurs réussissent et échouent à anticiper les conséquences de leurs actions pour ces observations.

La détection proposée par (Stolzmann, 1999) s'appuie sur ce constat : elle identifie ces observations par le biais de la spécification des éléments inchangés. La spécification des éléments inchangés intervient dans l'*expected case*

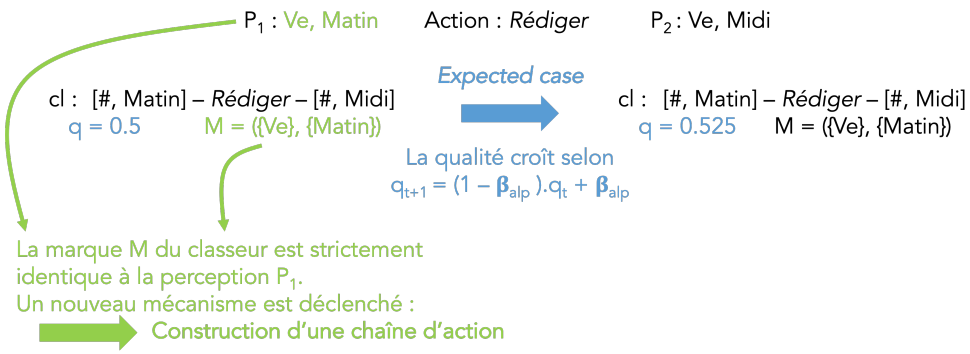


FIGURE 3.10 – Détection des perceptions déclenchant la construction des chaînes d'actions dans l'*Anticipatory Classifier System*.

de l'ALP, quand des classeurs ont anticipé avec succès les changements environnementaux, mais qu'ils ont échoué à anticiper ces changements en d'autres occasions. De tels classeurs sont marqués par les observations de leur environnement en ces occasions. Si la marque de ces classeurs correspond à l'observation courante de l'environnement, pour laquelle ces classeurs ont correctement puisqu'ils atteignent *expected case*, alors cette observation peut être associée au problème d'aliasing perceptif et la spécification des éléments inchangés ne peut être appliquée (*cf.* figure 3.10).

Construction des séquences comportementales

Si la détection du PAI est positive, un classeur possédant une séquence comportementale (*Behavioral Sequences (BSeq)*) est construit à partir du classeur cl_{t-1} qui a correctement anticipé l'état $s(t)$ depuis l'état $s(t-1)$ et de l'avant-dernier classeur cl_{t-2} sélectionné dans l'état précédent $s(t-2)$. Ces deux classeurs permettent la création d'un classeur avec une BSeq cl_{new}, potentiellement capable d'anticiper la situation $s(t)$ à partir de la situation $s(t-2)$, lorsque les actions décrites dans cl_{t-2} et cl_{t-1} sont réalisées à la suite (*cf.* figure 3.11) :

- le *Conséquent* de cl_{new} est construit à partir du chaînage des actions décrites par cl_{t-2}, puis avec celles de cl_{t-1} ;
- la *Condition* de cl_{new} est construite avec l'opérateur *passthrough* à partir des *Conditions* de cl_{t-1} et de cl_{t-2} selon *passthrough*($C_{\text{cl}_{t-1}}, C_{\text{cl}_{t-2}}, L$), où L désigne la longueur des *Conditions* ;
- l'*Anticipation* de cl_{new} est construite avec l'opérateur *passthrough* à

partir des *Anticipations* de cl_{t-2} et de cl_{t-1} selon *passthrough*($E_{\mathrm{cl}_{t-2}}$, $E_{\mathrm{cl}_{t-1}}$, L), où L désigne la longueur des *Anticipations*.

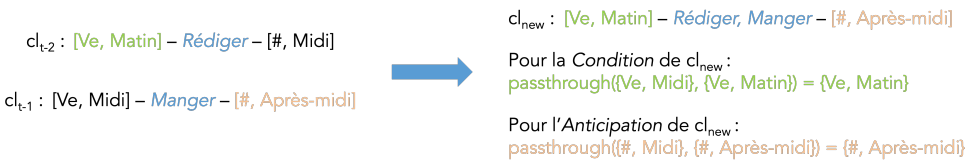


FIGURE 3.11 – Construction d'un classeur à séquence comportementale dans l'*Anticipatory Classifier System*.

Lorsqu'un classeur à BSeq est construit à partir de cl_{t-1} et de cl_{t-2}, ces deux classeurs peuvent eux-mêmes être des classeurs avec des BSeq. Le nombre d'actions inclus dans une séquence peut alors croître indéfiniment s'il n'est pas limité. Un nouveau paramètre $bs_{\max}$ est alors ajouté à ACS : si la longueur de la nouvelle séquence comportementale d'un classeur dépasse ce seuil $bs_{\max}$, il est abandonné. Un nouveau classeur à BSeq est ajouté à la population seulement s'il n'existe pas déjà dans la population, sinon sa qualité est augmentée comme si son anticipation était correcte.

Contrôle des séquences comportementales

ACS réalise son apprentissage à partir des changements environnementaux qu'il perçoit. Il favorise alors les classeurs capables de provoquer ces changements. L'emploi des séquences comportementales peut impliquer l'usage d'actions pouvant mutuellement s'annuler : il suffit d'imaginer une chaîne d'actions où ACS réaliserait un pas en avant, puis un pas en arrière.

Lorsqu'un classeur à BSeq est sélectionné par ACS pour interagir avec son environnement, le système garde temporairement en mémoire les observations intermédiaires et les compare entre elles. Si ACS perçoit une situation qu'il a déjà perçue lors de l'emploi de la séquence comportementale, alors il interrompt son interaction avec l'environnement et décroît directement la qualité du classeur à BSeq qui avait été sélectionné. Les processus de découverte de règles prennent ensuite place à partir de la dernière observation de l'environnement, que l'exécution de la chaîne ait été avortée ou non, puis le cycle d'apprentissage d'ACS continu.

3.2 Vers ACS 2

Les capacités d'apprentissage d'ACS ont évolué suite à l'introduction de nouvelles extensions et à l'évolution de composants préexistants, introduisant ACS 2 (Butz and Stolzmann, 2001). Deux premières différences peuvent déjà être notées entre ACS et ACS 2 :

- les séquences comportementales n'ont pas été intégrées ou adaptées à ACS 2 ;
- la liste des messages est supprimée du cycle d'apprentissage d'ACS 2, le système ne gardant que les deux dernières observations de son environnement en mémoire.

3.2.1 Sélection des actions et biais exploratoires

Les processus d'apprentissage d'ACS 2 sont appliqués à l'échelle d'une sous-population de classeurs, au lieu d'un unique classeur comme ACS. Cette modification nécessite de choisir une action pour une situation environnementale donnée, pour construire l'ensemble d'action qui comporte tous les classeurs décrivant l'action choisie et correspondant à cette situation.

La sélection d'action est réalisée par le biais de l'algorithme *ϵ-greedy policy*. Une action est choisie aléatoirement selon une probabilité ϵ ou bien, une action est choisie à partir des classeurs de l'ensemble d'appariement $[M]$ selon deux critères :

1. le classeur anticipe des changements perceptifs ;
2. le classeur possède la valeur de *fitness* la plus élevée.

Ce premier critère a pour but de favoriser les actions qui amènent à des changements environnementaux. Or, ACS 2 est capable de se doter et de maintenir des classeurs n'anticipant aucun changement (*cf.* section 3.2.2), contrairement à ACS qui les supprime de sa population *via* le *useless case*. Ce critère est nécessaire à ACS 2 afin de suivre l'idée que les actions du système ont des conséquences sur son environnement, alors qu'il n'est pas utile pour ACS.

L'*ϵ-greedy policy* est préférée à la *roulette wheel selection* car cette politique de sélection est algorithmiquement moins coûteuse, sans impacter négative-

ment les capacités d'apprentissage du système : une seule lecture des *fitness* des classeurs de l'ensemble d'appariement suffit à sélectionner une action.

Deux biais exploratoires ont également été introduits dans ACS 2 afin d'accélérer l'acquisition d'une représentation de l'environnement : le biais de rappel et le biais d'erreur (Butz, 2001). Le biais de rappel favorise la sélection d'une action ayant été la moins récemment utilisée par le système. Le biais d'erreur préfère les actions pour lesquelles les conséquences sont les moins connues : autrement dit, celles dont la qualité des classeurs est la plus faible.

Pour mettre en place le biais de rappel, un nouvel attribut est ajouté dans les classeurs : il s'agit d'une marque temporelle t_{alp} qui indique la dernière fois que le classeur est passé dans le processus d'apprentissage par anticipation. Cette marque est mise à jour au passage d'un classeur dans l'ALP, grâce à un compteur de pas de temps.

Pour mettre en place le biais d'erreur, la qualité moyenne Q_a des anticipations associées pour une action a est calculée selon l'équation 3.4. L'action qui est sélectionnée par le système avec ce biais est celle dont la qualité moyenne Q_a est minimale. Les qualités moyennes Q_a sont calculées pour chaque action possible. Avec ACS 2, des classeurs peuvent exister en plusieurs exemplaires, ce qui est décrit par la numérosité *num* des classeurs : un classeur, ou macro-classeur, peut exister en *num* exemplaires appelés micro-classeurs. Ce nouvel attribut *num* est mis à jour lorsqu'un nouveau classeur est ajouté dans la population par le biais des processus de découverte de nouvelles règles. ACS 2 adopte une nouvelle approche dont le but est de mieux contrôler l'insertion de nouveaux classeurs, en particulier dans les ensembles d'action, que nous abordons dans la section 3.2.5.

$$Q_a = \frac{\sum_{cl \in [M] \text{and } A_{\text{cl}}=a} q_{\text{cl}}.num_{\text{cl}}}{\sum_{cl \in [M] \text{and } A_{\text{cl}}=a} num_{\text{cl}}} \tag{3.4}$$

Ces deux biais interviennent quand le système tente de sélectionner une action aléatoirement selon une probabilité p_b. Puisque ces deux biais ont le même objectif, ils ont la même probabilité d'être sélectionné. Ainsi, si une action doit être choisie aléatoirement avec l'*ϵ-greedy policy* :

— une action est sélectionnée aléatoirement selon une probabilité $1-p_b$;
— le biais de rappel intervient avec une probabilité $p_b/2$;
— le biais d'erreur intervient avec une probabilité $p_b/2$.

La figure 3.12 schématise le processus de sélection d'action d'ACS 2.

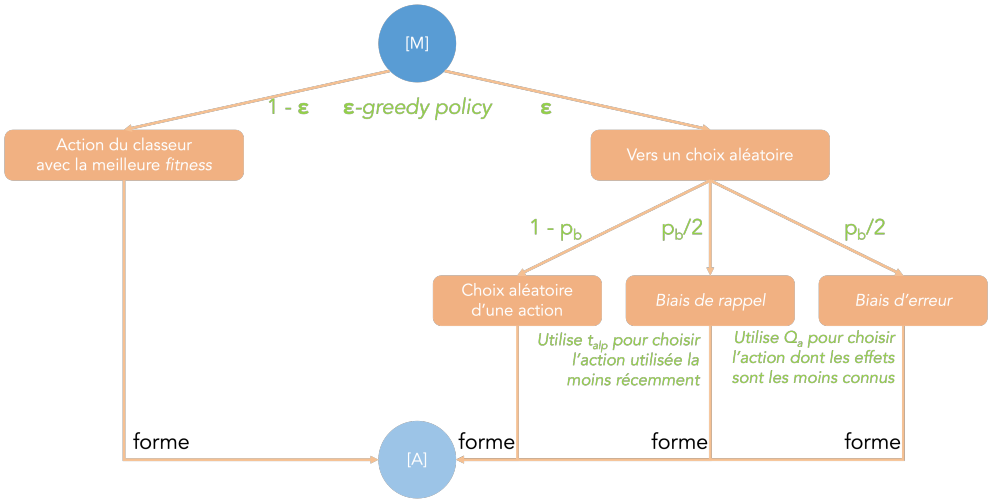


FIGURE 3.12 – Sélection d'une action dans ACS 2.

3.2.2 Processus d'apprentissage par anticipation

Une première modification est réalisée afin de préserver la sémantique des *Anticipations* des classeurs. Un classeur anticipe correctement une situation si seuls les changements environnementaux sont décrits dans son anticipation, et que ceux-ci correspondent à la situation à anticiper. Si un attribut perceptif est décrit dans une *Anticipation*, celui-ci indique obligatoirement un changement perceptif et doit donc être différent de l'attribut perceptif correspondant dans la *Condition*. Suite à cette modification, la spécification des éléments inchangés ne spécifie plus les attributs des *Anticipations*, mais uniquement les attributs des *Conditions* des classeurs.

L'ALP met ensuite à jour trois nouveaux attributs au sein des classeurs, en plus de leur *Condition*, de leur *Anticipation*, de leur qualité et de leurs marques :

— la marque temporelle t_{alp} introduite dans la section précédente ;
— l'expérience *exp* d'un classeur qui désigne le nombre de fois où un classeur est intervenu dans l'ALP ;

— la fréquence *aav* de mise à jour d'un classeur (*application average*) qui estime la fréquence à laquelle le classeur est mis à jour par l'ALP ou autrement dit, fait partie de l'ensemble d'action $[A]$.

L'expérience *exp* d'un classeur est incrémentée de 1 lorsque celui-ci entre dans l'ALP. Cette valeur permet de savoir si un classeur a été éprouvé par ACS 2 : un classeur est considéré comme expérimenté si son expérience dépasse un seuil θ_{exp} défini en amont de l'apprentissage. L'expérience est utilisée non seulement pour mettre à jour la fréquence *aav* et aussi lorsqu'un classeur est inséré dans la population (*cf.* section 3.2.5).

La fréquence *aav* de mise à jour d'un classeur est calculée selon l'équation 3.5 (Venturini, 1994), à partir de son expérience *exp*, de sa marque temporelle t_{alp}, du pas d'apprentissage β_{alp} de l'ALP et du compteur de pas de temps t. Cette équation assure une adaptation rapide de la fréquence dès qu'un classeur est créé et assure sa mise à jour au cours de l'apprentissage. Cette fréquence est en particulier utilisée par un nouveau processus de découverte de nouvelles règles, le mécanisme de généralisation génétique, qui est présenté dans la section 3.2.4.

$$aav_{\text{cl}} \leftarrow \begin{cases} aav_{\text{cl}} + \frac{t - t_{\text{alp}_{\text{cl}}} - aav_{\text{cl}}}{exp_{\text{cl}}} & si\ exp_{\text{cl}} < 1/\beta_{\text{alp}} \\ aav_{\text{cl}} + \beta_{\text{alp}}(t - t_{\text{alp}_{\text{cl}}}) & sinon \end{cases} \tag{3.5}$$

Pour permettre à ACS 2 de faire évoluer des classeurs n'anticipant aucun changement perceptif, le processus d'apprentissage par anticipation ne comporte plus de *useless case*. L'ALP est alors constitué de l'*expected case* qui peut faire intervenir la spécification des éléments inchangés, du *correctable case* et du *not correctable case*. Puisque ACS 2 utilise les ensembles d'action $[A]$, son ALP est appliqué pour chacun des classeurs de cet ensemble, en lieu et place d'un unique classeur.

Une dernière modification concerne la population initiale de classeurs qui est maintenant vide. Pour assurer qu'il existe un classeur pour chaque situation environnementale, un processus de recouvrement est ajouté à ACS 2 (*cf.* chapitre 2 section 2.2.4). Le recouvrement intervient si un ensemble d'action construit par le système est vide, ou si aucun classeur d'un ensemble $[A]$ n'a réussi à anticiper la transition environnementale courante. Le cas échéant,

un classeur est généré afin de décrire la transition environnementale perçue par le système, à l'image de la figure 3.13 :

- le *Conséquent* de ce nouveau classeur décrit l'action choisie par le système et pour laquelle $[A]$ est vide.
- pour chaque différence entre les deux observations, la *Condition* et l'*Anticipation* du classeur sont spécifiées avec ces différences.
- les autres attributs du classeur sont initialisés à leur valeur par défaut, ou prennent pour valeur le pas de temps courant.

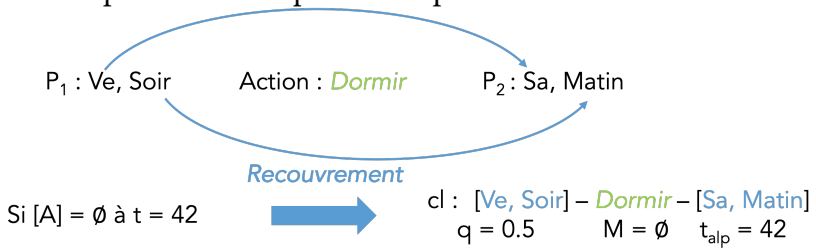


FIGURE 3.13 – Processus de recouvrement d'ACS 2 déclenché si par exemple, un ensemble d'action est vide.

Tous les classeurs générés lors du processus d'apprentissage par anticipation ne sont pas directement insérés dans la population de classeurs ou dans l'ensemble d'action : un classeur nouvellement construit peut déjà exister dans la population. L'insertion des classeurs est détaillée dans la section 3.2.5.

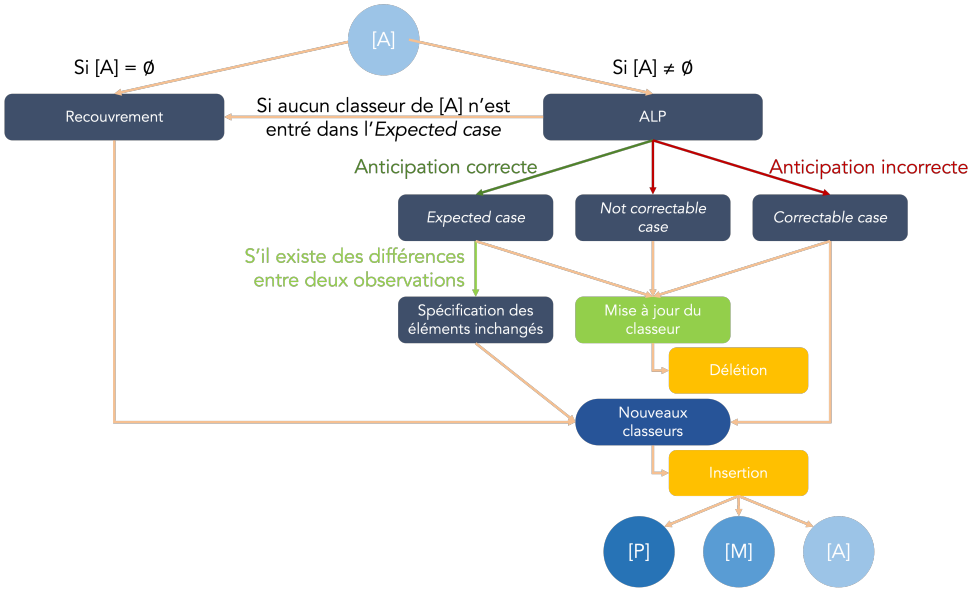


FIGURE 3.14 – Processus d'apprentissage par anticipation d'ACS 2.

La figure 3.14 schématise l'ensemble du processus d'apprentissage par anticipation d'ACS 2.

3.2.3 Spécification des éléments inchangés

La spécification des éléments inchangés est modifiée en introduisant deux types de différences (*cf.* figure 3.15) :

— il y a une différence nette si un attribut perceptif de la situation environnementale n'appartient pas à l'ensemble associé d'attributs de la marque généralisée ;

— il y a une différence floue si un attribut perceptif de la situation environnementale appartient à l'ensemble associé d'attributs de la marque généralisée, ce dernier contenant également d'autres attributs (autrement, ils seraient identiques).

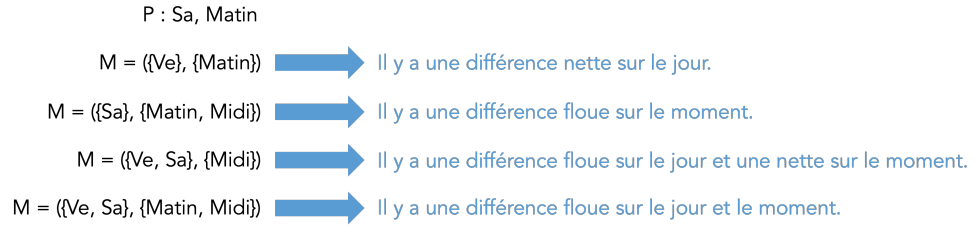


FIGURE 3.15 – Différences nettes et floues entre différentes marques pour une perception donnée.

La spécification des éléments inchangés utilise ces deux types différences quand elle peut générer un nouveau classeur. Elle favorise d'abord l'usage de l'une des différences nettes choisie aléatoirement pour spécifier la *Condition* d'un nouveau classeur. Elle n'utilise les différences floues que s'il n'y a pas de différences nettes entre la marque d'un classeur et la perception courante. Si elle doit utiliser les différences floues, toutes celles-ci sont utilisées pour indiquer les attributs perceptifs à spécifier dans la *Condition* d'un nouveau classeur, à l'image de la figure 3.16.

Un nouveau paramètre $u_{\max}$ est également introduit : il contrôle le nombre maximal d'attributs spécifiables dans les *Conditions* des classeurs générés par la spécification des éléments inchangés. Si le nombre d'attributs spécifiés atteint ce nombre $u_{\max}$, il est nécessaire de généraliser des attributs spécifiés d'un classeur par le symbole générique #, ou d'abandonner des attributs qui seraient à spécifier à partir d'une différence nette ou de plusieurs différences floues. La spécification des éléments inchangés fonctionne selon l'algorithme 3.2.1, qui prend en argument le classeur *cl* arrivant à cette étape de l'apprentissage, des différences floues ou une différence nette *diff* et $u_{\max}$.

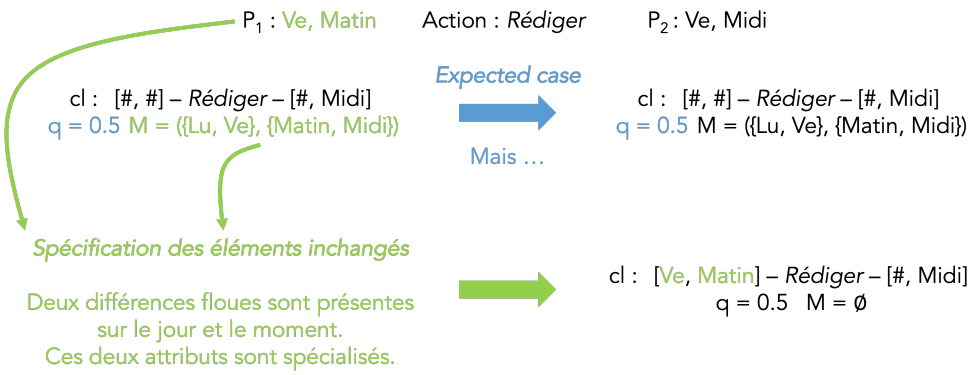


FIGURE 3.16 – Utilisation des différences floues par le mécanisme de spécification des éléments d'ACS.

Algorithme 3.2.1 Spécification des éléments inchangés d'ACS 2

```
1 : function SUC(cl, diff, u_max, t)
2 :     spe_cl ← number of specialized attributes in Condition of cl
3 :     spe_diff ← number of specialized attributes in diff
4 :     child ← copy of cl
5 :     if spe_cl ≥ u_max then
6 :         Randomly remove specific attribute in child's Condition
7 :         spe_cl ← spe_cl − 1
8 :         while spe_cl + spe_diff > u_max do
9 :             if spe_cl > 0 and random number in [0, 1[ < 0.5 then
10 :                Randomly remove specific attribute in child's Condition
11 :                spe_cl ← spe_cl − 1
12 :            else
13 :                Randomly remove specific attribute in diff
14 :                spe_diff ← spe_diff − 1
15 :            end if
16 :        end while
17 :    else
18 :        while spe_cl + spe_diff > u_max do
19 :            Randomly remove attribute in diff
20 :            spe_diff ← spe_diff − 1
21 :        end while
22 :    end if
23 :    Specify child with diff
24 :    Initialize other attributes of child such as exp, t_alp
25 :    return child
```

3.2.4 Généralisation génétique

Le mécanisme de généralisation génétique a été introduit par (Butz et al., 2002). Il permet de contre-balancer la pression de spécification induite par processus d'apprentissage par anticipation, afin de faire apparaître dans les *Conditions* des classeurs les *points d'attention* liés à l'usage de ces classeurs. Autrement dit, il est possible d'ignorer les attributs conditionnels qui ne sont pas spécifiés, ceux-ci ne caractérisant pas les transitions environnementales décrites. Couplée aux précédents ajouts, la généralisation génétique permet à ACS 2 de se doter de classeurs capables de décrire un plus grand nombre de situations environnementales pour lesquelles la réalisation d'une action mène aux mêmes conséquences.

Deux pressions antagonistes sont ainsi incluses dans ACS 2, lui permettant de trouver un équilibre dans la construction des représentations conditionnelles des classeurs. Ces représentations étant plus compactes, elles sont plus facilement manipulables, tant pour être réutilisées par le système en de nouvelles situations environnementales, que pour être expliquées.

Boucle génétique

A l'image des algorithmes génétiques desquels il est issu, ce mécanisme s'appuie sur une sélection de classeurs de l'ensemble d'action, où leurs *Conditions* sont modifiées par des opérateurs de croisement et de mutation, afin de générer de nouveaux classeurs. Le choix de classeurs dans l'ensemble d'action $[A]$ permet de s'assurer que les classeurs construits par les opérations génétiques correspondent toujours aux transitions environnementales décrites par $[A]$.

Contrairement à de nombreux autres systèmes de classeurs, ce mécanisme de découverte de règles n'est pas le principal élément permettant à ACS 2 (ou aux ALCS) de construire sa population de classeurs, puisqu'il s'agit de l'ALP. Son usage est régulé par un paramètre θ_{ga} qui caractérise la fréquence à laquelle ce mécanisme doit être utilisé dans un ensemble d'action. La généralisation génétique fonctionne selon l'algorithme 3.2.2 où chacune des étapes est décrite en détail dans les paragraphes suivants, à l'exception de l'insertion de nouveaux classeurs qui est décrite dans la section 3.2.5. L'algorithme

3.2.2 prend en argument :

- les ensembles de classeurs $[A]$, $[M]$ et $[P]$, afin de pouvoir insérer les classeurs nouvellement créés ;
- la fréquence d'application du mécanisme de généralisation θ_{ga} ;
- le compteur de pas de temps t ;
- les taux de croisement χ et de mutation μ, respectivement utilisés par les opérateurs de croisement et de mutation.

Algorithme 3.2.2 Mécanisme de généralisation génétique d'ACS 2

```
1 : function ApplyGeneticGeneralization([A], [M], [P], θ_ga, χ, μ, t)
2 :     if Genetic generalization should be applied according to θ_ga then
3 :         Update t_ga of all cl of [A]
4 :         Select two classifiers cl_1 and cl_2 in [A]
5 :         child_1 ← copy of cl_1 ; child_2 ← copy of cl_2
6 :         Initialize other attributes of child_1 and child_2
7 :         Mutate child_1 ; Mutate child_2
8 :         if random number in [0, 1[ < χ then
9 :             Crossover between child_1 and child_2
10 :            Update qualities and rewards of child_1 and child_2
11 :        end if
12 :        Update qualities of child_1 and child_2
13 :        Delete classifiers in [A] to enable insertion of children in [A]
14 :        if child_1's Condition is specialized by at least one attribute then
            ▷ From alg 3.2.7
15 :            InsertFromGA(child_1, θ_exp, θ_r, [A], [M], [P])
16 :        end if
17 :        if child_2's Condition is specialized by at least one attribute then
            ▷ From alg 3.2.7
18 :            InsertFromGA(child_2, θ_exp, θ_r, [A], [M], [P])
19 :        end if
20 :    end if
```

Fréquence d'application

La généralisation génétique est applicable sur un ensemble d'action si le délai moyen depuis la dernière application de la généralisation sur cet ensemble est supérieur à la fréquence d'application θ_{ga}. Ce délai est calculé à partir d'un marqueur temporel t_{ga} ajouté à chaque classeur de la population et pondéré par la numérosité *num* des classeurs. t_{ga} est mis à jour avec le compteur de pas de temps t dès que la généralisation génétique a été appliquée sur un

classeur. L'équation 3.6 décrit le calcul de ce délai et sa comparaison à θ_{ga}, à partir des classeurs de l'ensemble d'action $[A]$, le compteur de pas de temps t et la fréquence θ_{ga}.

$$t - \frac{\sum_{cl \in [A]} t_{\text{ga}_{\text{cl}}}.num_{\text{cl}}}{\sum_{cl \in [A]} num_{\text{cl}}} > \theta_{\text{ga}} \tag{3.6}$$

Sélection des classeurs *parents*

La *Roulette Wheel Selection* est utilisée pour sélectionner les deux classeurs à partir desquels de nouveaux classeurs sont générés. Cette sélection s'appuie sur la qualité q des classeurs élevée au cube, afin de creuser les écarts entre des classeurs dont les qualités sont proches. Un même classeur peut également être choisi deux fois. L'algorithme 3.2.3 décrit la sélection d'un classeur d'un ensemble d'action $[A]$.

Algorithme 3.2.3 Sélection d'un classeur *parent* dans $[A]$ dans ACS 2

```
1 : function SelectClassifier([A])
2 :     q_total ← 0
3 :     for each cl ∈ [A] do
4 :         q_total ← q_total + q_cl^3
5 :     end for
6 :     rand ← q_total× random number in [0, 1[
7 :     q_total ← 0
8 :     for each cl ∈ [A] do
9 :         q_total ← q_total + q_cl^3
10 :        if rand < q_total then
11 :            return cl
12 :        end if
13 :    end for
```

Mutation d'un classeur

L'opération de mutation consiste à transformer selon une probabilité μ chacun des attributs spécifiés d'une *Condition* en un symbole générique #. Pour rappel, un attribut généralisé décrit par # peut correspondre à toutes les valeurs possibles pour cet attribut perceptif. Si un attribut d'une *Condition* d'un classeur est généralisé alors que l'attribut correspondant est spécifié

dans son *Anticipation*, il est sous-entendu que le symbole générique ne prendra pas la valeur spécifiée dans l'*Anticipation* du classeur, sans quoi les anticipations deviendraient incorrectes : # correspond alors à tous les symboles possibles d'un attribut perceptif à l'exception de ceux décrits par des changements dans l'attribut correspondant d'une *Anticipation*.

Croisement entre classeurs

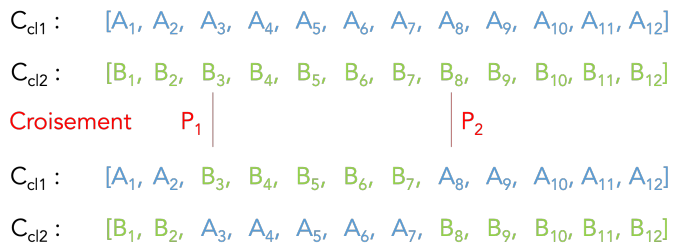


FIGURE 3.17 – Illustration d'un croisement en deux points dans les *Conditions* de deux classeurs.
Deux points de croisement P_1 et P_2 ont été sélectionnés aléatoirement afin d'intervertir les attributs des Conditions *des classeurs cl_1 et cl_2. Cette interversion est réalisé à partir du point P_1 jusqu'au point P_2 exclu.*

L'opération de croisement est réalisée selon une probabilité χ à partir de deux classeurs dont les *Anticipations* sont identiques, afin de s'assurer que ceux-ci décrivent les mêmes transitions environnementales. Si elles sont identiques, un croisement en deux points est opéré : deux attributs de croisement sont sélectionnés au hasard, afin d'intervertir tous les attributs entre les deux classeurs compris entre ces points de croisement (le premier étant inclus et le second exclus) tel qu'illustré sur la figure 3.17.

Suppression de classeurs

Afin de réguler l'expansion de la population de classeurs d'ACS 2, la taille de l'ensemble d'action $[A]$ est contrainte par un paramètre θ_{as}. Le nombre de micro-classeurs composant $[A]$ ne peut pas excéder θ_{as} lorsque le mécanisme de généralisation génétique tente d'ajouter des micro-classeurs à cet ensemble. Si le seuil θ_{as} devait être atteint lors de l'insertion de nouveaux classeurs par le mécanisme de généralisation génétique, il est nécessaire de supprimer des micro-classeurs de $[A]$, voire un (macro-)classeur lorsque son

attribut *num* devient nul. Un classeur complètement supprimé de $[A]$ est supprimé de $[P]$ et aussi de $[M]$ s'il appartient à cet ensemble.

Algorithme 3.2.4 Suppression de n micro-classeurs dans $[A]$ dans ACS 2

```
1 : function DeleteMicroClassifiers([A], [M], [P], n)
2 :     while n + Σ_{cl∈[A]} num_cl > θ_as do
3 :         μcl_del ← ∅
4 :         for each micro classifier μcl ∈ [A] do
5 :             if random number in [0, 1[ < 1/3 then
6 :                 if μcl_del == ∅ then
7 :                     μcl_del ← μcl
8 :                 else
9 :                     if q_μcl_del − q_μcl > 0.1 then
10 :                        μcl_del ← μcl
11 :                    end if
12 :                    if |q_μcl_del − q_μcl| ≤ 0.1 then
13 :                        if M_μcl_del == ∅ and M_μcl ≠ ∅ then
14 :                            μcl_del ← μcl
15 :                        else if aav_μcl_del < aav_μcl and
                                    (M_μcl_del == ∅ or M_μcl ≠ ∅) then
16 :                            μcl_del ← μcl
17 :                        end if
18 :                    end if
19 :                end if
20 :            end if
21 :        end for
22 :        if μcl_del ≠ ∅ then
23 :            if num_μcl_del > 1 then
24 :                num_μcl_del ← num_μcl_del − 1
25 :            else
26 :                Remove μcl_del from [A], [M] and [P]
27 :            end if
28 :        end if
29 :    end while
```

La suppression de ces micro-classeurs est décrite par l'algorithme 3.2.4 où n désigne le nombre de classeurs construits par le mécanisme de généralisation génétique et le nombre de classeurs à supprimer de $[A]$, $[M]$ et $[P]$. Les micro-classeurs à supprimer sont sélectionnés à la manière d'un processus de sélection par tournoi (Miller et al., 1995) : des individus sont aléatoirement choisis dans une population afin de les mettre en compétition pour déterminer ceux devant être sélectionnés. Ici, en moyenne, un tiers des micro-

classeurs de l'ensemble d'action entrent en compétition, afin de déterminer des micro-classeurs à supprimer. Quand un micro-classeur μcl_{del} marqué pour suppression est en compétition avec un autre micro-classeur μcl :

- μcl est préféré pour suppression si sa qualité est significativement plus faible que celle de μcl_{del}
- s'ils ont une qualité similaire, μcl est préféré pour suppression s'il est marqué alors que μcl_{del} ne l'est pas ;
- s'ils ont une qualité similaire et que la fréquence *aav* de μcl est plus grande que celle de μcl_{del}, μcl est préféré pour suppression s'il a une marque ou si μcl_{del} n'a pas de marque.

3.2.5 Subsomption et ajout de classeurs

Subsomption

La subsomption d'ACS 2 intervient dans les processus de découverte de nouvelles règles afin de contrôler l'ajout de classeurs dans les ensembles d'action. Les processus de découverte de nouvelles règles sont l'ALP et le mécanisme de généralisation génétique. La subsomption cherche à identifier si les transitions environnementales décrites par un classeur nouvellement créé sont déjà décrites par un classeur existant dans les ensembles d'action. Puisque la subsomption opère sur des ensembles d'action, les classeurs correspondent au moins à la même situation environnementale et à une même action représentée dans leur *Conséquent* : c'est pourquoi seules les généralités des *Conditions* sont comparées et qu'il n'y a pas de comparaison sur les *Conséquents*. Un classeur cl_1 est alors subsumé par un autre classeur cl_2 si (*cf.* algorithme 3.2.5) :

- la *Condition* de cl_2 est plus générale que celle de cl_1 (ou autrement dit, si les attributs génériques # sont plus nombreux) ;
- les *Anticipations* de cl_1 et cl_2 sont identiques, indiquant qu'ils décrivent au moins la même transition environnementale ;
- cl_2 n'est pas marqué ($M_{\text{cl}_2} = \varnothing$) ;
- cl_2 est un classeur fiable ($q_{\text{cl}_2} > \theta_{\text{r}}$) ;
- cl_2 est un classeur expérimenté ($exp_{\text{cl}_2} > \theta_{\text{exp}}$).

Algorithme 3.2.5 Subsomption de classeurs dans les ensembles d'action d'ACS 2

```
1 : function IsSubsumed(cl_1, cl_2, θ_exp, θ_r)
2 :     if number of # in C_cl2 ≥ number of # in C_cl1 then
3 :         if E_cl1 == E_cl2 then
4 :             if M_cl2 == ∅ then
5 :                 if q_cl2 > θ_r then
6 :                     if exp_cl2 > θ_exp then
7 :                         return True
8 :                     end if
9 :                 end if
10 :            end if
11 :        end if
12 :    end if
13 :    return False
```

Ajout de classeurs à partir de l'ALP

Un classeur créé par l'ALP est ajouté dans un ensemble d'action (et la population de classeurs) uniquement si (*cf.* algorithme 3.2.6) :

- il n'existe pas un classeur le subsumant ;
- il existe un autre classeur dont la *Condition* et l'*Anticipation* sont identiques à celui-ci.

Un classeur à insérer peut également être ajouté dans l'ensemble d'appariement $[M]$ si sa condition correspond à la dernière observation de l'environnement, parce que cet ensemble $[M]$ est construit avant que les différents processus de découverte de règles interviennent. Si un classeur préexistant cl_{old} prévient l'insertion d'un nouveau classeur cl dans un ensemble d'action, la qualité de cl_{lod} est augmentée de la même manière que si son anticipation est correcte, puis cl est abandonné.

Algorithme 3.2.6 Insertion de classeurs à partir de l'ALP d'ACS 2

```
1 : function InsertFromALP(cl, β_alp, θ_exp, θ_r, [A], [M], [P])
2 :     cl_old ← ∅
3 :     for each cl_A ∈ [A] do
        ▷ From alg 3.2.5
4 :         if IsSubsumed(cl, cl_A, θ_exp, θ_r) then
5 :             if cl_old == ∅ or C_cl_A is more general than C_cl_old then
6 :                 cl_old ← cl_A
7 :             end if
8 :         end if
9 :     end for
10 :    if cl_old == ∅ then
11 :        for each cl_A ∈ [A] do
12 :            if C_cl_A = C_cl and E_cl_A == E_cl then
13 :                cl_old ← cl_A
14 :            end if
15 :        end for
16 :    end if
17 :    if cl_old == ∅ then
18 :        Insert cl in [A] and [P]
19 :        Insert cl in [M] if cl matches last observation
20 :    else
21 :        q_cl_old ← (1 − β_alp) q_cl_old + β_alp
22 :        Discard cl
23 :    end if
```

Ajout de classeurs à partir du processus de généralisation génétique

Un classeur créé par le processus de généralisation génétique est ajouté à la population de classeurs de façon analogue à l'ajout depuis l'ALP (*cf.* algorithme 3.2.7). Cet ajout diffère du précédent sur un principal élément : s'il existe un classeur cl_{old} qui prévient l'insertion d'un nouveau classeur cl, la qualité de cl_{old} n'est pas modifiée, la numérosité de cl_{old} est augmentée de 1 s'il n'est pas marqué et cl est abandonné.

Algorithme 3.2.7 Insertion de classeurs depuis la généralisation génétique d'ACS 2

```
1 : function InsertFromGA(cl, θ_exp, θ_r, [A], [M], [P])
2 :     cl_old ← ∅
3 :     for each cl_A ∈ [A] do
        ▷ From alg 3.2.5
4 :         if IsSubsumed(cl, cl_A, θ_exp, θ_r) then
5 :             if cl_old == ∅ or C_cl_A is more general than C_cl_old then
6 :                 cl_old ← cl_A
7 :             end if
8 :         end if
9 :     end for
10 :    if cl_old == ∅ then
11 :        for each cl_A ∈ [A] do
12 :            if C_cl_A == C_cl and E_cl_A == E_cl then
13 :                cl_old ← cl_A
14 :            end if
15 :        end for
16 :    end if
17 :    if cl_old == ∅ then
18 :        Insert cl in [A] and [P]
19 :        Insert cl in [M] if cl matches last observation
20 :    else
21 :        if M_cl_old == ∅ then
22 :            num_cl_old ← num_cl_old + 1
23 :        end if
24 :        Discard cl
25 :    end if
```

3.2.6 Processus de rétribution

La récompense r des classeurs s'inspire du Q-Learning de (Watkins and Dayan, 1992) et est mise à jour selon l'équation 3.7 où :

- β_{rl} désigne le pas d'apprentissage ;
- ρ_{t-1} la récompense reçue à l'instant $t-1$ par le système ;
- γ le facteur d'actualisation compris entre 0 et 1 qui valorise les récompenses obtenues sur le court terme ou sur le long terme ;
- les classeurs appartenant à l'ensemble d'appariement $[M]_t$ qui sont ceux dont la condition correspond à la situation perçue au temps t ;
- le nombre L d'attributs perceptifs contenus dans les *Conditions* et les *Anticipations* des classeurs.

La *fitness* est aussi dans ACS 2 le produit de la qualité q d'un classeur par la récompense estimée r. Puisqu'ACS 2 fait évoluer des classeurs ne pouvant anticiper aucun changement, son processus de rétribution s'appuie explicitement sur des classeurs de $[M]_t$ dont l'*Anticipation* en décrit au moins un.

$$r \leftarrow r + \beta_{\mathrm{rl}}(\rho_{t-1} + \gamma \max_{\substack{cl \in [M]_t \text{and} \\ E_{\mathrm{cl}} \neq \{\#\}^L}} (q_{\mathrm{cl}}.r_{\mathrm{cl}}) - r) \tag{3.7}$$

Quand les classeurs d'ACS 2 estiment le gain d'une action sur une durée contrôlée par le facteur d'actualisation γ, ils deviennent aussi capables d'estimer le gain immédiat ir de leurs actions. Cette information additionnelle est calculée de manière analogue à la récompense r, si la valeur γ est nulle dans l'équation 3.7. Son calcul est décrit par l'équation 3.8, où β_{rl} désigne le pas d'apprentissage et ρ_{t-1} la récompense reçue à l'instant $t-1$ par le système.

$$ir \leftarrow ir + \beta_{\mathrm{rl}}(\rho_{t-1} - ir) \tag{3.8}$$

3.2.7 Cycle d'apprentissage d'ACS 2

Le cycle complet d'apprentissage d'ACS 2 est initialisé avec une action choisie au hasard parmi celles que le système peut faire et avec une première observation de l'environnement. L'action choisie est réalisée, puis ACS 2 itère le cycle suivant (*cf.* figure 3.18) :

1. De nouvelles observations sont perçues et les récompenses sont reçues de l'environnement. Si l'objectif d'apprentissage est atteint, les processus d'apprentissage par anticipation, par renforcement et de généralisation génétique sont appliqués une dernière fois sur le nouvel ensemble d'action.
2. Le processus d'appariement est appliqué sur la population de classeurs $[P]$, permettant de former le sous-ensemble d'appariement $[M]$ où les *Conditions* de ces classeurs correspondent à la nouvelle observation.
3. L'ALP est appliqué sur l'ensemble d'action $[A]$ qui a été précédem-

ment formé par le système.

4. Le processus de rétribution est appliqué sur l'ensemble d'action $[A]$ qui a été précédemment formé par le système.
5. Le processus de généralisation génétique peut être appliqué sur l'ensemble d'action $[A]$ qui a été précédemment formé par le système.
6. Une action est choisie à partir des classeurs de $[M]$.
7. Un nouvel ensemble d'action est formé à partir des classeurs de $[M]$ dont les *Conséquents* décrivent l'action sélectionnée.
8. ACS 2 interagit avec son environnement en réalisant l'action sélectionnée.

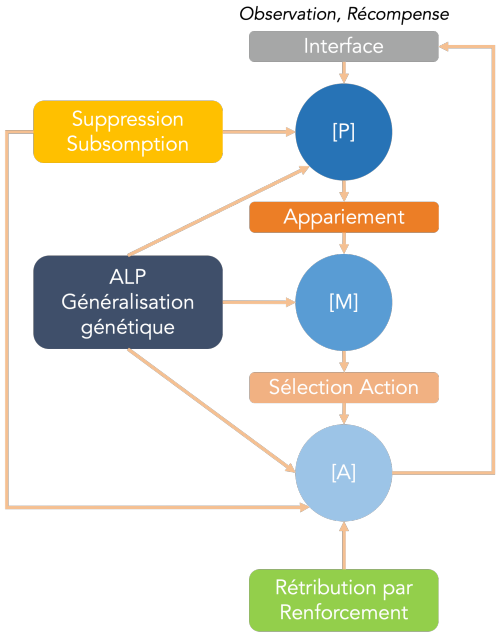


FIGURE 3.18 – Cycle d'apprentissage d'*Anticipatory Classifier System* 2.
Les principaux composants d'ACS 2 et leurs interactions sont décrits dans cette illustration qui s'appuie la présentation des systèmes de classeurs de la section 2.2 et la synthèse proposée par la figure 2.9.

3.2.8 Paramètres d'ACS 2

ACS 2 dispose de différents paramètres à définir en amont d'un apprentissage afin de configurer :

— la sélection des actions et des biais exploratoires, avec la probabilité d'exploration ϵ de l'*ϵ-greedy policy* et la probabilité de biaiser l'exploration p_b ;

- la subsomption de classeurs, avec les seuils de fiabilité θ_{r} et d'expérience θ_{exp} des classeurs ;
- le processus d'apprentissage par anticipation, avec le pas d'apprentissage β_{alp}, le seuil d'inadéquation θ_{i} et le nombre maximal d'attributs conditionnels spécifiés $u_{\max}$ des classeurs ;
- le processus de généralisation génétique, avec la fréquence d'application θ_{ga}, la taille maximale d'un ensemble d'action θ_{as}, le taux de croisement χ et le taux de mutation μ ;
- le processus de rétribution par renforcement, avec le pas d'apprentissage β_{rl} et le facteur d'actualisation γ.

L'ensemble de ces paramètres peuvent être laissés à leurs valeurs empiriques par défaut (détaillées ci-dessous), ACS 2 étant robuste aux divers réglages de paramètres possibles (Butz and Stolzmann, 2001). Pour une tâche d'apprentissage donnée, la configuration de ces paramètres impacte principalement la vitesse à laquelle le système est capable de se doter d'un modèle de son environnement et de résoudre sa tâche.

Sélection des actions et biais exploratoires

L'apprentissage d'ACS 2 est réalisé en deux phases :

- une phase d'exploration où le système explore son environnement, afin qu'il puisse se doter de classeurs décrivant toutes les transitions environnementales observées ;
- une phase d'exploitation où le système cherche à résoudre sa tâche d'apprentissage directement.

Durant la phase d'exploration, ϵ tend vers 1, ce qui permet d'accélérer la vitesse à laquelle les transitions environnementales sont apprises. Les actions sont alors choisies le plus souvent aléatoirement ou selon l'un des deux biais exploratoires inclus, qui favorisent également l'acquisition de ces transitions. La probabilité p_b de biaiser l'exploration est fixée par défaut à 0.5, où chaque biais a ensuite une chance sur deux d'être sélectionné. (Butz, 2001) montre que l'utilisation seule de l'un des deux biais est contre-productif, car il empêche le système d'acquérir toutes les transitions d'un environnement en négligeant certaines parties de celui-ci. L'utilisation des deux biais est

alors favorisée, car elle permet de toutes les acquérir, indépendamment de p_b : p_b influe uniquement sur la vitesse à laquelle elles sont apprises.

Durant la phase d'exploitation, ϵ tend vers 0 afin de préférer les actions permettant effectivement de résoudre la tâche d'apprentissage.

La configuration de ϵ est le plus souvent réalisée de manière empirique, selon le problème qui doit être résolut par ACS 2.

Paramètres de subsomption

La subsomption utilise les seuils de fiabilité θ_{r} et d'expérience θ_{exp} des classeurs. Le seuil de fiabilité θ_{r}, compris entre 0 et 1, détermine si un classeur peut faire partie du modèle de l'environnement construit par le système. Plus ce seuil est élevé, plus ACS 2 a besoin de temps pour construire les classeurs qui intégreront ce modèle, ces derniers étant plus fiables, anticipant correctement des changements environnementaux. Par défaut, θ_{r} a pour valeur 0.9.

Le seuil d'expérience θ_{exp} détermine directement si un classeur peut être utilisé pour subsumer d'autres classeurs. Une valeur empirique de ce seuil est de 20, indiquant qu'un classeur est passé au moins 20 fois dans le processus d'apprentissage par anticipation. Un seuil moins élevé pourrait réduire la qualité globale du modèle environnemental ou ralentir l'acquisition de ce modèle : des classeurs trop généraux (par conséquent plus propices aux erreurs d'anticipation) seraient plus aisément maintenus dans la population et pourraient prévenir l'insertion de classeurs plus adéquats. Un seuil plus élevé ralentirait la vitesse à laquelle le modèle environnemental est construit, les classeurs devant accéder plus souvent à l'ALP.

Paramètres du processus d'apprentissage par anticipation

Le pas d'apprentissage β_{alp} de l'ALP est une valeur réelle positive fixée couramment à 0.05. Plus cette valeur est élevée, plus vite les attributs des classeurs dépendant de ce pas (q et aav) convergent, mais ils sont sujets à plus de variations. A l'inverse, un pas d'apprentissage plus faible permet une convergence plus fine et moins bruitée de ces attributs.

Le seuil d'inadéquation θ_{i} détermine à partir de quel moment un classeur

est supprimé de la population : ce seuil est alors fixé à une faible valeur, usuellement 0.1, et strictement inférieur à θ_{r}.

Le nombre d'attributs des *Conditions* des classeurs qui peuvent être spécifiés lors de l'ALP est contraint par $u_{\max}$. Une valeur par défaut d'$u_{\max}$ est le nombre total d'attributs compris dans une observation, permettant ainsi de spécifier au besoin l'ensemble des attributs conditionnels des classeurs.

Paramètres du processus de généralisation génétique

La fréquence d'application θ_{ga} de la généralisation génétique est empiriquement fixée à 100. L'idée principale est de laisser du temps à l'ALP pour d'abord adapter les classeurs à l'environnement puisque la généralisation s'appuie sur la qualité des classeurs. Les classeurs, peu importe par quels processus ils sont construits, doivent idéalement être éprouvés par le processus d'apprentissage par anticipation avant de passer par le mécanisme de généralisation. Une fréquence plus élevée permet de s'assurer que suffisamment de temps ait été laissé, mais ralentit la vitesse à laquelle le modèle environnemental est généralisé. Une fréquence plus faible permet de contraindre plus rapidement la taille de la population de classeurs, en limitant le dépassement du nombre de micro-classeurs θ_{as} possibles dans un ensemble d'action. Mais en contraignant plus rapidement les ensembles d'action, l'insertion et le maintien de classeurs qui seraient plus adaptés à leur environnement et plus généraux sont plus délicats, ceux-ci n'ayant pu être suffisamment expérimentés ou à cause de la présence de classeurs trop généraux.

Le paramétrage de la taille maximale du nombre de micro-classeurs θ_{as} des ensembles d'action a des conséquences similaires à θ_{ga} sur l'apprentissage d'ACS 2. Un seuil trop faible peut causer la suppression des classeurs nécessaires à la représentation environnementale ou à la résolution de la tâche. Un seuil élevé permet le maintien et l'évolution d'un plus grand nombre de classeurs, au prix d'un accroissement important de la population de classeurs se traduisant par une vitesse d'apprentissage plus faible. Empiriquement, le seuil θ_{as} est fixé à 20, celui-ci ayant été fonctionnel pour de nombreux problèmes.

Le généralisation génétique d'ACS 2 s'appuie principalement sur une muta-

tion directe des attributs conditionnels des *Conditions* des classeurs. L'opérateur de croisement joue ainsi un rôle secondaire par rapport à l'opérateur de mutation. Le taux de croisement χ, compris entre 0 et 1, est fixé à 0.8 qui est une valeur standard de la littérature sur les algorithmes génétiques (Butz, 2001). Au contraire, le taux de mutation μ, lui aussi compris entre 0 et 1, est fixé empiriquement à 0.3,ce qui est plus élevé que les taux de mutation employés traditionnellement. Une valeur plus faible de μ réduit la vitesse à laquelle les classeurs sont généralisés. Une valeur trop élevée réduit également la vitesse à laquelle le modèle est généralisé : un nombre plus important de classeurs trop généraux est introduit dans la population de classeurs, ce qui implique des calculs supplémentaires pour les spécifier à nouveau afin qu'ils s'adaptent à leur environnement.

Paramètres du processus de rétribution par renforcement

Le pas d'apprentissage du processus de rétribution par renforcement fonctionne de manière analogue à celui de l'ALP. Ce pas est lui aussi fixé à 0.05.

Le facteur d'actualisation γ, compris entre 0 et 1, permet d'indiquer à quel point les récompenses futures issues de l'environnement influence le comportement actuel du système. Plus cette valeur est proche de 1, plus l'influence de ces récompenses est importante et inversement. Une valeur couramment utilisée de γ est 0.95.

3.3 Revue des systèmes de classeurs à anticipation

Les principaux systèmes de classeurs à anticipation sont introduits dans les sections suivantes selon qu'ils puissent être associés à ACS, à ACS 2 ou à aucun de ces deux ALCS. Nous motiverons notre choix d'ACS 2 dans le section 3.3.4 afin de concevoir une intelligence artificielle autonome et explicable pour des environnements incertains.

3.3.1 Variants d'ACS

Avant de parvenir à ACS 2, le système ACS a connu différentes propositions qui peuvent inclure des évolutions proposées dans ACS 2 ou de nouvelles extensions.

Prédictions améliorées par les probabilités

Les Prédictions améliorées par les probabilités (*Probability-Enhanced Predictions (PEP)*) étoffent les capacités d'anticipation des classeurs en leur permettant de décrire des changements perceptifs antagonistes dans leurs *Anticipations* (Butz et al., 2000). Une PEP consiste en un tableau associatif dont les clés désignent les attributs qui sont anticipés, et dont les valeurs décrivent les probabilités d'anticiper chacun des attributs. Chaque attribut des *Anticipations* des classeurs est alors remplacé par une PEP. La figure 3.19 donne deux exemples d'*Anticipations* qui contiennent des PEP.

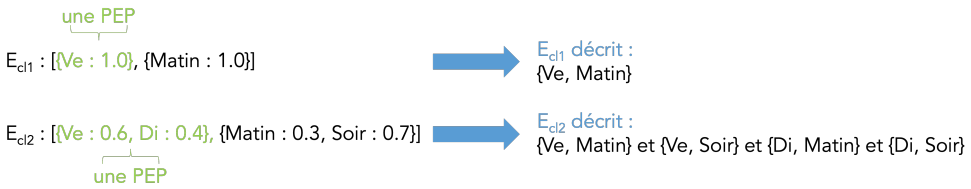


FIGURE 3.19 – Illustration des prédictions améliorées par les probabilités dans l'*Anticipation* d'un classeur.
*Un attribut d'une PEP peut être combiné avec chacun des attributs des autres PEP de l'*Anticipation *du classeur pour déterminer les situations environnementales anticipées.*

Les PEP renforcent les capacités d'ACS à gérer la stochasticité des environnements. Dans de tels environnements, différentes situations environnementales sont atteignables si des attributs perceptifs varient aléatoirement au cours de l'apprentissage ou si la réalisation d'une action est non-déterministe. Les PEP permettent alors au système de se doter de classeurs fiables en toutes ces situations. (Butz et al., 2000) montrent que les PEP permettent effectivement à ACS de se doter d'une population de classeurs fiables qui décrit toutes les transitions environnementales.

(Butz et al., 2000) intègrent le mécanisme des PEP dans un ACS qui :

— possède le mécanisme de généralisation génétique d'ACS 2 ;

- possède le mécanisme de subsomption d'ACS 2 ;
- possède le processus de rétribution d'ACS 2 ;
- applique l'ALP d'ACS sur des ensembles d'action ;
- ne possède plus la liste de messages.

Ils modifient ensuite leur système en :

- ajoutant le mécanisme de détection du PAI de (Stolzmann, 1999) afin de marquer les classeurs à utiliser pour construire un classeur avec plusieurs anticipations ;
- ajoutant un mécanisme de construction de ces classeurs qui s'appuie sur les classeurs précédemment marqués ;
- permettant la mise à jour des probabilités des PEP d'un classeur lorsque celui-ci anticipe correctement les changements environnementaux.

Le mécanisme de détection du PAI cherche à identifier les situations environnementales pour lesquelles les classeurs ne peuvent plus être spécifiés par ALP et pour lesquelles ils parviennent et échouent à anticiper les changements perçus. Ce mécanisme détecte si différents changements perceptifs sont à anticiper pour une même situation, indépendamment de la source d'incertitude : ces changements pourront être décrits par des PEP. Tous les classeurs comportent un nouvel attribut booléen *ee* (*enhanced effect*) qui indique si le classeur peut être utilisé pour construire un classeur dont l'*Anticipation* est améliorée. *ee* est initialisé à *False* pour tous les classeurs, est modifié à *True* si la détection est positive pour un classeur et est modifié à *False* si un classeur est marqué par plusieurs situations environnementales.

Si l'attribut *ee* d'un classeur cl_1 prend la valeur *True*, un nouveau classeur cl_{new} dont l'*Anticipation* comporte plusieurs changements perceptifs peut être généré. Un second classeur cl_2 est recherché dans l'ensemble d'action courant tel que l'attribut *ee* de cl_2 vaut *True* et les marques de cl_1 et cl_2 soient strictement identiques. Le nouveau classeur cl_{new} fusionne les deux *Anticipations* de cl_1 et de cl_2 en combinant chacune des PEP en une unique PEP : les valeurs de probabilités sont sommées entre les symboles identiques puis normalisées. La *Condition* de cl_{new} est également spécifiée de telle sorte que tout attribut ayant été spécifié dans les *Conditions* de cl_1 et de cl_2 le soit également. La figure 3.20 schématise la construction du nouveau classeur cl_{new}. Les autres attributs de ce classeur sont initialisés à leur valeur par défaut ou

marqués par la valeur du compteur de pas de temps.

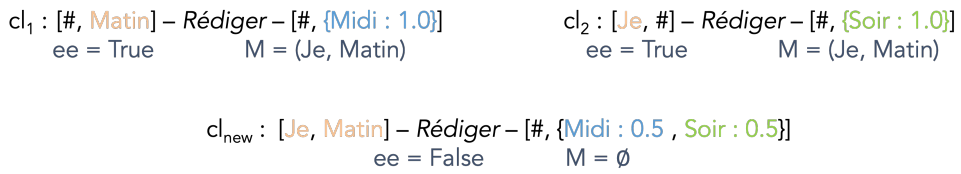


FIGURE 3.20 – Création d'un classeur avec des prédictions améliorées par les probabilités dans son *Anticipation*.
La Condition *de* cl_{new} *est spécifiée sur les deux attributs, car le moment du jour est précisé dans la* Condition *de* cl_1 *et le jour est précisé dans la* Condition *de* cl_2. *L'*Anticipation *de* cl_{new} *ne prédit aucun changement sur le jour, et décrit un changement dans le moment du jour où il y a* 50% *de chances que le matin passe au midi ou au soir. Les classeurs* cl_1 *et* cl_2 *ne sont pas supprimés lors de la création de* cl_{new}.

Les probabilités des PEP d'un classeur sont mises à jour si ce classeur entre dans l'*expected case* du processus d'apprentissage par anticipation. Les probabilités des attributs perceptifs effectivement observées sont mises à jour en deux temps selon les équations 3.9, où p_i est la probabilité d'un attribut i d'une PEP et β_{pep} désigne le pas d'apprentissage associé à ces probabilités.

$$\begin{aligned} p_i &\leftarrow p_i + \beta_{\text{pep}}(1 - p_i) \\ p_i &\leftarrow \frac{p_i}{\sum\limits_{i \in |pep|} p_i} \end{aligned} \tag{3.9}$$

Séquences comportementales

(Métivier and Lattaud, 2002) présentent le seul autre ALCS qui utilise des séquences comportementales. Ils s'appuient sur un ACS qui est différent de celui-ci de (Stolzmann, 1999) sur différents points :

- il applique l'ALP d'ACS sur des ensembles d'action ;
- il intègre le processus de rétribution d'ACS 2 ;
- il intègre les mécanismes de subsomption et d'insertion de classeurs provenant de l'ALP issus d'ACS 2 en les appliquant sur l'ensemble de la population de classeurs au lieu des ensembles d'action ;
- il ne possède plus la liste de messages.

(Métivier and Lattaud, 2002) utilisent les mêmes mécaniques de détection

du PAI et de construction de séquences comportementales que (Stolzmann, 1999). En revanche, ils adaptent le mécanisme de contrôle des séquences comportementales aux ensembles d'action que leur système construit. Si leur ACS doit réaliser une séquence d'actions, le système garde temporairement en mémoire les observations intermédiaires dans une liste. Si une nouvelle observation est déjà présente dans cette liste lors de l'exécution d'une séquence d'actions, la qualité de tous les classeurs de l'ensemble d'action courant est décrémentée sans que la séquence ne soit interrompue.

Ils montrent que les séquences comportementales permettent à leur système de résoudre leur tâche d'apprentissage plus rapidement : entraîné sur 4 nouveaux environnements dans lesquels ils doivent atteindre la sortie, leur ACS est capable de l'atteindre plus rapidement que si ces séquences n'étaient pas utilisées et de se doter d'une politique de décision stable au cours du temps. Mais l'emploi des séquences comportementales a généré un grand nombre de classeurs au sein de la population impactant de fait : la compréhensibilité globale de la population et l'exécution du système, car plus de classeurs impliquent plus de ressources pour réaliser les différents calculs.

Planification d'actions et apprentissage latent

(Stolzmann and Butz, 1999) introduit un mécanisme de planification d'actions dans ACS, lui permettant d'accélérer l'apprentissage latent d'une représentation environnementale. Cette planification s'appuie sur un chaînage des *Conditions* et des *Anticipations* des classeurs afin de trouver une séquence de classeurs permettant d'atteindre une situation environnementale cible (indiquée au préalable au système). Ce chaînage des classeurs est réalisé à partir des classeurs fiables de la population (ceux pour lesquels les qualités q sont supérieures à θ_{r}) et dans la limite d'un nombre maximal de classeurs pouvant être chaînés. Il existe trois différents chaînages possibles :

- à partir de la situation dans laquelle se trouve le système, un chaînage avant permet de calculer récursivement les situations atteignables jusqu'à atteindre la situation environnementale cible ou que le nombre maximal de classeurs chaînés soit atteint ;
- à partir de la situation cible que le système doit atteindre, un chaînage arrière permet de calculer récursivement les situations à l'origine de

celle-ci jusqu'à atteindre la situation environnementale courante ou que le nombre maximal de classeurs chaînés soit atteint ;
— un chaînage qui combine les deux approches avant et arrière ci-dessous.

La planification d'actions intervient de manière cyclique : ACS explore son environnement durant un nombre d'étapes défini en amont avant d'utiliser la planification pour tenter d'atteindre une situation cible. Si une séquence de classeurs permet effectivement d'atteindre la situation cible à partir de la situation courante, les qualités des classeurs de la séquence sont augmentées et une nouvelle situation cible peut lui être fournie. Sinon, il bascule à nouveau dans une phase d'exploration durant un nombre défini d'étapes, devant permettre à ACS de compléter son modèle de représentation de l'environnement.

La planification d'actions peut être couplée à un générateur d'objectifs, c'est-à-dire un générateur de situations environnementales qu'un système doit atteindre. Des tels générateurs ont été développés pour différents environnements d'apprentissage et ne sont donc pas intégrés à ACS. (Stolzmann and Butz, 1999) dessinent des pistes de conception permettant d'en intégrer un au sein d'ACS. Ces générateurs permettent en particulier de donner des objectifs qui pourraient être statistiquement plus rares que d'autres, ou de favoriser l'exploration de situations environnementales d'intérêt pour la résolution d'une tâche d'apprentissage, permettant *in fine* d'accélérer l'acquisition d'un modèle environnemental complet par ACS.

Planification d'actions et renforcement

(Stolzmann et al., 2000) montrent que leur ACS peut être comparé à un système cognitif dont le comportement est similaire à celui de rats au travers de différentes expériences. En particulier, ils étudient deux mécanismes de planification d'actions s'intégrant dans une tâche d'apprentissage par renforcement :

— le *One-step Mental Acting*, qui consiste à apprendre à partir de l'exécution d'actions hypothétiques. Des connexions entre différents classeurs sont recherchées pour propager leurs récompenses sans directement réaliser les actions décrites dans ces classeurs ;

— le *Lookahead-Winner Algorithm*, qui consiste à introduire un biais dans la sélection d'une action à partir des anticipations faisant suite à la réalisation d'une action et des récompenses associées. La sélection d'une action intègre alors plus d'éléments contextuels afin de choisir une action dont les conséquences seraient plus adéquates à la résolution de la tâche d'apprentissage.

(Stolzmann et al., 2000) utilisent dans ce but un ACS qui : applique ses mécanismes de découverte de règle sur des ensembles d'action ; modifie son processus de sélection d'action avec une *roulette wheel selection* à partir des récompenses moyennes pour chaque action décrite par les classeurs de l'ensemble d'appariement.

Le *One-step Mental Acting* est déclenché de manière cyclique, quand un nombre prédéfini d'actions a été réalisé par cet ACS. Ce processus de planification intervient alors en trois étapes :

— un classeur fiable *cl* est choisi aléatoirement au sein de la population ;
— un ensemble de classeurs $[L]$ est formé où les changements décrits dans l'*Anticipation* des classeurs de $[L]$ correspondent à la *Condition* de *cl* et où les *Conditions* des classeurs de $[L]$ correspondent à la *Condition* de *cl* si les attributs perceptifs ne changent pas (# dans une *Anticipation*) ;
— la récompense r du classeur *cl* est rétropropagée à tous les classeurs de l'ensemble $[L]$ par le *Bucket Brigade Algorithm* (cas où il n'y a pas de récompenses environnementales de l'équation 3.3)

Enfin, le *Lookahead-Winner Algorithm* intervient lors de la sélection d'une action par cet ACS, qui la modifie ainsi :

— les prédictions de récompense moyenne p_a pour chaque action a décrite par les classeurs de l'ensemble d'appariement sont calculées ;
— pour chaque action a décrite par les classeurs de l'ensemble d'appariement :
  — la situation s_{next} suivant la réalisation de l'action a est prédite avec le classeur cl_a qui possède la plus haute qualité pour l'action a ;
  — si au moins un changement est décrit par cl_a, le classeur cl_{next} dont la *Condition* correspond à s_{next} et dont la qualité est la plus élevée est identifié ;

- la prédiction de récompense moyenne p_a associée à l'action a est modifiée à partir de la récompense $r_{\mathrm{cl}_{\mathrm{next}}}$, de la qualité q_{cl_a} et du facteur d'actualisation γ selon $p_a \leftarrow \frac{p_a + \gamma . q_{\mathrm{cl}_a} . r_{\mathrm{cl}_{\mathrm{next}}}}{1 + \gamma . q_{\mathrm{cl}_a}}$;
- une action est finalement sélectionnée par une *roulette wheel selection* à partir des nouvelles prédictions de récompense moyenne.

Ce mécanisme peut être adapté afin d'explorer les anticipations faisant suite à la réalisation de plusieurs actions. Cependant, planifier la réalisation de plusieurs actions accroît de manière exponentielle le nombre de situations environnementales à prédire.

Apprentissage onirique

(Holley et al., 2004) ont proposé une extension à ACS où celui-ci réalise indistinctement son apprentissage dans un environnement réel et dans un environnement créé à partir d'une image instantanée de sa population de classeurs, à l'image d'un environnement onirique. Les situations perçues dans cet environnement onirique sont uniquement calculées à partir des *Conditions* et des *Anticipations* des classeurs, que les situations *rêvées* existent ou non. Le système ACS en lui-même est alors inchangé : seules les interactions avec ces environnements sont redéfinies. (Holley et al., 2004) ont alors montré qu'ACS est en mesure de se doter d'une représentation complète des environnements *réels* en interagissant moins avec celui-ci grâce aux environnements oniriques, accélérant *de facto* l'apprentissage d'une représentation environnementale.

3.3.2 Extensions d'ACS 2

Planification d'actions

Les mécanismes de planification d'actions accélérant l'apprentissage latent d'une représentation environnementale ont été intégrés tels quels dans ACS 2 par (Unold et al., 2019) : les résultats obtenus confirment leur apport. Les auteurs ont également introduit de nouveaux générateurs d'objectifs pour divers environnements.

(Butz and Hoffmann, 2002) ont aussi intégré le *One-step Mental Acting* et

le *Lookahead-Winner Algorithm* dans ACS 2 avec quelques modifications, puisque les processus de rétribution entre ACS et ACS 2 ne sont plus les mêmes. Le *One-step Mental Acting* d'ACS 2 sélectionne d'abord un classeur fiable *cl* dans la population qui prédit au moins un changement environnemental. Au lieu de former un ensemble de classeurs dont l'*Anticipation* peut être associée à la *Condition* de *cl*, un ensemble de classeurs dont les *Conditions* peuvent être associées à l'*Anticipation* de *cl* est formé de manière analogue. La *fitness*[2] la plus élevée est identifiée dans l'ensemble formé afin de la rétropropager dans la prédiction de récompense de *cl* par le biais du processus de rétribution. Le *Lookahead-Winner Algorithm* synthétise dans un tableau noté *AA* la *fitness* la plus élevée pour chacune des actions *a* réalisables par le système. Pour chaque action *a*, s'il existe un classeur cl_{next} ayant la plus haute qualité dans la situation résultante à l'action *a* et dont l'*Anticipation* décrit au moins un changement perceptif, $AA[a]$ est modifiée selon $AA[a] \leftarrow \frac{AA[a]+\gamma.q_{\text{cl}_{\text{next}}}.q.r}{1+\gamma.q_{\text{cl}_{\text{next}}}}$. L'action pour laquelle la valeur dans *AA* est la plus élevée est enfin sélectionnée.

Représentations continues dans ACS 2

(Kozlowski and Unold, 2019) couplent dans ACS 2 deux techniques pour que ce système réalise son apprentissage lorsque ses données d'apprentissage sont composées de valeurs réelles. Ils opèrent d'abord une normalisation des valeurs réelles sur un nombre de bits prédéfinis, afin de les ramener dans l'intervalle $[0; 1]$. Puis, ils utilisent dans les classeurs une représentation des attributs qui décrit directement des intervalles de valeurs discrètes par les bornes de ces intervalles, de façon non ordonnée. Leur ACS 2, nommé en l'occasion RACS (*Real-valued Anticipatory Classifier System*), est modifié de façon à ce que :

- le symbole générique # corresponde au plus grand intervalle possible dans une *Condition* ;
- pour chaque attribut réel, le processus de recouvrement créé un intervalle discret à partir de la valeur réelle observée, cette valeur observée étant au préalable bruitée uniformément selon un paramètre ϵ_{noise} ;
- l'opérateur de mutation du processus de généralisation accroît ou réduit les intervalles discrets selon un bruit uniforme paramétré par ϵ_{mute} ;

2. Il s'agit du produit de la qualité par le renforcement.

— le processus de subsomption tient compte des intervalles présents dans les *Conditions* et *Anticipations* des classeurs.

Cependant, ils constatent que les populations de classeurs construites par RACS augmentent de manière incontrôlée à cause d'un problème de généralisation, de recouvrement des classeurs et de paramétrage des différents bruits ou de la discrétisation. Des mécanismes prenant en compte les *Anticipations* et de nouveaux opérateurs génétiques sont nécessaires pour intégrer et faire évoluer avec succès des représentations continues dans les ALCS.

(Kozlowski and Unold, 2020) comparent trois techniques biaisant l'exploration d'un environnement s'appuyant sur l'ϵ-*greedy selection*, pour des environnements où les données d'apprentissage sont composées de valeurs réelles. ACS 2 n'est pas modifié : seule une discrétisation des valeurs réelles est réalisée durant l'apprentissage. ACS 2 est capable de se doter d'une représentation complète de ces environnements, mais dans une moindre mesure de résoudre ses tâches d'apprentissage. Les trois biais étudiés permettent effectivement d'accélérer l'acquisition d'un modèle de l'environnement, mais à des vitesses différentes selon les environnements étudiés. Par ailleurs, les populations de classeurs construites par ACS 2 ont connu un accroissement d'autant plus fort que la discrétisation opérée est fine, impactant *de facto* la compréhensibilité des solutions développées : un traitement algorithmique afin de réduire la taille des populations après apprentissage pourrait alors résoudre cet inconvénient. Enfin, le processus de rétribution d'ACS 2 n'a pas permis de résoudre une tâche d'apprentissage dans un environnement qui a été parfaitement modélisé par la population de classeurs : ACS 2 aurait des difficultés à propager les récompenses environnementales sur de longues séquences d'actions qui pourraient être résolues avec de nouveaux mécanismes de rétribution.

Suite à ce dernier constat, (Kozłowski and Unold, 2021) proposent un nouveau processus de rétribution : ils cherchent à maximiser la moyenne des récompenses successivement obtenues par les classeurs à partir du *R-learning* (Schwartz, 1993), au lieu de chercher à maximiser le total des récompenses successivement obtenues avec un processus adapté du *Q-Learning*. Les politiques de décision formées par leur système échelonnent plus uniformément les récompenses moyennes estimées pour chacun des classeurs, facilitant en conséquence le choix de l'action la plus à même de résoudre la tâche d'ap-

prentissage.

(Kozlowski and Unold, 2022) montrent enfin qu'ACS 2 est capable de se doter d'une représentation complète d'environnements à valeurs réelles lors d'un apprentissage latent et avec une discrétisation de ces valeurs réelles. ACS 2 a également été évalué avec d'autres systèmes de classeurs à anticipation sur ces tâches d'apprentissage latent : ils ont montré que les populations de classeurs construites par ACS 2 sont plus générales, compactes et décrivent plus fidèlement leurs environnements que celles des systèmes auxquels il a été comparé.

Intégration d'ALCS à OpenAI Gym

(Kozlowski and Unold, 2018) ont d'abord proposé une version en *Python* d'ACS 2 dans un dépôt *GitHub* public, basée sur une description algorithmique fournie par (Butz and Stolzmann, 2001). Ce dépôt s'est ensuite élargi à d'autres ALCS avec, entre autres, des versions d'ACS 2 avec les extensions présentées et un ACS. Dans le même temps, (Kozlowski and Unold, 2018) ont mis en place une interface permettant d'utiliser les ALCS avec *OpenAI Gym* (Brockman et al., 2016a). *OpenAI Gym* est une bibliothèque *Python* de référence, en code source ouvert, permettant de développer et comparer des algorithmes d'apprentissage par renforcement en fournissant une interface commune entre ces algorithmes et les environnements dans lesquels ils peuvent réaliser leur apprentissage.

3.3.3 Autres systèmes de classeurs à anticipation

Yet Another Classifier System

YACS (*Yet Another Classifier System*) est un système de classeurs à anticipation qui propose une mise en œuvre différente d'ACS et d'ACS 2 des principes du contrôle comportemental anticipatif dans ses mécanismes de découverte de règles (Gerard et al., 2002).

Les classeurs de YACS sont toujours composés d'un tuple contenant une *Condition*, un *Conséquent* et une *Anticipation*, mais ils intègrent des traces

d'une longueur configurable qui indiquent les échecs et succès à anticiper les prochaines situations environnementales, ainsi que les situations précédent ces échecs et succès.
Ces traces permettent au processus d'apprentissage par anticipation de YACS de construire ses classeurs en commençant d'abord par indiquer les changements environnementaux dans leurs *Anticipations*, puis en spécifiant de manière ciblée leurs *Conditions* à partir des traces. Si une trace *oscille*, c'est-à-dire qu'elle indique qu'un classeur échoue et parvient à correctement anticiper une situation, un attribut général # de la *Condition* est spécifié avec chaque valeur possible de cet attribut, créant par la même occasion autant de classeurs que de valeurs possibles. Le choix d'un attribut conditionnel d'un classeur à spécifier est réalisé à partir de scores propres à ce classeur et à chaque attribut conditionnel spécifiable, indiquant à quel point la spécification de l'attribut en question permettrait à YACS de se doter de classeurs plus à même de représenter fidèlement son environnement.

Enfin, le processus de rétribution de YACS s'appuie sur une liste qui contient l'ensemble des situations environnementales effectivement perçues au cours de son apprentissage et les actions réalisées en ces situations. Une récompense est prédite pour chacune des paires {situation, action} de cette liste à partir de l'algorithme *Value Iteration* de (Bellman, 1957).

YACS est différent d'ACS et ACS 2 par : son processus de généralisation qui s'appuie sur des scores analogues à ceux liés à la spécification pour déterminer quels attributs conditionnels peuvent être généralisés (Gerard and Sigaud, 2001) ; son processus de rétribution qui s'appuie sur les transitions environnementales apprises par ses classeurs, au contraire d'ACS et d'ACS 2 qui n'en ont pas besoin ; ses mécaniques de découverte de règles qui décorrèlent explicitement la construction des *Anticipations* et des *Conditions* des classeurs ; son incapacité à réaliser un apprentissage dans des environnements incertains (Gerard et al., 2002).

Modular Anticipatory Classifier System

MACS *Modular Anticipatory Classifier System* est un système de classeurs à anticipation conçu à partir de YACS dans le but de représenter des changements perceptifs réguliers suite à l'interaction du système avec son environ-

nement (Gérard and Sigaud, 2001). MACS se diffère de YACS par la structure de l'*Anticipation* de ses classeurs et son processus d'apprentissage par anticipation qui a été adapté en conséquence. Ces différences lui permettent effectivement d'accroître ses capacités à généraliser : les connaissances acquises dans ses populations de classeurs sont exploitées plus rapidement pour accélérer l'acquisition d'un modèle environnemental complet. Sa construction de politique de décision est similaire à YACS, de même que les mécaniques de généralisation ou de spécification des *Conditions* des classeurs (Gérard et al., 2005).

Un nouveau symbole ? est introduit dans les *Anticipations* des classeurs de MACS à la place du symbole # : si # désigne l'absence de changement d'un attribut perceptif, ? indique qu'un classeur ne fournit aucune information relative à la présence ou à l'absence de changement pour cet attribut. Autrement dit, un classeur ignore les possibles changements perceptifs d'un attribut dont la valeur est ?.
MACS fait évoluer des classeurs dont l'*Anticipation* n'est composée que d'un unique attribut différent de ce nouveau symbole ?, même s'ils peuvent en contenir théoriquement plusieurs. Un classeur fiable, à l'image de YACS, est un classeur capable de décrire un changement perceptif dans son *Anticipation* suite à la réalisation de l'action décrite dans son *Conséquent* dans l'ensemble des situations où il peut être utilisé. L'évaluation de la fiabilité d'un classeur n'est plus assurée par une trace, mais en comptabilisant les nombres d'anticipations correctes et erronées, ainsi que les dernières situations où ces anticipations ont été correctes et erronées. Les traces restent présentes dans les classeurs de MACS car elles permettent de spécialiser et généraliser les *Conditions* des classeurs comme YACS.
Une description fiable d'une transition environnementale est alors assurée par un ensemble de classeurs fiables décrivant eux-mêmes une partie de cette transition, contrairement à YACS, à ACS ou à ACS 2. L'ensemble des classeurs qui correspondent à une situation environnementale et ayant le même *Conséquent* est utilisé pour déterminer le ou les situations anticipées.

(Gérard and Sigaud, 2003) proposent ensuite avec MACS une nouvelle technique d'exploration de l'environnement dont le but est de maximiser l'information apportée par chacune des interactions du système avec son environnement. Cette technique ne nécessite aucun paramètre et s'appuie sur

les connaissances déjà acquises dans les populations de classeurs : elle serait alors applicable aux autres ALCS. Ils couplent et hiérarchisent dans un même processus de sélection d'action : le gain en information apporté par la réalisation d'une action ; le gain immédiat reçu de l'environnement ; le biais de rappel qui favorise l'action ayant été la moins récemment utilisée par le système. Ils montrent alors que ce processus d'exploration permet au système d'adapter rapidement sa politique de décision si l'environnement change durant la résolution de la tâche d'apprentissage.

XCS Anticipatory Classifier System

(Butz and Goldberg, 2003) proposent de découpler la construction d'une représentation de l'environnement de celle d'une politique de décision dans un nouveau ALCS : XACS désigne le couplage d'un ACS 2 pour apprendre une représentation environnementale et d'un système de classeurs XCS pour apprendre une politique de décision. À chaque fois que le processus de rétribution d'ACS 2 aurait dû être appelé, XCS est appelé : il s'appuie sur les *Anticipations* des classeurs d'ACS 2 pour déterminer la prochaine situation et rétropropager dans ses propres classeurs les prédictions de récompenses associées. XACS a été capable de construire des politiques de décision plus adaptée à la résolution des tâches d'apprentissage qu'ACS 2.

En effet, ACS 2 construit d'abord des représentations de ses environnements, avant de développer des politiques de décision s'appuyant sur ces représentations. Les politiques de décision sont ainsi conditionnées par les représentations environnementales, ce qui peut poser problème : des classeurs peuvent parfaitement décrire un ensemble de transitions environnementales pour lesquelles les récompenses à prédire ne sont pas identiques. Les récompenses de ces classeurs sont alors instables et leur rétropropagation peut fausser la construction d'une politique de décision adéquate à la résolution de la tâche d'apprentissage.

3.3.4 ACS 2 pour des environnements incertains

La mise en place d'une intelligence artificielle autonome et explicable pour des environnements incertains présuppose de discuter de l'usage des sys-

tèmes de classeurs à anticipation que nous avons présentés. Le choix d'un ALCS peut se faire entre ACS, ACS 2, YACS, MACS, XACS ou bien d'une nouvelle proposition de mise en œuvre du contrôle comportemental anticipatif qui caractérise tous ces systèmes. Le cadre des systèmes de classeurs est suffisamment malléable pour adapter des extensions d'un ALCS à un autre ALCS. Il semble ainsi plus pertinent de s'appuyer sur les capacités des ALCS existants à évoluer dans de tels environnements, afin de déterminer si de nouvelles extensions ou si une nouvelle proposition d'ALCS seraient nécessaires. Nous proposons alors de nous appuyer sur ACS 2.

ACS 2 par rapport à ACS

Le choix d'ACS 2 par rapport à ACS permet de disposer d'une population de classeurs qui représente l'ensemble des transitions environnementales déjà observées par le système, que des changements perceptifs soient associés ou non à ces transitions. Les représentations environnementales construites par ACS 2 permettent ainsi de renseigner plus d'informations sur le comportement appris par ce système durant la résolution de ses tâches d'apprentissage. Par ailleurs, les mécanismes de généralisation génétique et de subsomption permettent à ACS 2 d'assurer un contrôle plus fin de l'évolution de sa population de classeurs en limitant l'accroissement du nombre de ses classeurs, ce qui facilite l'explicabilité des représentations environnementales et des politiques de décision développées durant l'apprentissage. Bien que les capacités d'ACS 2 à évoluer dans des environnements incertains soient plus faibles qu'ACS, les séquences comportementales et les prédictions améliorées par les probabilités n'ayant jamais été intégrées à ACS 2, la proximité de ces deux ALCS ne prévient pas leur intégration.

ACS 2 par rapport à YACS et MACS

YACS et MACS gardent tous deux en mémoire toutes les observations de leurs environnements : soit au travers d'une liste leur permettant de construire leurs politiques de décision ; soit au travers des traces propres à chaque classeur et de longueur à définir par l'utilisateur. Les processus de spécification et de généralisation des *Conditions* de leurs classeurs s'appuient en particulier sur ces mémoires des observations. Conceptuellement, garder explicite-

ment en mémoire toutes ces observations semble difficilement concevable, d'autant plus si les environnements sont incertains.

D'une part, ces systèmes ne sont pas capables de prendre des décisions pour des situations jamais observées, qu'elles soient nouvelles si l'environnement est volatil, ou que l'exploration de l'environnement n'ait permis de les atteindre. Qui plus est, la construction d'une situation à anticiper avec MACS fait intervenir un ensemble de classeurs, ce qui complexifie la prédiction d'une récompense si elle devait être faite par ses classeurs : cette prédiction serait-elle réalisée au niveau de chacun des classeurs, au niveau de chaque ensemble de classeurs utilisé pour prédire une anticipation, ou autrement ?

D'autre part, la construction des classeurs de YACS et MACS est fortement dépendante de la configuration de la longueur des traces de leurs classeurs. Les processus de spécification et de généralisation de ces systèmes ne sont déclenchés que si les traces de leurs classeurs sont complètes. Il est alors nécessaire que les différentes situations décrites par les *Conditions* des classeurs soient atteintes, ce qui est fortement dépendant des processus d'exploration des environnements ainsi que des propriétés des environnements. En particulier, l'emploi de ces traces peut être délicat si les environnements sont volatils et stochastiques : un équilibre potentiellement inaccessible est à déterminer entre une longueur de traces suffisamment faible qui permettrait de vite adapter les classeurs à leurs environnements et en même temps, une longueur de traces assez élevée qui permettrait de rassembler les informations nécessaires pour adapter les classeurs à leurs environnements de manière fiable. Toutefois, l'idée d'un calcul d'indices permettant de contrôler la généralisation ou la spécification d'attributs perceptifs des *Conditions* pourrait être adaptée afin de fournir plus d'informations sur l'intérêt des attributs pour décrire les situations environnementales.

Enfin, l'emploi de ces systèmes nécessite des connaissances à propos des environnements où ces systèmes réalisent leur apprentissage, ce qui est dû à l'opérateur de spécification utilisé. Cet opérateur spécifie un attribut général # de la *Condition* d'un classeur et construit autant de classeurs que de valeurs que l'attribut à spécifier peut avoir (Dorigo, 1993). L'ensemble des valeurs que peut prendre chaque attribut perceptif doit ainsi être connu en amont d'un apprentissage, ce qui est à nouveau difficilement concevable pour des

environnements incertains à cause de leur volatilité : toute nouvelle situation peut ne pas avoir été envisagée avant de réaliser un apprentissage. De plus, si un attribut perceptif peut prendre un grand nombre de valeurs, un mécanisme plus fin de spécialisation tel que celui proposé par ACS 2 permet d'éviter la construction d'un nombre important de classeurs et par extension, de limiter l'accroissement de la population de classeurs.

ACS 2 par rapport à XACS

XACS n'est adapté aux environnements incertains que si ACS 2 est capable de se doter d'une représentation environnementale complète dans ce type d'environnements. Il semble alors plus naturel de commencer par étudier les capacités d'ACS 2 à évoluer dans ces environnements, avec par exemple les séquences comportementales ou les prédictions améliorées par les probabilités issues d'ACS, avant d'étudier différents processus de rétribution.

3.4 Applications des systèmes de classeurs à anticipation

Les systèmes de classeurs à anticipation ont été utilisés pour résoudre de nombreux problèmes, qu'il s'agisse de problèmes de classification ou de problèmes d'apprentissage par renforcement, de problèmes à données discrètes ou de problèmes à données réelles. Le tableau 3.1 résume les problèmes où les ALCS ont été employés.

Monk's problems

Monk's problems (UCI, 1992) désigne un ensemble de problèmes de classification binaire (0 ou 1) à données discrètes. Chaque donnée des jeux d'apprentissages est constituée de six attributs où :

- le premier, le second et le quatrième attribut peuvent valoir 1, 2 ou 3 ;
- le troisième et le sixième attribut peuvent valoir 1 ou 2 ;
- le cinquième attribut peut valoir 1, 2, 3 ou 4.

Il s'agit alors, pour un algorithme d'apprentissage automatique, de trouver

	Problèmes à données discrètes	Problèmes à données réelles
Problèmes de classification	*Monk's problem* Multiplexeur *Voting Record* *Wisconsin Breast Cancer*	*Checkerboard* Multiplexeur
Problèmes d'apprentissage par renforcement	*Finite State World* *Frozen Lake* Go *Gripper* Labyrinthes Taxi	*CartPole* *Corridor* *Grid*

TABLE 3.1 – Applications des systèmes de classeurs à anticipation selon le type de problème et la nature des données associées à ces problèmes.

les règles permettant de déterminer si une donnée correspond à la classe 1 à partir de ces six attributs. Plusieurs variants de ce problème existent. Par exemple, une donnée est de classe 1 dans *Monk's 2* si uniquement deux attributs valent 1. Dans *Monk's 3*, une donnée est de classe est 1 lorsque le cinquième attribut est égal à 3 et le quatrième attribut est égal à 1, ou le cinquième attribut n'est pas égal à 4 et le deuxième attribut n'est pas égal à 3. (Unold and Tuszyński, 2008) ont montré que les systèmes de classeurs à anticipation sont capables de résoudre ces problèmes, bien qu'ils soient initialement destinés pour des tâches d'apprentissage à plusieurs étapes.

Voting Record

Voting Record (UCI, 1987) est un problème de classification binaire à données discrètes, où il est possible de corréler l'appartenance à un parti (*D* démocrate ou *R* républicain) à partir de votes (*y* oui, *n* non ou ? inconnu) sur 16 attributs différents. À nouveau, (Unold and Tuszyński, 2008) ont montré que les ALCS sont capables de résoudre ce problème.

Wisconsin Breast Cancer

Wisconsin Breast Cancer (Wolberg, 1992) est un problème de classification binaire à données discrètes, où il est possible de déterminer le caractère bénin ou malin de cellules liées au cancer du sein à partir de 9 attributs. Ces attributs détaillent différentes caractéristiques des cellules sur une échelle de valeurs

entières allant de 1 à 10. A nouveau, (Unold and Tuszyński, 2008) ont montré que les ALCS sont capables de résoudre ce problème.

Multiplexeur

Le problème du multiplexeur est un problème de classification binaire pouvant être à données discrètes ou à données réelles. Une donnée d'un multiplexeur est composée de deux sous-ensembles de données : un premier associé à l'adressage et un second associé aux registres. Les valeurs associées à l'adressage permettent d'identifier un registre et de transmettre en sortie le contenu du registre désigné. Un système d'apprentissage automatique doit alors apprendre à corréler ces ensembles d'adressage vers les ensembles de registres. La figure 3.21 illustre par exemple un multiplexeur discret à 11 bits et son comportement au travers de deux exemples.

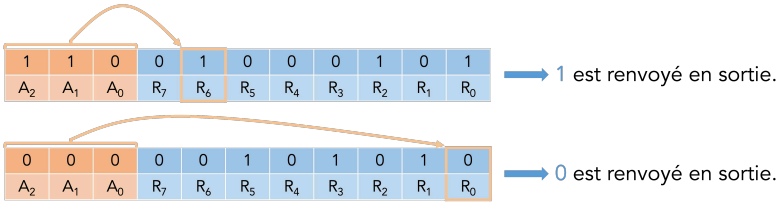


FIGURE 3.21 – Illustration du comportement d'un Multiplexeur discret à 11 bits.
Par exemple, dans le multiplexeur à 11 bits, trois bits d'adressage (A_0, A_1 et A_2) pointent vers un bit de registre dont la valeur correspondra à la sortie (R_0, R_1, ..., R_7). Si $A_0 = A_1 = A_2 = 0$, alors le registre R_0 est désigné et son contenu est renvoyé.

Les problèmes de multiplexeur sont des problèmes classiquement utilisés pour vérifier les capacités d'apprentissage des systèmes de classeurs. Pour les ALCS, les multiplexeurs à données discrètes ont été utilisés par (Butz et al., 2002) et (Kozlowski and Unold, 2018), et ceux à données réelles ont été employés par (Kozlowski and Unold, 2019), (Kozlowski and Unold, 2020) et (Kozlowski and Unold, 2022). Les ALCS sont capables de résoudre ces problèmes pour des multiplexeurs de faibles dimensions. Cependant, ces problèmes ont permis de mettre en lumière les difficultés des ALCS à faire évoluer des populations de classeurs utilisant des représentations continues, ainsi qu'à maîtriser l'évolution de leur population de classeurs plus les espaces de recherche relatifs aux données sont grands.

Checkerboard

Checkerboard est un problème de classification binaire à données réelles. Un espace fini de dimension n est scindé en des hypercubes de mêmes côtés. Chaque hypercube possède une couleur blanche ou noire en alternance sur chacune des dimensions. La figure 3.22 illustre un *Checkerboard* de dimension 2 scindé en 2 sur chacune de ces dimensions. Un algorithme d'apprentissage doit alors apprendre à déterminer la couleur en toute position de cet espace. Lors d'expériences pour permettre aux ALCS de gérer des données à valeurs réelles, (Kozlowski and Unold, 2019) montrent que les ALCS sont capables d'apprendre des règles à partir de ces environnements. Cependant, les ALCS n'ont pas été capables de faire évoluer des classeurs de confiance et leurs populations croient de façon incontrôlée.

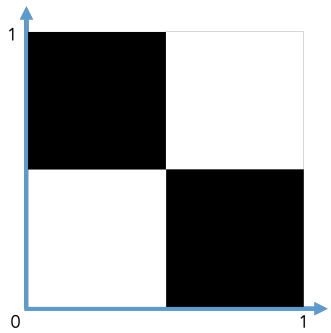


FIGURE 3.22 – Illustration d'un *Checkerboard* de dimension 2 scindé en 2 sur chacune de ces dimensions.

Finite State World

Finite State World désigne un ensemble d'environnements à longues séquences d'actions nécessaires pour atteindre une récompense. Ces environnements sont constitués de nœuds, chacun identifié par une unique étiquette, et d'arcs étiquetés par les actions qui permettent de lier les différents nœuds. La figure 3.23 illustre un *Finite State World* de longueur n où deux actions sont possibles.

(Kozłowski and Unold, 2021) utilisent ces environnements pour étudier différents mécanismes de renforcement intégrés dans les systèmes de classeurs à anticipation. Les ALCS ont alors été capables d'atteindre les performances optimales pour un *Finite State World* de longueur 10 : ils sont capables de

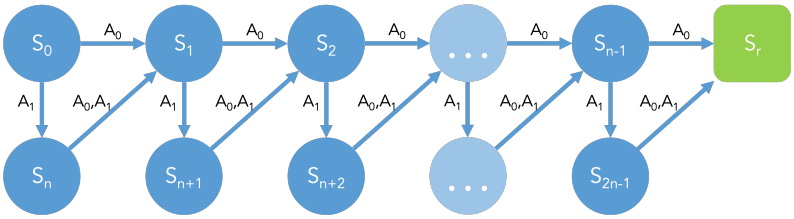


FIGURE 3.23 – Illustration d'un *Finite State World* de longueur n.
Les effets des actions A_0 et A_1 sont décrits par les arcs de transitions : par exemple, A_0 permet de passer de S_0 à S_1.

parcourir le chemin le plus court pour atteindre la récompense.

Frozen Lake

Les environnements *Frozen Lake* sont des environnements sous forme de grille en deux dimensions où un agent contrôle un personnage pouvant évoluer sur un lac gelé. Ce lac gelé est constitué de cellules sur lesquelles il peut marcher, d'autres où il tomberait à l'eau (entraînant un échec et la fin d'une tentative pour atteindre son objectif) et de cellules cibles où il doit se rendre. La particularité de ces environnements est que les actions réalisées par un agent sont incertaines : l'action effectivement réalisée peut être différente de celle choisie initialement, parce qu'un agent peut glisser à cause du *gel* en tentant de se déplacer.

(Kozlowski and Unold, 2018) ont testé les capacités d'un ALCS à évoluer dans ces environnements. Cependant, leurs essais ont été infructueux puisque leur ALCS ne disposait pas de mécanismes lui permettant de gérer l'incertitude de ces environnements, en particulier la stochasticité.

Go

Des tentatives avec le jeu de Go existent également dans la littérature (Kozlowski and Unold, 2018). Le jeu de Go est un jeu de stratégie chinois à deux joueurs qui se jouent avec des pierres noires et blanches. Le but est de capturer le plus grand nombre de pierres adverses et de cases sur une grille où ces pierres sont jouées. Cette grille est traditionnellement constituée d'un quadrillage de 19 lignes verticales et 19 lignes horizontales, qui déterminent 361 intersections sur lesquelles les pierres peuvent être jouées. La grille peut

être réduite à 13 lignes verticales et 13 lignes horizontales ou 9 lignes verticales et 9 lignes horizontales pour des parties plus rapides. Apprendre à un algorithme à jouer au jeu de Go est une tâche particulièrement difficile tant la combinatoire de ce problème est importante, excluant toute recherche exhaustive de solutions. Les tentatives d'apprentissage du jeu de Go réalisées par (Kozlowski and Unold, 2018) ont ainsi été infructueuses sur un plateau de 9 par 9 lignes, dont les causes peuvent tant résider dans l'approche pour réaliser l'apprentissage, que dans les capacités de l'ALCS utilisé pour faire évoluer sa population de classeurs.

Gripper

Les problèmes du *Gripper*, aussi appelés *Block World* ou *Hand Eye Coordination* désignent des problèmes qui consistent à déplacer des blocs vers des positions objectif par le biais d'un bras robotique. Ces problèmes ont permis d'étudier les capacités des systèmes de classeurs à modéliser ces problèmes, ainsi qu'à les résoudre, au travers des mécaniques de généralisation et de planification d'actions. (Stolzmann and Butz, 1999), (Butz, 2001), (Butz et al., 2002) et (Unold et al., 2019) montrent alors que les ALCS sont effectivement capables de résoudre cette tâche d'apprentissage et comment ces mécaniques permettent d'accroître leurs capacités d'apprentissage.

Taxi

L'environnement taxi est une grille prédéfinie en deux dimensions et illustrée par la figure 3.24. Un agent pilote un taxi dont le but est de réceptionner des passagers sur l'une des quatre cellules dédiées, et de l'amener le plus rapidement possible à sa destination représentée par une autre de ces quatre cellules (donc différente de celle où le passager a été réceptionné).

Cet environnement a été étudié par (Unold et al., 2019). Leur ALCS n'est cependant pas capable de se doter d'une représentation complète de cet environnement, ni d'accomplir la tâche d'apprentissage : leurs résultats indiquent que leur agent ne réalise pas d'actions pour lesquelles il serait pénalisé, sans pour autant parvenir à conduire à bon port le passager.

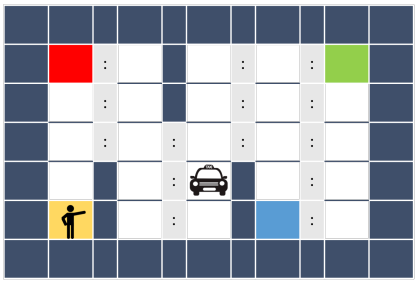


FIGURE 3.24 – Illustration de l'environnement Taxi.
Un passager est représenté par la silhouette d'un être humain. Le taxi est représenté par la face avant d'un véhicule. Les chemins sont illustrés par des cellules blanches. Les murs sont représentés par des cellules foncées. Les cellules grises permettent le passage du taxi entre deux chemins : elles ne peuvent pas contenir de passager ou de véhicule. Les cellules rouges, jaunes, vertes et bleues désignent les cellules où un taxi peut réceptionner ou déposer un passager.

Corridor et *Grid*

Le problème du *Corridor* est un problème à une dimension où un système doit se déplacer le plus possible vers la droite. Ce problème a été initialement posé avec des valeurs réelles comprises entre 0 et 1, qu'il est possible de discrétiser en n valeurs. Le déplacement d'un agent vers la droite ou la gauche correspond alors à incrémenter ou décrémenter d'un pas (prédéfini en amont) la position courante de l'agent. Le problème du *Grid* est le pendant du *Corridor* avec deux dimensions. Ces problèmes ont été utilisés par (Kozlowski and Unold, 2020), (Kozłowski and Unold, 2021) et (Kozlowski and Unold, 2022) lorsque ces derniers ont étudié les mécanismes de renforcement de leurs ALCS, ainsi que différentes techniques d'exploration des environnements et leurs capacités à accumuler des connaissances au sein des populations sans renforcement.

CartPole

Le problème du *CartPole* est un problème d'apprentissage par renforcement classique, consistant à maintenir en équilibre un bâton sur un chariot en ne pouvant déplacer ce chariot que vers la droite ou la gauche. La figure 3.25 illustre l'ensemble les éléments constitutifs ce de problème. Seuls (Kozlowski and Unold, 2020) ont tenté, sans succès, de résoudre ce problème avec un système de classeurs à anticipation : leur ALCS n'est pas capable de maintenir pendant suffisamment de « temps » ce bâton en équilibre sur le chariot.

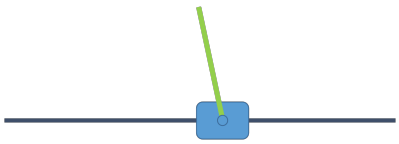


FIGURE 3.25 – Illustration du problème du *CartPole*.
Un mobile bleu peut glisser sur un axe horizontal (représenté en bleu gris) afin de maintenir en équilibre un bâton vert, ce dernier étant fixé sur le chariot par un pivot.

Labyrinthes

Les labyrinthes font partie des environnements ayant été les plus utilisés lors du développement des systèmes de classeurs à anticipation. Ces environnements ont été utilisés par exemple par (Stolzmann et al., 1998), (Stolzmann and Butz, 1999), (Butz, 2001), (Butz et al., 2002), ou encore (Kozlowski and Unold, 2022). Les labyrinthes prennent le plus souvent la forme de grilles en deux dimensions, où chaque cellule de la grille correspond à un élément du labyrinthe tel qu'un mur, un objectif ou encore un chemin. Les buts des ALCS dans les labyrinthes sont de se doter d'une représentation de ces environnements et d'être capables d'atteindre les cellules objectifs en un nombre minimum d'actions. Un exemple de labyrinthe est illustré par la figure 3.26.

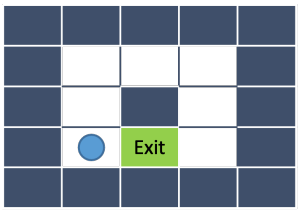


FIGURE 3.26 – Exemple de labyrinthe comprenant trois types de cellules.
Un agent est ici représenté par le point bleu. La cellule objectif est représentée par une cellule verte. Les chemins sont illustrés par des cellules blanches. Les murs sont représentés par des cellules foncées.

La section 4.1 aborde plus en détail ces environnements puisqu'ils ont été majoritairement utilisés dans le cadre des contributions de cette thèse. Par ailleurs, les environnements *Frozen Lake* peuvent être considérés comme des cas particuliers de labyrinthes où les actions réalisables par un agent sont incertaines.

ACS 2 a été le système de classeurs à anticipation principalement utilisé pour résoudre ces différents problèmes d'apprentissage, ce qui a permis de mieux cerner les avantages et limites liés à son utilisation.

3.5 Avantages et limites d'ACS 2

3.5.1 Gestion de l'incertitude

ACS 2 est capable d'évoluer dans des environnements incertains dans une certaine mesure. (Butz, 2001) montre que cet ALCS est capable d'adapter sa population de classeurs, autrement dit sa représentation de l'environnement et sa politique de décision, si la situation à atteindre dans un environnement est dynamiquement modifiée. ACS 2 s'adapte à de nouvelles situations environnementales en spécifiant au besoin des classeurs de sa population avec son processus d'apprentissage par anticipation, en construisant de nouveaux classeurs avec son processus de recouvrement ou même, sans aucune modification de sa population, grâce à son mécanisme de généralisation qui permet de former des points d'attention qui caractérisent des ensembles de situations environnementales. Cependant, peu d'études ont été réalisées si les tâches d'apprentissage venaient à être modifiées ou si les changements des environnements étaient plus complexes (par exemple plus nombreuses ou plus variées) : des analyses plus approfondies sont alors nécessaires pour clarifier les capacités d'apprentissage d'ACS 2 dans des environnements volatils.

Deux mécanismes ont été développés pour améliorer les capacités d'apprentissage d'ACS dans des environnements stochastiques : les séquences comportementales (BSeq) et les prédictions améliorées par les probabilités (PEP). Ceux-ci ont été étudiés dans un nombre très limité d'environnements et jamais intégrés à ACS 2. Ces deux approches permettent à ACS de gérer des formes différentes de stochasticité et à des fins différentes : les BSeq permettent de mettre en place une politique de décision face au problème d'aliasing perceptif ; les PEP permettent de construire une représentation environnementale quand les observations de l'environnement sont bruitées. Si ces approches sont intégrées dans ACS 2, les résultats obtenus avec ACS pourraient être généralisés au travers d'un banc de test comprenant des environnements aux caractéristiques variées. Ce banc permettrait alors de vérifier entre autres : l'impact des longueurs des séquences comportementales sur la construction des politiques de décision et des populations de classeurs ; l'impact du couplage des BSeq et des PEP avec les mécanismes de générali-

sation et de subsomption de classeurs d'ACS 2 ; la cohérence des probabilités des PEP avec les environnements. Ces mécanismes n'ont par ailleurs jamais été utilisés simultanément dans un unique ALCS, ce qui pourrait encore plus accroître les capacités d'ACS 2 à évoluer dans des environnements stochastiques.

3.5.2 Contrôle des populations de classeurs

Le contrôle des populations de classeurs est réalisé dans ACS 2 par le biais de son mécanisme de généralisation génétique et de son processus de subsomption. Ces derniers limitent l'accroissement de la population de deux façons différentes : en évitant d'ajouter des classeurs qui décriraient des transitions environnementales déjà connues et fiables ; en limitant la taille des ensembles d'action construits par le système, ce qui limite au maximum la taille globale de la population de classeurs selon le produit du nombre de situations environnementales différentes par le nombre d'actions réalisables. Cependant, un contrôle plus fin de sa population peut être nécessaire, en particulier si de nouveaux mécanismes sont ajoutés à ACS 2 ou si un environnement d'apprentissage présente un nombre important de situations possibles. Par exemple, il est admis que les séquences comportementales induisent une forte hausse du nombre de classeurs (Métivier and Lattaud, 2002), car le nombre de combinaisons d'actions réalisables suit une évolution exponentielle dépendant de la longueur maximale de ces séquences. Un accroissement de la population de classeurs entraîne une multiplicité des règles qui peut devenir un frein à l'explicabilité inhérente à ce système. Les techniques d'extraction de connaissance, potentiellement couplées à des techniques de contrôle en lien avec ces nouveaux mécanismes, peuvent alors favoriser l'explicabilité des solutions développées par ACS 2.

3.5.3 Processus de rétribution

La mise en place d'une politique de décision dans ACS 2 est dépendante de son modèle de l'environnement, ce qui peut alors poser des difficultés si son modèle environnemental est représentable de manière plus généra-

lisée que la politique de décision permettant effectivement de résoudre la tâche d'apprentissage. Pour pallier cette limite, l'emploi d'un second système de classeurs dédié à l'apprentissage des politiques décisionnelles (Butz and Goldberg, 2003), l'adaptation de techniques d'apprentissage par renforcement (Kozłowski and Unold, 2021) ou d'un opérateur de spécification (Dorigo, 1993) ont été proposés. D'autres solutions sont également possibles en considérant la mesure de l'adéquation d'un classeur à son environnement. La *fitness* des classeurs d'ACS 2 s'appuie en particulier sur le produit de la *force* des prédictions des récompenses par la qualité des anticipations. La *précision* des prédictions des récompenses pourrait être préférée à leur *force* pour s'assurer de leur exactitude, ce qui fournirait de nouveaux indicateurs à propos de la mise en place des politiques de décision. Ces indicateurs participeraient aussi à la conception de nouveaux processus de rétribution qui seraient potentiellement plus adéquats aux environnements dont les récompenses environnementales peuvent être incertaines.

3.5.4 Dilemme exploration-exploitation

Le dilemme exploration-exploitation désigne le compromis entre l'acquisition de nouvelles connaissances et l'utilisation des connaissances acquises pour résoudre une tâche d'apprentissage (Berger-Tal et al., 2014). Ce dilemme reste un problème ouvert et dépasse le cadre des systèmes de classeurs. Dans le cas des ALCS, les recherches autour de ce dilemme ont mené à de nouvelles techniques qui cherchent à tirer parti des anticipations de ces systèmes comme (Gérard and Sigaud, 2003) ou (Unold et al., 2019). Les anticipations peuvent permettre une adaptation plus rapide des populations de classeurs dans des environnements dynamiques (Butz et al., 2003), ce qui peut simplifier la mise en place d'un compromis entre exploration et exploitation : une acquisition plus rapide des connaissances limite les risques d'une exploitation infructueuse due à un manque de connaissances. Dans le cas d'ACS 2 (comme pour XACS (Butz and Goldberg, 2003)), ce dilemme est double puisqu'une représentation complète de son environnement est recherchée, ainsi qu'une politique de décision construite à partir de cette dernière. Mais toutes les techniques proposées (planification, biais de rappel, biais d'erreur, apprentissage onirique, …) n'ont pas encore été étudiées simultanément, ou

unifiées dans une unique approche dans le but de rendre le système plus autonome dans l'acquisition et l'utilisation de ses connaissances.

3.5.5 Deux principes à déprécier

Une limitation importante d'ACS 2 concerne les changements que les classeurs décrivent dans leur *Anticipation* : tous ceux-ci sont associés à la réalisation de l'action décrite par le classeur et réalisée par le système. Or, tous les changements perçus peuvent ne pas être du fait de l'activité du système, à l'image du temps qui passe ou de la météo qui évolue. Il peut donc être difficile de savoir quels sont effectivement les changements causés par l'interaction d'ACS 2 avec son environnement, ce qui est d'autant plus délicat si une approche multiagent est considérée. De nouveaux mécanismes sont alors nécessaires pour identifier l'origine des changements perçus.

En conjonction avec la limitation précédente, ACS 2 intègre un biais d'apprentissage qui favorise les classeurs qui anticipent des changements perceptifs. Autrement dit, ACS 2 cherche à employer des actions qui vont avoir des conséquences perceptibles sur son environnement. Ce biais intervient lors de la sélection d'une action qui serait la plus adéquate à la résolution d'une tâche d'apprentissage et lors de la rétropropagation des récompenses des classeurs. Or, il peut tout autant aider à la résolution d'une tâche d'apprentissage qu'il peut prévenir sa résolution. Par exemple, ce biais peut favoriser les actions qui vont permettre au système d'atteindre des situations différentes dans des labyrinthes, quand la tâche d'apprentissage consiste à atteindre la sortie de ces environnements. À l'inverse, ce biais peut poser problème lors de la résolution du problème du *cartpole*, puisque le bâton doit être maintenu en équilibre et ne plus bouger sur le chariot. La présence de ce biais d'apprentissage doit alors a minima être prise en considération quand ACS 2 doit réaliser un apprentissage, et son impact effectif devrait être analysé plus en détail.

3.5.6 Contributions

La mise en place d'une intelligence artificielle autonome et explicable pour des environnements incertains s'appuie alors sur ACS 2, le système de classeurs à anticipation de référence (Pätzel et al., 2020). Cependant, il n'intègre pas d'extensions lui permettant de se doter de modèles complets de transitions ou de politiques de décision adaptées dans des environnements incertains.

Nous proposons alors d'intégrer les séquences comportementales à ACS 2 afin d'accroître sa capacité à se doter de règles lui permettant effectivement de résoudre ses tâches d'apprentissage dans des environnements incertains, en particulier s'il est confronté au problème d'aliasing perceptif. L'intégration des BSeq à ACS 2 est l'objet du chapitre 5.

Nous proposons également d'intégrer les prédictions améliorées par les probabilités à ACS 2 pour qu'il puisse construire des représentations complètes d'environnements incertains. De telles représentations ont un objectif double : une description complète de l'interaction du système avec son environnement permet de mieux cerner les *causes* et les *conséquences* liées aux actions réalisées par le système ; ces représentations sont utilisées pour construire les politiques de décision et peuvent donc favoriser la mise en place de politiques permettant de résoudre les tâches d'apprentissage. L'intégration des PEP à ACS 2 est l'objet du chapitre 6.

Les séquences comportementales et les prédictions améliorées par les probabilités sont ensuite couplées pour la première fois dans un unique ALCS. Ce couplage est proposé à partir des résultats présentés dans les chapitres 5 et 6 qui permettent de mettre en lumière des axes d'amélioration pour chacune de ces extensions. Les capacités d'apprentissage, d'autonomie et d'explicabilité du système résultant de ce couplage sont l'objet du chapitre 7.

Nous mettions également en avant le besoin de contrôle des populations de classeurs qui sont élaborées par les ALCS, en particulier quand ils sont en mesure de construire de nouveaux classeurs avec les BSeq ou les PEP. Nous proposons dans le chapitre 8 le premier algorithme de compression des populations de classeurs adapté aux ALCS afin de limiter la multiplicité des clas-

seurs qui peut devenir un frein à l'explicabilité intrinsèque de ces systèmes.

Avant de présenter ces contributions, nous introduisons le banc de test d'environnements labyrinthiques que nous avons mis en place pour étudier nos modèles dans le chapitre 4 suivant. Les labyrinthes ont été choisis pour leur malléabilité et leur diversité, permettant en conséquence de disposer aisément d'un grand nombre d'environnements aux propriétés différentes et de contrôler l'incertitude inhérente. Ils ont également été choisis parce qu'il s'agit d'environnements de référence dans le domaine des systèmes de classeurs, ce qui facilite une comparaison entre différents systèmes de classeurs. Le protocole expérimental permettant l'étude des systèmes de classeurs à anticipation est aussi présenté dans ce chapitre, puisqu'il s'appuie sur ce banc de test.

Deuxième partie

Contributions

CHAPITRE 4

Protocole d'évaluation expérimental des systèmes de classeurs à anticipation

Ce chapitre présente les environnements d'apprentissage, les principales métriques d'évaluation et le protocole expérimental des systèmes de classeurs à anticipation pour évaluer les capacités d'apprentissage de ces systèmes. Ce protocole est commun à tous les systèmes de classeurs à anticipation évalués, afin de permettre une étude la plus équitable possible entre ces différents systèmes.

Sommaire

4.1 Environnements labyrinthiques

Historiquement, les environnements labyrinthiques sont les environnements principalement utilisés en tant que problèmes d'apprentissage par renforcement de référence pour les systèmes de classeurs (Bagnall and Zatuchna, 2005).

4.1.1 Définition des labyrinthes

Les labyrinthes sont décomposables en un ensemble de cellules prenant le plus souvent la forme d'une grille en deux dimensions. Chacune de ces cel-

lules correspond à un élément du labyrinthe, tel qu'un mur, un chemin, une sortie, un obstacle, des objets, des agents, des compositions d'éléments *etc.* Un agent peut réaliser différentes actions au sein d'un labyrinthe qui vont définir ses interactions avec chacune des cellules : il est alors possible de représenter un labyrinthe comme un graphe de transition où à chaque nœud correspond une cellule et à chaque arc correspond une action réalisable par un agent dans ces environnements. D'autres éléments permettent de constituer des labyrinthes sans que ces derniers correspondent à une cellule : il est possible d'avoir des éléments qui sont définis par les arcs de transition. Par exemple, la figure 4.1 illustre deux exemples de labyrinthes en présence de deux murs différents : un mur peut à la fois correspondre à une cellule, ou bien à un élément prévenant le passage d'une cellule de type chemin à une autre (auquel cas il peut correspondre à l'absence d'un arc entre deux nœuds).

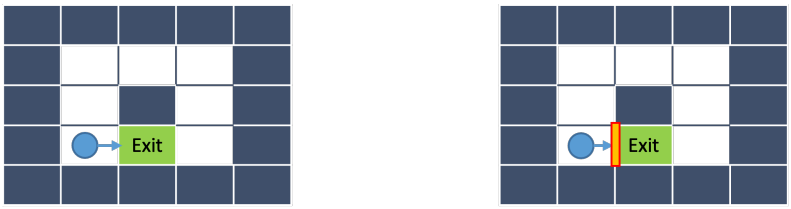


Figure 4.1 – Exemple de labyrinthes comprenant trois types de cellules. *Un agent est ici représenté par le point bleu et essaie d'aller vers la droite pour atteindre la sortie (cellule verte). Les chemins sont illustrés par des cellules blanches. Les murs sont représentés par des cellules bleues grises ou par un bloc orange entre deux cellules, rendant l'action de l'agent impossible à réaliser s'il tente de les traverser.*

Ces environnements sont facilement manipulables pour tester les capacités d'apprentissage des systèmes de classeurs, et permettent aussi de faciliter les comparaisons entre différents systèmes de classeurs. De nouvelles cellules et interactions peuvent être facilement définies afin de concevoir de nouveaux labyrinthes et de nouvelles tâches d'apprentissages. Et selon les tâches d'apprentissage, les renforcements attribués à chacune des transitions entre les cellules sont facilement manipulables. C'est pourquoi il s'agit d'environnements de référence pour tester les capacités d'apprentissage de nombreux systèmes, dont les systèmes de classeurs.

Des propriétés propres à ces environnements peuvent alors être définies, indépendamment d'une tâche d'apprentissage ou des capacités d'un agent. Le nombre total de cellules permet de définir la taille d'un labyrinthe. La den-

sité d'un labyrinthe désigne le rapport du nombre de cellules de type *mur* sur l'ensemble des cellules. Des cellules peuvent avoir une dynamique leur permettant de modifier leur type : une cellule de type *chemin* peut alors basculer vers le type obstacle et ainsi de suite.

Le but d'un agent dans ces environnements est traditionnellement d'atteindre en un nombre minimum d'actions une cellule destination pouvant être désignée comme la sortie, et ceci à partir de n'importe quelle cellule où un agent peut évoluer. Cette cellule destination a été nommée de différentes manières dans la littérature : par exemple, (Bagnall and Zatuchna, 2005) associent à cette cellule de la nourriture, dont le but pour un agent est également de l'atteindre aussi vite que possible. Les capacités des agents permettent alors de caractériser plus en détail ces environnements. Les capacités des agents désignent les actions qu'ils peuvent réaliser au sein de ces environnements, ainsi que leur perception.

4.1.2 Capacités des agents

Les agents peuvent réaliser *a minima* 4 ou 8 différentes actions dans les labyrinthes, à savoir les quatre directions cardinales avec ou sans les quatre directions intercardinales, leur permettant ainsi de se mouvoir. Ces actions permettent de définir la distance moyenne minimale à la sortie dans chacun des labyrinthes. À partir de chacune des cellules, il est possible de déterminer la plus petite séquence d'actions permettant d'atteindre la sortie. Cette distance moyenne minimale à la sortie peut fournir une information quant à la difficulté de la tâche à résoudre : plus cette distance moyenne est grande, plus la tâche d'apprentissage est difficile. Intuitivement, il est plus facile d'apprendre à atteindre la sortie d'un labyrinthe si une unique action est nécessaire, que d'apprendre à atteindre cette sortie alors qu'une vingtaine d'actions est nécessaire.

Par ailleurs, il est possible de bruiter la réalisation des actions : une autre action peut être réalisée par un agent selon une probabilité définie en amont. Ce bruit permet de modéliser de l'incertitude au sein des environnements labyrinthiques, et de complexifier la résolution de la tâche d'apprentissage par l'introduction de stochasticité. Le bruit des actions n'augmente pas la

distance moyenne minimale à la sortie, celle-ci ne dépendant que des actions possiblement réalisables et de la configuration des labyrinthes. En revanche, il est possible d'approximer le nombre moyen minimal d'actions nécessaires pour parvenir à la sortie en tenant compte de cette incertitude : ce nombre est la distance moyenne minimale à la sortie sous incertitude.

Les perceptions des agents désignent leur capacité à se former une image de leur environnement. Leurs capacités perceptives peuvent être limitées : ils peuvent être incapables de percevoir l'ensemble de leur environnement, mais uniquement les cellules adjacentes. Par exemple, leur perception peut être réduite au voisinage de Moore d'ordre 1 à la position courante, comme illustrée par la figure 4.2. Il est alors dit que les labyrinthes sont des environnements partiellement observables. Ce voisinage de Moore d'ordre 1 est ici représenté par une séquence de symboles, en commençant par la cellule au-dessus de la position courante, puis en représentant les autres cellules en allant dans le sens des aiguilles d'une montre. Cette description d'une perception est utilisée avec tous les ALCS que nous étudions dans les chapitres suivants.

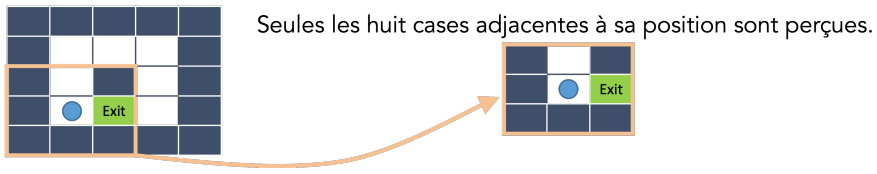


Figure 4.2 – Perception d'un agent limitée au voisinage de Moore d'ordre 1 à la position courante dans un labyrinthe.
La représentation des labyrinthes est identique à celle décrite par la figure 4.1. Cette perception est décrite par la séquence suivante : {Chemin, Mur, Exit, Mur, Mur, Mur, Mur, Mur}.

Atteindre la sortie de labyrinthes partiellement observables peut être plus délicat que si les agents avaient à leur disposition une perception complète de ceux-ci. Des observations partielles peuvent amener un agent à devoir gérer le problème d'aliasing perceptif (en anglais Perceptual Aliasing Issue - PAI). Le PAI survient quand un agent n'est pas capable de distinguer deux situations environnementales distinctes à partir de ses capacités perceptives, parce que les perceptions de ces situations sont identiques, comme l'illustre la figure 4.3. Le PAI complexifie alors la mise en place d'une politique de décision permettant d'atteindre le plus rapidement la sortie : pour cette même

perception, l'agent doit idéalement aller à droite ou à gauche selon la situation environnementale dans laquelle il se trouve réellement.

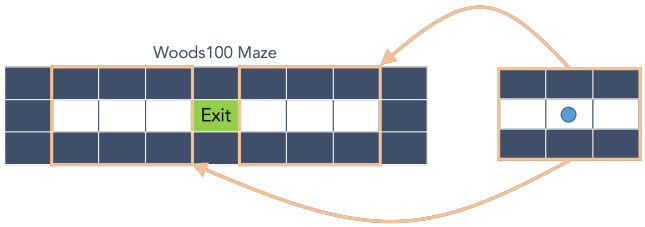


FIGURE 4.3 – Illustration du problème d'aliasing perceptif dans le labyrinthe Woods100.
La représentation des labyrinthes est identique à celle décrite par la figure 4.1. Un agent qui a cette perception de l'environnement peut se trouver en deux situations environnementales bien distinctes dans l'environnement Woods100. Ces deux situations environnementales sont perçues par le biais de cette séquence de symboles : {Mur, Mur, Chemin, Mur, Mur, Mur, Chemin, Mur}.

Les perceptions peuvent également être bruitées par l'ajout d'éléments perceptifs inexistants ou par la randomisation d'éléments perceptifs, introduisant à nouveau de la stochasticité dans les environnements labyrinthiques. Des perceptions bruitées ou le problème d'aliasing perceptif sont alors à prendre en compte pour déterminer la distance moyenne minimale à la sortie sous incertitude.

4.1.3 Complexité des labyrinthes

Quand un agent doit atteindre le plus rapidement possible la sortie des labyrinthes, la complexité des labyrinthes permet de caractériser la difficulté d'un agent à résoudre cette tâche d'apprentissage (Bagnall and Zatuchna, 2005). La complexité dépend principalement des caractéristiques des labyrinthes (taille, densité, distance moyenne à la sortie, structure) et des capacités des agents à interagir (actions possibles, actions bruitées) ou percevoir leur environnement (observations partielles, observations bruitées, problème d'aliasing perceptif).

Par exemple, il est d'autant plus difficile d'atteindre la sortie d'un labyrinthe en un nombre minimal d'actions que la taille de ces environnements est grande, l'enchaînement des actions à réaliser étant d'autant plus conséquent. De même, une faible densité complique la résolution de cette tâche, car il

réduit le nombre de repères permettant à un agent de faciliter sa navigation. Autrement dit, il peut être plus souvent confronté au PAI l'empêchant de distinguer des situations environnementales différentes.

Le PAI peut avoir des conséquences différentes sur l'apprentissage d'un agent. (Bagnall and Zatuchna, 2005) montrent que le PAI peut être catégorisé en quatre différents types, selon les distances à la sortie de deux situations environnementales distinctes et les actions à réaliser pour se rapprocher de celle-ci dans ces situations. Ces quatre types de PAI illustrés par la figure 4.4 et sont :

- le pseudo PAI, si les distances à la sortie sont identiques et les actions à réaliser sont également identiques en ces situations ;
- le PAI de type I, si les distances à la sortie sont différentes et les actions à réaliser sont identiques en ces situations ;
- le PAI de type II, si les distances à la sortie et les actions à réaliser sont différentes en ces situations ;
- le PAI de type III, si les distances à la sortie sont identiques et les actions à réaliser sont différentes en ces situations.

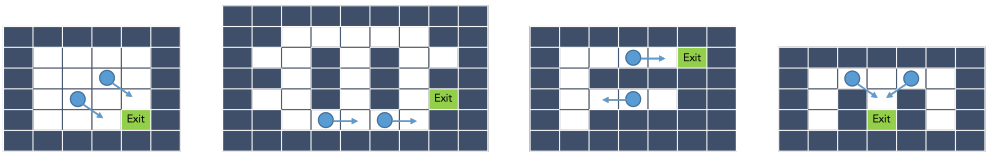


Figure 4.4 – Illustration des quatre types du problème d'aliasing perceptif dans les labyrinthes.
Les situations où un système est confronté au problème d'aliasing perceptif sont dénotées par les marques bleues. Les actions à réaliser afin de se rapprocher idéalement de la sortie sont indiquées par des flèches bleues.

Le PAI représente un défi supplémentaire pour les systèmes de classeurs à anticipation, puisque ces derniers doivent également anticiper l'état qui suit ces situations particulières. Ces types de PAI sont alors distinguables en deux sous-types, selon que les situations environnementales à anticiper soient identiques ou différentes.

(Bagnall and Zatuchna, 2005) mesurent la complexité des labyrinthes de telle sorte à prendre en compte l'ensemble des facteurs pouvant influencer l'apprentissage. Pour ce faire, ils réalisent le rapport entre le nombre d'actions

effectuées en moyenne par un agent utilisant le *Q-learning* de (Watkins and Dayan, 1992) pour atteindre la sortie de ces environnements, sur la distance moyenne minimale à la sortie. Les perceptions de leur agent sont limitées aux huit cellules adjacentes à sa position, quand il peut atteindre chacune de ces cellules. Seule l'incertitude relative au PAI peut être présente. Ils ont réalisé ces mesures sur différents labyrinthes, dont les résultats sont reportés plus loin dans la table 4.1 : la complexité est globalement plus élevée pour les environnements ayant des situations où le PAI est de type III, puis globalement pour ceux disposant de PAI de type II et enfin, ceux disposant de PAI de type I.

Cette augmentation de la complexité s'explique par le bruit généré par le PAI sur la fonction de valeur des renforcements qu'un agent estime. Le bruit généré est d'autant plus important qu'il empêche l'agent de déterminer l'action lui permettant de résoudre le plus efficacement sa tâche. Par exemple, dans le cas du pseudo PAI et du PAI de type I, les actions à réaliser dans les situations correspondantes sont identiques : un agent peut alors déterminer quelle est cette action à réaliser dans toutes ces situations. Le pseudo PAI et le PAI de type I sont ainsi plus simples à gérer que ceux du type II et III. Et le PAI de type III est plus dur à gérer que celui de type II, puisque la distance n'est pas utilisable pour discerner quelle est l'action à réaliser dans chacune des situations environnementales.

4.2 Métriques d'évaluation

Les métriques d'évaluation des capacités d'apprentissage peuvent être séparées en deux catégories selon les propriétés étudiées des ALCS :

- les métriques d'évaluation des représentations environnementales.
- les métriques d'évaluation des tâches d'apprentissage.

4.2.1 Évaluation des représentations environnementales

Les métriques d'évaluation des représentations environnementales désignent différentes mesures sur les populations de classeurs construites par les ALCS.

Ces mesures permettent de discuter de la qualité des représentations, indépendamment de la résolution de la tâche d'apprentissage.

Ces mesures intègrent classiquement :

- la taille de la population de (macro-)classeurs.
- la taille de la population de classeurs fiables où un classeur est fiable quand sa qualité dépasse le seuil θ_r défini par l'utilisateur en amont de l'apprentissage.
- la spécificité moyenne des classeurs fiables où un classeur est fiable quand sa qualité dépasse le seuil θ_r défini par l'utilisateur en amont de l'apprentissage.
- le ratio de connaissance, qui désigne le rapport du nombre de transitions correctes apprises par au moins un classeur fiable sur toutes les transitions possibles. Seules les transitions qui conduisent à des changements environnementaux sont dénombrées.

En plus de ces métriques, nous proposons l'*Erreur Moyenne sur la Prédiction de l'Anticipation (EMPA)* afin de fournir une information supplémentaire quant à la qualité des représentations environnementales générées par les ALCS. Les ALCS sont capables de construire des classeurs prédisant plusieurs anticipations : avec les PEP, pour chaque élément anticipé est également calculé une probabilité d'occurrence de cet élément. Le ratio de connaissance fournit alors des informations relatives à l'occurrence d'une anticipation, quand *l'Erreur Moyenne sur la Prédiction de l'Anticipation* donne des informations sur l'adéquation environnementale de leurs probabilités d'occurrence, comme illustré par la figure 4.5.

L'erreur moyenne sur la prédiction de l'anticipation est calculée comme suit : pour chaque transition possible dans un environnement à l'aide d'une unique action, nous calculons d'abord les probabilités théoriques associées à chaque attribut perceptif, compte tenu des états atteignables à partir d'une situation donnée et de l'incertitude environnementale. L'incertitude environnementale dépend des environnements dans lesquels les ALCS réalisent leur apprentissage, ainsi que de paramètres utilisateur permettant d'induire de la stochasticité dans ces environnements. Un exemple de calcul de ces probabilités est illustré par la figure 4.6.

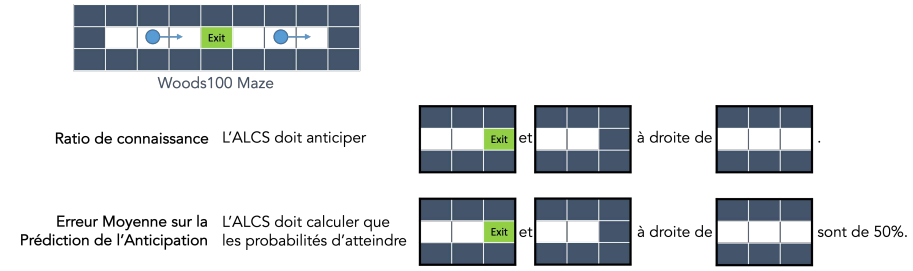


FIGURE 4.5 – Illustration des informations portées par le ratio de connaissances et par l'Erreur Moyenne sur la Prédiction de l'Anticipation dans le labyrinthe Woods100.
La perception d'un agent est identique dans les deux positions bleues, si sa perception est limitée au voisinage de Moore d'ordre 1 à la position courante. Une même perception correspond alors à deux situations environnementales distinctes, ce qui désigne le problème d'aliasing perceptif (PAI). Un classeur d'un ALCS doit alors idéalement anticiper les deux situations atteignables en allant à droite. Les probabilités d'anticiper l'une ou l'autre de ces situations dépendent alors des caractéristiques environnementales. Ici, seule la structure du labyrinthe est considérée pour déterminer qu'il y a 1 chance sur 2 d'anticiper l'une ou l'autre des situations.

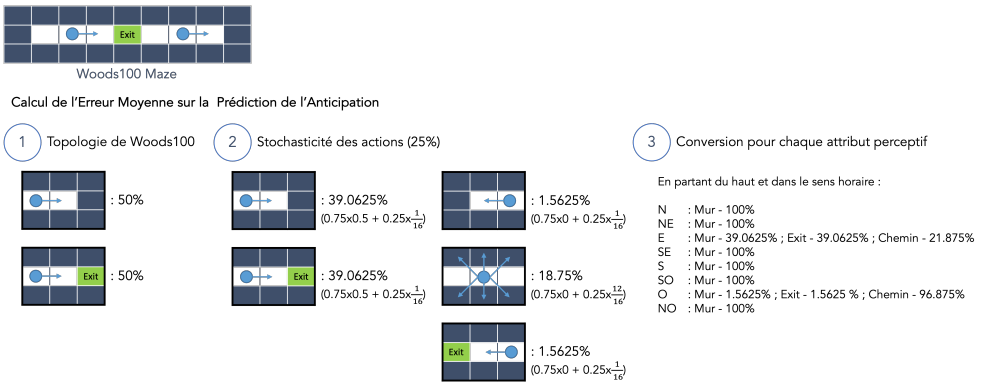


FIGURE 4.6 – Exemple de calcul des probabilités nécessaires à l'Erreur Moyenne sur la Prédiction de l'Anticipation dans le labyrinthe Woods100.
Ce calcul est réalisé quand les actions ont 25% d'être choisies aléatoirement pour la situation correspondant au PAI et si un agent va à droite à partir de cette situation. Les probabilités d'anticiper chacun des états atteignables sont d'abord calculées en prenant en compte la structure du labyrinthe. La stochasticité des actions est ensuite utilisée pour calculer les probabilités d'atteindre chacun des états. Enfin, ces probabilités permettent de trouver celles associées à chaque attribut perceptif.

Les classeurs les plus expérimentés correspondant à chaque transition sont ensuite utilisés pour calculer les différences entre ces probabilités théoriques et les probabilités d'occurrence du classeur. Les classeurs les plus expérimentés :

1. sont identifiés au sein de la sous-population de classeurs fiables (sinon,

au sein de l'ensemble de la population) ;

2. ont une structure conditionnelle correspondant à la transition environnementale étudiée ;
3. et le produit de leur expérience par leur qualité élevée au cube est maximal : discriminer les classeurs seulement sur leur qualité induirait que des classeurs nouvellement créés puissent être utilisés pour réaliser cette mesure, quand bien même leur adéquation à l'environnement n'aurait pas été éprouvée par le système ; c'est pourquoi leur expérience intervient dans ce calcul. Par ailleurs, la qualité des classeurs est élevée au cube pour agrandir les écarts entre les qualités de classeurs, afin de donner plus d'importance aux classeurs les plus fiables.

Ces différences sont finalement accumulées et moyennées sur le nombre de transitions possibles, avant d'être divisées par la taille d'une anticipation, pour obtenir l'erreur moyenne sur la prédiction de l'anticipation. La fin du calcul de l'EMPA pour une transition est illustrée par la figure 4.7.

Woods100 Maze

Calcul de l'Erreur Moyenne sur la Prédiction de l'Anticipation

En partant du haut et dans le sens horaire :

N : Mur - 100%
NE : Mur - 100%
E : Mur - 39.0625% ; Exit - 39.0625% ; Chemin - 21.875%
SE : Mur - 100%
S : Mur - 100%
SO : Mur - 100%
O : Mur - 1.5625% ; Exit - 1.5625 % ; Chemin - 96.875%
NO : Mur - 100%

Si le classeur le plus expérimenté anticipe :

N : Mur - 100%
NE : Mur - 100%
E : Mur - 42% ; Exit - 42% ; Chemin - 16%
SE : Mur - 100%
S : Mur - 100%
SO : Mur - 100%
O : Mur - 3% ; Exit - 5% ; Chemin - 92%
NO : Mur - 100%

L'EMPA est alors de :

(|100% - 100% | +
| 100% - 100% | +
| 39.0625% - 42%| + | 39.0625% - 42%| + |21.875% - 16%| +
| 100% - 100% | +
| 100% - 100% | +
| 100% - 100% | +
| 1.5625% - 3%| + | 1.5625% - 5%| + |96.875% - 92%| +
| 100% - 100% |) / 8

= 2.6875 %

Figure 4.7 – Exemple de calcul de l'Erreur Moyenne sur la Prédiction de l'Anticipation dans le labyrinthe Woods100, si les actions ont 25% d'être choisies aléatoirement pour la situation correspondant au PAI et si un agent va à droite.

4.2.2 Évaluation des tâches d'apprentissage

Les métriques d'évaluation des tâches d'apprentissage désignent des mesures permettant de discuter de l'usage et des performances des systèmes de classeurs à anticipation par rapport à la tâche d'apprentissage. Ces mesures dépendent alors principalement de la tâche d'apprentissage et des environnements étudiés.

Dans le cas de labyrinthes en deux dimensions, il s'agit le plus souvent d'atteindre le plus rapidement possible la sortie de ces labyrinthes, sachant qu'un agent ne peut bouger que d'une case à la fois dans le voisinage de Moore d'ordre 1 de la position courante. Lorsqu'un agent tente d'atteindre la sortie d'un labyrinthe, le nombre d'étapes nécessaires pour atteindre la sortie est relevé.

Dans le cas du problème *CartPole*, il s'agit de maintenir un bâton en équilibre le plus longtemps possible sur un mobile, sachant qu'un agent ne peut bouger le mobile que vers la gauche ou la droite à chaque pas de temps. Lorsqu'un agent tente de maintenir en équilibre le bâton sur un mobile, le nombre de pas de temps sans que le bâton soit tombé est relevé.

4.3 Protocole expérimental

4.3.1 Constitution d'un banc de test

Le protocole expérimental utilisé avec tous les systèmes de classeurs à anticipation s'appuie sur un banc de test de 23 labyrinthes de structures et de complexités variées (comme défini dans la section 4.1.3). Ces labyrinthes sont décrits en détail dans l'annexe A.

Ces labyrinthes sont classés dans le tableau 4.1 par complexité suivant un ordre décroissant (Bagnall and Zatuchna, 2005). Le type de PAI d'un labyrinthe est défini comme le type le plus complexe présent dans l'environnement, ce qui n'exclut pas la présence d'autres types de PAI. Cet ensemble de labyrinthes est non exhaustif : les labyrinthes retenus sont ceux dont la structure est clairement décrite dans la littérature. Ce banc de test peut être étendu à d'autres environnements labyrinthiques grâce à l'interface proposée par (Kozlowski and Unold, 2018) avec *OpenAI Gym* (Brockman et al., 2016b).

Ce banc de test de labyrinthes permet ainsi d'étudier les capacités des ALCS à gérer différentes formes d'incertitudes, en particulier de stochasticité. La présence du PAI permet de confronter les ALCS à des perceptions bruitées

Labyrinthe	Distance à la sortie	Complexité	Type de PAI
MazeE2 (Bagnall and Zatuchna, 2005)	2.27	251	III
Woods101.5 (Zang et al., 2015)	2.8	> 225	III
Maze10 (Zang et al., 2015)	4.56	> 150	III
Woods102 (Zang et al., 2015)	2.77	> 150	III
Woods100 (Métivier and Lattaud, 2002)	2	> 150	III
Woods101 (Bagnall and Zatuchna, 2005)	2.7	> 150	III
MazeF4 (Stolzmann, 1999)	4.3	> 125	II
MazeE1 (Métivier and Lattaud, 2002)	2.45	> 100	III
Maze7 (Zang et al., 2015)	4.11	> 100	II
MazeB (Arai and Sycara, 2001)	3.5	< 10	I
Littman89 (Loch and Singh, 1998)	3.77	< 10	I
MiyazakiA (Miyazaki and Kobayashi, 1999)	3.05	< 10	I
MiyazakiB (Miyazaki and Kobayashi, 1999)	3.35	< 10	II
MazeD (Arai and Sycara, 2001)	2.75	< 10	I
Littman57 (Bagnall and Zatuchna, 2005)	3.71	1	I
Cassandra4x4 (Bagnall and Zatuchna, 2005)	2.27	1	Pseudo
Maze4 (Butz et al., 2000)	3.5	$1*$	∅
Maze5 (Butz, 2001)	4.61	$1*$	∅
MazeA (Arai and Sycara, 2001)	4.23	$1*$	∅
MazeF1 (Stolzmann, 1999)	1.8	$1*$	∅
MazeF2 (Stolzmann, 1999)	2.5	$1*$	∅
MazeF3 (Stolzmann, 1999)	3.38	$1*$	∅
Woods14 (Butz et al., 2002)	9.5	$1*$	∅

Table 4.1 – Principales caractéristiques des environnements labyrinthiques du banc de test décrites dans (Bagnall and Zatuchna, 2005).
$*$ *indique qu'aucune valeur n'est renseignée parce que les labyrinthes correspondants n'ont pas le problème d'aliasing perceptuel. Nous pouvons nous attendre à ce que leur complexité soit de 1, un agent employant le* Q-learning *pouvant atteindre de manière optimale la sortie des labyrinthes en toute position. Les valeurs précises n'étant pas données pour tous les environnements, seules les plages de valeurs sont indiquées.*

et insuffisantes pour déterminer l'état exact dans lequel se trouve un agent. Le PAI permet également de bruiter les récompenses issues des labyrinthes, puisqu'une même récompense peut être fournie pour des états à distances différentes de la sortie. Les résultats des actions des ALCS peuvent également devenir incertains en paramétrant une probabilité qui déterminera si l'action réalisée par un agent est tirée aléatoirement parmi celles qui sont possibles, ou bien choisie selon la politique de sélection d'actions de l'agent.

Le but des ALCS dans ces labyrinthes est de construire une représentation complète et précise de leur environnement, en se déplaçant d'une case à la fois dans le voisinage de Moore d'ordre 1 de la position courante, tout en essayant d'atteindre la sortie le plus rapidement possible. Leurs capacités perceptives sont limitées aux huit cellules adjacentes à chaque position (qui

sont donc celles atteignables par les systèmes). Leurs positions de départ dans les labyrinthes sont aléatoires et distinctes de la sortie.

Les systèmes de classeurs à anticipation sont ainsi comparés entre eux au travers de ce banc de test et d'un protocole d'apprentissage commun.

4.3.2 Protocole d'apprentissage

Pour chaque labyrinthe du banc de test, 30 cycles d'apprentissage et de test sont réalisés. Chacun de ces cycles est constitué d'essais, qui désignent un nombre limité d'actions (100 actions en l'occurrence) qu'un agent peut réaliser dans un labyrinthe tant qu'il n'en a pas atteint la sortie. Un cycle est alors constitué de plusieurs essais qui permettent d'explorer l'environnement et d'exploiter les classeurs afin de résoudre la tâche d'apprentissage selon deux phases :

1. **Exploration** : 5000 essais qui doivent permettre aux ALCS de se doter d'une représentation de l'environnement propice à la résolution de la tâche d'apprentissage. Les hyperparamètres relatifs à ces essais sont définis ainsi : ϵ est fixé à 0.8 pour la politique ϵ-*greedy* de sélection des actions ; les pas d'apprentissage des composants d'apprentissage par renforcement β_{rl} et d'apprentissage par anticipation β_{alp} sont fixés à leurs valeurs classiquement utilisées dans la littérature 0.05 (Butz and Stolzmann, 2001). L'ensemble des métriques d'évaluation des populations de classeurs sont collectées à la fin de ces essais.
2. **Exploitation** :
   (a) 500 essais dont le but est de permettre aux ALCS d'initier une politique de décision propre à résoudre la tâche d'apprentissage. Les hyperparamètres associés sont définis ainsi : $\epsilon = 0.2$, $\beta_{\mathrm{rl}} = 0.05$ et $\beta_{\mathrm{alp}} = 0.0$. Seules les récompenses et par extension, l'adéquation des classeurs à la tâche d'apprentissage peuvent alors être modifiées. Ce paramétrage d'ϵ permet au système de pouvoir essayer dans une moindre mesure d'autres potentielles solutions.
   (b) 500 essais dont le but est de permettre aux ALCS de stabiliser les récompenses des classeurs et donc, de stabiliser les politiques de décision trouvées. Les hyperparamètres associés sont définis ainsi :

$\epsilon = 0.0$, $\beta_{\text{rl}} = 0.05$ et $\beta_{\text{alp}} = 0.0$.

(c) 500 essais où le nombre d'étapes nécessaires pour atteindre la sortie des labyrinthes est mesuré. Les hyperparamètres associés sont définis ainsi : $\epsilon = 0.0$, $\beta_{\text{rl}} = 0.05$ et $\beta_{\text{alp}} = 0.0$.

Les autres paramètres des ALCS qui ne sont pas décrits dans ce protocole expérimental sont fixés à leurs valeurs classiquement utilisées dans la littérature (Butz and Stolzmann, 2001) : ces valeurs permettent de résoudre différentes tâches d'apprentissage dont celles associées aux labyrinthes. Tout paramètre particulier à un système de classeurs à anticipation, et qui serait non explicité dans ce protocole, est décrit dans le chapitre propre à celui-ci.

Afin d'étudier les capacités des ALCS à gérer différentes formes d'incertitudes, les 30 cycles d'apprentissage et de tests ont été chacun réalisés avec et sans actions bruitées. Ainsi, un agent a une probabilité de 25% de réaliser une action choisie aléatoirement parmi celles possibles, quand ce bruit est mis en place.

Toutes les métriques d'évaluation des capacités d'apprentissage des systèmes de classeurs à anticipation sont moyennées sur les 30 cycles pour chaque environnement labyrinthique. Les moyennes résultantes sont comparées entre elles grâce au *test t de Welch* où le calcul des degrés de liberté est approché par *l'équation de Welch-Satterthwaite*, avec un seuil de signification α fixé à 0.05 (Fagerland and Sandvik, 2009). À nouveau, toutes métriques propres à un ALCS et non explicitées dans ce protocole commun sont présentées dans leurs chapitres respectifs.

Enfin, une copie profonde de la population de classeurs est réalisée à la fin des 5000 essais exploratoires, afin de comparer les effets d'une compression de la population de classeurs sur les capacités d'apprentissage des ALCS (les résultats sont présentés dans le chapitre 8). Ainsi, les étapes 2.a, 2.b et 2.c d'un cycle d'apprentissage et de test sont réalisées sur la population de classeurs originelle et sur la population de classeurs copiée puis compressée.

La figure 4.8 synthétise les différentes étapes du protocole expérimental présenté dans cette section.

Les chapitres suivants détaillent les différents systèmes de classeurs à anticipation que nous avons développés, ainsi que leurs algorithmes, et l'algo-

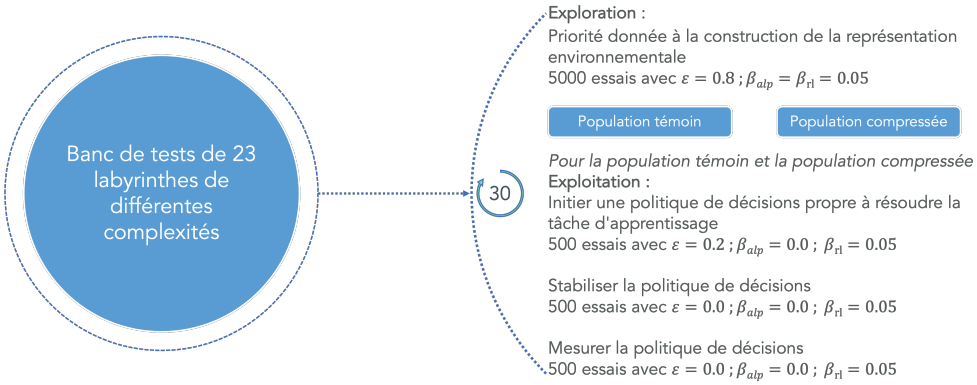


FIGURE 4.8 – Synthèse des différentes étapes du protocole expérimental commun à tous les systèmes de classeurs à anticipation évalués.
Ce protocole est répété sur l'ensemble du banc de test avec **et** *sans la stochasticité des actions, afin d'étudier les capacités des ALCS à gérer simultanément différentes formes d'incertitudes.*

rithme de compression des populations de classeurs de ces systèmes. Nous proposons également un dépôt GitHub dans lequel tous nos systèmes de classeurs et le banc de tests de labyrinthes sont disponibles à l'adresse suivante : https://github.com/RomainHappy/Reasoning-with-ALCS.

CHAPITRE 5

BACS : Accroître l'autonomie avec des séquences comportementales

L'ajout des séquences comportementales (BSeq) au système de classeurs à anticipation ACS 2 doit lui permettre d'accroître son autonomie dans des environnements incertains. De nouveaux classeurs contiennent des chaînes d'actions et décrivent les conséquences résultant de la réalisation de ces chaînes. Les BSeq doivent permettre à ACS 2 de traverser des situations environnementales pour lesquelles il ne serait pas en mesure de prendre une décision adaptée à la résolution de sa tâche d'apprentissage à cause du problème d'aliasing perceptif. Nous montrerons que Behavioral Anticipatory Classifier System (BACS), l'ALCS qui résulte de cet ajout à ACS 2, est capable de résoudre des tâches d'apprentissage qui lui seraient inaccessibles sans les séquences comportementales.

Sommaire

Les travaux présentés dans ce chapitre ont fait l'objet de publications : (Orhand et al., 2020b) et (Orhand et al., 2020a).

5.1 Intégrer les séquences comportementales dans ACS 2

L'ajout des séquences comportementales dans ACS 2 nécessite de déterminer les situations dans les lesquelles elles doivent être construites, comment les classeurs intégrant ces séquences sont générés et comment l'emploi des BSeq est contrôlé. Cependant, il est aussi nécessaire d'adapter les politiques de sélection d'action et de découverte de règles qui sont propres à ACS 2, puisqu'elles sont amenées à interagir avec les classeurs comportementaux (*i.e.* les classeurs dont le *Conséquent* décrit une séquence comportementale).

5.1.1 Adapter les politiques de sélection d'action

Les politiques de sélection permettant l'exploration de l'environnement ou l'exploitation des connaissances sont, dans ACS, des politiques de sélection de classeurs, ce qui permet de conserver en mémoire les classeurs qui ont été sélectionnés. L'avant-dernier classeur et le dernier classeur sélectionnés sont utilisés pour construire un nouveau classeur contenant une séquence comportementale quand le PAI est détecté. Cependant, ACS 2 ne s'appuie plus directement sur les classeurs, mais sur les actions que le système est en mesure de réaliser : il n'y a donc plus de classeurs conservés en mémoire pour construire les classeurs comportementaux. Les politiques de sélection de classeurs de BACS modifient alors celles d'ACS 2 de telle sorte à ce qu'un classeur soit systématiquement renvoyé, permettant de conserver en mémoire le pénultième classeur sélectionné :

- si une action est choisie aléatoirement, un classeur est choisi aléatoirement dans un ensemble constitué de la population courante de classeurs enrichie de classeurs génériques pour chacune des actions possibles (leurs *Conditions* et *Anticipations* ne contiennent que des symboles génériques #). Cet enrichissement est nécessaire, car la population initiale de classeurs de BACS, comme ACS 2, est vide et il permet à BACS de pouvoir sélectionner n'importe laquelle des actions ;
- si une action est choisie à partir du biais de rappel, un classeur générique est en priorité renvoyé pour une action décrite par aucun clas-

seur, sinon le classeur dont la marque temporelle t_{alp} est la plus faible est renvoyé ;
- si une action est choisie à partir du biais d'erreur, l'action unique de qualité moyenne Q_a minimale est identifiée afin de sélectionner aléatoirement un classeur dont la première action décrite dans son *Conséquent* correspond à celle de qualité minimale ;
- si une action est choisie à partir de sa *fitness*, le classeur ayant la plus grande *fitness* est retourné.

Le biais d'erreur favorise les actions qui permettent à BACS de franchir les situations relatives au PAI en s'appuyant sur les classeurs qui n'ont pas de BSeq pour potentiellement sélectionner un classeur comportemental. Étant donné la nature de ces situations (le système ne peut pas prédire l'état suivant), la qualité des classeurs réguliers ne peut pas augmenter de façon stable. Les qualités moyennes Q de ces actions seront par conséquent plus faibles que celles des autres actions. BACS peut donc exploiter ce biais pour construire des séquences comportementales quand il détecte le PAI et les éprouver ensuite.

5.1.2 Détecter le problème d'aliasing perceptif

La détection du PAI, illustrée par la figure 5.1, s'opère de la même manière que (Stolzmann, 1999) et (Métivier and Lattaud, 2002) : le système cherche à détecter les situations pour lesquelles des classeurs anticipent correctement les prochaines situations environnementales et en même temps, échouent à les anticiper. La détection ne cherche pas alors à déterminer directement à quel type de PAI ou d'incertitude le système peut être confronté. Cette détection prend place pour chaque classeur entrant dans l'*expected case* du processus d'apprentissage par anticipation, donc quand leur anticipation est correcte, et s'appuie sur leurs marques pour déterminer quand leur anticipation a été incorrecte. Si la marque d'un classeur entrant dans ce cas correspond strictement à la situation observée, alors BACS considère qu'il subit le problème d'aliasing perceptif. Il déclenche alors la construction d'un classeur à séquence comportementale à partir du classeur ayant permis la détection du PAI et du classeur précédemment sélectionné par BACS.

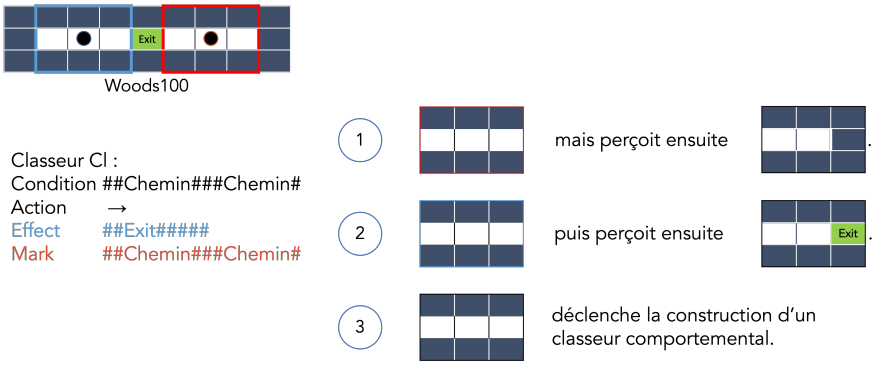


FIGURE 5.1 – Détection du problème d'aliasing perceptif avec BACS pour un environnement labyrinthique.
Dans le labyrinthe Woods100*, les perceptions des deux situations environnementales marquées par un point noir sont identiques, si la perception d'un agent est limitée à son voisinage de Moore d'ordre 1. Le classeur en exemple anticipe la sortie du labyrinthe à sa droite. Il échoue d'abord à anticiper cette sortie et est donc marqué par la situation où il subit le PAI. Par la suite, il anticipe correctement cette sortie et la comparaison de son observation de la situation courante et de sa marque est identique, ce qui déclenche la construction d'un classeur comportemental.*

5.1.3 Construire les classeurs comportementaux

Un classeur comportemental est construit à partir de chaque classeur cl_{t-1} de l'ensemble d'action dont la détection du PAI est positive et du classeur cl_{t-2} précédemment sélectionné par BACS, seulement si :

- cl_{t-1} anticipe au moins un changement perceptif ;
- cl_{t-2} anticipe au moins un changement perceptif et n'est pas marqué ;
- la longueur de la séquence comportementale à construire est inférieure au paramètre utilisateur $bs_{\max}$ qui définit la longueur maximale des BSeq.

Ces conditions permettent de limiter la création de classeurs comportementaux qui n'entraîneraient pas de changements perceptifs et par extension, ne permettraient pas de traverser les situations où BACS est confronté au PAI. La vérification sur la marque du classeur cl_{t-2} permet aussi d'éviter l'utilisation de classeurs dont l'*Anticipation* a été au moins une fois erronée, limitant par conséquent la construction de classeurs comportementaux dont les *Anticipations* seraient inadéquates.

Si ces conditions sont remplies, un classeur comportemental est créé, comme illustré sur la figure 5.2, où tous ses attributs sont mis à leur valeur par défaut

sauf :

- sa *Condition*, correspondant à la *Condition* de cl_{t-2} ;
- son *Conséquent*, correspondant à celui de cl_{t-2} chaîné à celui de cl_{t-1} ;
- son *Anticipation*, correspondant au résultat de l'opérateur *passthrough*[1] entre cl_{t-2} et cl_{t-1} *passthrough*($E\mathrm{cl}_{t-2}$, $E_{\mathrm{cl}_{t-1}}$), puis en remplaçant les attributs spécifiés par le symbole # si l'attribut correspondant de la *Condition* est identique à celui-ci.

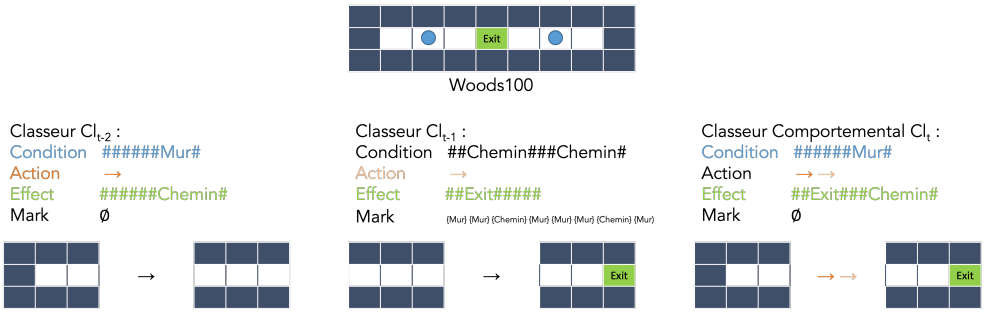


FIGURE 5.2 – Construction d'un classeur comportemental par BACS. *Dans le labyrinthe* Woods100*, les classeurs comportementaux permettent de traverser les situations environnementales décrites par les deux points bleus afin d'atteindre la sortie, comme celui construit en exemple pour la situation associée au point gauche.*

5.1.4 Adapter l'apprentissage par anticipation

Une fois que les classeurs comportementaux ont été générés, ils doivent pouvoir être mis à jour au travers du processus d'apprentissage par anticipation. L'ALP de BACS est différent selon qu'il est appliqué à des classeurs comportementaux ou des classeurs réguliers (non comportementaux).

L'ALP de BACS est similaire à celui d'ACS 2 pour les classeurs non comportementaux : la détection du PAI et la construction des classeurs comportementaux ont été ajoutées dans l'*expected case*. Ce dernier prend alors un argument supplémentaire qui est le classeur précédemment sélectionné pour construire les classeurs comportementaux.
L'ALP des classeurs réguliers peut créer des classeurs comportementaux qui ne peuvent être directement ajoutés aux ensembles d'actions, puisque leur *Conséquent* décrit nécessairement des séquences d'actions différentes d'une

1. L'opérateur *passthrough* est présenté par l'algorithme 3.1.1 de la section 3.1.3 du chapitre 3.

unique action. Un nouveau classeur à BSeq construit par l'ALP des classeurs réguliers est alors ajouté par le biais du processus de subsomption à l'ensemble de la population. L'algorithme 5.1.1 décrit l'insertion d'un classeur cl dans un ensemble $[S]$ de classeurs pouvant être un ensemble d'action, un ensemble d'appariement ou l'ensemble de la population de classeurs. Il prend également en argument les seuils d'expérience θ_{exp} et de fiabilité θ_{r} nécessaires à la subsomption, ainsi que le pas d'apprentissage β_{alp}. La subsomption [2] est également modifiée pour vérifier que les *Conséquents* de deux classeurs sont identiques et que leurs *Conditions* ne précisent pas d'attributs spécifiés antagonistes afin de permettre l'ajout d'un classeur dans un ensemble $[S]$ différent d'un ensemble d'action. L'algorithme 5.1.1 ne décrit pas l'ajout d'un nouveau classeur dans d'autres ensembles que $[S]$, qu'il s'agisse par exemple de l'ensemble des classeurs de la population ou d'un ensemble d'appariement si la *Condition* du nouveau classeur correspond à la perception associée avec l'ensemble d'appariement.

Algorithme 5.1.1 Insertion de classeurs à partir de l'ALP de BACS

```
1 : function InsertFromALP(cl, β_alp, θ_exp, θ_r, [S])
2 :     cl_old ← ∅
3 :     for each cl_S ∈ [S] do
4 :         if IsSubsumed(cl, cl_S, θ_exp, θ_r) then
5 :             if cl_old == ∅ or C_cl_S is more general than C_cl_old then
6 :                 cl_old ← cl_S
7 :             end if
8 :         end if
9 :     end for
10 :    if cl_old == ∅ then
11 :        for each cl_S ∈ [S] do
12 :            if C_cl_S == C_cl and E_cl_S == E_cl then
13 :                cl_old ← cl_S
14 :            end if
15 :        end for
16 :    end if
17 :    if cl_old == ∅ then
18 :        Insert cl in [S]
19 :    else
20 :        q_cl_old ← (1 − β_alp) q_cl_old + β_alp
21 :        Discard cl
22 :    end if
```

2. Fonction *IsSubsumed* définie par l'algorithme 3.2.5 de la section 3.2.5 du chapitre 3

L'algorithme 5.1.2 fournit le pseudo-code complet de l'ALP des classeurs réguliers de BACS, où $[A]$, $[M]$, $[P]$ sont les différents ensembles de classeurs, $cl_{\mathrm{t}-2}$ la classeur précédemment sélectionné par BACS au temps $t-2$, p_{t-1}, p_t sont les observations de l'environnement avant et après la réalisation de l'action a, t le pas de temps courant et θ_{exp}, θ_i, θ_{r} les seuils d'expérience, d'inadéquation et d'adéquation des classeurs.

Algorithme 5.1.2 ALP des classeurs réguliers de BACS

function ALP($[A]$, $[M]$, $[P]$, $cl_{\mathrm{t}-2}$, p_{t-1}, a, p_t, β_{alp}, θ_{exp}, θ_i, θ_{r}, $u_{\max}$, t)
  for each $cl \in [A]$ **do**
    Update $t_{alp_{\mathrm{cl}}}$ with t and increase exp_{cl} by 1
    $cl_{\mathrm{new}} \leftarrow \emptyset$
    oneCorrectAnticipation $\leftarrow$ *False*
    if cl anticipates correclty p_t from p_{t-1} **then**
      $cl_{\mathrm{new}} \leftarrow$ BACS *expected case* with $cl_{\mathrm{t}-2}$, cl, p_{t-1}, β_{alp}, $u_{\max}$, t
      oneCorrectAnticipation $\leftarrow$ *True*
    else
      if cl is correctable **then**
        $cl_{\mathrm{new}} \leftarrow$ *correctable case* of ACS 2 with cl, p_{t-1}, p_t, β_{alp}, t
      else
        not correctable case of ACS 2 with cl, p_{t-1}, β_{alp}
      end if
    end if
    if $q_{\mathrm{cl}} < \theta_i$ **then**
      Remove cl from $[A]$, $[M]$ and $[P]$
    end if
    if $cl_{\mathrm{new}} \neq \emptyset$ **then**
      if cl_{new} is a behavioral classifier **then**
        InsertFromALP(cl_{new}, β_{alp}, θ_{exp}, θ_{r}, $[P]$)
      else
        InsertFromALP(cl_{new}, β_{alp}, θ_{exp}, θ_{r}, $[A]$)
      end if
    end if
  end for
  if *oneCorrectAnticipation* == *False* **then**
    Build cl_{new} by *Covering* with p_{t-1}, a, p_t, t
    InsertFromALP(cl_{new}, β_{alp}, θ_{exp}, θ_{r}, $[A]$)
  end if

En revanche, l'ALP des classeurs comportementaux comporte trois principales différences avec l'ALP des classeurs réguliers. D'une part, il n'y a pas de processus de recouvrement si aucun classeur comportemental d'un ensemble

d'action ne parvient à correctement anticiper la prochaine situation. Seule la détection du PAI peut déclencher la création de ces classeurs, car nous souhaitons éviter que les classeurs comportementaux soient utilisés dans des situations où ils ne seraient pas nécessaires.

D'autre part, l'*expected case* des classeurs comportementaux est identique à celui d'ACS 2 : il n'intègre pas la détection du PAI et la construction de nouveaux classeurs comportementaux. L'absence de ces mécanismes permet à BACS de contrôler la création de nouvelles séquences comportementales dans l'*expected case* des classeurs réguliers : les actions ne peuvent être ajoutées qu'une seule à la fois et en bout de chaîne ; un nouveau classeur comportemental n'est construit que s'il n'a pas permis de traverser les situations sujettes au PAI. Les classeurs comportementaux sont tout de même spécifiables par la spécification des éléments inchangés. Le *Not correctable case* et le *correctable case* sont identiques entre les classeurs réguliers et les classeurs comportementaux.

Enfin, l'ALP des classeurs comportementaux intègre une étape préliminaire similaire au *useless case* d'ACS : si les observations avant et après la réalisation des BSeq sont identiques, toutes les qualités de ces classeurs sont diminuées de la même manière que si leur *Anticipation* est incorrecte. Ainsi, BACS favorise les classeurs comportementaux qui entraînent des changements perceptifs. Ces changements sont attendus puisqu'une succession d'actions est réalisée pour traverser les situations sujettes au PAI. Si aucun changement n'est perçu suite à l'usage d'une séquence comportementale, la séquence peut contenir des actions superflues qui ne permettent pas cette traversée ou ne pas être suffisamment longue pour la rendre possible. Empiriquement, l'utilisateur de BACS peut suivre l'évolution de l'apprentissage pour analyser les situations dans lesquelles le PAI est détecté et aussi, adapter le seuil $bs_{\max}$ si celui-ci est limitant. Autrement, BACS intègre un mécanisme qui lui permet automatiquement de discriminer les actions superflues des BSeq : ce mécanisme est présenté dans la section suivante.

L'algorithme 5.1.3 fournit le pseudo-code complet de l'ALP des classeurs comportementaux de BACS, où $[A]$, $[M]$, $[P]$ sont les différents ensembles de classeurs, p_{t-1}, p_t sont les observations de l'environnement entre deux réalisations d'actions, t le pas de temps courant et $\theta_{\exp}$, θ_i, θ_{r} les seuils d'ex-

périence, d'inadéquation et d'adéquation des classeurs.

Algorithme 5.1.3 ALP des classeurs comportementaux de BACS

```
1 : function ALPBSeq([A], [M], [P], p_{t-1}, p_t, θ_exp, θ_i, θ_r, u_max, t)
2 :     for each cl ∈ [A] do
3 :         Update t_{alp_cl} with t and increase exp_cl by 1
4 :         cl_new ← ∅
5 :         if p_{t-1} == p_t then
6 :             q_cl ← (1 − β_alp) q_cl
7 :         else if cl anticipates correclty p_t from p_{t-1} then
8 :             cl_new ← expected case of ACS 2 with cl, p_{t-1}, β_alp, u_max, t
9 :         else
10 :            if cl is correctable then
11 :                cl_new ← correctable case of ACS 2 with cl, p_{t-1}, p_t, β_alp, t
12 :            else
13 :                not correctable case of ACS 2 with cl, p_{t-1}, β_alp
14 :            end if
15 :        end if
16 :        if q_cl < θ_i then
17 :            Remove cl from [A], [M] and [P]
18 :        end if
19 :        if cl_new ≠ ∅ then
20 :            InsertFromALP(cl_new, β_alp, θ_exp, θ_r, [S])
21 :        end if
22 :    end for
```

5.1.5 Discriminer les séquences comportementales superflues

Lorsque BACS utilise une séquence comportementale, il doit exécuter chaque action de la séquence avant d'appliquer les processus d'apprentissage. Ces actions peuvent conduire à une boucle : par exemple, dans les labyrinthes, une boucle peut être caractérisée par un pas en avant, puis un pas en arrière, puis un pas en avant, et ainsi de suite. Afin d'éviter de telles boucles, (Métivier and Lattaud, 2002) enregistrent les observations dans une liste temporaire pour détecter les situations déjà observées lors de l'exécution d'une séquence : le cas échéant, la qualité des classeurs comportementaux est autant de fois diminuée que le système observe à nouveau une situation, sans interrompre l'exécution de la séquence. BACS intègre ce mécanisme discrimina-

toire afin que BACS défavorise les classeurs proposant de telles boucles.

5.1.6 Éviter la généralisation des classeurs comportementaux

Les classeurs comportementaux ont pour objectif de traverser les situations où le PAI est détecté : ils n'ont pas vocation à être utilisés en dehors de ces situations. Une spécification de ces classeurs par le processus d'apprentissage par anticipation ne leur permet pas d'être employés dans de nouvelles situations, au contraire du mécanisme de généralisation génétique. BACS ne généralise pas les classeurs comportementaux : BACS vérifie la composition des ensembles d'action pour prévenir l'application de la généralisation génétique s'il s'agit de classeurs comportementaux. L'ALP et la discrimination mise en place permettent exclusivement de maintenir, de faire évoluer ou de supprimer ces classeurs.

5.2 Capacités de BACS

5.2.1 Méthodologie

Les métriques d'évaluation et le protocole expérimental détaillés dans le chapitre 4 doivent permettre d'évaluer les capacités d'apprentissage de BACS en répondant aux questions suivantes :

1. BACS se comporte-t-il comme ACS 2 dans les environnements où il n'est pas sujet à l'incertitude environnementale ?
2. BACS est-il capable d'atteindre plus rapidement qu'ACS 2 la sortie des labyrinthes dans lesquels il est confronté au PAI ?
3. Est-ce que les séquences comportementales ont un impact sur la capacité de BACS à construire une représentation d'environnements dans lesquels il est confronté au PAI ?
4. Quelle est l'influence de l'incertitude relative aux résultats des actions sur ACS 2 et BACS ?

Les capacités d'apprentissage de BACS sont d'abord comparées avec celles d'ACS 2 dont il est dérivé. Les nombres d'étapes nécessaires pour atteindre la sortie des labyrinthes permettent de comparer les politiques de décision mises en place par ces systèmes et de répondre aux deux premières interrogations. Les ratios de connaissance, les tailles des populations de classeurs et les spécificités moyennes des classeurs fiables réguliers et comportementaux permettent de comparer les représentations environnementales développées par ces systèmes, complétant la réponse à la première interrogation.
Pour mesurer l'influence générale du paramètre $bs_{\max}$ de BACS, toutes ces métriques ont été relevées pour deux valeurs différentes de ce paramètre : les séquences comportementales peuvent être de longueur maximale 2 ou de longueur maximale 3. Les résultats obtenus quand $bs_{\max} = 2$ et $bs_{\max} = 3$ correspondent respectivement à BACS-2 et BACS-3. Les environnements du banc de test sur lequel nous nous appuyons ne nécessitent pas l'emploi de valeurs plus élevées du paramètre $bs_{\max}$. La grande majorité des labyrinthes du banc ne nécessite que l'usage de BSeq de longueur 2 pour résoudre la tâche d'apprentissage. Les résultats obtenus avec BACS-2 et BACS-3 permettent de comparer leurs capacités à trouver une composition des BSeq à même d'atteindre la sortie, et aussi de comparer les conséquences sur le développement des représentations suite à leur emploi.
Toutes ces métriques peuvent également être utilisées pour répondre à la dernière interrogation puisqu'elles ont été relevées avec et sans l'incertitude associée aux résultats des actions.

Les résultats obtenus par BACS sont présentés dans la section 5.2.2, puis comparés avec les ALCS utilisant aussi les séquences comportementales dans la section 5.2.3. Ils sont ensuite discutés dans la section 5.3.

5.2.2 Résultats

BACS se comporte-t-il comme ACS 2 dans les environnements où il n'est pas sujet à l'incertitude environnementale ?

Les ratios de connaissance des populations de classeurs d'ACS 2, de BACS-2 et de BACS-3 sont strictement identiques et maximaux : les trois systèmes sont parvenus à apprendre une représentation complète de leur environne-

ment puisque toutes les transitions environnementales sont décrites par au moins un classeur fiable.
Les tailles des populations de classeurs, et donc des représentations environnementales, ne présentent pas de différences statistiques significatives à l'exception de trois cas : les populations de BACS-2 sont plus petites que celles d'ACS 2 dans l'environnement *Maze5*, alors que celles de BACS-3 sont aussi grandes que BACS-2 et ACS 2 ; les populations de BACS-3 sont plus grandes que celles de BACS-2 dans les labyrinthes *MazeF3* et *Woods14*, alors que celles d'ACS 2 sont aussi grandes que BACS-2 et BACS-3.
Les spécificités moyennes des classeurs ne présentent pas de différences significatives sauf dans *Maze5*, où les classeurs de BACS-2 et BACS-3 sont aussi généraux les uns que les autres et plus généraux qu'ACS 2.
Nous pouvons donc dire que les modèles de transitions appris par ces systèmes de classeurs sont équivalents. Les différences relevées peuvent s'expliquer par l'emploi du mécanisme de généralisation génétique : ce dernier peut introduire une légère variation sur les tailles ou spécificités des classeurs due au hasard inhérent à son fonctionnement. Nous fournissons en annexe B.1 des figures illustrant ces résultats.

Enfin, les politiques de décision construites par ACS 2, BACS-2 et BACS-3 ne présentent pas de différences significatives. Ni BACS-2, ni BACS-3 n'ont créé ou utilisé de séquences comportementales dans ces environnements, ce qui conforte ce résultat. Ainsi, BACS se comporte comme ACS 2 dans les environnements où il n'est pas sujet à l'incertitude environnementale.

BACS est-il capable d'atteindre plus rapidement qu'ACS 2 la sortie des labyrinthes dans lesquels il est confronté au PAI ?

La figure 5.3 donne les nombres moyens d'étapes nécessaires à BACS et ACS 2 pour atteindre la sortie des labyrinthes, quand ils sont confrontés au problème d'aliasing perceptif. Pour 9 des 16 environnements, BACS-2 et BACS-3 mettent en place des politiques de décision équivalentes et plus efficaces qu'ACS 2. Pour *Woods101*, *Woods101.5* et *Woods102*, BACS-2 est plus performant que BACS-3, ce dernier étant lui plus performant qu'ACS 2. Pour les 4 labyrinthes restants :

— ACS 2, BACS-2 et BACS-3 atteignent aussi rapidement l'un que l'autre

la sortie de *MazeB* ;

— ACS 2 est aussi performant que BACS-2 pour atteindre la sortie de *Cassandra4x4*, ces deux systèmes étant plus rapides que BACS-3 ;

— ACS 2 est aussi performant que BACS-3 pour atteindre la sortie de *MiyazakiB*, ces deux systèmes étant plus rapides que BACS-2 ;

— ACS 2 est plus performant que BACS-2, ce dernier étant lui plus performant qu'BACS-3, pour atteindre rapidement la sortie de *MazeD*.

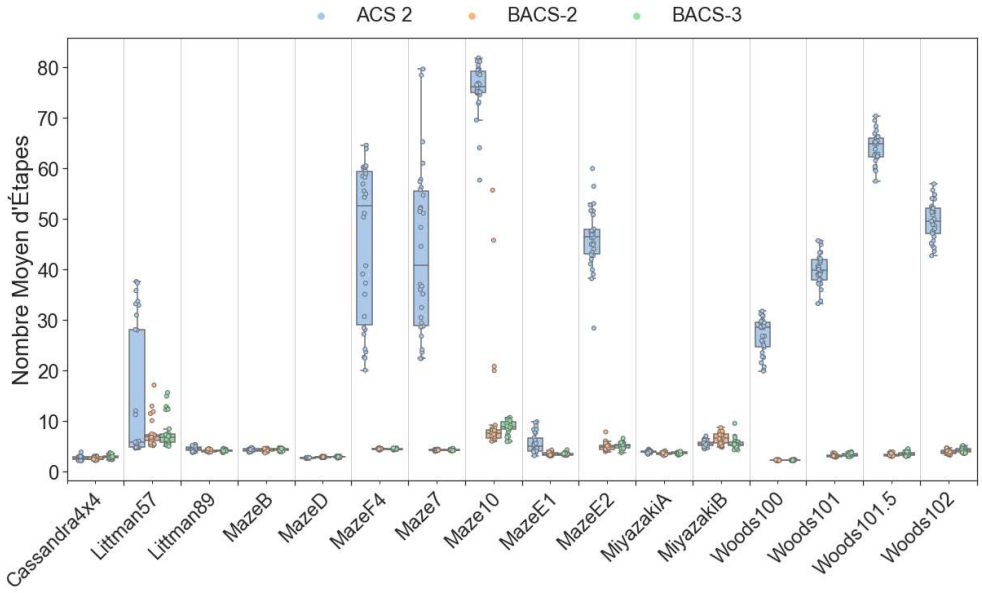


FIGURE 5.3 – Nombres moyens d'étapes de BACS et ACS 2 pour atteindre la sortie des environnements avec PAI et sans les actions incertaines.

À l'exception d'un environnement, BACS est capable d'atteindre la sortie de 12 labyrinthes plus rapidement qu'ACS 2 quand il est confronté au PAI, ou sinon aussi rapidement que ce dernier dans les 3 labyrinthes restants : l'efficacité des BSeq est ainsi démontrée. Cependant, les performances de BACS-2 et BACS-3 sont différentes pour 6 environnements, ce qui montre que le paramètre $bs_{\max}$ influence la capacité de BACS à résoudre ces tâches d'apprentissage.

Est-ce que les séquences comportementales ont un impact sur la capacité de BACS à construire une représentation d'environnements dans lesquels il est confronté au PAI ?

Les ratios de connaissance ne présentent pas de différences entre les trois

ALCS pour 12 des 16 environnements. Les ratios d'ACS 2 sont plus élevés que ceux de BACS pour *MiyazakiB*, *Maze7* et *MazeF4*, ceux de BACS-2 et BACS-3 étant équivalents. Les ratios de connaissance atteints par ACS 2 et BACS-3 sont équivalents et plus élevés que ceux de BACS-2 pour le dernier environnement, *MiyazakiA*.

La figure 5.4 illustre les tailles des différentes populations de classeurs. Pour tous les labyrinthes, les populations de classeurs les plus petites sont celles d'ACS 2. Les populations de classeurs les plus grandes sont celles de BACS-3 pour 11 environnements, ou sont de tailles équivalentes à celles de BACS-2 pour les 5 environnements restants. Au sein d'un même labyrinthe, l'accroissement des populations de classeurs de BACS-2 à BACS-3 est de l'ordre de 60%, celles d'ACS 2 à celles BACS-2 de 300% et celles d'ACS 2 à BACS-3 de 450%.

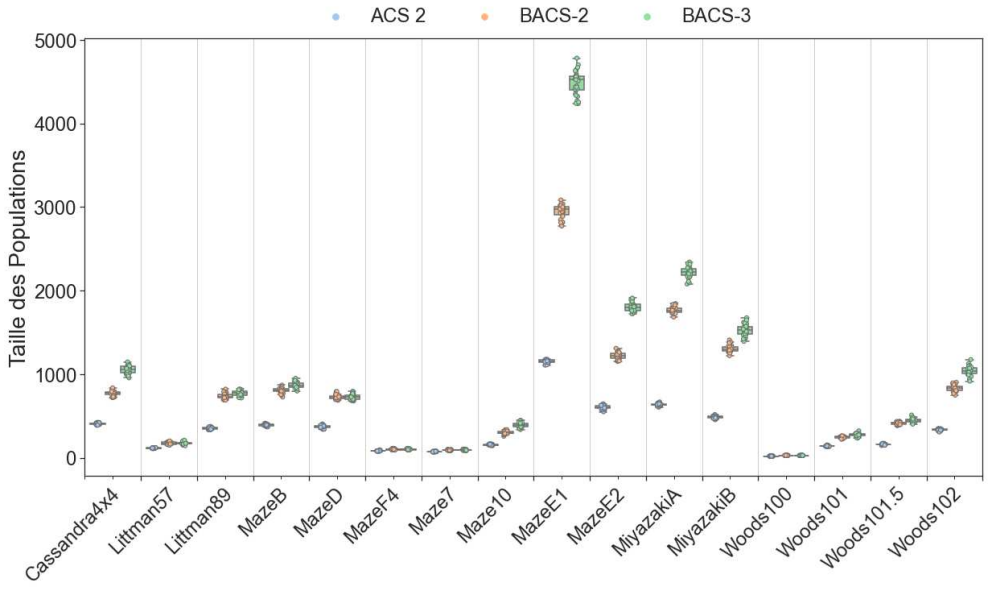


FIGURE 5.4 – Tailles des populations de classeurs de BACS et ACS 2 dans les environnements avec PAI et sans les actions incertaines.

La figure 5.5 illustre les spécificités moyennes des classeurs réguliers fiables. Ces spécificités ne présentent pas de différences entre les trois systèmes pour 3 environnements. Dans 10 labyrinthes, les populations les plus générales sont celles d'ACS 2 et les plus spécifiques celles de BACS-3 pour 4 de ces 10 labyrinthes, sinon celles de BACS-3 et BACS-2 sont aussi spécifiques. Dans *Maze7* et *MazeF4*, les populations les plus générales sont celles de BACS-2 et BACS-3. Enfin, les classeurs réguliers de BACS-3 dans *MazeE2* sont les plus

généraux par rapport à ceux de BACS-2 et ACS 2 qui sont équivalents.
En résumé, les spécificités moyennes des classeurs réguliers fiables sont les plus faibles pour ACS 2 dans 13 des 16 labyrinthes, quand celles liées à BACS-2 sont plus faibles que celles de BACS-3 dans 6 environnements ou équivalentes dans 10 environnements.

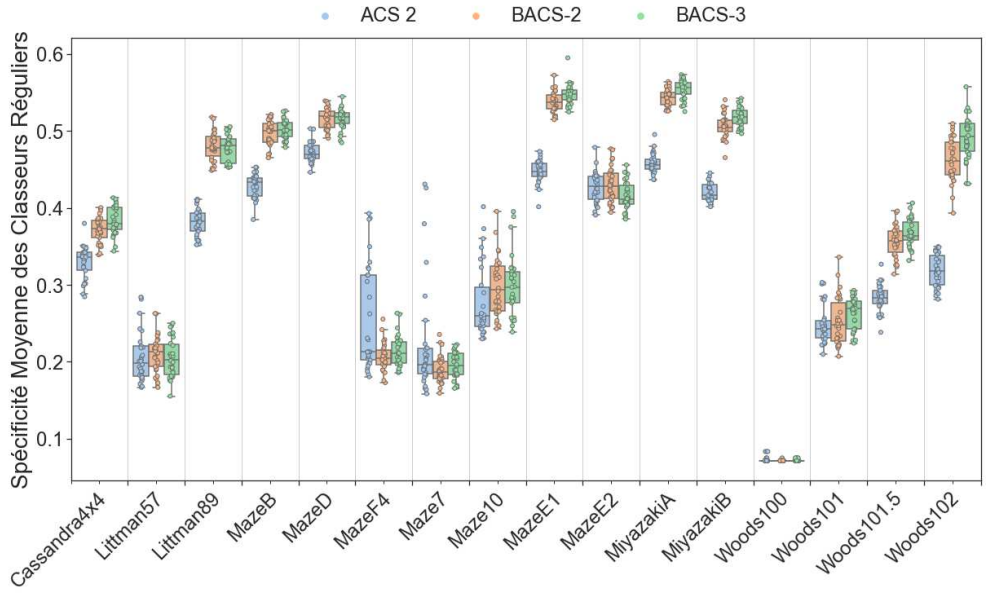


FIGURE 5.5 – Spécificités moyennes des classeurs réguliers de BACS et ACS 2 dans les environnements avec PAI et sans les actions incertaines.

Enfin, les spécificités des classeurs comportementaux de BACS-2 sont équivalentes à celles de BACS-3 dans 13 environnements, sinon plus faibles dans les labyrinthes 3 restants. Les figures illustrant les spécificités moyennes des classeurs comportementaux, ainsi que les ratios de connaissance atteints, sont fournies dans l'annexe B.2.

Quelle est l'influence de l'incertitude relative aux résultats des actions sur ACS 2 et BACS ?

Nous nous appuyons sur l'ensemble du banc de tests pour analyser les effets des actions incertaines sur chacun des ALCS, tant sur les représentations environnementales que les politiques de décision développées. Les actions incertaines ont des effets similaires sur la construction des représentations environnementales dans ACS 2, BACS-2 et BACS-3 :

— tous les ratios de connaissance chutent fortement, à l'image de la

figure 5.6 qui présente les ratios atteints par BACS-2 ;

— les nombres de classeurs dans les populations augmentent fortement, étant 11× plus nombreux dans *MazeA* avec ACS 2, ou 12× plus nombreux dans *Maze5* avec BACS-2 et BACS-3, ce qui est montré par la figure 5.7 pour BACS-2 ;

— les spécificités moyennes des classeurs réguliers augmentent toutes, tel qu'illustré par la figure 5.8 pour BACS-2.

Les figures pour ACS 2 et BACS-3 sont similaires à celles données pour BACS-2 : elles sont données dans l'annexe B.3.

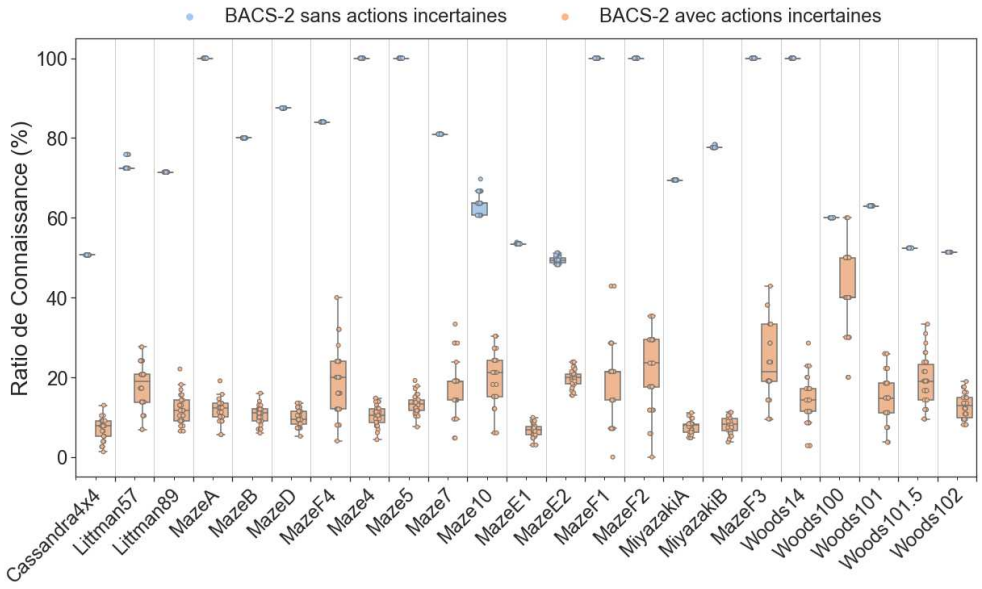


FIGURE 5.6 – Ratios de connaissance atteints par BACS-2 avec et sans l'incertitude liée à la réalisation des actions dans tous les environnements.

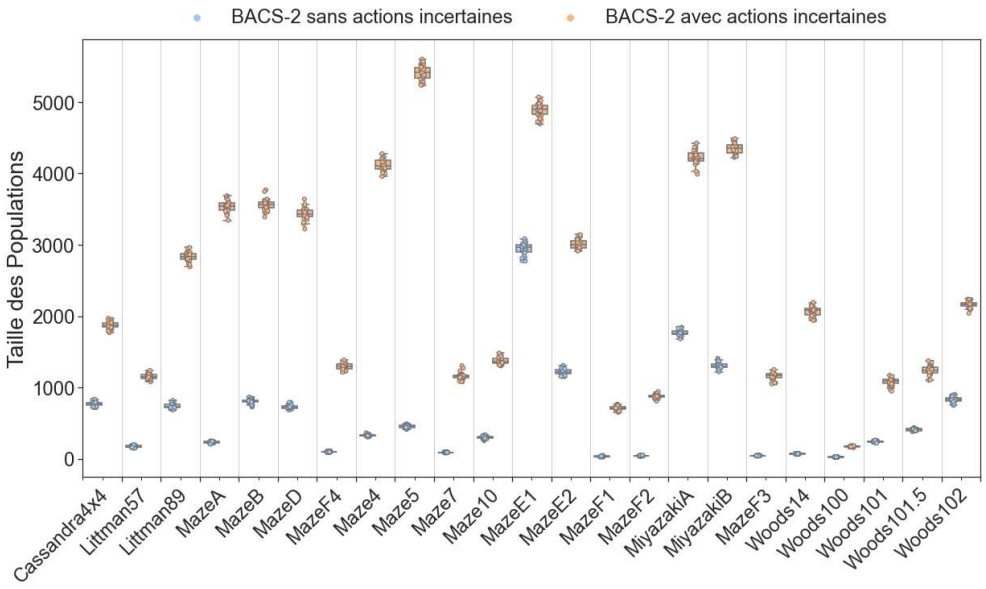


FIGURE 5.7 – Tailles moyennes des populations de classeurs de BACS-2 avec et sans l'incertitude liée à la réalisation des actions dans tous les environnements.

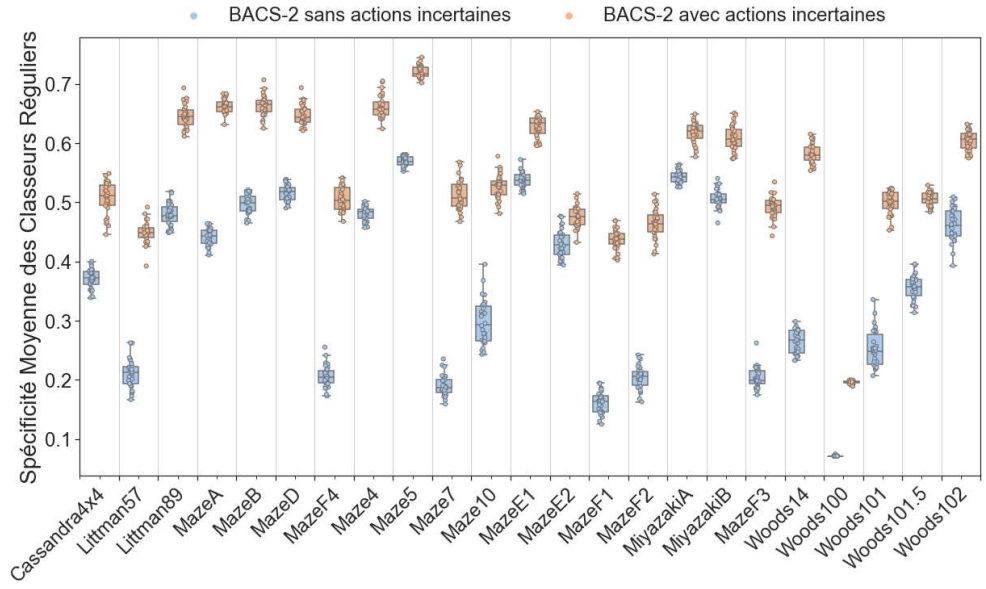


FIGURE 5.8 – Spécificités moyennes de classeurs réguliers fiables de BACS-2 avec et sans l'incertitude liée à la réalisation des actions dans tous les environnements.

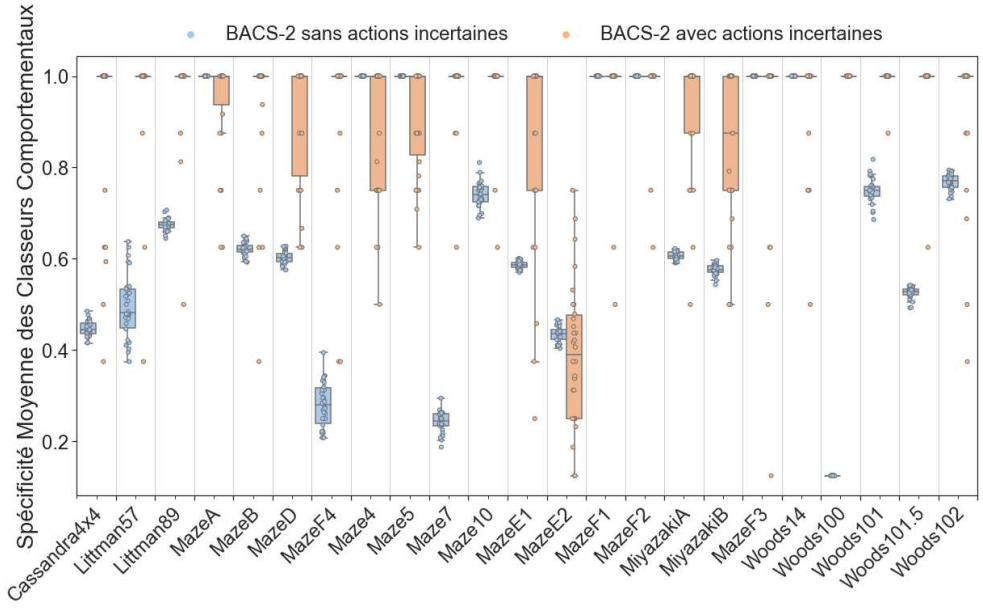


FIGURE 5.9 – Spécificités moyennes de classeurs comportementaux fiables de BACS-2 avec et sans l'incertitude liée à la réalisation des actions dans tous les environnements.

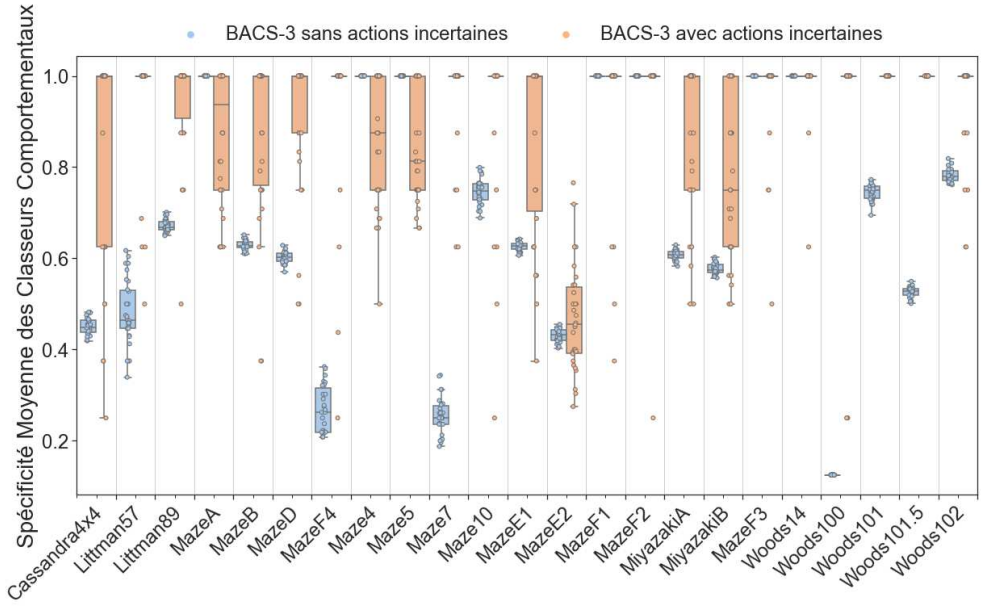


FIGURE 5.10 – Spécificités moyennes de classeurs comportementaux fiables de BACS-3 avec et sans l'incertitude liée à la réalisation des actions dans tous les environnements.

Si les actions sont incertaines, les spécificités des classeurs comportementaux de BACS-2 sont plus élevées pour 14 des 23 environnements, équivalentes pour 5 labyrinthes et plus faibles pour les 4 derniers (*cf.* figure 5.9). Dans

BACS-3, cette mesure est plus élevée pour 15 des 23 environnements, équivalente pour 4 labyrinthes et plus faible pour les 4 restants (*cf.* figure 5.10).

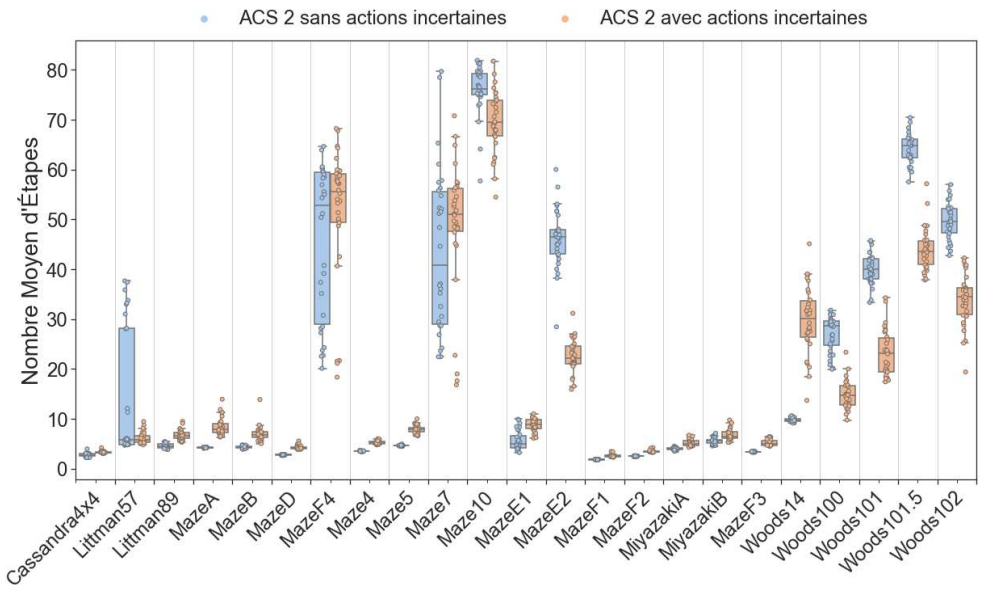


FIGURE 5.11 – Nombres moyens d'étapes d'ACS 2 pour atteindre la sortie des labyrinthes, avec et sans l'incertitude liée à la réalisation des actions.

Dans BACS-3, les nombres moyens d'étapes nécessaires pour atteindre la sortie des labyrinthes sont augmentés à cause des actions incertaines dans 22 environnements et sont plus faibles pour *Littman57*. Cette mesure augmente également avec BACS-2 dans 21 labyrinthes, sont plus faibles pour *Littman57* ou sont équivalents dans *MiyazakiB*. Nous donnons en annexe B.4 les figures détaillées de cette métrique pour BACS-2 et BACS-3. Dans ACS 2, les nombres moyens d'étapes augmentent dans 14 labyrinthes, sont équivalents dans 2 labyrinthes et diminuent dans les 7 restants (*cf.* figure 5.11).

En résumé, l'incertitude relative aux résultats des actions réduit les capacités des ALCS à se doter de classeurs fiables, augmente le nombre de classeurs dans les populations et accroît les pressions de spécification sur ces populations. Elle semble améliorer la mise en place des politiques de décision d'ACS 2 quand il est confronté au PAI, sinon elle complexifie leur élaboration.

Les capacités de BACS et d'ACS 2 peuvent aussi être comparées entre elles. Nous remarquons alors que l'emploi des BSeq avec les actions incertaines impacte les capacités de BACS à se doter d'une représentation des environne-

ments. À cause des actions incertaines, les ratios de connaissance sont désormais les plus élevés pour ACS 2 dans 18 des 23 environnements (dont 3 sans PAI) ou équivalents à ceux de BACS dans les 5 restants, alors qu'ils sont identiques pour 19 environnements sans les actions incertaines. Les séquences comportementales de BACS préviennent la création de classeurs fiables si les actions sont incertaines. La figure 5.12 donne les ratios de connaissance atteints par les trois ALCS.

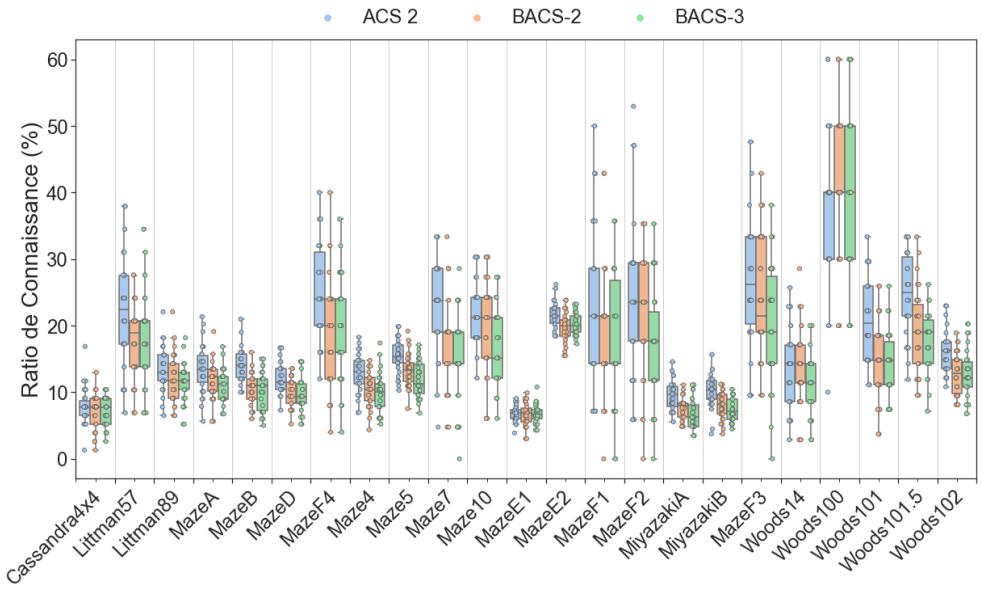


FIGURE 5.12 – Ratios de connaissance d'ACS 2, de BACS-2 et BACS-3 avec l'incertitude liée à la réalisation des actions pour le banc de test de labyrinthes.

Les tailles de populations de classeurs sont les plus faibles pour ACS 2 dans tous les environnements, celles de BACS-2 étant aussi plus faibles que celles de BACS-3. L'impact des BSeq est le plus évident pour les environnements sans PAI, puisque les tailles des populations de classeurs ne sont plus identiques à cause de l'apparition de classeurs à BSeq.
Les spécificités moyennes des classeurs réguliers d'ACS 2 sont aussi les plus élevées dans 11 labyrinthes (dont 4 sans PAI), équivalentes à BACS-2 et BACS-3 dans 8 autres labyrinthes (dont 2 sans PAI) et celles de BACS-3 sont les plus faibles dans les 4 restants. Sans action incertaine, elles étaient identiques dans les environnements sans PAI et celles d'ACS 2 étaient les plus faibles pour 13 labyrinthes. Les BSeq influencent la généralisation des classeurs réguliers en leur évitant d'être spécifiés avec les actions incertaines.

Les spécificités moyennes des classeurs comportementaux de BACS-2 et de BACS-3 sont équivalentes dans 22 environnements, celles de BACS-2 plus faibles pour le dernier. Les actions incertaines induisent alors une hausse équivalente de ces spécificités (figures 5.9 et 5.10) sans qu'une différence soit marquée par rapport à la longueur des séquences comportementales.
Les figures des tailles des populations de classeurs et des spécificités moyennes des classeurs sont données en annexe B.5.

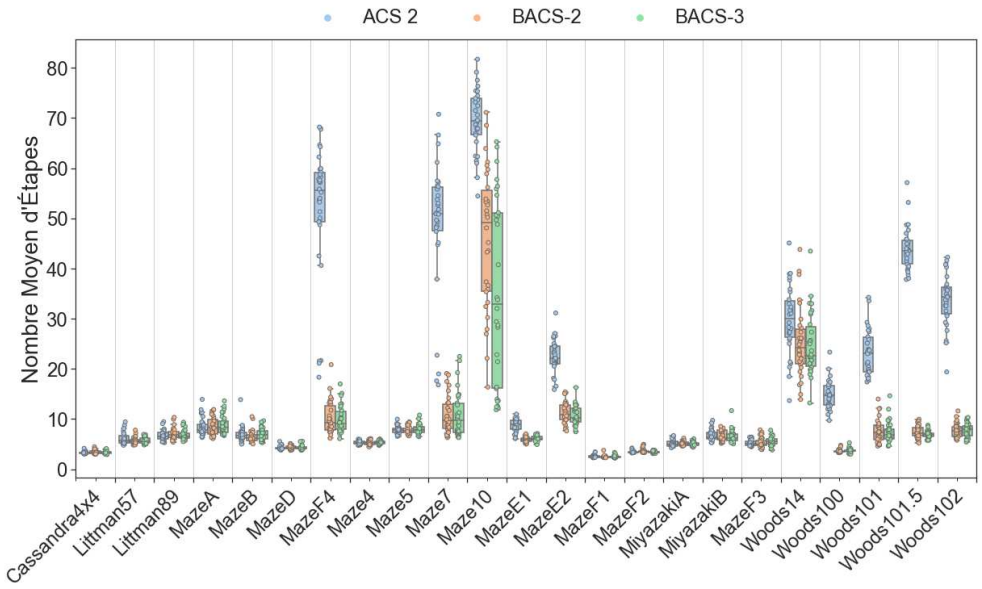


FIGURE 5.13 – Nombres moyens d'étapes d'ACS 2 et de BACS pour atteindre la sortie des labyrinthes, avec l'incertitude liée à la réalisation des actions.

Dans les environnements sans PAI, les nombres moyens d'étapes nécessaires d'ACS 2 et de BACS sont identiques pour 6 environnements, BACS-2 et BACS-3 étant plus rapides dans le dernier (*Woods14*). Dans les labyrinthes avec PAI, BACS est plus rapide qu'ACS 2 à atteindre la sortie pour 9 labyrinthes, ou est aussi rapide que ce dernier dans les 7 restants. La figure 5.13 donne le détail de cette métrique pour l'ensemble du banc de test. Les BSeq permettent alors de résoudre plus efficacement la tâche d'apprentissage, même avec les actions incertaines.

5.2.3 Comparaison avec d'autres systèmes

Le tableau 5.1 permet de comparer les capacités des systèmes de classeurs qui utilisent les séquences comportementales, quand ils doivent mettre en place

une politique de décision leur permettant de résoudre leur tâche d'apprentissage dans des environnements sans actions incertaines. Cette comparaison s'appuie sur l'ACS de (Stolzmann, 1999) noté ACS-S, l'ACS de (Métivier and Lattaud, 2002) ACS-M et BACS, avec pour témoin ACS 2. Bien que les protocoles expérimentaux soient différents, il est possible de vérifier que les BSeq permettent à ces systèmes de construire des politiques de décision à même de résoudre efficacement leur tâche d'apprentissage. Une comparaison entre ACS-S et BACS montre que leurs capacités sont au moins équivalentes, de même entre ACS-M et BACS. Ces résultats permettent de confirmer l'apport bénéfique des BSeq, mais ne permettent pas de comparer les représentations environnementales construites pour lesquelles il y a trop peu de données chiffrées dans les publications des auteurs : (Métivier and Lattaud, 2002) indique seulement que son ALCS analogue à BACS-3 converge vers une population de 2500 classeurs, contre 1800 en moyenne pour BACS-3.

	ACS-S		ACS-M			ACS 2		BACS-2		BACS-3	
$bs_{\max}$	1	2	1	2	3	1		2		3	
MazeE1	∅		$\mu = 4$	$\mu = 3.33$	∅	$\mu = 5.7$	$\sigma = 1.9$	$\mu = 3.5$	$\sigma = 0.3$	$\mu = 3.5$	$\sigma = 0.2$
MazeE2	∅		∅	$\mu = 6.5$	$\mu = 6.5$	$\mu = 46.2$	$\sigma = 6.0$	$\mu = 5.0$	$\sigma = 0.8$	$\mu = 5.1$	$\sigma = 0.6$
MazeF1	$\mu = 1.8$	∅	∅			$\mu = 1.8$	$\sigma = 0.0$	$\mu = 1.8$	$\sigma = 0.0$	$\mu = 1.8$	$\sigma = 0.0$
MazeF2	$\mu = 2.5$	∅	∅			$\mu = 2.5$	$\sigma = 0.1$	$\mu = 2.5$	$\sigma = 0.0$	$\mu = 2.5$	$\sigma = 0.0$
MazeF3	$\mu = 3.38$	∅	∅			$\mu = 3.37$	$\sigma = 0.1$	$\mu = 3.37$	$\sigma = 0.1$	$\mu = 3.37$	$\sigma = 0.1$
MazeF4	∅	$\mu = 4.5$	∅			$\mu = 45.6$	$\sigma = 15.5$	$\mu = 4.5$	$\sigma = 0.1$	$\mu = 4.5$	$\sigma = 0.1$
Woods100	∅		$\mu = 400$	$\mu = 2.33$	∅	$\mu = 27.0$	$\sigma = 3.6$	$\mu = 2.33$	$\sigma = 0.0$	$\mu = 2.33$	$\sigma = 0.1$

TABLE 5.1 – Nombres moyens d'étapes pour atteindre la sortie des labyrinthes avec les ALCS utilisant les séquences comportementales et ACS 2 en guide de témoin, sans actions incertaines.
L'ALCS de (Stolzmann, 1999) est désigné par l'acronyme ACS-S, et celui de (Métivier and Lattaud, 2002) par ACS-M. Pour ACS-S, les valeurs indiquées sont celles des optimima *vers lesquelles les courbes données semblent converger. Pour ACS-M, les moyennes μ indiquées sont issues du papier où elles ont été données explicitement. Aucun écart-type σ n'est donné pour ACS-S et ACM-M puisqu'ils n'ont pas été fournis.*

5.3 Discussion

5.3.1 Des classeurs plus nombreux

L'usage des séquences comportementales induit une augmentation du nombre de classeurs dans toutes les populations développées par BACS (figure 5.4), et qui est d'autant plus forte qu'un environnement est incertain : c'est-à-

dire s'il y a plus de situations où BACS est confronté au problème d'aliasing perceptif ; si ces situations nécessitent l'emploi de plusieurs actions pour les traverser parce que ces situations se suivent ; si l'exécution d'une action est incertaine. Pour ce dernier cas, le mécanisme de détection du PAI ne différencie par les situations propres au PAI des autres formes de stochasticité possibles comme les actions incertaines. De nombreux classeurs comportementaux sont ainsi générés par BACS dans toutes les situations possibles d'un environnement, alors même que l'emploi des BSeq ne serait pas nécessaire. Par ailleurs, BACS ne s'appuie pas sur le seuil θ_{as} de contrôle de la taille des ensembles d'action introduit dans les algorithmes génétiques : cela peut laisser BACS donner plus de poids aux classeurs comportementaux dans ses politiques de sélection de classeurs, mais au prix d'un accroissement des populations de classeurs.

Pour autant, l'accroissement des populations de BACS est limité par la création des classeurs comportementaux qui doivent nécessairement s'appuyer sur un classeur non marqué : autrement dit, un classeur qui n'a pas échoué à anticiper les prochaines situations environnementales. Ce mécanisme est alors très limitant lorsque l'exécution des actions est incertaine, les classeurs ne pouvant anticiper l'ensemble des situations qu'ils peuvent atteindre.

5.3.2 Spécificité des classeurs et actions incertaines

Les séquences comportementales impactent aussi les spécificités des populations de classeurs réguliers de BACS puisque :

- sans les actions incertaines, ces populations sont plus générales pour ACS 2 que BACS ;
- à l'inverse avec les actions incertaines, ces populations sont plus générales pour BACS qu'ACS 2.

L'usage des séquences comportementales permettent à BACS de réaliser de « nouvelles » actions, puisque ces chaînes d'actions sont considérées et réalisées comme un bloc. Pour un nombre donné de fois où les processus de découverte de nouvelles règles sont applicables, BACS fait entrer des classeurs réguliers et des classeurs comportementaux, alors qu'ACS 2 ne fait entrer que des classeurs réguliers. Les classeurs réguliers de BACS entrent

ainsi moins souvent que les classeurs réguliers d'ACS 2 dans le processus de généralisation, ce qui permet d'expliquer les différences observées sans les actions incertaines. Ces différences sont d'ailleurs plus importantes si les classeurs comportementaux sont plus nombreux.

Si les actions sont incertaines, les anticipations des classeurs sont plus difficiles à construire puisque plusieurs situations doivent être anticipées. Or, les classeurs ne sont pas capables d'anticiper plusieurs situations. Ils sont alors plus propices aux erreurs d'anticipation, que BACS et ACS 2 vont ensuite tenter de spécifier afin de les prévenir, ce qui justifie l'augmentation globale des spécificités.
Par ailleurs, le mécanisme de généralisation génétique cherche à généraliser les classeurs réguliers. Mais quand les actions sont incertaines, un classeur a plus de chance de mal anticiper les prochaines situations et d'être en conséquence spécifié, d'autant plus s'il a été généralisé. La pression de généralisation renforce alors la pression de spécification, ce qui est corrélé par le fait que les classeurs réguliers de BACS sont plus généraux que ceux d'ACS 2 : ceux de BACS subissent une pression de généralisation plus faible que ceux d'ACS 2 à cause des BSeq.

Enfin, les spécificités des classeurs comportementaux sont globalement équivalentes entre BACS-2 et BACS-3, pouvant indiquer que la longueur des séquences n'a pas d'influence sur cette métrique, avec ou sans les actions incertaines. Les actions incertaines impliquent une forte augmentation de cette métrique qui est souvent maximale, comme pour les classeurs réguliers (figures 5.9 et 5.10). Les classeurs comportementaux sont complètement spécifiés puisque plus il y a d'actions à réaliser, plus il y a de chances pour qu'une action différente de celle escomptée soit effectivement réalisée, entraînant de fait des erreurs d'anticipation. La longueur des BSeq a donc un effet sur les spécificités des classeurs comportementaux : leur équivalence relève du fait que ces classeurs sont complètement spécifiés à cause des actions incertaines. En revanche, sans les actions incertaines, la longueur des BSeq ne semble pas avoir d'influence sur leur spécificité. Indépendamment de $bs_{\max}$, les classeurs comportementaux subissent des pressions de spécification équivalentes à tous les classeurs, et une pression de généralisation très faible puisqu'ils n'entrent pas dans le processus associé : pour qu'un classeur comportemental plus général soit créé, ses parents doivent avoir été généralisés et non

marqués.

5.3.3 Des représentations incomplètes

Les séquences comportementales ne permettent pas de décrire directement les transitions environnementales à partir des situations où le problème d'aliasing perceptif est présent : c'est pourquoi les ratios de connaissance n'atteignent pas les 100% dans les environnements avec le PAI. Elles apportent tout de même de nouvelles connaissances à leur propos quand ces classeurs sont fiables. Par exemple, il est possible de déterminer ces transitions en croisant les classeurs comportementaux qui traversent ces situations, avec les classeurs qui anticipent ces situations : l'ensemble des situations intermédiaires d'une séquence comportementale peut ainsi être reconstruit.

Si les actions sont incertaines, nous indiquions que BACS et ACS 2 sont amenés à anticiper plusieurs situations environnementales alors qu'ils n'en ont pas la capacité. La qualité des classeurs avec les actions incertaines est plus faible : c'est pourquoi les ratios de connaissance chutent tous. Quand BACS et ACS 2 essaient d'apprendre une transition, ils essaient d'apprendre celle qui serait statistiquement la plus observée et pour laquelle un classeur aurait le plus de chance d'être fiable. Mais les erreurs d'anticipation dues à l'incertitude sont trop nombreuses pour permettre à un classeur d'être fiable de manière stable.

Enfin, nous indiquions que les ratios de connaissance d'ACS 2 sont globalement plus élevés que ceux de BACS (5.12) lorsque les actions sont incertaines, ce qui est causé par la présence des BSeq dans BACS. La mise en place de classeurs fiables avec cette forme de stochasticité nécessite au moins plus de ressources calculatoires, ce dont dispose ACS 2 relativement à BACS qui doit faire évoluer en plus ses classeurs comportementaux.

5.3.4 Séquences comportementales et PAI

Les résultats de BACS et ACS 2 permettent de mettre en évidence un lien entre les différents types du problème d'aliasing perceptif et les capacités de

ces systèmes à se doter de politiques de décision à même de résoudre leur tâche d'apprentissage.

Les capacités d'ACS 2 et de BACS sont équivalentes dans les environnements où s'ils sont confrontés au PAI de type I avec et sans les actions incertaines (figures 5.3 et 5.13). Le PAI de type I intervient lorsqu'un système perçoit de la même manière deux situations environnementales distinctes, ces situations étant à distances différentes de la sortie et où, pour chacune d'elles, une même action est à réaliser pour se rapprocher idéalement de la sortie (action *optimale*). ACS 2 reçoit, en ces situations particulières, des renforcements différents pour toutes les actions réalisables qui vont être agrégés par le processus de rétribution. ACS 2 est alors capable de déterminer l'action optimale puisque les renforcements calculés pour cette action sont les plus grands possibles. Il est néanmoins délicat pour ACS 2 de stabiliser l'emploi de cette action optimale dans sa politique de décision, comme dans *Littman57* (figure 5.3) : l'emploi de cette action dépend de la qualité d'anticipation des classeurs associés qui est instable et dépend aussi des actions passées du système. Les BSeq offrent alors la possibilité de développer une politique de décision plus stable, puisqu'un système les utilisant ne serait pas confronté au bruit induit par le PAI sur les valeurs de renforcement et sur les valeurs de qualité. Un raisonnement similaire est applicable pour le pseudo PAI où le bruit des renforcements serait plus faible.

En revanche, les BSeq permettent à BACS de gérer le PAI de type II et III, là où ACS 2 ne parvient pas à identifier quelles actions doivent être réalisées. Ces formes de PAI sont celles où ACS 2 éprouve le plus de difficultés puisque des actions différentes, potentiellement contradictoires, sont à réaliser dans les situations présentant ces formes. S'il ne peut éviter ces situations comme dans Woods100, ACS 2 se retrouve alors dans des situations de blocage : il réalise des actions s'annulant successivement jusqu'à ce que les valeurs de renforcement aient suffisamment évoluées pour en faire d'autres. Les actions incertaines semblent à ce propos intervenir pour aider ACS 2 à sortir de ces situations de blocage (figure 5.11), puisqu'une action différente de celles qui mènent et perpétuent ce blocage est réalisable à tout moment.

5.3.5 Des séquences comportementales sous-optimales

Les séquences comportementales permettent globalement d'améliorer les capacités de BACS à se doter en toute autonomie de politiques de décision plus efficaces pour résoudre sa tâche d'apprentissage (figures 5.3 et 5.13). Elles restent néanmoins perfectibles.

BACS ne favorise pas les BSeq les plus courtes : c'est pourquoi BACS-2 est plus performant que BACS-3 dans *Woods101*, *Woods101.5* et *Woods102*. En effet, le biais d'erreur impacte les politiques de décision de BACS pour les situations proches de celles associées au PAI : BACS cherche d'autant plus à employer des séquences comportementales qu'elles sont nombreuses dans la population, indépendamment de leur utilité, comme illustré par la figure 5.14. Lier l'adéquation des classeurs de BACS avec le nombre d'actions dans leur *Conséquent* permettrait alors d'atténuer ce mauvais usage des séquences.

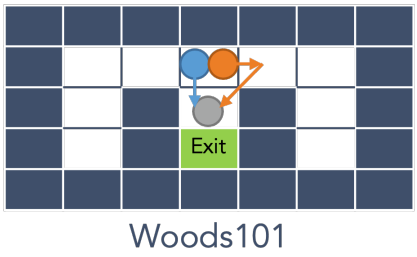


FIGURE 5.14 – Différentes séquences d'actions de BACS dans *Woods101* afin de se rapprocher de la sortie.
Dans le labyrinthe Woods101, *il est possible d'atteindre la situation grise aux portes de la sortie à partir de la situation environnementale bleue et orange avec deux séquences d'actions différentes : soit directement en allant vers le sud pour la séquence bleue ; soit en allant à l'est vers la situation associée au PAI de type III, puis au sud-ouest pour la séquence orange. La séquence la plus courte n'est pas favorisée par BACS.*

Permettre à BACS de se doter de séquences comportementales plus longues peut l'amener à se doter de politiques de décision moins efficaces mais plus stables. Au contraire, trop limiter la longueur des séquences comportementales peut prévenir la résolution de la tâche d'apprentissage ou induire plus d'instabilité dans la construction des politiques de décision. Cet effet apparaît clairement pour le labyrinthe *Maze10*. BACS-2 est passé 20 fois sous la barre de 8 actions en moyennes pour atteindre la sortie de cet environnement, contre 5 pour BACS-3. Mais BACS-3 ne passe pas au-dessus de la barre des 11 actions moyennes, contre 4 fois pour BACS-2 allant jusqu'à atteindre 55

actions en moyenne. Cette instabilité des résultats est appuyée par l'écart-type associé au nombre moyen d'étapes qui est de 11.35 pour BACS-2 et de 1.28 pour BACS-3. L'instabilité de BACS-2 ne permet pas de différencier statistiquement ses performances dans *Maze10* de celles de BACS-3, ce qui apparaît si ses valeurs extrêmes ne sont plus considérées : BACS-2 ($\mu = 7.42, \sigma = 0.83$) est alors effectivement plus rapide que BACS-3 ($\mu = 8.92, \sigma = 1.28$) à atteindre la sortie avec une valeur p de $3e - 06$.

Des séquences plus longues réduisent les écarts de performance entre BACS-2 et BACS-3 avec les actions incertaines : BACS-2 atteint plus rapidement la sortie des labyrinthes que BACS-3 1 fois au lieu de 4, et BACS-3 est plus rapide que BACS-2 2 fois au lieu d'1. Ce résultat met en lumière deux effets. D'un côté, les populations de classeurs de BACS-3 sont relativement plus proches de celles de BACS-2. Bien que celles de BACS-3 soient toujours les plus grandes, l'écart entre les populations est au maximum de l'ordre de 10% (contre 60% sans les actions incertaines). Il y a donc plus de chances que BACS-3 construise et utilise des BSeq équivalentes à celles de BACS-2, ce qui explique les similarités entre les capacités des deux systèmes à atteindre la sortie. Ce point est encore plus appuyé qu'il est difficile pour BACS-3 de construire des classeurs comportementaux de longueur maximale, puisque des séquences plus longues réduisent la probabilité qu'un classeur anticipe correctement les changements perceptifs.
D'un autre côté, BACS-3 semble capable de mieux gérer les actions incertaines que BACS-2 puisque ses performances se rapprochent de celles de BACS-2, voire sont meilleures. Des BSeq plus longues permettent à BACS d'accroître ses chances de se rapprocher de la sortie en prenant la direction qui y mène. L'adéquation des classeurs s'appuie autant sur la qualité que sur les renforcements estimés. Avec les actions incertaines, la qualité de l'ensemble des classeurs est fortement instable puisqu'ils ne peuvent fidèlement anticiper les changements perceptifs : le processus de rétribution occupe alors une place plus importante pour déterminer quels classeurs prennent part dans la politique de décision construite.

BACS discrimine les classeurs dont les séquences comportementales font boucler le système entre des situations à priori identiques. Ce mécanisme semble efficace puisque la capacité à atteindre rapidement la sortie de BACS-2 est équivalente ou meilleure à celle de BACS-3 pour la majorité des environ-

nements du banc de test (14 labyrinthes avec les actions incertaines, sinon 15). Cependant, il relèverait plutôt du mécanisme de rétribution de BACS d'adapter complètement ses politiques de décision en fonction des BSeq pour deux raisons. D'une part, réduire la qualité d'un classeur alors que ses anticipations sont correctes introduit une incohérence relative à la lecture de cet attribut : la qualité d'un classeur ne devrait être réduite que si ses anticipations sont erronées. D'autre part, il est des environnements où plusieurs états en lien avec le PAI sont juxtaposés. Ces environnements sont alors plus propices à ce que différentes situations dont l'observation est identique se succèdent, à l'image d'un *couloir* où la meilleure façon d'en sortir est de toujours continuer. BACS peut alors discriminer des classeurs qui proposeraient avec justesse d'avancer dans ce *couloir* parce qu'il ne percevrait pas de changements.

5.4 Synthèse

Les séquences comportementales ont été intégrées à ACS 2 pour accroître son autonomie dans des environnements incertains, en particulier quand il est confronté au problème d'aliasing perceptif. Le système résultant, *Behavioral Anticipatory Classifier System*(BACS), est capable de construire des classeurs dont les *Conséquents* sont composés de plusieurs actions. Les BSeq permettent de traverser directement des situations dans lesquelles il ne serait pas en mesure de prendre une décision adaptée à la résolution de sa tâche d'apprentissage.

L'intégration des BSeq avec BACS a nécessité une adaptation des politiques de sélection d'action en politiques de sélection de classeurs, un mécanisme de détection de l'incertitude qui décèle le PAI et un dispositif de construction et de contrôle des classeurs ayant des BSeq.

Les résultats montrent que les séquences comportementales permettent effectivement à BACS de se doter de politiques décisionnelles plus efficaces pour résoudre ses tâches d'apprentissage par rapport à ACS 2. Les BSeq sont en particulier pertinentes pour les PAI de type II et de type III qui préviennent la résolution de la tâche d'apprentissage dans ACS 2. Les BSeq permettent

aussi de stabiliser le comportement de BACS avec le PAI de type I. Enfin, les BSeq n'affectent pas négativement la construction des politiques de décision, indépendamment de l'incertitude environnementale. Cependant, l'emploi des séquences comportementales impacte les représentations environnementales : celles-ci sont plus volumineuses, spécifiques et ne permettent pas de construire des représentations complètes des environnements.

Le mécanisme des BSeq est améliorable :

- en introduisant un mécanisme qui favorise les séquences comportementales les plus courtes ;
- en contrôlant plus finement la création de ces classeurs uniquement lorsque le système est confronté au PAI ;
- en contrôlant l'accroissement des séquences comportementales avec des mécanismes de contrôle de la population, tels que celui introduit avec la généralisation génétique ou avec des techniques d'extraction de connaissances ;
- en permettant une généralisation indirecte et maîtrisée des classeurs comportementaux. Une généralisation directe ne reste pas souhaitée pour éviter que ces classeurs soient employés en des situations où ils ne seraient pas nécessaires. Une généralisation indirecte est déjà possible si les parents des classeurs à BSeq ont été généralisés avec succès. Une nouvelle approche peut s'inspirer de ce constat afin d'éviter une surspécialisation de ces classeurs, potentiellement synonyme de surapprentissage si des attributs conditionnels ont été spécifiés sans que cela ne soit nécessaire aux classeurs.

Dans les chapitres suivants, nous ne considérerons plus que BACS-3 sous l'acronyme BACS. Nous utiliserons au sein des ALCS qui utilisent des BSeq une longueur maximale $bs_{\max}$ de 3, puisque cette valeur permet de faire apparaître le plus de différences avec les pistes d'améliorations mises en lumière. Par exemple, c'est pour cette valeur de $bs_{\max}$ que les populations sont les plus grandes et qu'il y a plus de choix parmi les séquences d'actions possibles.

Une dernière voie d'amélioration serait de permettre aux classeurs d'apprendre à correctement anticiper plusieurs changements perceptifs antagonistes où un même attribut d'une *Anticipation* peut prendre des valeurs diffé-

rentes selon l'incertitude environnementale. Le système serait alors en mesure de se doter de représentations environnementales complètes, indépendamment de l'incertitude des environnements où il évolue. Nous proposons d'utiliser en ce sens le mécanisme des prédictions améliorées par les probabilités (PEP), que nous avons introduit dans ACS 2 afin de l'étudier, et qui est l'objet du chapitre suivant.

CHAPITRE 6

PEPACS : Accroître l'autonomie et l'explicabilité avec des anticipations améliorées

L'ajout des prédictions améliorées par les probabilités (PEP) au système de classeurs à anticipation ACS 2 doit lui permettre de se doter de représentations complètes des environnements incertains. Les PEP permettent à un ALCS d'anticiper plusieurs changements perceptifs décrits pour chacun des attributs perceptifs des Anticipations des classeurs, ce qui arrive dans des environnements incertains. Nous montrerons que le système résultant, Probability-Enhanced Predictions in Anticipatory Classifier System (PEPACS), peut fournir plus d'éléments explicatifs relatifs au comportement du système dans son environnement et aussi, améliorer ses capacités d'apprentissage quand il est confronté à l'incertitude environnementale.

Sommaire

Les travaux présentés dans ce chapitre ont fait l'objet d'une publication : (Orhand et al., 2020c).

6.1 Intégrer les PEP dans ACS 2

Le mécanisme de détection du PAI détecte plus que les situations où un ALCS est confronté au problème d'aliasing perceptif : ce mécanisme détecte différentes formes d'incertitude telles que les actions incertaines ou des observations bruitées. Le mécanisme de détection d'incertitude utilisé dans PEPACS est le même que celui de BACS (chapitre 5 section 5.1.2), de (Stolzmann, 1999) et de (Métivier and Lattaud, 2002). Il déclenche la construction de classeurs qui seront capables de décrire au travers des PEP tous les changements perceptifs à anticiper. Les classeurs améliorés par les PEP (ou classeurs à PEP) ne sont pas directement construits quand la détection est fructueuse. Chaque classeur d'un ensemble peut d'abord être marqué par un attribut *enhanced effect ee* à *True* lorsque la détection est positive, puis tous les classeurs d'un ensemble d'action dont l'attribut *ee* est *True* sont utilisés pour générer des classeurs à PEP. La construction des classeurs à PEP intervient ainsi à la toute fin de l'ALP.

6.1.1 Construction des classeurs à PEP

PEPACS essaie de générer un classeur à PEP pour chaque classeur d'un ensemble d'action dont l'attribut *ee* est *True*. Deux classeurs cl_1 et cl_2 sont nécessaires à la construction d'un classeur à PEP, où :

- les *Conditions* de cl_1 et cl_2 sont différentes, ainsi que leurs *Anticipations*, ce qui permet d'éviter la construction de classeurs superflus, car identiques à ceux existants ;
- les *Marques* de cl_1 et cl_2 sont identiques, ce qui permet d'éviter de s'appuyer sur des classeurs qui auraient été utilisés dans des situations différentes.

Un classeur à PEP cl_{pep} est créé à partir d'un premier classeur cl_1 et d'un classeur cl_2 choisi aléatoirement parmi ceux qui remplissent les conditions précédentes, comme illustré sur la figure 6.1, où tous ses attributs sont mis à leurs valeurs par défaut sauf :

- la *Condition* de cl_{pep} est spécifiée pour tous les attributs perceptifs spécifiés des *Conditions* de cl_1 et de cl_2 ;

— le *Conséquent* de cl_{pep} est celui de l'ensemble d'action ;
— l'*Anticipation* de cl_{pep} est construite de telle sorte que chacun de ses attributs perceptifs consiste en la fusion des attributs correspondants des *Anticipations* de cl_1 et de cl_2. Si deux attributs à fusionner de cl_1 et de cl_2 n'indiquent pas de changement, alors l'attribut correspondant de cl_{pep} est $\#$. Si l'un des attributs de cl_1 ou de cl_2 correspond au symbole générique $\#$, la pénultième observation est utilisée pour déterminer le symbole à fusionner. Une PEP est construite pour chaque attribut perceptif si les attributs correspondants des *Anticipations* de cl_1 et de cl_2 (voire de la pénultième observation) sont différents. Pour chaque symbole d'une nouvelle PEP, les valeurs de probabilité associées aux symboles sont additionnées à partir d'une valeur par défaut 1 si le symbole ne vient pas d'une PEP, sinon à partir des valeurs respectives des PEP de cl_1 et de cl_2, puis normalisées ;
— la récompense de cl_{pep} est la récompense moyenne de cl_1 et de cl_2 ;
— la qualité de cl_{pep} est fixée au maximum entre la qualité moyenne de cl_1 et de cl_2 et la valeur initiale par défaut (0.5).

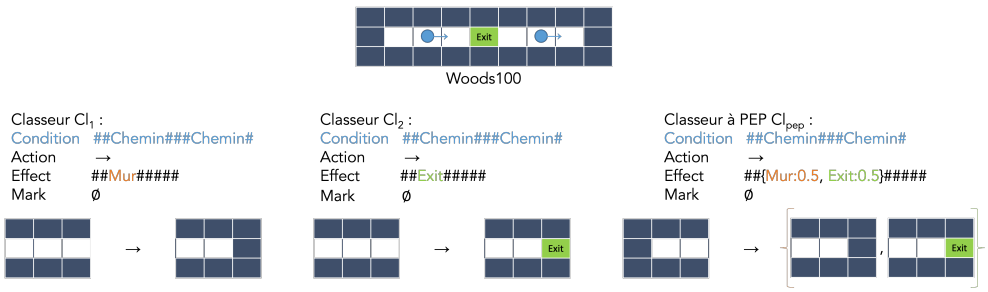


FIGURE 6.1 – Construction d'un classeur à PEP dans *Woods100*.
Dans le labyrinthe Woods100*, il est possible d'atteindre la sortie ou un cul-de-sac à partir des situations bleues en allant vers la droite. Un classeur amélioré par les PEP peut être construit par PEPACS pour apprendre à anticiper ces deux situations environnementales.*

Chaque nouveau classeur cl_{pep} n'est pas directement ajouté à la population de classeurs de PEPACS : il est inséré à celle-ci par le biais de la fonction d'ajout de classeurs de l'ALP décrite par l'algorithme 3.2.6 (chapitre 3 section 3.2.5). Ainsi, un nouveau classeur qui existerait déjà dans la population ou qui serait subsumé par un autre classeur de la population est abandonné. La subsomption de classeurs à PEP est décrite dans la section 6.1.3. L'algorithme 6.1.1 fait la synthèse du processus de génération des classeurs à PEP, de la sélection de classeurs générant les classeurs à PEP jusqu'à leur insertion dans

la population. L'algorithme 6.1.1 s'appuie sur l'avant-dernière observation de l'environnement σ_{t-1}, l'ensemble d'action $[A]$, l'ensemble d'appariement $[M]$, la population $[P]$ de classeurs, le pas d'apprentissage β_{alp} de l'ALP, le seuil d'expérience θ_{exp} et le seuil de fiabilité θ_r.

Algorithme 6.1.1 Processus de génération des classeurs à PEP de PEPACS

function APPLYPEPBUILDINGPROCESS(σ_{t-1},β_{alp},θ_{exp},θ_{r},$[A]$,$[M]$,$[P]$)
  $list_1 \leftarrow \emptyset$
  for each $cl_A \in [A]$ **do**
    if $ee_{\text{cl}_A} == True$ **then**
      Append cl_A to $list_1$
    end if
  end for
  if length of $list_1 \geq 2$ **then**
    for each $cl_1 \in list_1$ **do**
      $list_2 \leftarrow \emptyset$
      for each $cl \in list_1$ **do**
        if $C_{\text{cl}} \neq C_{\text{cl}_1}$ **and** $E_{\text{cl}} \neq E_{\text{cl}_1}$ **and** $M_{\text{cl}} == M_{\text{cl}_1}$ **then**
          Append cl to $list_2$
        end if
      end for
      if length of $list_2 > 0$ **then**
        Choose cl_2 randomly amongst $list_2$
        Build cl_{pep} from cl_1 and cl_2 with σ_{t-1}
        Insert cl_{pep} in $[A]$ and $[P]$ (or $[M]$) with alg 3.2.6
      end if
    end for
  end if

6.1.2 Adapter les processus de découvertes de règles

Tout d'abord, quand un classeur à PEP est copié, toutes les PEP de son *Anticipation* sont spécifiées pour uniquement correspondre à l'observation courante. La copie d'un classeur intervient principalement quand le mécanisme de généralisation génétique essaie de généraliser un classeur ou quand un classeur est spécifié par l'ALP. Cette modification du processus de copie doit principalement permettre d'éviter que les classeurs à PEP deviennent surgénéraux. Un classeur à PEP est surgénéral s'il est utilisé dans des situations qui ne nécessitent pas de PEP : il décrit donc plusieurs changements perceptifs superflus.

Ensuite, un classeur avec des PEP *cl* anticipe bien des changements perceptifs entre la situation courante décrite par P_t et la situation précédente décrite par P_{t-1} si pour chaque attribut perceptif e de l'*Anticipation* de *cl* :

- si e est une PEP, alors le symbole décrit par l'attribut perceptif correspondant de P_t est contenu dans e ;
- sinon e doit décrire le changement environnemental observé entre les attributs correspondants de P_t et P_{t-1} : s'ils sont identiques e doit valoir $\#$, sinon il doit avoir la valeur de celui de P_t.

Les PEP permettent qu'un attribut perceptif soit spécifié dans une *Condition* d'un classeur et que sa valeur soit en même temps présente dans la PEP associée d'une *Anticipation*. Alors, un classeur contenant au moins une PEP décrit toujours des changements perceptifs puisque plusieurs symboles sont spécifiquement décrits dans les *Anticipations* de ces classeurs : ils n'utilisent pas le symbole générique $\#$ marquant l'absence de changement dans les PEP.

Un classeur avec des PEP qui anticipe correctement la prochaine situation environnementale atteint l'*expected case* de l'ALP : les probabilités des PEP sont mises à jour selon les équations 3.9 détaillées précédemment dans le chapitre 3 section 3.3.1. Cette mise à jour s'appuie en particulier sur un pas d'apprentissage β_{pep} qui doit être défini en amont de l'apprentissage. Par défaut, ce pas vaut 0.01 pour équilibrer la vitesse à laquelle les probabilités vont converger vers leurs valeurs finales et l'approximation des valeurs réelles de ces probabilités.

Excepté la mise à jour des probabilités des classeurs à PEP et la détection de l'incertitude dans l'*expected case* de l'ALP, l'ALP de PEPACS est fortement similaire à celui d'ACS 2. La spécification introduite dans la copie ne permet juste pas au *Correctable case* de l'ALP de spécifier l'*Anticipation* d'un classeur avec des PEP : seule la construction des classeurs à PEP définie précédemment permet d'étoffer les PEP avec de nouvelles clés. Les classeurs à PEP sont ajoutés à la population de classeurs de la même façon que les classeurs ne contenant pas de PEP. L'ajout de classeurs s'appuyant sur un processus de subsomption, ce dernier est modifié pour prendre en compte les PEP.

6.1.3 Adapter le processus de subsomption

L'ALP et la généralisation génétique utilisent tous deux un mécanisme de subsomption afin de contrôler l'ajout de nouveaux classeurs. La subsomption nécessite d'ajuster la façon dont un classeur amélioré par les PEP peut subsumer ou être subsumé par un autre classeur, l'un des critères de subsomption portant sur les *Anticipations* des classeurs. Afin de prendre en compte les classeurs à PEP, la subsomption doit déterminer si un attribut d'une *Anticipation* est inclus dans une autre. Si E_{new} est l'*Anticipation* d'un nouveau classeur et E_{sub} l'*Anticipation* d'un classeur candidat à la subsomption, soit e_{new} (resp. e_{sub}) un attribut de E_{new} (resp. E_{sub}). E_{sub} subsume E_{new} si pour chaque attribut :

- si e_{new} et e_{sub} sont des PEP, alors e_{sub} doit contenir tous les symboles de e_{new} ;
- si seulement e_{sub} est une PEP, alors e_{sub} doit contenir le symbole décrit par e_{new} ;
- si seulement e_{new} est une PEP, alors E_{sub} ne peut pas subsumer E_{new} ;
- sinon, les deux symboles décrits par e_{new} et e_{sub} doivent être identiques.

6.2 Capacités de PEPACS

6.2.1 Méthodologie

Les métriques d'évaluation et le protocole expérimental détaillés dans le chapitre 4 doivent permettre d'évaluer les capacités d'apprentissage de PEPACS en répondant aux questions suivantes :

1. PEPACS se comporte-t-il comme ACS 2 dans les environnements où il n'est pas sujet à l'incertitude environnementale ?
2. PEPACS est-il capable de se doter de représentations environnementales complètes, indépendamment de l'incertitude environnementale ?
3. Le mécanisme des prédictions améliorées par les probabilités permet-il de décrire fidèlement les différentes probabilités des transitions environnementales ?

4. Comment les PEP influencent-elle la mise en place de politiques de décision dans des environnements incertains ?
5. PEPACS est-il capable d'éviter de construire des classeurs à PEP surgénéraux ?

Les capacités d'apprentissage de PEPACS sont d'abord comparées avec celles d'ACS 2 dont il est dérivé. Les nombres d'étapes nécessaires pour atteindre la sortie des labyrinthes permettent de comparer les politiques de décision mises en place par ces systèmes, apportant une réponse à la première et à la dernière interrogation. Les ratios de connaissance, les tailles des populations et les spécificités moyennes des classeurs fiables permettent de comparer les représentations environnementales développées par ces systèmes, complétant ainsi la réponse à la première interrogation et fournissent les éléments de réponse à la seconde interrogation.
Il est possible de savoir si toutes les transitions environnementales sont anticipées par les ratios de connaissance. Cependant, ces ratios n'indiquent pas si les probabilités calculées dans les PEP sont cohérentes avec les environnements où évoluent PEPACS : c'est pourquoi nous avons introduit et utilisons les Erreurs Moyennes sur la Prédiction de l'Anticipation (EMPA, chapitre 4 section 4.2.1) pour répondre à la troisième interrogation. ACS 2 sert ici de contrôle puisqu'il ne peut pas décrire plus d'une transition environnementale. Les EMPA d'ACS 2 ne seront pas maximales mais refléteront une erreur sur la prédiction de l'anticipation si seule la transition qui a le plus de chance d'aboutir est considérée lors du calcul de cette métrique.
Quand les actions incertaines ne sont pas actives dans les environnements, le mécanisme des PEP ne doit être actif que pour les situations où PEPACS est confronté au problème d'aliasing perceptif. Nous avons alors dénombré tous les classeurs fiables qui contiennent des PEP et qui correspondent à d'autres situations que celles où il y a le PAI : il s'agit des classeurs surgénéraux. Ce relevé a été réalisé pour PEPACS et une version de PEPACS sans la spécification introduite dans la copie de classeurs. Avec ce relevé, nous pouvons savoir si PEPACS parvient à éviter le construction de classeurs surgénéraux et ainsi répondre à la quatrième interrogation.

Les résultats de PEPACS sont présentés dans la section 6.2.2, comparés avec différents ALCS dans la section 6.2.3 et discutés dans la section 6.3.

6.2.2 Résultats

PEPACS se comporte-t-il comme ACS 2 dans les environnements où il n'est pas sujet à l'incertitude environnementale ?

Les ratios de connaissance des populations de classeurs d'ACS 2 et de PEPACS sont strictement identiques et maximaux : les deux systèmes sont parvenus à apprendre une représentation complète de leur environnement puisque toutes les transitions environnementales sont décrites par au moins un classeur fiable.
Les tailles des populations de classeurs, et donc des représentations environnementales, ne présentent pas de différences statistiques significatives à l'exception de deux cas : les populations de PEPACS sont plus grandes que celles d'ACS 2 dans les environnements *Maze4* et *Maze5*.
Les spécificités moyennes des classeurs ne présentent pas de différences significatives sauf dans *Maze4*, *Maze5* et *MazeA* où les classeurs de PEPACS sont moins généraux que ceux d'ACS 2.
Les modèles de transitions appris par ces systèmes de classeurs sont équivalents malgré les différences relevées. Celles-ci peuvent s'expliquer par l'emploi du mécanisme de généralisation génétique : il peut introduire une variation sur les tailles ou spécificités des classeurs due au hasard inhérent à son fonctionnement. PEPACS n'a utilisé à aucun moment des classeurs améliorés, ce qui conforte ce résultat. Nous fournissons en annexe C.1 des figures illustrant ces résultats.

Enfin, toutes les politiques de décision construites par ACS 2 et PEPACS ne présentent de différences significatives. Ainsi, PEPACS se comporte comme ACS 2 dans les environnements où il n'est pas sujet à l'incertitude environnementale.

PEPACS est-il capable de se doter de représentations environnementales complètes, indépendamment de l'incertitude environnementale ?

Tous les ratios de connaissance atteints par PEPACS sont maximaux pour tous les environnements avec le problème d'aliasing perceptif et si les actions

ne sont pas incertaines. En particulier, il n'y a pas un seul labyrinthe où ce ratio n'a pas atteint les 100% au cours des différents apprentissages. PEPACS, contrairement à ACS 2 est donc capable de se doter de représentations environnementales complètes lorsqu'il est confronté au PAI et que le résultat des actions est certain. Par ailleurs, toutes les représentations construites par PEPACS sont strictement plus petites que celles d'ACS 2 : par exemple, elles peuvent être réduites jusqu'à 50% des tailles moyennes de celles d'ACS 2 dans *MazeE1*. Elles sont aussi plus spécifiques que celles d'ACS 2 dans 11 des 16 labyrinthes avec le problème d'aliasing perceptif, équivalentes dans 4 autres environnements et plus générale dans *MazeE2*. Les figures C.4 à C.6 de l'annexe C.2 montrent les ratios de connaissance, les tailles des populations et les spécificités moyennes des classeurs fiables lorsque les actions ne sont pas incertaines.

Quand les actions sont incertaines, les ratios moyens de connaissance de l'ensemble des 23 labyrinthes du banc de test atteignent a minima les 98%, étant même maximaux pour 12 labyrinthes à chaque apprentissage (*cf.* figure 6.2). Il n'y a pas d'environnement où ces ratios n'auraient jamais atteint un ratio de connaissance de 100%. Autrement dit, PEPACS est capable de se doter d'une représentation complète de son environnement indépendamment de l'incertitude associée à celui-ci.

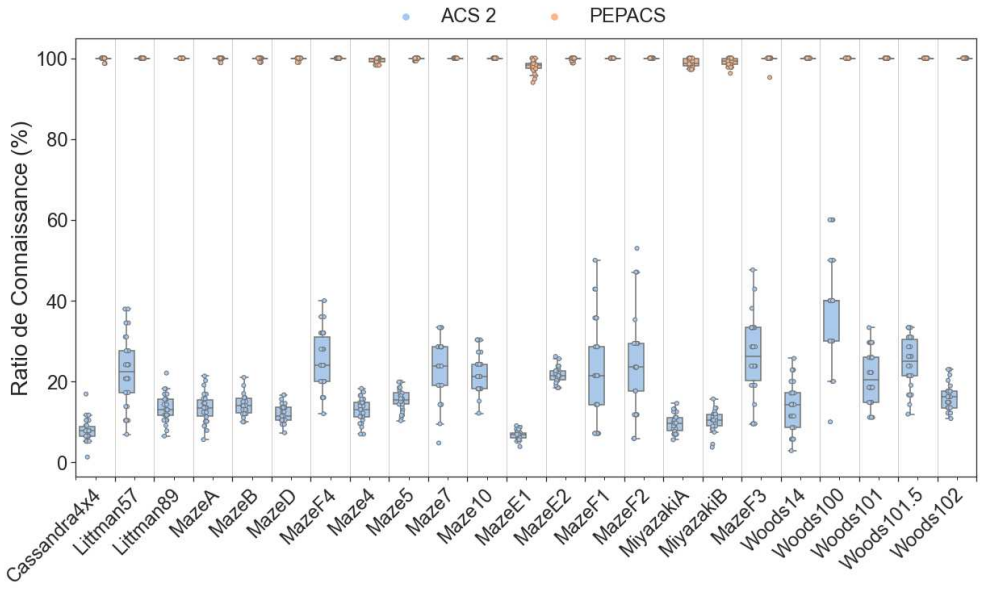


FIGURE 6.2 – Ratios de connaissance des populations de classeurs de PEPACS et ACS 2 dans tous les environnements avec les actions incertaines.

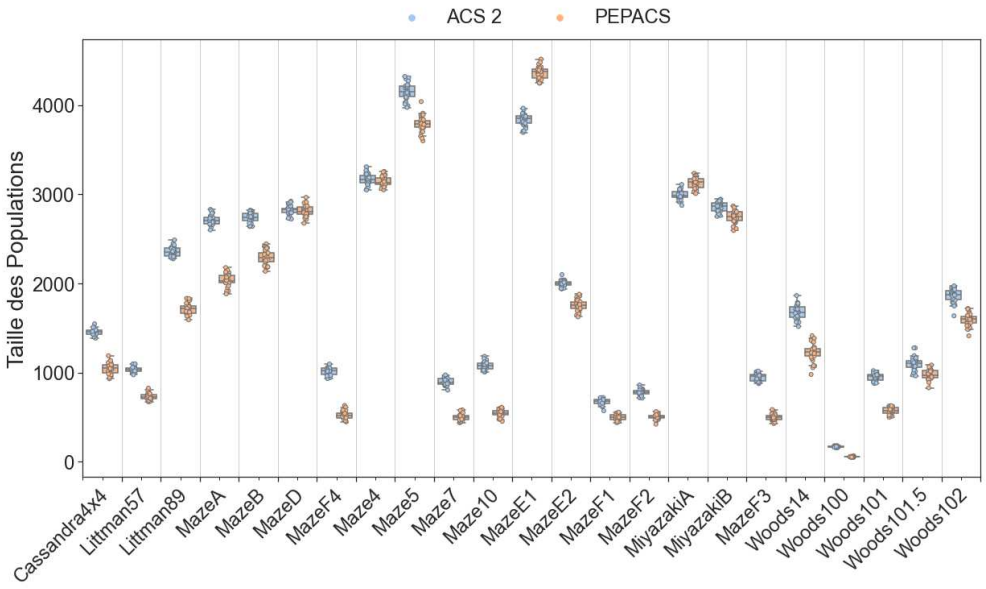


FIGURE 6.3 – Tailles des populations de classeurs de PEPACS et ACS 2 dans tous les environnements avec les actions incertaines.

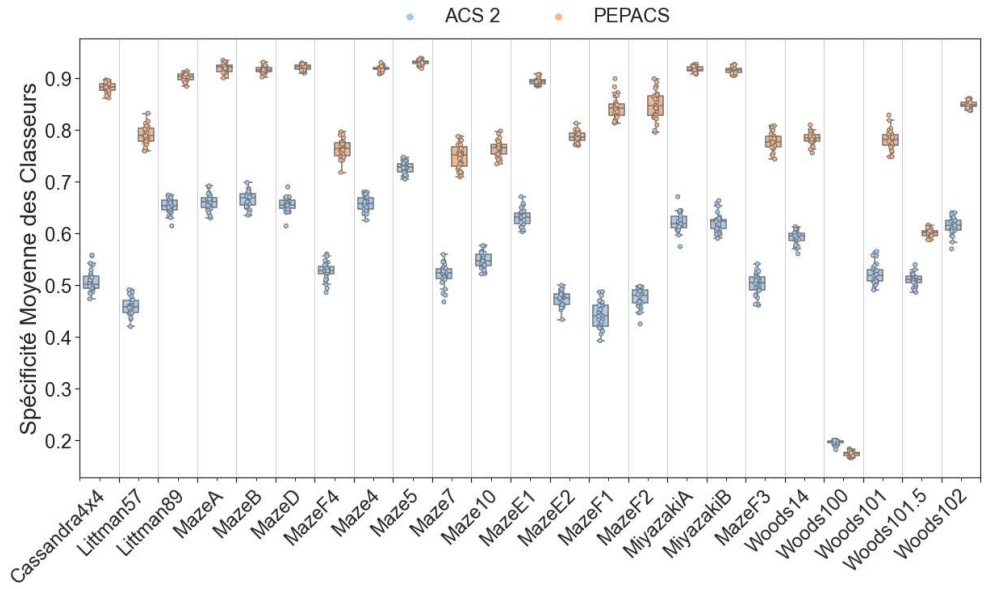


FIGURE 6.4 – Spécificités moyennes des classeurs fiables de PEPACS et ACS 2 dans tous les environnements avec les actions incertaines.

Les représentations construites par PEPACS avec les actions incertaines sont plus petites dans 19 environnements, de tailles équivalentes pour *MazeD* et *Maze4* et plus grandes pour *MazeE1* et *MiyazakiA* (*cf.* figure 6.3). Les spécificités moyennes des classeurs fiables de PEPACS sont plus élevées pour 22 labyrinthes, étant uniquement plus faibles pour *Woods100* (*cf.* figure 6.4).

Le mécanisme des prédictions améliorées par les probabilités permet-il de décrire fidèlement les différentes probabilités des transitions environnementales ?

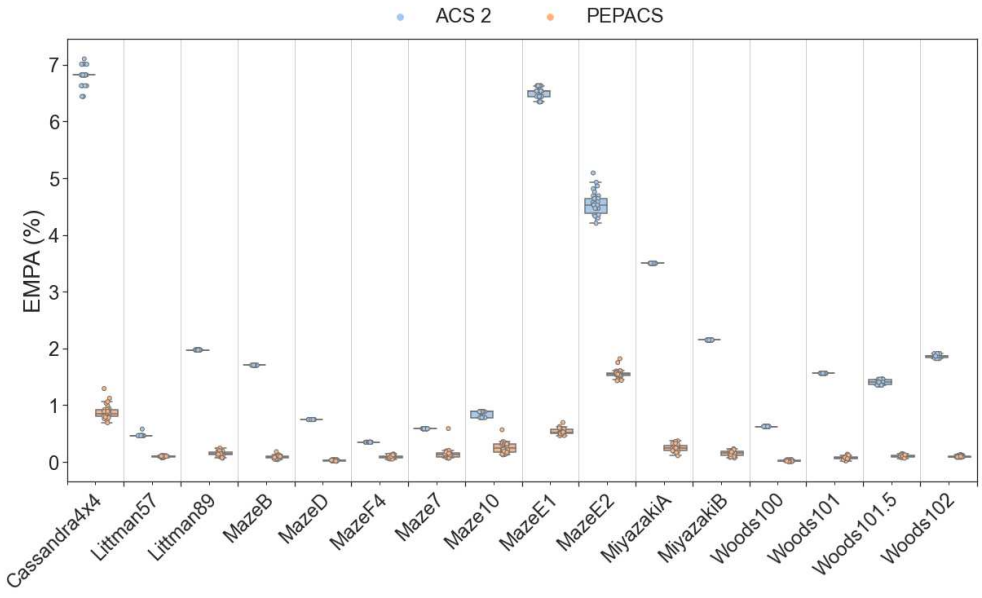


FIGURE 6.5 – Erreurs Moyennes sur la Prédiction de l'Anticipation de PEPACS et ACS 2 dans les environnements avec PAI et sans les actions incertaines.

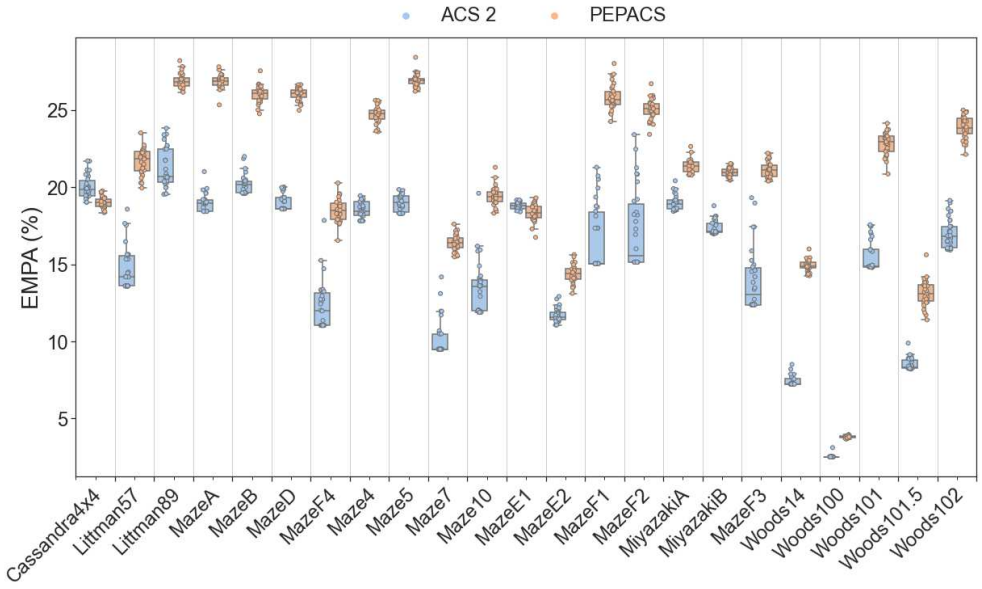


FIGURE 6.6 – Erreurs Moyennes sur la Prédiction de l'Anticipation de PEPACS et ACS 2 dans tous les environnements avec les actions incertaines.

Les Erreurs Moyennes sur la Prédiction de l'Anticipation (EMPA) de PEPACS sont les plus faibles dans tous les environnements avec PAI et sans les actions

incertaines (cf. figure 6.5). Dans ces conditions, PEPACS décrit plus fidèlement les probabilités des transitions environnementales qu'ACS 2.

En revanche, les EMPA d'ACS 2 sont les plus faibles pour 21 labyrinthes du banc de test et plus élevées pour *Casssandra4x4* et *MazeE1* (cf. figure 6.6). Autrement dit, le mécanisme des PEP ne permet plus de prédire les probabilités des différentes transitions environnementales avec les actions incertaines. Les PEP ne permettent également pas d'anticiper quelle situation environnementale aurait le plus de chance de se produire suite à la réalisation d'une action, puisque les EMPA sont globalement plus élevées que celles d'ACS 2.

Woods100 Maze

En partant du haut et dans le sens horaire,
seuls l'Est et l'Ouest sont indiqués puisque ce sont les seuls où il y a un changement.

Si le taux de bruitage est de 0% :

Est : Mur - 50% ; Exit - 50%
Ouest : Chemin - 100%

Si le taux de bruitage est de 1% :

Est : Mur - 49.5625% ; Exit - 49.5625% ; Chemin - 0.875%
Ouest : Mur - 0.0625% ; Exit - 0.0625 % ; Chemin - 99.875%

Si le taux de bruitage est de 10% :

Est : Mur - 45.625% ; Exit - 45.625% ; Chemin - 8.75%
Ouest : Mur - 0.625% ; Exit - 0.625 % ; Chemin - 98.75%

Si le taux de bruitage est de 25% :

Est : Mur - 39.0625% ; Exit - 39.0625% ; Chemin - 21.875%
Ouest : Mur - 1.5625% ; Exit - 1.5625 % ; Chemin - 96.875%

FIGURE 6.7 – Transition environnementale avec PAI dans *Woods100*. *Pour les* 4 *taux de bruit des actions (*0%, 1%, 10%, 25%*), nous indiquons également quelles sont les probabilités théoriques d'anticiper chacun des attributs perceptifs.*

Pour comprendre ce qu'il se passe quand les actions sont incertaines, nous avons simulé l'apprentissage d'une transition environnementale avec cette forme d'incertitude en modifiant le taux de bruit des actions et le pas d'apprentissage β_{pep} des PEP. La transition étudiée est celle associée au PAI dans l'environnement *Woods100* et représentée par la figure 6.7 : PEPACS apprendrait à anticiper les changements perceptifs en allant à droite à partir de cette situation. Les taux de bruit des actions utilisés sont de 0%, 1%, 10% et 25% et les pas d'apprentissage β_{pep} de 0.001, 0.01 et 0.1. Cette transition est répétée 250000 fois pour chacune des combinaisons de ces paramètres. L'EMPA associée à cette transition est suivie tout au long de ces répétitions et moyennée sur une fenêtre de 1000 pas de temps afin de mettre en évidence l'évolution de l'erreur.
La figure 6.8 illustre l'évolution de l'EMPA associée à la transition étudiée pour chacune des combinaisons possibles des taux de bruit des actions et des pas d'apprentissage des PEP de PEPACS. L'EMPA de la transition oscille

sans converger quand les actions sont incertaines ; cette oscillation étant plus importante si le pas d'apprentissage β_{pep} est plus élevé, la valeur de l'EMPA étant aussi plus élevée si le taux de bruit est élevé.

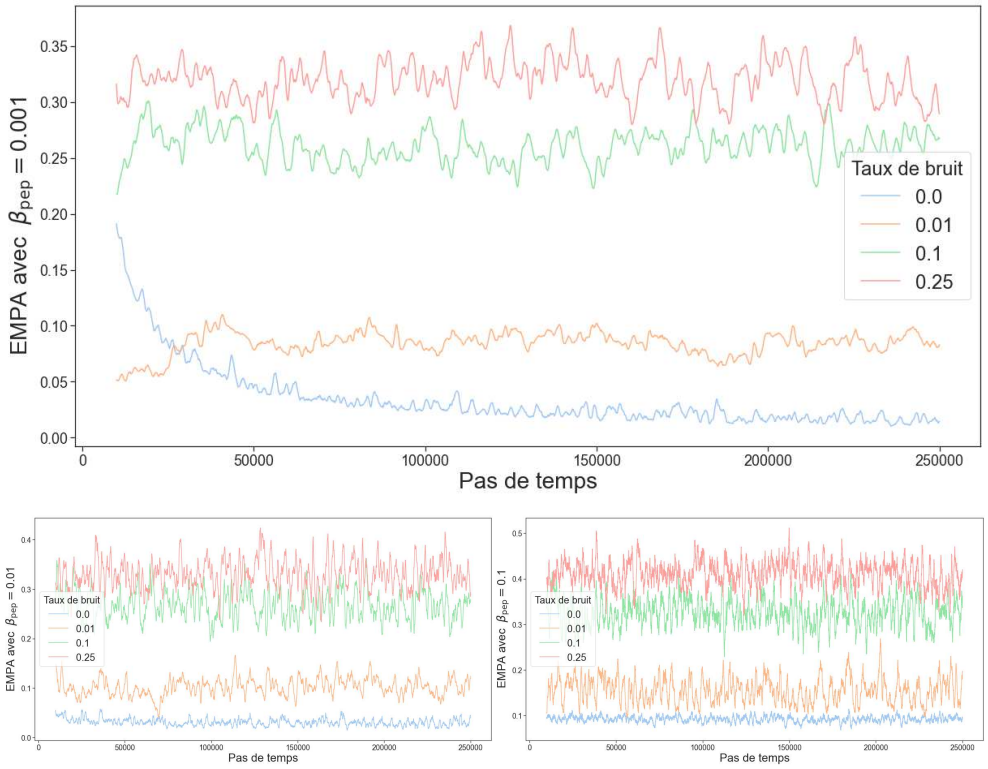


FIGURE 6.8 – Évolution de l'Erreur Moyenne sur la Prédiction de l'Anticipation de la transition environnementale liée au PAI dans le labyrinthe *Woods100* avec des PEP.

Comment les PEP influencent-elle la mise en place de politiques de décision dans des environnements incertains ?

La figure 6.9 donne les nombres moyens d'étapes nécessaires à PEPACS et ACS 2 pour atteindre la sortie des labyrinthes quand ils sont confrontés au problème d'aliasing perceptif mais pas aux actions incertaines. Pour 13 des 16 environnements, PEPACS met en place des politiques de décision plus efficaces qu'ACS 2. Les politiques de décision de PEPACS sont équivalentes à celles d'ACS 2 dans les trois environnements restants : *Maze7*, *MazeF4* et *MazeE2*.

La figure 6.10 donne les nombres moyens d'étapes nécessaires à PEPACS et ACS 2 pour atteindre la sortie de tous les labyrinthes quand les actions

sont incertaines. Pour 20 des 23 environnements, PEPACS met en place des politiques de décision plus efficaces qu'ACS 2. Les politiques de décision de PEPACS sont équivalentes à celles d'ACS 2 dans *MazeF2* et moins efficaces dans *MazeE2* et *Woods100*.

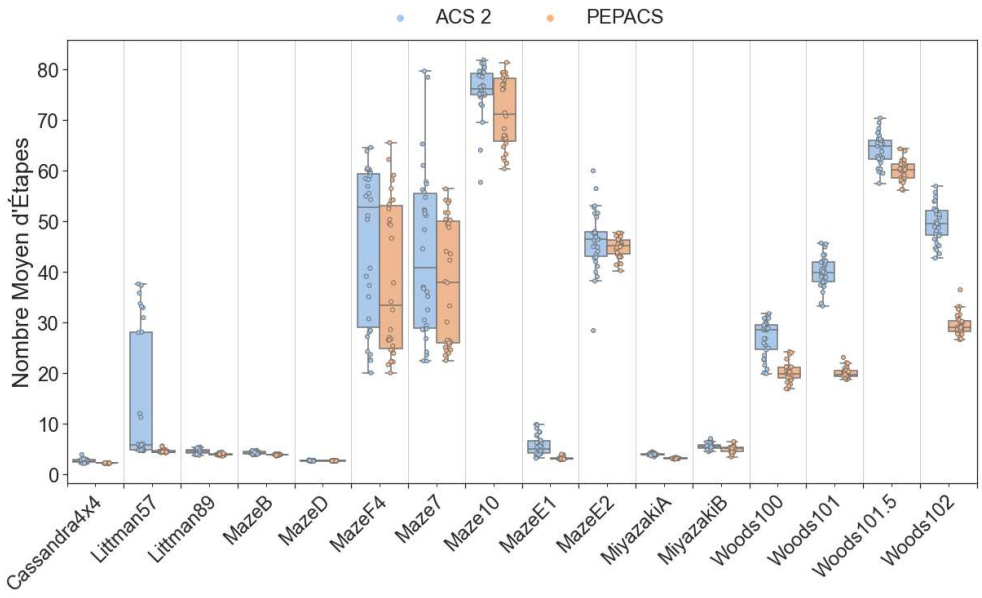


FIGURE 6.9 – Nombres moyens d'étapes de PEPACS et d'ACS 2 pour atteindre la sortie des labyrinthes avec PAI du banc de test sans les actions incertaines.

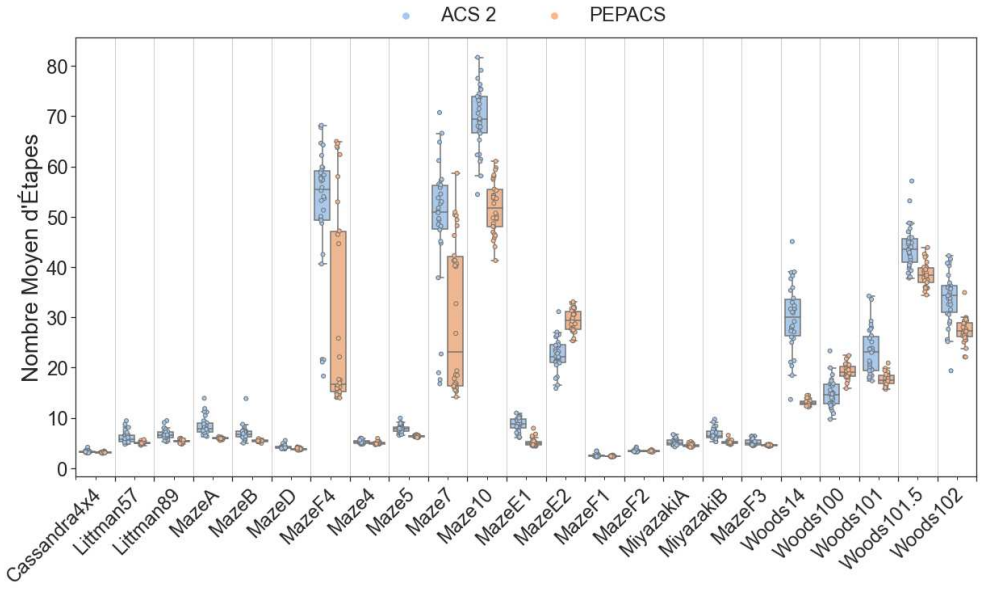


FIGURE 6.10 – Nombres moyens d'étapes de PEPACS et d'ACS 2 pour atteindre la sortie des labyrinthes avec les actions incertaines.

PEPACS est-il capable d'éviter de construire des classeurs à PEP surgénéraux ?

Dans les environnements avec les problèmes d'aliasing perceptif, la spécification introduite dans la copie de classeurs de PEPACS lui permet de fortement limiter la construction de classeurs généraux (*cf.* figure 6.11) : ce mécanisme réduit effectivement leur construction dans 15 des 16 environnements, sachant que dans le dernier aucun classeur surgénéral n'est construit. Grâce à cette spécification, PEPACS ne développe dans le pire des cas que 2 classeurs surgénéraux dans *Maze10* sur une population moyenne de 82 classeurs (soit un ratio de 0.03%) et il développe en moyenne 0.73 classeur surgénéral dans cet environnement (soit un ratio de 0.01%).

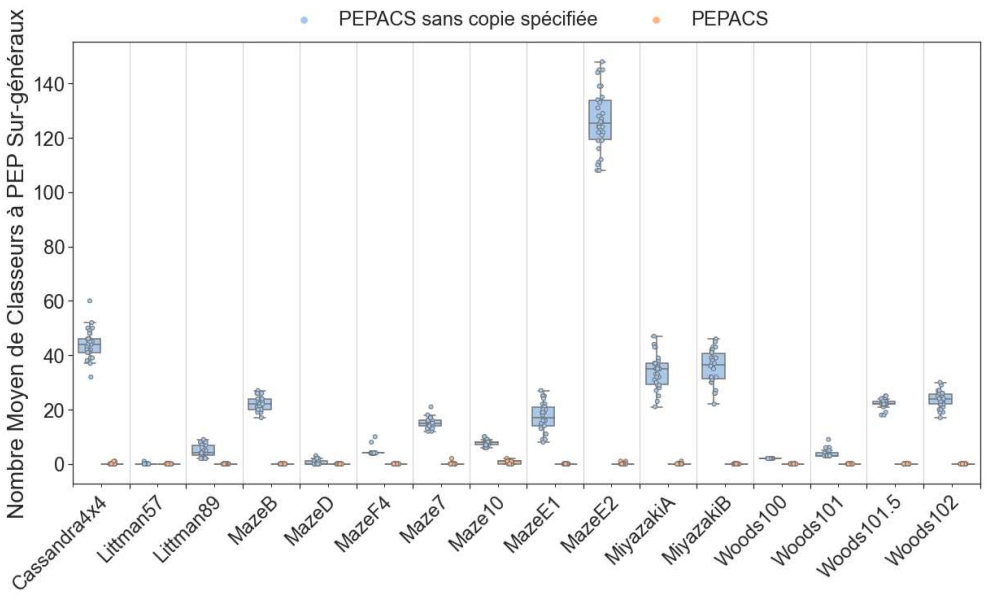


FIGURE 6.11 – Nombres moyens de classeurs à PEP surgénéraux de PEPACS dans les environnements avec PAI et sans les actions incertaines, avec et sans la spécification introduite dans la copie de classeurs.

La présence de classeurs surgénéraux dans PEPACS a différents impacts sur la construction d'une représentation environnementale ou la mise en place d'une politique de décision. Leur présence n'empêche pas de construire une représentation complète (il n'y a pas de différences statistiques), mais peut induire une légère instabilité des ratios de connaissance qui n'est pas présente lorsque ces classeurs sont absents : PEPACS sans cette contrainte n'a pas été capable d'atteindre un ratio maximal de connaissance à 6 reprises dans 4 environnements différents, au contraire de PEPACS avec la copie spé-

cifiée. Globalement, les représentations construites par PEPACS sans la copie spécifiée sont plus petites et plus générales : les tailles des populations sont plus petites dans 11 des 16 labyrinthes avec PAI et plus générales dans 14 d'entre eux. Les figures illustrant ces trois résultats sont données dans l'annexe C.3. En revanche, les représentations de PEPACS avec la copie spécifiée sont plus cohérentes avec leurs environnements, puisque les EMPA sont plus faibles dans 15 des 16 labyrinthes (*cf.* figure 6.12).

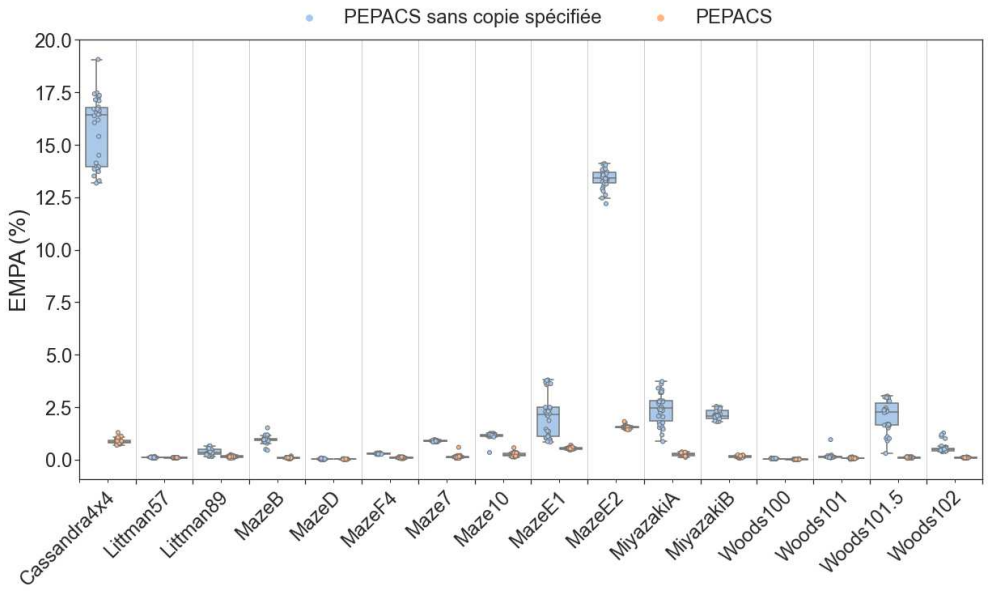


FIGURE 6.12 – Erreurs Moyennes sur la Prédiction de l'Anticipation de PEPACS dans les environnements avec PAI et sans les actions incertaines, avec et sans la spécification introduite dans la copie de classeurs.

Enfin, lorsque PEPACS ne parvient pas à prévenir la construction de classeurs surgénéraux, les politiques de décision qu'il met en place sont moins efficaces pour résoudre la tâche d'apprentissage dans la moitié des environnements du banc de test ou équivalentes dans l'autre moitié (cf. figure 6.13). Elles sont en particulier moins efficaces si les environnements confronte PEPACS au PAI de type II ou de type III (6 des 8 labyrinthes).

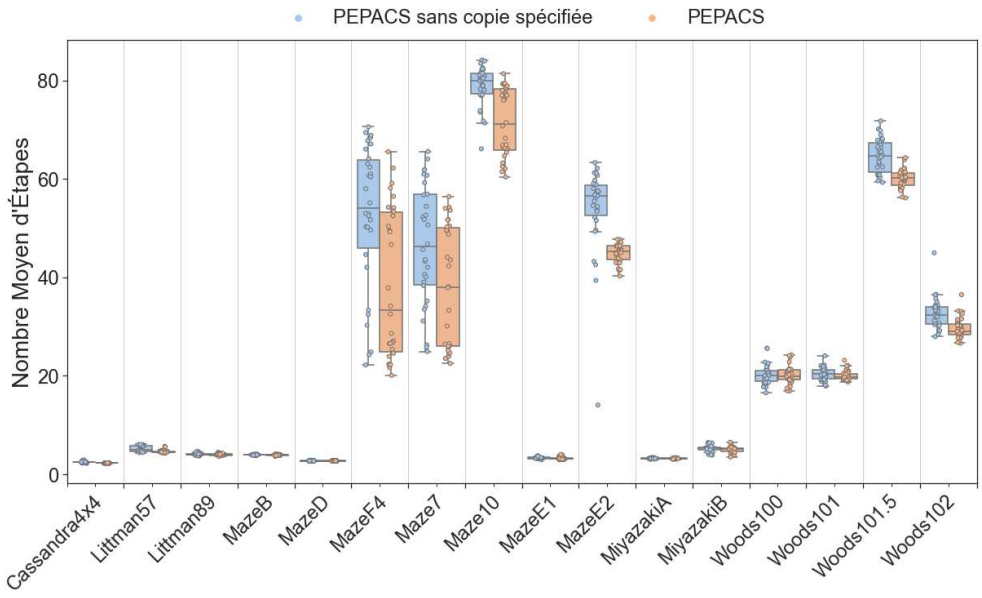


FIGURE 6.13 – Nombres moyens d'étapes de PEPACS pour atteindre la sortie des labyrinthes avec PAI du banc de test sans les actions incertaines, avec et sans la spécification introduite dans la copie de classeurs.

6.2.3 Comparaison avec d'autres systèmes

Deux comparaisons sont envisageables : PEPACS avec le seul ACS qui utilise les PEP (Butz et al., 2000) et que nous notons ACS-PEP ; PEPACS avec BACS et ACS 2, afin de voir l'influence des PEP dans la construction de politiques décisionnelles par rapport aux séquences comportementales.

Il est possible de vérifier que ces deux systèmes sont capables de se doter d'une représentation environnementale complète sous l'influence des actions incertaines dans *Maze4*. Cependant, il n'est pas possible de pousser plus loin la comparaison : il n'y a pas de données chiffrées permettant de comparer les tailles ou spécificités des représentations environnementales ou les politiques de décision développées ; les auteurs n'ont pas évalué si les probabilités calculées par ACS-PEP sont cohérentes avec leur environnement.

Une comparaison complète entre PEPACS et BACS est possible grâce à notre protocole d'évaluation expérimental. ACS 2 figure sur les illustrations suivantes comme système témoin.

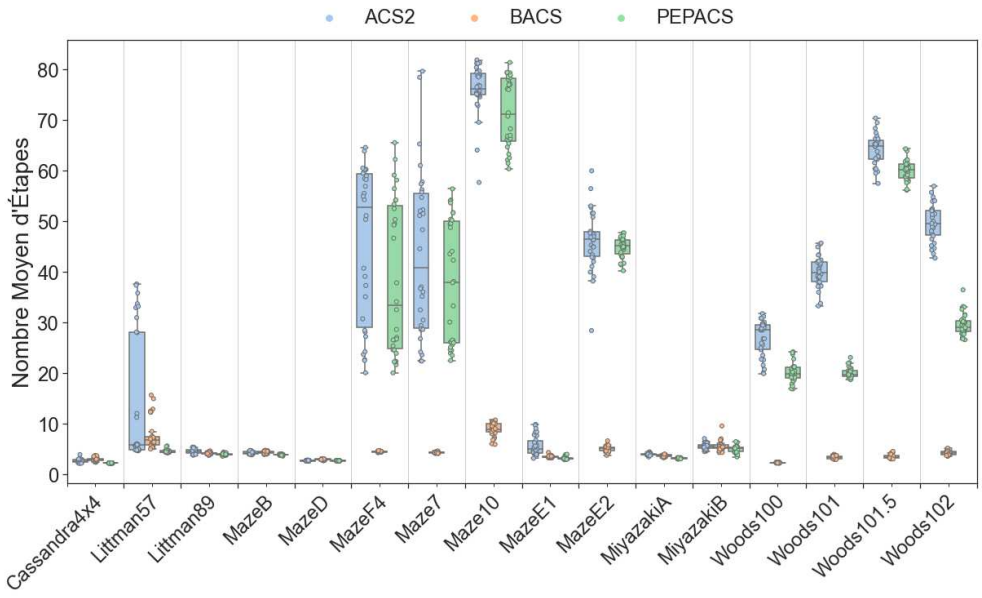


FIGURE 6.14 – Nombres moyens d'étapes de PEPACS, BACS et ACS 2 pour atteindre la sortie des labyrinthes avec PAI sans les actions incertaines.

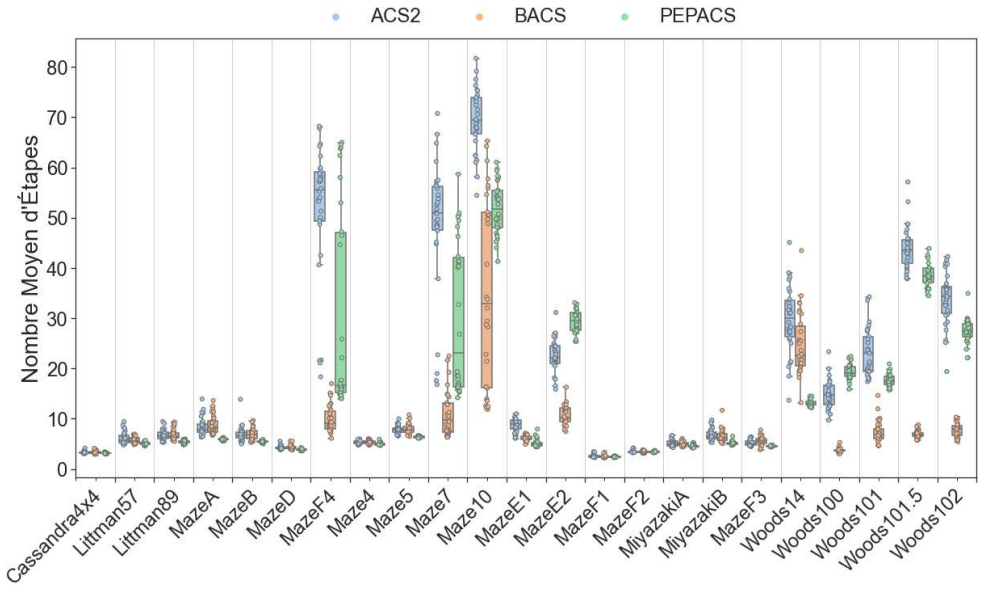


FIGURE 6.15 – Nombres moyens d'étapes de PEPACS, BACS et ACS 2 pour atteindre la sortie des labyrinthes avec les actions incertaines.

Les figures 6.14 et 6.15 illustrent respectivement les nombres moyens d'étapes nécessaires pour atteindre les sorties des labyrinthes avec PAI sans les actions incertaines et celles de tous les labyrinthes avec les actions incertaines. Sans les actions incertaines, BACS est l'ALCS le plus rapide à atteindre la sortie dans 8 des 16 environnements avec le PAI (*Maze7*, *Maze10*, *MazeE2*, *MazeF4*,

Woods100, *Woods101*, *Woods101.5* et *Woods102*), sinon PEPACS est le plus rapide. Avec les actions incertaines, BACS est à nouveau le plus rapide dans les 8 mêmes labyrinthes précédents, PEPACS étant le plus rapide dans 13 environnements ou aussi rapide qu'ACS 2 dans les deux labyrinthes restants. On peut donc constater que BACS est globalement plus efficace que les autres ALCS s'il est confronté au PAI de type II ou de type III, alors que PEPACS est plus efficace que les autres ALCS s'il est confronté au pseudo PAI ou à celui de type I, indépendamment des actions incertaines.

L'annexe C.4 donne les figures relatives à la construction des représentations environnementales de PEPACS, BACS et ACS 2. Ces figures confortent les résultats déjà présentés dans la section précédente où :

- les PEP permettent à PEPACS de se doter d'une représentation complète de son environnement contrairement à BACS et ACS 2 ;
- les EMPA de PEPACS sont les plus faibles si les actions ne sont pas incertaines ;
- les EMPA de BACS et ACS 2 sont globalement les plus faibles si les actions sont incertaines (21 des 23 environnements) ;
- les représentations les plus petites sont construites par PEPACS, celles de BACS étant toujours les plus grandes ;
- les classeurs fiables les plus spécifiques sont ceux de PEPACS si les actions sont incertaines, sinon il s'agit de ceux de BACS.

6.3 Discussion

6.3.1 Des représentations complètes mais des anticipations incohérentes

PEPACS est capable de construire des représentations complètes des environnements incertains où chaque transition environnementale est au moins décrite par un classeur fiable. Cependant, deux problèmes sont soulevés par l'utilisation des prédictions améliorées par les probabilités en lien avec l'explicabilité des classeurs :

1. elles ne permettent pas de savoir clairement quelles ont été les situa-

tions environnementales réellement perçues ou anticipées par le système ;

2. elles introduisent une incohérence dans la lecture des *Anticipations* du classeur.

Un attribut perceptif d'une *Anticipation* d'un classeur peut être constitué d'une PEP : c'est-à-dire qu'il est composé d'un tableau associatif où chaque clé désigne une valeur possible de l'attribut en question et la valeur relative à la clé la probabilité d'occurrence associée. Or, les *Anticipations* des classeurs sont composées de plusieurs attributs perceptifs et donc potentiellement de plusieurs PEP. Si l'*Anticipation* d'un classeur est composée de plusieurs PEP, plusieurs situations environnementales sont anticipables et dénombrables en combinant toutes les valeurs prises par chacun des attributs perceptifs. La figure 6.16 propose un exemple où toutes les situations anticipées possibles sont dénombrées pour un classeur à PEP. Le classeur donné dans cet exemple peut avoir été construit uniquement à partir de 2 des 4 situations anticipées : par exemple, la numéro 1 et la numéro 3 ou sinon, la numéro 2 et la numéro 4. Ce classeur anticiperait deux autres situations décrites par des PEP qui pourraient ne pas exister dans l'environnement. Il peut alors devenir délicat de savoir quelles situations environnementales ont été effectivement anticipées par le système et par extension, d'expliquer son comportement. Par exemple, un classeur qui contiendrait un nombre important de PEP dans son *Anticipation* est amené à anticiper un nombre exponentiel de situations environnementales : savoir comment un tel classeur a été créé est difficile à expliquer sans savoir quelles situations ont été réellement anticipées. Un raffinement des PEP permettant d'être plus précis sur les situations anticipées quand le système est confronté à différentes formes d'incertitude résoudrait cette limite.

À l'origine, les *Anticipations* des classeurs d'ACS 2 décrivent des changements perceptifs. PEPACS étant conçu à partir d'ACS 2, nous pouvons nous attendre à ce que les *Anticipations* des classeurs de PEPACS ne décrivent que ces changements. Autrement dit, si un attribut perceptif est spécifié dans une *Anticipation*, c'est que sa valeur est nécessairement différente de la valeur de l'attribut perceptif associé dans la *Condition* d'un classeur, exception faite du symbole générique $\#$ dans la *Condition*. Or, les PEP de PEPACS modifient la lecture des classeurs puisqu'il est possible qu'une même valeur soit spéci-

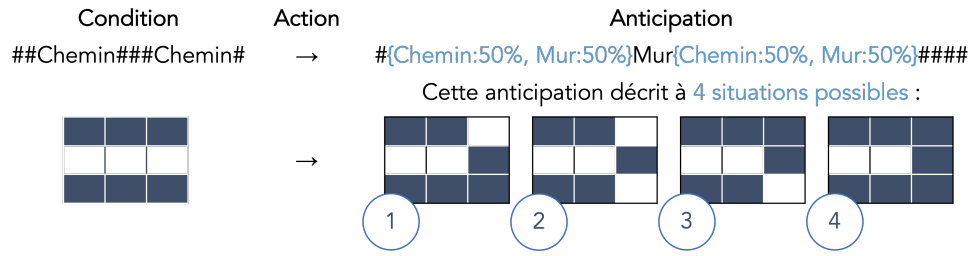


FIGURE 6.16 – Dénombrement des situations anticipées à partir d'un classeur comprenant deux PEP.
Les deux PEP portent sur les attributs perceptifs observés en haut à droite et en base à droite. Elles peuvent être combinées pour déterminer quelles sont les situations anticipées par ce classeur : un chemin avec un mur (situation 1), un chemin avec un chemin (situation 2), un mur avec un chemin (situation 3) et un mur avec un mur (situation 4).

fiée dans un attribut d'une *Condition* d'un classeur et dans une PEP en lien avec cet attribut dans l'*Anticipation* de ce classeur. Deux valeurs identiques seraient alors spécifiées sans qu'un changement ne soit décrit. Cette incohérence pourrait être résolue dans une itération du mécanisme des PEP si le symbole générique # des *Anticipations* indiquant l'absence d'un changement par rapport à un attribut perceptif conditionnel était employé.

Enfin, nous pouvons remarquer que le biais d'erreur inclus dans les politiques de sélection d'action de PEPACS joue un rôle dans la construction des représentations environnementales. Face à différentes formes d'incertitude où le système a des difficultés à anticiper l'état suivant, la qualité des classeurs ne peut pas augmenter de façon stable. Les qualités moyennes Q des actions où le système à de telles difficultés seront par conséquent plus faibles que celles des autres actions. PEPACS peut alors exploiter ce biais pour favoriser la construction de classeurs à PEP qui pourront devenir fiables dans les situations incertaines.

6.3.2 Des probabilités inadaptées à l'environnement

PEPACS est capable de déterminer quelles sont les probabilités d'anticiper les changements perceptifs quand il ne subit pas l'incertitude liée à la réalisation des actions (figure 6.5). En revanche, s'il subit cette incertitude, les probabilités calculées au sein des PEP ne sont plus adaptées à l'environnement (figure 6.6).

Les probabilités des PEP reflètent davantage les dernières observations des situations anticipées par un classeur que l'ensemble des situations anticipées par celui-ci depuis sa création. Cette sensibilité aux dernières observations est accrue selon le pas d'apprentissage β_{pep}. La simulation de l'apprentissage d'une transition de *Woods100* (figure 6.8) permet de confirmer cette analyse puisqu'une oscillation des Erreurs Moyennes sur la Prédiction de l'Anticipation de PEPACS est observée pour chaque tracé, en particulier ceux où les actions sont bruitées. Les actions incertaines accentuent aussi cette sensibilité aux dernières observations puisqu'il y a plus de chances d'observer des situations différentes si le taux de bruit des actions est élevé, encore plus si les actions incertaines sont couplées avec le problème d'aliasing perceptif. Une nouvelle mécanique pour calculer les probabilités des PEP est donc nécessaire pour que ces probabilités soient cohérentes avec l'environnement où le système réalise son apprentissage.

6.3.3 Effets de la copie spécifiée de PEPACS

Le mécanisme de copie spécifiée des *Anticipations* des classeurs à PEP permet effectivement de limiter l'apparition de classeurs surgénéraux (figure 6.11). Bien que la présence des classeurs surgénéraux dans les populations de PEPACS induit aussi des représentations environnementales plus petites et plus générales, de tels classeurs ne sont pas souhaités puisqu'ils sont utilisés dans des situations où les PEP ne sont pas nécessaires pour décrire les changements perceptifs anticipés, en plus de bruiter le calcul des probabilités des PEP (figure 6.12).

La présence de ces classeurs pose aussi un autre problème qui concerne l'élaboration des politiques de décision. Une représentation environnementale utilisant de tels classeurs a plus de chance d'être trop générale par rapport à une politique de décision. Les classeurs surgénéraux engloberaient alors des situations pour lesquelles les récompenses obtenues seraient complètement différentes, ce qui bruiterait la mise en place des politiques de décision et pourrait réduire leur efficacité. Cet effet est observé sur la figure 6.13 où la contrainte mise en place sur la copie permet à PEPACS d'être globalement le plus rapide à atteindre la sortie des labyrinthes.

La copie spécifiée introduite dans PEPACS répond donc à son objectif. Mais elle présente un inconvénient important dans la mise en place des classeurs à PEP de PEPACS, capable de complexifier le développement d'une représentation générale et compacte d'un environnement. Si un classeur à PEP devait être généralisé, le classeur résultant du processus de généralisation génétique ne contient aucune PEP. Si ce dernier vient à être utilisé dans des situations incertaines, *a minima* celles du classeur à PEP à son origine, les PEP doivent à nouveau être complètement reconstruites. Autrement dit, l'information contenue originellement dans un classeur à PEP à généraliser n'est pas pleinement exploitée. Il serait alors intéressant d'adopter une approche moins abrupte permettant de tirer parti de ces informations tout en limitant l'émergence de classeurs surgénéraux : les processus s'appuyant sur la copie de classeurs pourraient gagner en efficacité avec à la clé, par exemple, une meilleure généralisation des classeurs à PEP.

6.3.4 Influence des PEP sur les politiques de décision

PEPACS est capable d'atteindre plus rapidement la sortie des labyrinthes du banc de test qu'ACS 2 (figures 6.9 et 6.10). Les PEP impactent la mise en place des politiques décisionnelles de PEPACS, en particulier quand le problème d'aliasing perceptif peut survenir : PEPACS est capable de construire des classeurs fiables dans ces situations. Ce sont ces classeurs fiables qui permettent à PEPACS d'être plus rapide qu'ACS 2, et même BACS, quand il est confronté au pseudo PAI ou au PAI de type I. Pour ces deux types de PAI, l'action à réaliser dans les situations concernées est la même pour se rapprocher idéalement de l'objectif. PEPACS peut ainsi découvrir quelle est cette action et quels sont les classeurs les plus adéquats par le biais de son processus de rétribution, sans que l'adéquation des classeurs à la tâche d'apprentissage ne soit bruitée par la qualité des classeurs. Néanmoins, quand PEPACS est confronté au PAI de type II ou de type III pour lesquels des actions différentes sont nécessaires dans les situations concernées, il est moins performant que BACS.

6.4 Synthèse du chapitre

Les prédictions améliorées par les probabilités ont été intégrées à ACS 2 pour accroître son autonomie et son explicabilité dans des environnements incertains. Le système résultant, *Probability-Enhanced Predictions in Anticipatory Classifier System* (PEPACS), est capable de construire des classeurs dont les *Anticipations* sont composées de plusieurs situations. Les PEP permettent de décrire différentes situations en précisant dans un tableau associatif toutes les valeurs possiblement prises par un attribut perceptif anticipé et les probabilités d'occurrence associées.

L'intégration des PEP avec PEPACS a nécessité un mécanisme de détection de l'incertitude, un dispositif de construction des classeurs ayant des PEP, une adaptation des processus de découverte de nouvelles règles et une adaptation du processus de subsomption.

Les résultats montrent que les prédictions améliorées par les probabilités permettent effectivement à PEPACS de se doter de représentations environnementales complètes par rapport à ACS 2, tout en prévenant la construction de classeurs surgénéraux et inadaptés aux environnements. Les PEP permettent aussi de mettre en place des politiques décisionnelles plus efficaces, en particulier quand PEPACS est confronté au pseudo PAI et au PAI de type I, avec et sans les actions incertaines. Cependant, les probabilités calculées dans les PEP ne sont pas cohérentes avec les environnements explorés par le système, surtout si ceux-ci sont incertains.

Des pistes d'améliorations des PEP ont alors été dégagées :

- la représentation des PEP peut être raffinée afin de pouvoir déterminer quelles sont les situations environnementales effectivement anticipées par le système ;
- le mécanisme de calcul des probabilités peut être modifié afin que ces probabilités deviennent cohérentes avec les environnements explorés ;
- un nouveau mécanisme de prévention des classeurs surgénéraux peut être proposé afin de tirer parti des informations contenues dans les classeurs à PEP dans la mise en place des représentations environne-

mentales.

Une dernière voie d'amélioration serait de coupler les séquences comportementales avec les prédictions améliorées par les probabilités. Un tel couplage n'a jamais été proposé et devrait permettre à un système de classeur à anticipation de bénéficier des avantages de ces deux approches qui se révèlent être complémentaires : un système serait alors en mesure de se doter de représentations environnementales complètes et de résoudre efficacement ses tâches d'apprentissage, indépendamment de l'incertitude des environnements où il évolue. Le chapitre suivant propose un nouveau système de classeurs à anticipation où les mécanismes des PEP et des BSeq ont été respectivement raffinés, puis couplés.

CHAPITRE 7

BEACS : Coupler les séquences de BACS et les anticipations de PEPACS

Les séquences comportementales et les prédictions améliorées par les probabilités sont deux approches complémentaires permettant à ACS 2 d'évoluer dans des environnements incertains. Chacune de ces approches sont améliorées et couplées dans un nouveau système de classeurs à anticipation basé sur ACS 2 et nommé en l'occasion Behavioral Enhanced Anticipatory Classifier System (BEACS). Nous montrerons que les capacités d'apprentissage et l'explicabilité de BEACS sont donc plus importantes que celles de de BACS et PEPACS : BEACS est capable de se doter de représentations environnementales et de politiques décisionnelles plus adaptées aux environnements incertains que BACS et PEPACS.

Sommaire

Les travaux présentés dans ce chapitre ont fait l'objet de publications : (Orhand et al., 2021) et (Orhand et al., 2022).

7.1 Des anticipations basées sur l'expérience

Le mécanisme des PEP de PEPACS a d'abord été modifié en des Prédictions Améliorées par l'Expérience (*Enhanced Predictions through Experience (EPE)*). Les PEP de PEPACS et les EPE de BEACS ont tous les deux le même objectif qui de permettre aux classeurs d'anticiper plusieurs situations environnementales : ils diffèrent en revanche dans les représentations utilisées. Les EPE modifient la représentation des *Anticipations* des classeurs dans le but de savoir exactement quelles ont été les situations anticipées par les classeurs et de pouvoir déterminer avec fiabilité les probabilités d'anticiper ces situations. BEACS intègre dans un premier temps ce nouveau mécanisme des EPE dans un ACS 2 tout en prévoyant les interactions des EPE avec le mécanisme des séquences comportementales (BSeq).

7.1.1 Une nouvelle représentation

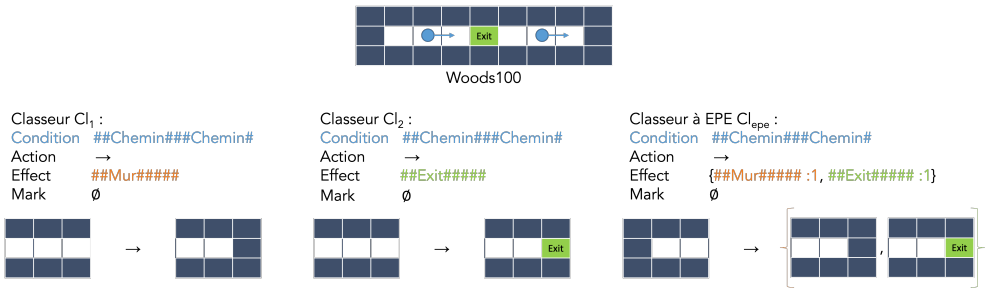


FIGURE 7.1 – Classeur à PEP et classeur à EPE dans *Woods100*.
Dans le labyrinthe Woods100*, il est possible d'atteindre la sortie ou un cul-de-sac en allant vers la droite à partir des situations bleues qui sont perçues de la même façon (PAI). PEPACS et BEACS sont capables de se doter d'un classeur dont l'*Anticipation *est améliorée pour décrire cette transition.*

Les *Anticipations* des classeurs de BEACS sont toutes constituées d'une EPE. Une EPE consiste en un tableau associatif dont les clés sont des situations anticipées décrites par plusieurs attributs perceptifs et dont les valeurs sont le nombre d'occurrences où ces situations ont été correctement anticipées. La figure 7.1 illustre deux classeurs de PEPACS et de BEACS qui décrivent les deux situations atteignables depuis la situation incertaine bleue avec une *Anticipation* améliorée (*i.e.* la sortie ou un cul-de-sac). Le classeur de BEACS décrit complètement les deux situations atteignables à partir de la situation

incertaine, là où le classeur de PEPACS n'utilise une PEP que sur l'attribut perceptif changeant. Par conséquent, une EPE fournit des informations plus détaillées qu'une *Anticipation* contenant plusieurs PEP, puisque chaque situation anticipée est explicitement décrite et comptabilisée. Nous appelons classeurs à EPE, les classeurs dont l'EPE est composée de plusieurs situations anticipées : un classeur dont l'EPE ne contiendrait qu'une seule situation anticipée se comporte exactement comme un classeur régulier d'ACS 2.

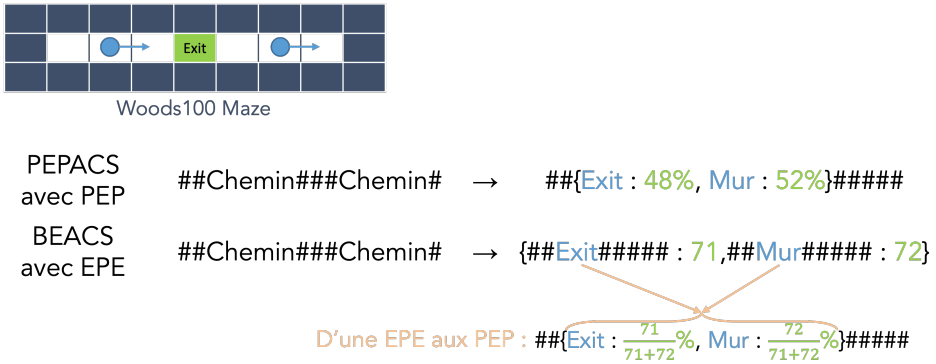


FIGURE 7.2 – Tranformation d'un classeur à EPE en un classeur à PEP dans *Woods100*.
Un seul changement perceptif est décrit par le troisième attribut perceptif : la sortie et le mur des deux situations anticipées de l'EPE sont combinées en une PEP où les probabilités d'occurrence sont calculées en normalisant les compteurs associés.

De plus, les EPE ne nécessitent pas la mise en place d'un pas d'apprentissage dédié contrairement aux PEP : plus BEACS peut interagir avec un environnement, plus ce dernier sera en mesure de construire des classeurs à EPE dont les probabilités seront cohérentes avec l'environnement. Il est aussi possible de transformer une *Anticipation* à EPE en une *Anticipation* à PEP, quand la transformation inverse ne l'est pas puisque des PEP peuvent décrire des situations qui ne seraient jamais anticipées. Une *Anticipation* à EPE est transformée en une *Anticipation* avec des PEP si (*cf.* figure 7.2) :

- pour chacun des attributs perceptifs, tous les symboles sont agglomérés dans une PEP ;
- puis, les compteurs sont normalisés pour calculer les probabilités d'occurrence de chacun des attributs des nouvelles PEP.

La mesure de l'Erreur Moyenne sur la Prédiction de l'Anticipation (EMPA) est ainsi utilisable pour comparer les probabilités des *Anticipations* des classeurs de BEACS et de PEPACS.

7.1.2 Construction des classeurs à EPE

Le mécanisme de détection d'incertitude utilisé dans BEACS est le même que celui de PEPACS et de BACS (chapitre 5 section 5.1.2). Il déclenche la construction de classeurs qui sont capables de décrire au travers de leur EPE toutes les situations à anticiper. Les classeurs à EPE ne sont pas directement construits quand la détection est fructueuse. Comme PEPACS, chaque classeur d'un ensemble d'action peut être marqué par un attribut *enhanced effect ee* à *True* lorsque la détection de l'incertitude est positive, puis tous les classeurs d'un ensemble d'action dont l'attribut *ee* est *True* sont utilisés pour générer des classeurs à EPE. La construction des classeurs à EPE intervient ainsi à la toute fin de l'ALP.

Les classeurs à EPE de BEACS sont construits de façon plus fine que les classeurs à PEP de PEPACS. Tous les classeurs de BEACS intègrent un nouvel attribut *uM* qui est une *Marque incertaine* : elle est utilisée pour guider l'évolution des classeurs à EPE pour l'unique situation incertaine décrite dans cette marque. Les marques incertaines *uM* sont vides pour tous les classeurs, sauf pour les classeurs à EPE où cette marque a pour valeur la perception associée à la situation incertaine qui a déclenché leur création. BEACS essaie de générer un seul classeur à EPE pour chaque classeur d'un ensemble d'action dont l'attribut *ee* est *True*. Deux classeurs cl_1 et cl_2 sont nécessaires à la construction d'un classeur à EPE, où :

- l'*Anticipation* de cl_1 n'est pas incluse dans celle de cl_2 et inversement, ce qui permet d'éviter la construction de classeurs superflus qui décriraient des changements environnementaux déjà décrits par cl_1 ou cl_2. Une *Anticipation* d'un classeur cl_1 n'est pas incluse dans celle de cl_2 s'il existe au moins une situation anticipée qui n'est pas décrite dans l'*Anticipation* de cl_2 ;
- les *Marques* de cl_1 et cl_2 sont identiques, ce qui permet d'éviter de s'appuyer sur des classeurs qui auraient été utilisés dans des situations différentes ;
- les *Marques incertaines* de cl_1 et cl_2 sont vides ou correspondent à la situation qui a déclenché la construction des classeurs à EPE qui décrivent plusieurs situations anticipées. Les marques incertaines sont utilisées pour cadrer la construction de classeurs à EPE de telle sorte

qu'un nouveau classeur à EPE corresponde à la même situation incertaine que celle des classeurs dont il est issu.

Un classeur à EPE cl_{epe} est créé à partir d'un premier classeur cl_1 et d'un classeur cl_2 remplissent les conditions précédentes, comme illustré sur la figure 7.3, où tous ses attributs sont mis à leurs valeurs par défaut sauf :

- la *Condition* de cl_{epe} est spécifiée pour tous les attributs perceptifs spécifiés des *Conditions* de cl_1 et de cl_2 ;
- le *Conséquent* de cl_{epe} est celui de l'ensemble d'action ;
- l'*Anticipation* de cl_{epe} est construite à partir de celle de cl_1 en y agglomérant l'*Anticipation* de cl_2. Si une situation anticipée de cl_2 est décrite dans cl_1, les nombres d'occurrences de cette situation sont sommés dans cl_{epe}. Sinon, cette situation anticipée de cl_2 est ajoutée avec son nombre d'occurrences dans cl_{epe} ;
- la récompense de cl_{epe} est la récompense moyenne de cl_1 et de cl_2 ;
- la qualité de cl_{epe} est fixée au maximum entre la qualité moyenne de cl_1 et de cl_2 et la valeur initiale par défaut (0.5) ;
- la *marque incertaine* de cl_{epe} contient la perception p_{t-1} de la situation incertaine, c'est-à-dire celle avant la réalisation d'une action par le système et qui a déclenché la construction de cl_{epe}.

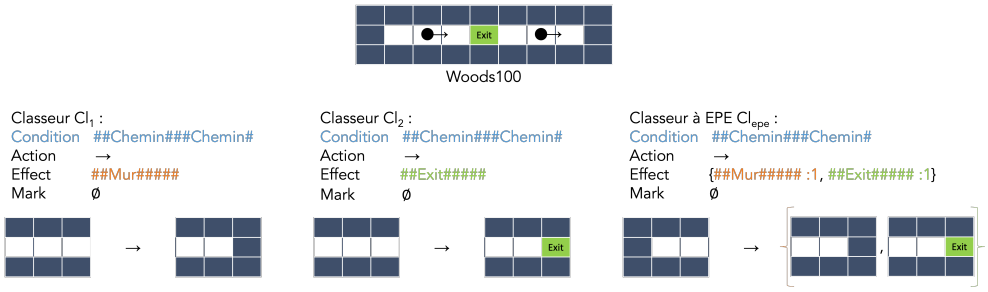


FIGURE 7.3 – Construction d'un classeur à EPE dans *Woods100*.
Dans le labyrinthe Woods100*, il est possible d'atteindre la sortie ou un cul-de-sac à partir de la situation noire en allant vers la droite. Un classeur amélioré par une EPE peut être construit par BEACS pour apprendre à anticiper ces deux situations environnementales.*

Chaque nouveau classeur cl_{epe} n'est pas directement ajouté à la population de classeurs de BEACS : il est inséré à celle-ci par le biais de la fonction d'ajout de classeurs à partir de l'ALP décrite par l'algorithme 5.1.1 (chapitre 5 section 5.1.4). Ainsi, un nouveau classeur qui existerait déjà dans la population ou qui serait subsumé par un autre classeur de la population est abandonné.

Les EPE de BEACS, contrairement aux PEP de PEPACS, permettent aussi de plus respecter la sémantique du symbole # qui correspond :

- à toutes les valeurs possibles d'un attribut perceptif dans une *Condition*, sauf celles spécifiés dans l'attribut associé d'une *Anticipation* ;
- à un absence de changement perceptif dans une *Anticipation*.

Le mécanisme de généralisation génétique peut soulever une ambiguïté sur les valeurs prises par les attributs perceptifs conditionnels valant # et dont l'attribut anticipatif associé est spécifié : si toutes les valeurs conditionnelles possibles peuvent se transformer en l'attribut anticipatif, indiquant donc un changement perceptif, alors l'attribut conditionnel peut en même temps prendre la valeur de l'attribut anticipatif auquel cas il n'y aurait pas de changement perceptif. C'est pourquoi il est sous-entendu qu'un attribut conditionnel de valeur # correspond à toutes les valeurs pouvant être prises par celui-ci, excepté les valeurs qui ont été spécifiées dans les attributs associés d'une *Anticipation*. Cette ambiguïté est renforcée quand, au travers des PEP ou des EPE, un attribut anticipatif peut désigner plusieurs changements perceptifs et en même temps une absence de changement. BEACS limite cette ambiguïté dans la lecture des classeurs en généralisant des attributs conditionnels seulement si les attributs anticipatifs associés ne désignent pas en même temps une absence de changement et un changement.

Pour ce faire, un nouvel attribut, la trace de mutation μT (*mutation trace*), est ajouté à tous les classeurs afin de garder une trace des attributs perceptifs dont l'*Anticipation* indique une absence de changement et un changement. μT est une liste de booléens où chaque booléen correspond à un attribut perceptif et indique par *False* si l'anticipation de cet attribut contient # avec d'autres valeurs de cet attribut. μT_{cl} est ainsi initialisé à *True* pour chaque attribut et est mis à jour selon l'algorithme 7.1.1 à la création d'un classeur à EPE ou lorsqu'un classeur à EPE apprend à anticiper une nouvelle situation.

μT_{cl} permet aussi de réduire la pression de généralisation des classeurs à EPE, limitant leur usage aux situations incertaines pour lesquelles ils ont été créés et de fait, l'apparition de classeurs surgénéraux dont la présence n'est pas souhaitée. L'opérateur de mutation est donc modifié de façon à tenir compte de cette trace μT_{cl} et aussi, des classeurs comportementaux. Son pseudo-code complet est alors fourni par l'algorithme 7.3.3 dans la section 7.3.3.

Algorithme 7.1.1 Mise à jour de la trace des mutations des classeurs de BEACS

```
1 : function UpdateMutationTrace(cl, L)
2 :     for i ← 0 to L − 1 do
3 :         symbols ← []
4 :         for each effect ∈ E_cl do
5 :             if effect[i] ∉ symbols then
6 :                 Add effect[i] in symbols
7 :             end if
8 :         end for
9 :         μT_cl[i] ← # ∉ symbols or length of symbols == 1
10 :    end for
```

L'algorithme 7.1.2 fait la synthèse du processus de génération des classeurs à EPE, de la sélection de classeurs générant les classeurs à EPE jusqu'à leur insertion dans la population. L'algorithme 7.1.2 s'appuie sur l'avant-dernière observation de l'environnement p_{t-1}, l'ensemble d'action $[A]_{t-1}$ constitué pour la perception p_{t-1}, la longueur L des séquences d'attributs perceptifs, le pas d'apprentissage β_{alp} de l'ALP, le seuil d'expérience θ_{exp}, le seuil de fiabilité θ_r et enfin, le pas de temps t.

Algorithme 7.1.2 Processus de génération des classeurs à EPE dans BEACS

```
1 : function ApplyEPEBuildingProcess([A], p_{t−1}, L, β_alp, θ_exp, θ_r, t)
2 :     candidates ← []
3 :     for each cl_A ∈ [A]_{t−1} do
4 :         if ee_{cl_A} == True then
5 :             Append cl_A to candidates
6 :         end if
7 :     end for
8 :     for i from 0 to (length of candidates −1) do
9 :         for j from i (to length of candidates −1) do
10 :            cl_1 ← candidates[i] ; cl_2 ← candidates[j]
11 :            if M_{cl_1} == M_{cl_2} and
                   E_{cl_1} ⊈ E_{cl_2} and E_{cl_2} ⊈ E_{cl_1} and
                   (uM_{cl_1} == ∅ or uM_{cl_1} == p_{t−1}) and
                   (uM_{cl_2} == ∅ or uM_{cl_2} == p_{t−1})      then
12 :                Build cl_epe from cl_1, cl_2, p_{t−1}, t
13 :                UpdateMutationTrace(cl_epe, L)
14 :                InsertFromALP(cl_epe, β_alp, θ_exp, θ_r, [A]_{t−1})
15 :            end if
16 :        end for
17 :    end for
```

7.1.3 Apprentissage par anticipation des classeurs à EPE

Le processus d'apprentissage par anticipation de BEACS est modifié pour tenir compte des classeurs à EPE. Contrairement à PEPACS, BEACS ne s'appuie pas sur une spécification des *Anticipations* des classeurs lorsqu'ils sont copiés pour éviter l'émergence de classeurs surgénéraux et permet aussi aux classeurs à EPE d'être spécifiés dans le *Correctable case*.

Comme PEPACS et ACS 2 dont il est dérivé, l'ALP de BEACS est composé de l'*Expected case*, du *Correctable case* et du *Not correctable case*. Un classeur à EPE parvient dans l'*Expected case* s'il arrive à anticiper correctement la situation environnementale suivant la réalisation d'une action : il suffit pour cela que l'une des situations anticipées d'une EPE décrive uniquement les changements environnementaux (autrement dit, les attributs perceptifs qui ont changé entre deux perceptions) et que ceux-ci correspondent à la situation à anticiper. S'il n'existe pas de telle clé dans un classeur à EPE, alors il entre directement dans le *Correctable case* de l'ALP afin de spécifier sa structure conditionnelle. Le *Not correctable case* de BEACS est alors identique à celui d'ACS 2, ce dernier n'étant pas atteint pas des classeurs à EPE.

Un classeur à EPE est spécifié dans le *Correctable case* à partir des perceptions p_{t-1} et p_t qui précédent et suivent la réalisation d'une action et de la *marque incertaine uM* du classeur. Cette marque ne peut être vide pour un classeur à EPE puisque sa valeur est celle de la situation incertaine pour laquelle il a été construit. La spécification proposée dans BEACS présente plusieurs avantages :

- BEACS ne spécifie plus l'*Anticipation* des classeurs avec la perception courante lors d'une copie de classeurs, ce qui permet de ne pas avoir à reconstruire toute l'*Anticipation* si celle-ci contient plusieurs situations anticipées ;
- l'emploi des *marques incertaines* permet de restreindre l'usage des classeurs à EPE à la situation incertaine qui y est décrite lorsque la *Condition* de ces classeurs est spécifiée ;
- BEACS utilise aussi les *marques incertaines* pour directement apprendre les situations qu'ils ont échoué à anticiper dans les EPE.

Algorithme 7.1.3 Spécification des classeurs de BEACS

```
1 : function SPECIFY(cl, p_{t-1}, p_t, L)
2 :     if length of EPE of cl == 1 then
3 :         for i ← 0 to L − 1 do
4 :             if p_{t-1}[i] ≠ p_t[i] and E_cl[i] ≠ # then
5 :                 E_cl[i] ← p_t[i]
6 :                 C_cl[i] ← p_{t-1}[i]
7 :             end if
8 :         end for
9 :     else
10 :        if uM_cl ≠ p_{t-1} then
11 :            for i ← 0 to L − 1 do
12 :                if uM_cl[i] ≠ p_{t-1}[i] then
13 :                    C_cl[i] ← uM_cl[i]
14 :                    μT_cl[i] ← False
15 :                end if
16 :            end for
17 :        else
18 :            effect_new ← {#}^L
19 :            for i ← 0 to L − 1 do
20 :                if p_{t-1}[i] ≠ p_t[i] then
21 :                    effect_new[i] ← p_t[i]
22 :                    C_cl[i] ← p_{t-1}[i]
23 :                end if
24 :            end for
25 :            Add effect_new to E_cl
26 :            UpdateMutationTrace(cl_epe, L)
27 :        end if
28 :    end if
```

L'algorithme 7.1.3 détaille la spécification d'un classeur *cl* de BEACS avec les perceptions p_{t-1} et p_t et le nombre d'attributs L des séquences perceptives. Les lignes 11 à 16 décrivent la spécification d'une *Condition* d'un classeur à EPE si sa *marque incertaine* est différente de p_{t-1}. La trace de mutation μT_{cl} est aussi directement mise à jour afin de ne pas généraliser un attribut conditionnel qui a été spécifié en raison de la surgénéralité du classeur. Les lignes 18 à 26 décrivent alors la spécification d'une *Condition* d'un classeur à EPE et l'ajout d'une nouvelle situation anticipée à l'EPE. La trace de mutation μT_{cl} est mise à jour à partir de l'EPE résultante à l'ajout de la nouvelle situation anticipée, car les classeurs entrant dans ce bloc ne sont pas nécessairement surgénéraux.

L'*Expected case* de BEACS est aussi modifié afin de mettre à jour les compteurs des EPE, d'intégrer la détection de l'incertitude et de tirer parti des *marques incertaines* pour spécifier les classeurs *via* la spécification des éléments inchangés. Seuls les compteurs des classeurs à EPE, c'est-à-dire ceux qui anticipent plusieurs situations, sont mis à jour s'ils parviennent à correctement anticiper la situation décrite par la perception p_t : le compteur associé à cette anticipation est alors incrémenté de 1. Une situation incertaine est détectable avec le mécanisme de détection d'incertitude (chapitre 5 section 5.1.2) ou avec les *marques incertaines* classeurs à EPE : si une marque incertaine est identique à la situation décrite par la perception p_{t-1} qui précède la réalisation d'une action, alors la situation décrite par p_{t-1} est considérée comme incertaine. Enfin, pour éviter qu'un classeur à EPE ne soit spécifié pour une situation différente de celle décrite dans sa *marque incertaine*, la spécification des éléments inchangés d'un classeur à EPE s'appuie sur les différences entre sa marque M et sa *marque incertaine* uM.

Algorithme 7.1.4 *Expected case* des classeurs de BEACS

function EXPECTEDCASE(cl, p_{t-1}, p_t, β_{alp}, $u_{\max}$, t)
  uncertaintyDetected ← *False*
  if length of EPE of $cl > 1$ **then**
    Increment by 1 the counter of the anticipated state related to p_t
    if $uM_{\text{cl}} == p_{t-1}$ **then**
      uncertaintyDetected ← *True*
    end if
  end if
  if uncertainty is detected through p_{t-1} and M_{cl} **then**
    ee_{cl} ← *True*
    uncertaintyDetected ← *True*
  end if
  if length of EPE of $cl > 1$ **then**
    diff ← differences between M_{cl} and uM_{cl}
  else
    diff ← differences between M_{cl} and p_{t-1}
  end if
  if *diff* == ∅ **then**
    $q_{\text{cl}} \leftarrow (1 - \beta_{\text{alp}})q_{\text{cl}} + \beta_{\text{alp}}$
    return *uncertaintyDetected*, ∅
  end if
  return *uncertaintyDetected*, SUC(cl, *diff*, $u_{\max}$, t)

L'*Expected case* de BEACS est décrit par l'algorithme 7.1.4 qui prend en argument un classeur cl, les perceptions p_{t-1} et p_t, le pas d'apprentissage β_{alp} de l'ALP, le paramètre $u_{\max}$ de la spécification des éléments inchangés [1] et le pas de temps t. L'*Expected case* renvoie un booléen indiquant si la situation décrite par p_{t-1} est considérée comme incertaine par BEACS et un nouveau classeur peut-être construit par la spécification des éléments inchangés.

7.1.4 Adapter le processus de subsomption pour les EPE

La subsomption nécessite d'ajuster la façon dont les classeurs de BEACS peuvent subsumer ou être subsumés par un autre classeur. Toutes les classeurs de BEACS sont constitués d'une EPE qui contient une ou plusieurs situations anticipées : la subsomption doit alors déterminer si une EPE est incluse dans une autre EPE. Ainsi, l'*Anticipation* d'un premier classeur est contenue dans l'*Anticipation* d'un autre classeur si cette dernière contient toutes les situations anticipées décrites par l'EPE du premier classeur. BEACS assouplit également le critère portant sur les *Marques* des classeurs en permettant qu'un classeur marqué puisse en subsumer un autre si leurs *Marques* sont identiques et les autres conditions à la subsomption respectées.

Les classeurs à EPE sont spécifiables à partir de leur *marque incertaine*, auquel cas ils peuvent être ajoutés à l'ensemble de la population de classeurs si leur *Condition* ne correspond plus à la perception précédent la réalisation d'une action. Il est alors nécessaire de comparer les *Conditions* et les *Conséquents* des classeurs afin de vérifier s'il n'existe pas un classeur de la population qui subsumerait un classeur nouvellement créé :

- le *Conséquent* d'un classeur existant doit être identique à celui d'un nouveau classeur pour pouvoir le subsumer ;
- la *Condition* d'un classeur existant ne doit pas avoir d'attributs spécifiés qui sont différents des attributs spécifiés associés dans la *Condition* d'un nouveau classeur pour pouvoir le subsumer.

L'algorithme 7.1.5 décrit le test permettant de savoir si un classeur cl_1 est subsumé par un classeur cl_2. Il prend ainsi en argument les classeurs cl_1 et cl_2 et les seuils d'expérience θ_{exp} et de fiabilité θ_{r}.

1. La fonction SUC est décrite dans l'algorithme 3.2.1 de la section 3.2.3 du chapitre 3.

Algorithme 7.1.5 Subsomption de classeurs dans BEACS

```
1 : function IsSubsumed(cl_1, cl_2, θ_exp, θ_r)
2 :     if number of # in C_cl2 ≥ number of # in C_cl1 then
3 :         if C_cl2 does not specify a perceptive
              attribute different from the related specified one of C_cl2 then
4 :             if A_cl2 == A_cl1 then
5 :                 if E_cl2 ⊇ E_cl1 then
6 :                     if M_cl2 == ∅ or M_cl2 == M_cl1 then
7 :                         if q_cl2 > θ_r then
8 :                             if exp_cl2 > θ_exp then
9 :                                 return True
10 :                            end if
11 :                        end if
12 :                    end if
13 :                end if
14 :            end if
15 :        end if
16 :    end if
17 :    return False
```

7.1.5 Anticipation de changements et EPE

Comme ACS 2, BEACS cherche à employer des actions qui ont des conséquences perceptibles sur son environnement. Ce biais d'apprentissage intervient lors de la sélection d'une action et lors de la rétropropagation des récompenses des classeurs. La mise en place des EPE dans les classeurs de BEACS redéfinit ce en quoi consiste un classeur qui anticipe des changements perceptifs.

L'approche adoptée dans BEACS n'est pas de considérer qu'une EPE contenant plusieurs situations anticipées va nécessairement prédire *tout le temps* des changements perceptifs, contrairement à PEPACS et aux PEP. Il suffit d'imaginer pour cela un classeur qui tenterait d'aller dans un mur au sein d'un labyrinthe avec des actions incertaines. Sa position serait la majorité du temps inchangée et il ne percevrait aucun changement, sauf si l'une des autres actions est réalisée et pour laquelle la position du système dans le labyrinthe change. Un tel classeur prédit plusieurs situations à anticiper, mais la plupart du temps, il anticipe que la réalisation de l'action décrite dans son *Conséquent* n'entraîne aucun changement.

BEACS s'appuie alors sur les compteurs associés aux situations anticipées des EPE des classeurs pour déterminer qu'elle est la situation qui est la plus anticipée. Une fois que cette situation est déterminée, elle est analysée pour savoir si elle décrit au moins un changement sur l'un des attributs perceptifs, auquel cas BEACS considère que ce classeur anticipe des changements perceptifs suite à la réalisation de l'action décrite par son *Conséquent*. Autrement, si la situation la plus anticipée d'un classeur est uniquement composée de symbole #, alors BEACS considère que ce classeur n'anticipe pas de changements perceptifs.

Les EPE permettent à BEACS de décrire plusieurs situations environnementales dans l'*Anticipation* de ses classeurs. Les classeurs à EPE sont construits dès qu'une forme d'incertitude implique que la réalisation d'une action peut conduire à différentes situations, comme c'est le cas avec les actions incertaines ou le problème d'aliasing perceptif. BEACS utilise les séquences comportementales pour gérer cette dernière forme d'incertitude : elles doivent lui permettre de traverser les situations liées au problème d'aliasing perceptif. Il est souhaité que leur usage soit restreint en ces situations particulières, qui doivent alors être différenciées parmi toutes les situations qui seraient incertaines. Pour réaliser le couplage des EPE avec des séquences comportementales, BEACS s'appuie sur un mécanisme de détection du PAI qui conditionnerait la création de classeurs à BSeq. Ce mécanisme doit ainsi permettre à BEACS de contrôler plus précisément le moment où les classeurs à BSeq sont construits et au final, l'évolution de la population de classeurs.

7.2 Différencier le problème d'aliasing perceptif des formes d'incertitude

Le problème d'aliasing perceptif se produit dans des environnements partiellement observables lorsqu'un système ne peut pas différencier des situations qui sont réellement distinctes. Pour déterminer si une situation est liée au PAI, BEACS se concentre sur les situations anticipées décrites dans les EPE des classeurs : les situations atteignables à partir d'une situation liée au PAI et différentes de celle liée au PAI seraient plus nombreuses que le

nombre d'actions qui conduisent à des situations distinctes. Autrement dit, BEACS s'appuie sur l'hypothèse que les transitions environnementales à partir d'une situation liée au PAI sont plus nombreuses que les actions associées à ces transitions, indépendamment des autres formes d'incertitude telles que les actions incertaines. La figure 7.4 illustre deux exemples de comparaison pouvant permettre d'identifier le problème d'aliasing perceptif dans le labyrinthe *Woods100*.

La différentiation du PAI des autres formes d'incertitude est réalisé en plusieurs étapes où pour une situation donnée : elle est planifiée afin de laisser BEACS se doter de classeurs fiables et expérimentés ; ces classeurs sont retrouvés au sein de la population de classeurs ; toutes les situations anticipées et le nombre d'actions conduisant à des situations distinctes sont calculés et comparés.

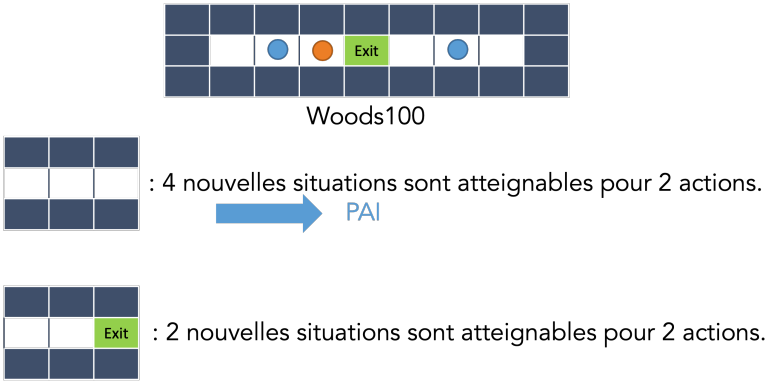


FIGURE 7.4 – Comparaison entre les situations atteignables distinctes de celle de départ et le nombre d'actions conduisant à des situations distinctes pour deux perceptions dans *Woods100*
Dans le labyrinthe Woods100, *il est possible d'atteindre deux culs-de-sac et deux situations proches de la sortie en allant vers la droite ou la gauche à partir des situations bleues liées au PAI. Seules la sortie du labyrinthe et une situation liée au PAI sont atteignables en allant à droite et à gauche pour une situation orange non liée au PAI.*

7.2.1 Planifier la détection du PAI

Alors que la détection de l'incertitude se fait à l'échelle d'un classeur, la détection du PAI se fait à l'échelle d'un *ensemble* de classeurs afin que BEACS puisse déterminer l'ensemble des situations anticipées à partir d'une situation donnée. Pour éviter des opérations inutiles, BEACS ne tente pas de

détecter le PAI dès que la détection de l'incertitude est fructueuse : la détection du PAI a lieu à la fin du processus d'apprentissage par anticipation. Comme BEACS a également besoin de temps pour adapter ses classeurs à son environnement et découvrir les transitions entre les situations, la détection du PAI est planifiée de façon similaire à la généralisation génétique.

La détection du PAI dépend d'un paramètre utilisateur θ_{pai} représentant un délai minimal entre deux détections et d'une marque temporelle t_{pai} ajoutée à chaque classeur indiquant la dernière utilisation du classeur dans la détection du PAI. Elle est appliquée sur un sous-ensemble $[M]_{\text{pai}}$ construit à partir de l'ensemble d'appariement $[M]$ correspondant à la perception p pour laquelle BEACS cherche à savoir s'il est confronté au PAI ou non : $[M]_{\text{pai}}$ contient les classeurs de $[M]$ qui ne sont pas des classeurs à BSeq, dont leur marque est vide ou correspond à la perception p et dont leur *marque incertaine* est vide ou correspond à la perception p.

La détection du PAI est réalisée si le délai moyen depuis la dernière application de la détection du PAI sur $[M]_{\text{pai}}$ est supérieur à la fréquence d'application θ_{pai}. Ce délai est calculé à partir d'un marqueur temporel t_{pai} ajouté à chaque classeur de la population et pondéré par la numérosité *num* des classeurs. t_{pai} est mis à jour avec le compteur de pas de temps t dès que la détection du PAI a été appliquée sur un classeur. L'équation 7.1 décrit le calcul de ce délai et sa comparaison à θ_{pai}, à partir des classeurs de l'ensemble $[M]_{\text{pai}}$, le compteur de pas de temps t et la fréquence θ_{pai}.

$$t - \frac{\sum_{cl \in [M]_{\text{pai}}} t_{\text{pai}_{\text{cl}}} . num_{\text{cl}}}{\sum_{cl \in [M]_{\text{pai}}} num_{\text{cl}}} > \theta_{\text{pai}} \tag{7.1}$$

7.2.2 Extraire les classeurs les plus expérimentés

Pour détecter le PAI, BEACS s'appuie sur l'ensemble $[M]_{\text{pai}}$ précédemment construit et cherche à extraire les classeurs les plus expérimentés pour chacune des actions : il s'agit des classeurs dont le produit de leur expérience par leur qualité élevée au cube est maximale. La puissance au cube est utilisée pour agrandir les écarts entre les qualités de classeurs, sans pour autant

que les qualités les plus élevées ne convergent aussi vers 0 puisqu'elles sont comprises entre 0 et 1 (auquel cas il n'y aurait plus d'écart entre les qualités). L'expérience est utilisée afin de s'appuyer sur des classeurs qui ont été éprouvés. Si les classeurs les plus expérimentés existent pour toutes les actions et sont aussi suffisamment expérimentés selon le seuil d'expérience θ_{exp} et fiables selon le seuil de fiabilité θ_{r}, alors l'ensemble des situations anticipées et le nombre d'actions conduisant à des situations distinctes sont calculés en utilisant les EPE de ces classeurs et la perception p pour laquelle BEACS cherche à savoir s'il est confronté au PAI ou non. L'algorithme 7.2.1 décrit l'extraction des classeurs les plus expérimentés de $[M]_{\text{pai}}$ en prenant en argument cet ensemble de classeurs, le nombre d'actions n_A réalisable par BEACS et les seuils d'expérience θ_{exp} et de fiabilité θ_{r}.

Algorithme 7.2.1 Extraction des classeurs les plus expérimentés pour la détection du PAI dans BEACS

```
1 : function IsEnoughInfoForPAIDetection([M]_pai, n_A, θ_r, θ_exp)
2 :     best_cls ← {∅}^{n_A}
3 :     for each cl ∈ [M]_pai do
4 :         if exp_cl > θ_exp and q_cl > θ_r then
5 :             if best_cls[A_cl] == ∅ then
6 :                 best_cls[A_cl] ← cl
7 :             else
8 :                 best ← best_cls[A_cl]
9 :                 if exp_cl * q_cl^3 > exp_best * q_best^3 then
10 :                    best_cls[A_cl] ← cl
11 :                end if
12 :            end if
13 :        end if
14 :    end for
15 :    for i ← 0 to n_A − 1 do
16 :        if best_cls[i] == ∅ then
17 :            return ∅
18 :        end if
19 :    end for
20 :    return best_cls
```

7.2.3 Détecter le PAI à l'aide des EPE

La détection du PAI calcule d'abord l'ensemble des situations anticipées et le nombre d'actions conduisant à des situations distinctes pour une perception

p donnée à partir de l'ensemble $best_{\text{cls}}$ des classeurs les plus expérimentés.

Pour retrouver toutes les situations anticipées à partir des EPE des classeurs de $best_{\text{cls}}$, l'opérateur *passthrough* [2] est utilisé en prenant en argument la perception p, une clé d'une EPE qui désigne un ensemble de changements perceptifs et la longueur L des séquences perceptives. Chaque situation anticipée calculée avec *passthrough* est gardée en mémoire dans une liste sans doublons. La longueur de cette liste, décrémentée de 1 si la situation décrite par p appartient à cette liste, permet alors de connaître le nombre de situations atteignables à partir d'une situation décrite par p et différentes de celle décrite par p. Cette longueur est comparée à terme avec le nombre d'actions conduisant à des situations distinctes.

Au début, toutes les actions conduisent à des situations distinctes entre elles et différentes de celle décrite par p. Ce nombre d'actions est ensuite mis à jour en déterminant d'abord pour chaque classeur de $best_{\text{cls}}$ quelle est la situation la plus anticipée à l'aide des compteurs des EPE. Une fois la situation la plus anticipée d'un classeur de $best_{\text{cls}}$ retrouvée, elle peut être ajoutée dans une liste permettant de déterminer ce nombre d'actions :

- si la situation la plus anticipée est celle décrite par p, le nombre d'actions recherché est décrémenté de 1 et elle est ajoutée à cette liste si elle n'y appartenait pas déjà ;
- si la situation la plus anticipée est déjà présente dans cette liste, le nombre d'actions recherché est décrémenté de 1 ;
- sinon, la situation la plus anticipée n'est pas présente dans celle liste et elle y est donc ajoutée.

Alors, si le nombre d'actions conduisant à des situations distinctes est plus petit que la longueur de la liste des situations atteintes à partir de p, la détection du PAI est fructueuse.

L'algorithme 7.2.2 décrit la détection du PAI et prend en argument : une perception p, la liste $best_{\text{cls}}$ des classeurs les plus expérimentés pour la perception p, le nombre n_A total d'actions réalisable par le système dans un environnement et la longueur L des séquences perceptives.

2. L'opérateur *passthrough* est présenté par l'algorithme 3.1.1 dans la section 3.1.3 du chapitre 3.

Algorithme 7.2.2 Détection du PAI dans BEACS

```
1 : function IsPAIDetected(best_cls, p, n_A, L)
2 :     nbrOfExpectedTransitions ← n_A
3 :     listOfAnticipatedStates ← []
4 :     listOfMostAnticipatedStates ← []
5 :     for each cl ∈ best_cls do
6 :         mostAnticipatedState ← ∅
7 :         occOfMostAnticipatedState ← 0
8 :         for each {anticipation, occ} ∈ E_cl do
9 :             completeAnticipatedState ← passthrough(p, anticipation, L)
                ▷ Complete if necessary the list of all anticipated states
10 :            if completeAnticipatedState ∉ listOfAnticipatedStates then
11 :                Add completeAnticipatedState in listOfAnticipatedStates
12 :            end if
                ▷ Find which state is the most anticipated one
13 :            if occOfMostAnticipatedState == 0 then
14 :                mostAnticipatedState ← completeAnticipatedState
15 :                occOfMostAnticipatedState ← occ
16 :            else if occ > occOfMostAnticipatedState then
17 :                mostAnticipatedState ← completeAnticipatedState
18 :                occOfMostAnticipatedState ← occ
19 :            end if
20 :        end for
            ▷ Complete the list of the most anticipated states or update the
    number of unique expected transitions different from p
21 :        if mostAnticipatedState == p then
22 :            nbrOfExpectedTransitions ← nbrOfExpectedTransitions −1
23 :            if p ∉ listOfMostAnticipatedStates then
24 :                Add p in listOfMostAnticipatedStates
25 :            end if
26 :        else if mostAnticipatedState ∈ listOfMostAnticipatedStates then
27 :            nbrOfExpectedTransitions ← nbrOfExpectedTransitions −1
28 :        else
29 :            Add mostAnticipatedState in listOfMostAnticipatedStates
30 :        end if
31 :    end for
        ▷ Compute the number of anticipated states
32 :    nbrOfAnticipatedStates ← length of listOfAnticipatedStates
33 :    if p ∈ listOfAnticipatedStates then
34 :        nbrOfAnticipatedStates ← nbrOfAnticipatedStates −1
35 :    end if
        ▷ Check for PAI
36 :    if nbrOfAnticipatedStates > nbrOfExpectedTransitions then
37 :        return True
38 :    end if
39 :    return False
```

7.2.4 Conditionner les classeurs comportementaux à la détection du PAI

L'algorithme 7.2.3 présente la boucle complète liée à la gestion du PAI dans BEACS, intégrant donc la planification de la détection du PAI, l'extraction des classeurs les plus expérimentés et la détection du PAI. Elle prend donc tous les arguments nécessaires à ces trois fonctions : la perception p_{t-1} précédent la réalisation d'une action et l'ensemble d'appariement M_{t-1} correspondant, le nombre d'actions n_A réalisable par BEACS, la longueur L des séquences perceptives, la fréquence d'application θ_{pai} de la détection du PAI, les seuils d'expérience θ_{exp} et de fiabilité θ_{r} et le pas de temps t.

BEACS garde en mémoire dans une liste *PAIMemory* l'ensemble des perceptions pour lesquelles le problème d'aliasing perceptif a été détecté. Cette liste est mise à jour à chaque détection en y ajoutant une nouvelle perception dont la détection est fructueuse ou en retirant de cette liste une perception qui n'est plus considérée comme liée au PAI, la détection étant négative. Cette liste *PAIMemory* est utilisée afin de renseigner un utilisateur sur les situations où BEACS considère qu'il est confronté au PAI et aussi, pour gérer les classeurs comportementaux qu'il est capable de construire. Les classeurs comportementaux de BEACS sont abordés dans la prochaine section.

BEACS marque les classeurs contenant une séquence comportementale avec la perception associée au PAI qui a déclenché leur construction dans un nouvel attribut, la marque PAI *paiM*. Si une situation précédemment détectée comme liée au PAI n'est plus liée au PAI, tous les classeurs comportementaux liés à cette situation sont retirés de la population de classeurs. Cela permet à BEACS de construire et de supprimer de manière adaptative des classeurs comportementaux au fur et à mesure que des perceptions liées au PAI sont détectées, évitant ainsi de garder des classeurs comportementaux non nécessaires. La boucle de gestion du PAI de BEACS prend alors des arguments supplémentaires : la liste *PAIMemory* des perceptions liées au PAI selon BEACS et la population de classeurs $[P]$, l'ensemble d'action $[A]_{t-1}$ construit pour p_{t-1} et l'ensemble d'appariement $[M]_t$ construit pour p_t qui suit la réalisation de l'action en p_{t-1} afin d'y supprimer des classeurs comportementaux

si nécessaire [3].

Algorithme 7.2.3 Boucle complète liée à la gestion du PAI dans BEACS

function PAILOOP($[A]_{t-1}$, $[M]_{t-1}$, $[M]_t$, $[P]$, p_{t-1}, n_A, L, *PAIMemory*, θ_r, θ_{exp}, θ_{pai}, t)
    $[M]_{\text{pai}} \leftarrow []$
    for each $cl \in [M]_{t-1}$ **do**
        if cl is not a behavioral classifier **then**
            if $M_{\text{cl}} == \emptyset$ **or** $M_{\text{cl}} == p_{t-1}$ **then**
                if $uM_{\text{cl}} == \emptyset$ **or** $uM_{\text{cl}} == p_{t-1}$ **then**
                    Add cl in $[M]_{\text{pai}}$
                end if
            end if
        end if
    end for
    if $t - \frac{\sum_{cl \in [M]_{\text{pai}}} t_{\text{pai}_{\text{cl}}}.num_{\text{cl}}}{\sum_{cl \in [M]_{\text{pai}}} num_{\text{cl}}} > \theta_{\text{pai}}$ **then**
        for each $cl \in [M]_{\text{pai}}$ **do**
            $t_{\text{pai}_{\text{cl}}} \leftarrow t$
        end for
        $best_{\text{cls}} \leftarrow$ *IsEnoughInfoForPAIDetection*($[M]_{\text{pai}}$, n_A, θ_r, θ_{exp})
        if $best_{\text{cls}} \neq \emptyset$ **then**
            if *IsPAIDetected*($best_{\text{cls}}$, p_{t-1}, n_A, L) == *True* **then**
                if $p_{t-1} \notin$ *PAIMemory* **then**
                    Add p_{t-1} in *PAIMemory*
                end if
            else
                if $p_{t-1} \in$ *PAIMemory* **then**
                    Remove p_{t-1} from *PAIMemory*
                    Remove all behavioral classifiers whose PAI mark *paiM* is p_{t-1} from $[A]_{t-1}$, $[M]_t$ and $[P]$
                end if
            end if
        end if
    end if

3. Les processus de découverte de nouvelles règles sont appliqués à l'instant t sur l'ensemble d'action construit à l'instant $t-1$ (section 3.2.7 du chapitre 3). L'ensemble d'appariement $[M]_t$ est formé à l'instant t avant que les processus de découverte de règles soient appliqués, afin de potentiellement y ajouter des classeurs nouvellement construits par ces processus. $[A]_{t-1}$ et $[M]_t$ sont donc les derniers sous-ensembles de classeurs qu'il convient de vérifier lorsque des classeurs doivent être supprimés.

7.3 Améliorer les séquences comportementales

BEACS est capable de se doter de classeurs contenant des séquences comportementales, c'est-à-dire des classeurs dont le *Conséquent* consiste en une chaîne d'actions afin de traverser les situations où il serait confronté au problème d'aliasing perceptif. Tous les classeurs de BEACS contiennent des EPE, y compris les classeurs comportementaux qui intègrent alors les changements introduits avec la conception des EPE.

7.3.1 Construire les classeurs comportementaux

Comme BACS, les politiques de sélection de classeurs de BEACS modifient celles d'ACS 2 de telle sorte à ce qu'un classeur soit systématiquement renvoyé, permettant de conserver en mémoire le pénultième classeur sélectionné qui est ensuite utilisé pour construire les classeurs comportementaux. Les modifications sur les politiques de sélection apportées à BACS se retrouvent donc dans celles de BEACS.

Les classeurs comportementaux sont construits si la détection du problème d'aliasing perceptif de BEACS détecte une perception qui y serait associée et qui est donc stockée en mémoire dans la liste *PAIMemory*. Pour générer les classeurs à BSeq, BEACS utilise l'avant-dernier classeur cl_{t-2} qu'il a sélectionné dans la situation décrite par la perception p_{t-2} et chacun des classeurs candidats qui ont anticipé avec succès la situation décrite par p_t depuis la situation décrite par p_{t-1}. Les classeurs candidats sont donc ceux qui ont atteint l'*Expected case* du processus d'apprentissage par anticipation et à partir desquels une forme d'incertitude a été détectée. Ils sont temporairement gardés en mémoire dans une liste *candidatesForBSeq* par BEACS, jusqu'à ce que toutes les *Anticipations* des classeurs de l'ensemble d'action courant soient passées dans l'ALP. À la fin de l'ALP, si BEACS a détecté que la perception p_{t-1} est liée au PAI (autrement dit, si p_{t-1} est dans la liste *PAIMemory*), il essaie de construire de nouveaux classeurs comportementaux cl_{bs} pour chaque classeur *cl* de *candidatesForBSeq* avec cl_{t-2} seulement si :

- *cl* anticipe au moins un changement perceptif ;
- cl_{t-2} anticipe au moins un changement perceptif et n'est pas marqué ;

— la longueur de la séquence comportementale à construire est inférieure au paramètre utilisateur $bs_{\max}$ qui définit la longueur maximale des BSeq.

Ces conditions, identiques à celles de BACS, permettent de limiter la création de classeurs comportementaux qui n'entraîneraient pas de changements perceptifs et par extension, ne permettraient pas de traverser les situations où BEACS est confronté au PAI. La vérification sur la marque du classeur cl_{t-2} permet aussi d'éviter l'utilisation de classeurs dont l'*Anticipation* a été au moins une fois erronée, limitant par conséquent la construction de classeurs à BSeq dont les *Anticipations* seraient inadéquates.

Si ces conditions sont remplies, un classeur comportemental cl_{bs} est créé, comme illustré sur la figure 7.5, où tous ses attributs sont mis à leur valeur par défaut sauf :

— la *Condition* qui correspond à celle de cl_{t-2} ;
— le *Conséquent* qui correspond à celui de cl_{t-2} chaîné à celui de cl ;
— l'*Anticipation* qui comprend une seule situation anticipée construite à partir des *Anticipations* de cl_{t-2} et de cl. Chaque attribut perceptif de la nouvelle situation anticipée est d'abord spécifié à partir de la perception courante p_t si un changement est présent dans les attributs perceptifs correspondants des situations décrites par l'EPE de cl_{t-2} ou par l'EPE de cl. Les attributs qui ont été spécifiés sont ensuite comparés avec les attributs perceptifs de la *Condition* de cl_{bs} dans le but les supprimer s'ils sont identiques à ceux de la *Condition*, puisque seuls des changements doivent être décrits dans une EPE.

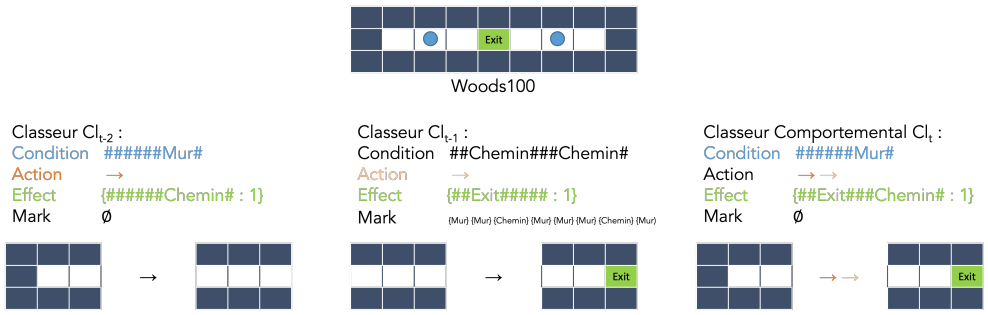


FIGURE 7.5 – Construction d'un classeur comportemental par BEACS. *Dans le labyrinthe* Woods100*, les classeurs comportementaux permettent de traverser les situations environnementales décrites par les deux points bleus afin d'atteindre la sortie, comme celui construit en exemple pour la situation associée au point gauche.*

Un nouveau classeur avec une séquence comportementale n'est pas directement ajouté à la population de classeurs par BEACS. BEACS s'appuie sur son processus de subsomption appliqué à l'ensemble d'appariement $[M]_{t-2}$ construit pour la perception p_{t-2} pour comparer les classeurs nouvellement construits avec ceux existants. Utiliser l'ensemble $[M]_{t-2}$ permet de ne pas avoir à comparer les nouveaux classeurs à séquence comportementale avec l'ensemble de la population. BEACS utilise alors l'algorithme 5.1.1 de BACS pour insérer ces nouveaux classeurs comportementaux dans sa population.

L'algorithme 7.3.1 décrit le processus de construction des classeurs comportementaux à partir de la liste *candidatesForBSeq* des classeurs candidats , la liste *PAIMemory* des perceptions liées au PAI, le classeur cl_{t-2}, les perceptions p_{t-1} et p_t, la longueur maximale $bs_{\max}$ des séquences comportementales, le pas d'apprentissage β_{alp} de l'ALP, les seuils d'expérience θ_{exp} et de fiabilité θ_{r}, l'ensemble d'appariement $[M]_{t-2}$ construit pour la perception p_{t-2} et le pas de temps t.

Algorithme 7.3.1 Processus de génération des classeurs comportementaux dans BEACS

```
1 : function ApplyBSeqBuildingProcess(candidatesForBSeq, PAIMemory,
    cl_{t-2}, p_{t-1}, p_t, bs_max, β_alp, θ_exp, θ_r, [M]_{t-2}, t)
2 :     if p_{t-1} ∈ PAIMemory and length of candidatesForBSeq > 0 then
3 :         for each cl ∈ candidatesForBSeq do
4 :             Build behavioral classifier cl_bs from cl_{t-2}, cl, bs_max, p_{t-1}, p_t, t
5 :             if cl_bs == ∅ then
6 :                 InsertFromALP(cl_bs, β_alp, θ_exp, θ_r, [M]_{t-2})
7 :             end if
8 :         end for
9 :     end if
```

7.3.2 Unifier l'apprentissage par anticipation

Le processus d'apprentissage par anticipation de BACS est séparé en deux sous-processus distincts, selon que les classeurs d'un ensemble d'action possèdent des séquences comportementales ou non. L'ALP des classeurs comportementaux de BACS est différent de celui des autres classeurs sur trois principaux points : il ne possède pas de processus de recouvrement ; il intègre un *Useless case* ; il ne détecte pas l'incertitude dans l'*Expected case*. À rebours

de BACS, BEACS unifie le processus d'apprentissage par anticipation de tous ses classeurs en un unique processus.

BEACS ne discrimine plus les classeurs comportementaux qui sont susceptibles de boucler entre des situations identiques. L'ALP de BEACS n'intègre plus de *Useless case* pour les classeurs à BSeq : il appartient au processus de rétribution d'adapter l'utilisation de tous les classeurs à la résolution de la tâche d'apprentissage, au lieu de diminuer leur qualité quand bien même leurs anticipations sont correctes. Suivant cette idée, BEACS n'intègre pas le mécanisme discriminatoire des séquences comportementales de BACS qui garde en mémoire les perceptions reçues lors de la réalisation d'une séquence afin de diminuer la qualité des classeurs si de nouvelles perceptions ne sont pas reçues.

L'*Expected case* des classeurs de BEACS est aussi identique à tous les classeurs, tout comme le *Correctable case* et le *Not correctable case*. Contrairement à BACS, BEACS est alors capable de détecter l'incertitude environnementale à partir des classeurs comportementaux et de construire de nouveaux classeurs capables de gérer l'incertitude détectée : si l'incertitude n'est pas liée au PAI, BEACS étoffe l'EPE des *Anticipations* afin d'apprendre à anticiper les différentes situations qui suivent la réalisation des actions ; sinon, BEACS essaie de construire de nouvelles séquences comportementales lui permettant de gérer le PAI avec plus de souplesse que BACS, puisqu'il peut s'appuyer sur tous les classeurs et ne plus avoir à construire de séquences en ajoutant une action une seule à la fois et en bout de chaîne.

BEACS, comme BACS, ne permet pas au processus de recouvrement de directement créer des classeurs comportementaux si aucun classeur comportemental d'un ensemble d'action ne parvient à correctement anticiper la prochaine situation. Il revient au processus de construction des classeurs à BSeq de créer de nouvelles séquences si les conditions décrites dans la section 7.3.1 sont réunies.

L'algorithme 7.3.2 décrit le processus d'apprentissage par anticipation de tous les classeurs de BEACS : il intègre alors la construction des classeurs à EPE, la détection du PAI et la construction des classeurs à BSeq.

Algorithme 7.3.2 Processus d'apprentissage par anticipation de BEACS

function ALP($[A]_{t-1}$, $[M]_{t-2}$, $[M]_{t-1}$, $[M]_t$, $[P]$, $cl_{\mathrm{t}-2}$, p_{t-1}, a, p_t, n_A, L, β_{alp}, *PAIMemory*, θ_{exp}, θ_i, θ_{r}, $u_{\max}$, $bs_{\max}$, t)
  if $bs_{\max} > 0$ **and** $cl_{\mathrm{t}-2} \neq \emptyset$ **then**
    candidatesForBSeq $\leftarrow$ []
  end if
  for each $cl \in [A]_{t-1}$ **do**
    Update $t_{alp_{\mathrm{cl}}}$ with t and increase exp_{cl} by 1
    $cl_{\mathrm{new}} \leftarrow \emptyset$
    oneCorrectAnticipation $\leftarrow$ *False*
    if cl anticipates correclty p_t from p_{t-1} **then**
      uncertaintyDetected, cl_{new} $\leftarrow$ BEACS *expected case* with cl, p_{t-1}, β_{alp}, $u_{\max}$, t
      oneCorrectAnticipation $\leftarrow$ *True*
      if *uncertaintyDetected* == *True* **and** $bs_{\max} > 0$ **and** $cl_{\mathrm{t}-2} \neq \emptyset$ **then**
        Add cl in *candidatesForBSeq*
      end if
    else
      if cl is correctable **then**
        $cl_{\mathrm{new}} \leftarrow$ *correctable case* of ACS 2 with cl, p_{t-1}, p_t, β_{alp}, t
      else
        not correctable case of ACS 2 with cl, β_{alp}
      end if
    end if
    if $q_{\mathrm{cl}} < \theta_i$ **then**
      Remove cl from $[A]_{t-1}$, $[M]_t$ and $[P]$
    end if
    if $cl_{\mathrm{new}} \neq \emptyset$ **then**
      if $C_{\mathrm{cl}_{\mathrm{new}}}$ corresponds to p_{t-1} **then**
        InsertFromALP(cl_{new}, β_{alp}, θ_{exp}, θ_{r}, $[A]_{t-1}$)
      else
        InsertFromALP(cl_{new}, β_{alp}, θ_{exp}, θ_{r}, $[P]$)
      end if
    end if
  end for
  if *oneCorrectAnticipation* == *False* **then**
    if $[A]_{t-1}$ is not an action set with behavioral sequences **then**
      Build cl_{new} by Covering from p_{t-1}, a, p_t, t
      InsertFromALP(cl_{new}, β_{alp}, θ_{exp}, θ_{r}, $[A]_{t-1}$)
    end if
  end if
  ApplyEPEBuildingProcess($[A]_{t-1}$, p_{t-1}, L, β_{alp}, θ_{exp}, θ_{r}, t)
  PAILoop($[A]_{t-1}$, $[M]_{t-1}$, $[M]_t$, $[P]$, p_{t-1}, n_A, L, *PAIMemory*, θ_r, θ_{exp}, θ_{pai}, t)
  ApplyBSeqBuildingProcess(*candidatesForBSeq*, *PAIMemory*, $cl_{\mathrm{t}-2}$, p_{t-1}, p_t, $bs_{\max}$, β_{alp}, θ_{exp}, θ_{r}, $[M]_{t-2}$, t)

7.3.3 Généraliser les classeurs comportementaux

BEACS généralise indirectement des classeurs à BSeq tout en évitant de créer des classeurs à BSeq qui correspondraient à des situations sans lien avec le PAI, là où BACS bloquait complètement la généralisation de ces classeurs. Les *Conditions* de deux classeurs à muter sont comparées : si un attribut perceptif est généralisé dans une *condition* et pas dans l'autre, l'attribut perceptif correspondant dans l'autre peut être généralisé. Comme les classeurs à BSeq contiennent des EPE, la trace de mutation μT est aussi vérifiée afin de savoir si un attribut conditionnel peut être généralisé ou non.

Algorithme 7.3.3 Mutation des classeurs de BEACS

```
1 : function MUTATE(cl_1, cl_2, μ, L)
2 :     for i ← 0 to L − 1 do
3 :         if C_cl1[i] ≠ # and C_cl2[i] == # then
4 :             if random number in [0, 1[ < μ and μT_cl1 == True then
5 :                 C_cl1[i] ← #
6 :             end if
7 :             continue
8 :         end if
9 :         if C_cl1[i] == # and C_cl2[i] ≠ # then
10 :            if random number in [0, 1[ < μ and μT_cl2 == True then
11 :                C_cl2[i] ← #
12 :            end if
13 :            continue
14 :        end if
15 :        if cl_1 is not a behavioral classifier and C_cl1[i] ≠ # and
              random number in [0, 1[ < μ and μT_cl1 == True then
16 :            C_cl1[i] ← #
17 :        end if
18 :        if cl_2 is not a behavioral classifier and C_cl2[i] ≠ # and
              random number in [0, 1[ < μ and μT_cl2 == True then
19 :            C_cl2[i] ← #
20 :        end if
21 :    end for
```

L'algorithme 7.3.3 décrit l'opérateur de mutation appliqué à deux classeurs cl_1 et cl_2 nouvellement générés par la généralisation génétique. Il prend en argument le taux de mutation μ et la longueur L des séquences perceptives. Si seul l'un des attributs conditionnels de cl_1 ou cl_2 peut être généralisé et que la trace de mutation permet cette généralisation, cet attribut est généralisé

selon une probabilité μ qu'il s'agisse d'un classeur à BSeq ou non. Sinon, un attribut conditionnel d'un classeur a une probabilité μ d'être généralisé si la trace de mutation permet cette généralisation et que ce classeur n'est pas un classeur comportemental.

Algorithme 7.3.4 Suppression de n micro-classeurs dans $[A]$ dans BEACS

```
1 : function DeleteMicroClassifiers([A], [M], [P], n)
2 :     while n + Σ_{cl∈[A]} num_cl > θ_as do
3 :         μcl_del ← random micro classifier of [A]
4 :         for each micro classifier μcl ∈ [A] do
5 :             if random number in [0, 1[ < 1/3 then
6 :                 if q_{μcl_del} − q_μcl > 0.1 then
7 :                     μcl_del ← μcl
8 :                 end if
9 :                 if |q_{μcl_del} − q_μcl| ≤ 0.1 then
10 :                    if M_{μcl_del} == ∅ and M_μcl ≠ ∅ then
11 :                        μcl_del ← μcl
12 :                    else if aav_{μcl_del} < aav_μcl and
                                (M_{μcl_del} == ∅ or M_μcl ≠ ∅) then
13 :                        μcl_del ← μcl
14 :                    end if
15 :                end if
16 :            end if
17 :        end for
18 :        if μcl_del ≠ ∅ then
19 :            if num_{μcl_del} > 1 then
20 :                num_{μcl_del} ← num_{μcl_del} − 1
21 :            else
22 :                Remove μcl_del from [A], [M] and [P]
23 :            end if
24 :        end if
25 :    end while
```

Par ailleurs, BEACS modifie le processus de délétion de micro-classeurs associé au mécanisme de généralisation afin de favoriser la mise en compétition de micro-classeurs lors du processus de sélection par tournoi. Le processus de délétion de micro-classeurs d'ACS 2 [4] n'initialise pas de variable permettant de directement réaliser un tournoi avec un autre micro-classeur. Il itère sur les micro-classeurs d'un ensemble d'action pour sélectionner un premier micro-classeur avec une probabilité de 30%. Puis, il doit sélectionner

4. Il est décrit dans la section 3.2.4 du chapitre 3

un second micro-classeur durant le même parcours avec aussi une probabilité de 30%. Si un second micro-classeur n'est pas sélectionné avant la fin de ce parcours, le premier micro-classeur est supprimé de l'ensemble d'action. Ce fonctionnement ne permet pas de mettre tous les micro-classeurs en compétition puisque tout dépend de la sélection du premier micro-classeur. BEACS choisit alors aléatoirement un premier micro-classeur avant d'itérer sur tous les micro-classeurs d'un ensemble d'action dans le but d'encourager la sélection par tournoi, où chaque micro-classeur a la même probabilité d'être sélectionné. La suppression des micro-classeurs par la généralisation génétique de BEACS est décrite par l'algorithme 7.3.4 où n désigne le nombre de classeurs construits par le mécanisme de généralisation génétique et le nombre de classeurs à supprimer de $[A]$, $[M]$ et $[P]$.

7.3.4 Promouvoir les petites séquences comportementales

Le processus de rétribution de BEACS est enfin modifié avec deux objectifs :

- promouvoir les séquences comportementales les plus petites ;
- améliorer les capacités du processus de rétribution face à l'incertitude environnementale.

Pour répondre à ces objectifs, BEACS s'appuie sur le *Double Q-Learning* (Hasselt, 2010). Quand le *Q-Learning* s'appuie sur un estimateur de la fonction de valeur Q pour construire ses politiques de décision, le *Double Q-Learning* utilise deux estimateurs dont la mise à jour est intriquée et où chacun des estimateurs s'appuie sur des ensembles d'interactions avec l'environnement différents. Contrairement au *Q-Learning*, le fait de mettre à jour les estimateurs à partir d'ensembles d'interactions différents permet au double *Q-Learning* de ne pas significativement surestimer la fonction de valeur Q d'un problème dans un environnement, en particulier si l'environnement est incertain. Par exemple, (Hasselt, 2010) montre au jeu de la roulette que le *Q-Learning* estime que chaque action rapporte en moyenne $10\$$ alors que chacune des actions devrait théoriquement faire perdre $0.053\$$, ce qui n'est pas réaliste. Le *Double Q-Learning* estimerait pour ce jeu une perte un peu plus importante, sous-estimant alors les valeurs des actions, ou équivalente à la perte théorique si les valeurs initiales des actions sont optimistes.

BEACS adapte l'estimation des récompenses des classeurs à la longueur des séquences d'actions afin de promouvoir l'utilisation des BSeq les plus courtes. Chaque classeur *cl* utilise deux estimateurs ($r_{A_{\text{cl}}}$ et $r_{B_{\text{cl}}}$) pour calculer sa prédiction de récompense r_{cl}, en utilisant les prédictions maximales de récompenses $maxP_A$ et $maxP_B$ au prochain pas de temps pour chaque estimateur, la récompense immédiate ρ, le facteur d'actualisation γ, le pas d'apprentissage par renforcement β_{rl}, la longueur maximale des séquences comportementales $bs_{\max}$ et une différence configurable ϵ_{rl}. Même si les deux estimateurs convergent vers la même valeur, ϵ_{rl} permet à BEACS d'être biaisé en faveur des séquences les plus courtes en introduisant un écart artificiel entre les prédictions de récompenses des deux estimateurs. La prédiction de récompense d'un classeur est mise à l'échelle, de sorte que la récompense la plus élevée d'un estimateur est donnée lorsque la séquence est constituée d'une action unique, tandis que la différence entre les deux estimateurs est utilisée pour diminuer la prédiction de récompense selon la longueur de la séquence d'actions. Les prédictions de récompense des classeurs sont mises à jour selon l'algorithme 7.3.5.

Algorithme 7.3.5 Processus de rétribution de BEACS

function UPDATEREWARD(cl, $maxP_A$, $maxP_B$, ρ, γ, β_{rl}, $bs_{\max}$, ϵ_{rl})
  if random number in $[0, 1[< 0.5$ **then**
    $r_{A_{\text{cl}}} \leftarrow r_{A_{\text{cl}}} + \beta_{\text{rl}}(\rho + \gamma maxP_B - r_{A_{\text{cl}}})$
  else
    $r_{B_{\text{cl}}} \leftarrow r_{B_{\text{cl}}} + \beta_{\text{rl}}(\rho + \gamma maxP_A - r_{B_{\text{cl}}})$
  end if
  $max_r \leftarrow max(r_{A_{\text{cl}}}, r_{B_{\text{cl}}})$
  $min_r \leftarrow min(r_{A_{\text{cl}}}, r_{B_{\text{cl}}})$
  $r_{\text{cl}} \leftarrow max_r - (max_r - min_r + \epsilon_{\text{rl}}) * \frac{length\ of\ A_{\text{cl}} - 1}{bs_{\max}}$

7.4 Capacités de BEACS

7.4.1 Paramètres de BEACS

BEACS intègre trois nouveaux paramètres à définir en amont d'un apprentissage en plus de ceux d'ACS 2 :

— une différence ϵ_{rl} utilisé par le processus de rétribution ;

— la longueur maximale des séquences comportementales $bs_{\max}$;
— le seuil de planification de la détection du PAI θ_{pai}.

La conception des EPE n'induit pas de nouveaux paramètres contrairement aux PEP de PEPACS qui nécessitent un pas d'apprentissage pour mettre à jour les probabilités incluses dans les PEP.

Le but du paramètre ϵ_{rl} est de créer une petite différence entre les deux estimateurs afin de pouvoir promouvoir les séquences comportementales les plus courtes. Elle n'a pas vocation à prendre de valeurs qui pourraient bruiter la mise en place d'une politique décisionnelle permettant à BEACS de résoudre sa tâche d'apprentissage. Nous avons alors empiriquement fixé ϵ_{rl} à la valeur $1\text{e}{-6}$.

Comme nous l'indiquions dans la section 5.4 du chapitre 5, nous utilisons une longueur maximale $bs_{\max}$ de 3 de telle sorte à pouvoir comparer les capacités de BEACS à se doter de représentations environnementales et de politiques décisionnelles par rapport à BACS. Par exemple, c'est pour cette valeur de $bs_{\max}$ que les populations de BACS sont les plus grandes et qu'il y a plus de choix parmi les séquences d'actions possibles (pour n actions possibles, il y a $n + n^2 + n^3$ séquences comportementales différentes possibles).

Nous avons empiriquement fixé le seuil de planification de la détection du PAI θ_{pai} à une valeur de 1000. Le choix d'une telle valeur pour ce seuil s'appuie sur l'idée que BEACS a besoin de temps pour que les processus de découverte de nouvelles règles construisent des classeurs fiables et expérimentés capables d'anticiper correctement les prochaines situations environnementales : le processus d'apprentissage par anticipation est appliqué à chaque pas de temps et le mécanisme de généralisation génétique est appliqué en moyenne tous les 100 pas de temps ($\theta_{\text{ga}} = 100$) pour un ensemble d'action donné. Ainsi, la détection du PAI est appliquée une fois sur un ensemble d'appariement quand la généralisation génétique peut être utilisée une dizaine de fois pour un ensemble d'action.
Un seuil θ_{pai} plus élevé permettrait au système de s'appuyer sur des classeurs potentiellement plus fiables et plus expérimentés, mais ralentirait en même temps l'acquisition d'une politique décisionnelle si des classeurs à séquences comportementales sont nécessaires pour résoudre une tâche d'apprentissage. A l'inverse, un seuil θ_{pai} plus faible peut accélérer l'acquisition

de classeurs à BSeq mais rendrait plus incertaine la détection du PAI celle-ci devant potentiellement s'appuyer sur des classeurs fiables et expérimentés à un instant donné : ces classeurs peuvent ne pas encore exister ou être incomplets à cause de situations environnementales non décrites dans les EPE bien que fiables et expérimentés.

7.4.2 Méthodologie

Les métriques d'évaluation et le protocole expérimental détaillés dans le chapitre 4 doivent permettre d'évaluer les capacités d'apprentissage de BEACS en répondant aux questions suivantes :

1. BEACS se comporte-t-il comme ACS 2 dans les environnements où il n'est pas sujet à l'incertitude environnementale ?
2. BEACS est-il capable de se doter de représentations environnementales complètes, indépendamment de l'incertitude environnementale ?
3. Le mécanisme des EPE permet-il de décrire fidèlement les différentes probabilités des transitions environnementales, indépendamment de l'incertitude environnementale ?
4. Comment le couplage des EPE et des BSeq influencent-elle la taille et la généralité des représentations environnementales dans des environnements incertains ?
5. BEACS est-il capable d'éviter de construire des classeurs à EPE surgénéraux ?
6. Comment le nouvel opérateur de mutation influence-t-il la construction des représentations environnementales ?
7. BEACS est-il capable de détecter les situations où il est confronté au problème d'aliasing perceptif quand le résultat des actions est incertain ?
8. Comment le couplage des EPE et des BSeq influencent-elle la mise en place de politiques de décision dans des environnements incertains ?
9. Est-ce que le *Double Q-Learning* permet à BEACS de se doter de politiques décisionnelles plus efficaces ?

L'ensemble de ces questions permettent de voir en quelle mesure BEACS est capable de tirer parti des avantages des séquences comportementales et du mécanisme des prédictions améliorées par l'expérience, successeur des prédictions améliorées par les probabilités, tout en essayant d'apporter une réponse aux pistes d'amélioration dégagées par l'usage de ces mécanismes dans leurs chapitres respectifs (chapitre 5 et chapitre 6). Les résultats de BEACS sont ainsi comparés avec BACS et PEPACS, ces derniers étant les systèmes de classeurs à anticipation à l'état de l'art sur les BSeq et les PEP. Nous fournissons en annexe D.1 une comparaison des capacités d'apprentissage de BEACS et d'ACS 2 dans les environnements incertains pour un lecteur désireux de savoir comment BEACS se comporte par rapport au système dont il est issu.

Comme pour BACS et PEPACS, les capacités d'apprentissage de BEACS sont d'abord comparées avec celles d'ACS 2 dont il est dérivé dans les environnements où il n'est pas sujet à l'incertitude environnementale. Les nombres d'étapes nécessaires pour atteindre la sortie des labyrinthes permettent de comparer les politiques de décision mises en place par ces systèmes : ces nombres sont donc utilisés pour comparer les politiques décisionnelles de BEACS et ACS 2. Les ratios de connaissance, les tailles des populations et les spécificités moyennes des classeurs fiables permettent de comparer les représentations environnementales développées par ces systèmes, complétant ainsi la réponse à notre 1re interrogation.

Les interrogations 2 à 5 étudient en détail les représentations construites par BEACS, BACS et PEPACS dans les environnements incertains. En particulier, les ratios de connaissance sont utilisés pour répondre à la 2e interrogation. Les Erreurs Moyennes sur la Prédiction de l'Anticipation (EMPA) sont utilisées pour répondre à la 3e interrogation. Les tailles des populations, les spécificités moyennes des classeurs réguliers (sans BSeq) et les spécificités moyennes des classeurs à BSeq sont utilisées pour répondre à la 4e interrogation. Tous les classeurs fiables qui contiennent des EPE et qui correspondent à d'autres situations que celles où il y a le PAI ont été dénombrés afin de répondre à la 5e interrogation. Les nombres résultants sont comparés à ceux de PEPACS avec et sans la spécification des PEP introduite dans la copie des classeurs.

Pour répondre à la 6e interrogation, toutes les métriques des précédentes interrogations sont utilisées où le nouvel opérateur de mutation de BEACS est comparé avec trois opérateurs de mutation : le premier permettant une généralisation directe des classeurs comportementaux et s'appuyant sur la trace de mutation des classeurs (dénommé par BEACS-DmuT) ; le second ne permettant pas une généralisation directe des classeurs comportementaux et ne s'appuyant pas sur la trace de mutation des classeurs (dénommé par BEACS-IndNomuT) ; le dernier étant celui d'ACS 2 où tous les classeurs peuvent être directement généralisés et où la trace de mutation n'est pas utilisée (dénommé par BEACS-DNomuT). Nous fournissons en annexe D.2 les algorithmes utilisés pour les opérateurs de mutation de BEACS-DmuT, BEACS-IndNomuT et de BEACS-DNomuT.

Pour répondre à la 7e interrogation, nous avons relevé à la fin de l'apprentissage de BEACS l'état de la mémoire des perceptions liées au PAI *PAIMemory* de BEACS. Connaissant pour chacun des environnements les situations liées ou non au PAI, le relevé de cette liste permet de déterminer la *balanced accuracy* du mécanisme de détection du PAI, qui est défini comme la moyenne de la sensibilité et de la spécificité des résultats de la détection ou autrement dit, la moyenne du taux de vrais positifs et du taux de vrais négatifs.

Les nombres d'étapes pour atteindre la sortie des labyrinthes sont utilisés pour comparer les politiques décisionnelles de BEACS, BACS et PEPACS, répondant ainsi à la 8e interrogation.

Pour enfin répondre à notre 9e et dernière interrogation, les nombres d'étapes pour atteindre la sortie des labyrinthes sont aussi utilisés pour comparer les politiques décisionnelles de BEACS avec le *Double Q-Learning* et les politiques décisionnelles de BEACS s'il s'appuie sur le processus de rétribution d'ACS 2 et par extension le *Q-Learning* (dénommé par BEACS-Q).

Les résultats obtenus par BEACS sont présentés dans la section 7.4.3. Ils sont ensuite discutés dans la section 7.5.

7.4.3 Résultats

BEACS se comporte-t-il comme ACS 2 dans les environnements où il n'est pas sujet à l'incertitude environnementale ?

Les ratios de connaissance des populations de classeurs d'ACS 2 et de BEACS sont strictement identiques et maximaux : les deux systèmes sont parvenus à apprendre une représentation complète de leur environnement puisque toutes les transitions environnementales sont décrites par au moins un classeur fiable.
Les tailles des populations de classeurs de BEACS, et donc des représentations environnementales, sont plus petites que celles d'ACS 2 dans tous les environnements sauf *Maze4* où elles ne présentent pas de différences significatives.
Les spécificités moyennes des classeurs sont plus petites que celles d'ACS 2 dans tous les environnements sauf *Maze4* et *Maze5* où elles sont plus grandes.
Les modèles de transitions appris par ces systèmes de classeurs sont globalement plus compacts et généraux que ceux d'ACS 2. Ces résultats sont exclusivement le fait du nouveau processus de suppression de micro-classeurs de la généralisation génétique de BEACS, puisque BEACS n'a utilisé à aucun moment des classeurs à BSeq ou des classeurs dont les EPE contiennent plusieurs situations anticipées. Nous fournissons en annexe D.3 des figures illustrant ces résultats.

Enfin, toutes les politiques de décision construites par ACS 2 et BEACS ne présentent de différences significatives. Ainsi, BEACS se comporte comme ACS 2 dans les environnements où il n'est pas sujet à l'incertitude environnementale.

BEACS est-il capable de se doter de représentations environnementales complètes, indépendamment de l'incertitude environnementale ?

Dans les environnements où BEACS est confronté au PAI et sans les actions incertaines, BEACS atteint au moins une fois les 100% dans 15 des 16 environnements, à l'exception de *MazeE2* où il atteint au plus un ratio de connais-

sance de 99.40% pour une moyenne de 97.7% (*cf.* figure 7.6). Les ratios de connaissance atteints par BEACS sont en particulier maximaux pour 11 des 16 labyrinthes. Les ratios de PEPACS présentent uniquement des différences significatives avec ceux de BEACS pour *Cassandra4x4* et *MazeE2*, où ils sont plus élevés. Tous les ratios de connaissance de BEACS sont plus élevés que ceux atteints par BACS ou sont équivalents à ceux atteints par PEPACS.

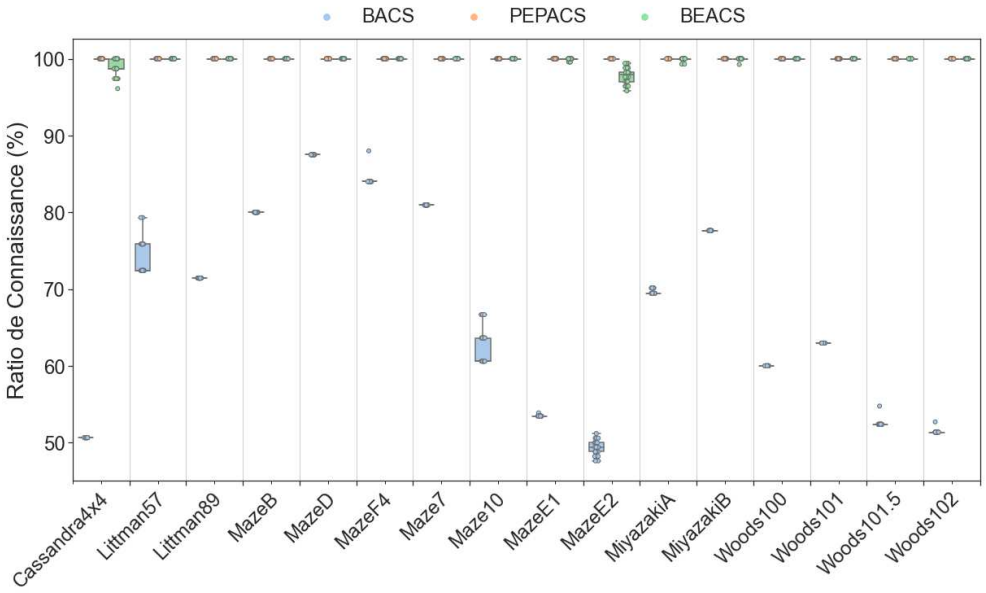


FIGURE 7.6 – Ratios de connaissance des populations de classeurs de BACS, PEPACS et BEACS dans les environnements avec PAI et sans les actions incertaines.

Quand le résultat des actions est incertain (*cf.* figure 7.7), les ratios de connaissance de BEACS sont à nouveau tous supérieurs à ceux atteints par BACS dans l'ensemble des environnements du banc de test. BEACS atteint au moins une fois les 100% dans 20 des 23 environnements, à l'exception de *MazeE1* (maximum de 97.41% pour une moyenne de 95.45%), *MazeE2* (maximum de 99.40% pour une moyenne de 97.04%) et *MiyazakiA* (maximum de 99.31% pour une moyenne de 95.27%). Les ratios de PEPACS présentent uniquement des différences significatives avec ceux de BEACS pour 7 environnements où ils sont plus élevés et pour *Maze4* où ils sont plus faibles.

BEACS semble ainsi capable de se doter de représentations environnementales complètes indépendamment de l'incertitude environnementale, néanmoins avec plus de difficulté que PEPACS.

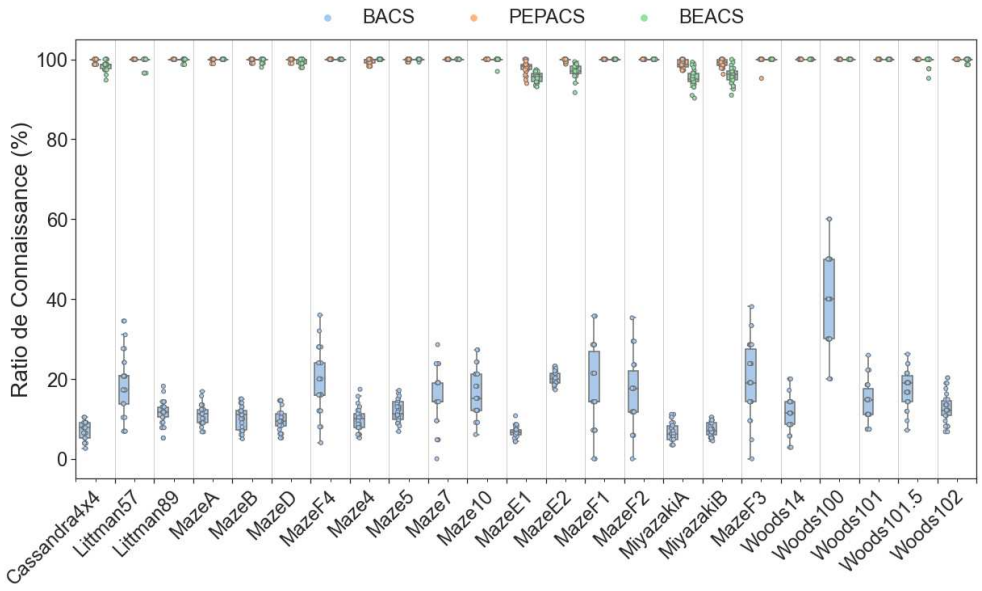


FIGURE 7.7 – Ratios de connaissance des populations de classeurs de BACS, PEPACS et BEACS dans tous les environnements avec les actions incertaines.

Le mécanisme des EPE permet-il de décrire fidèlement les différentes probabilités des transitions environnementales, indépendamment de l'incertitude environnementale ?

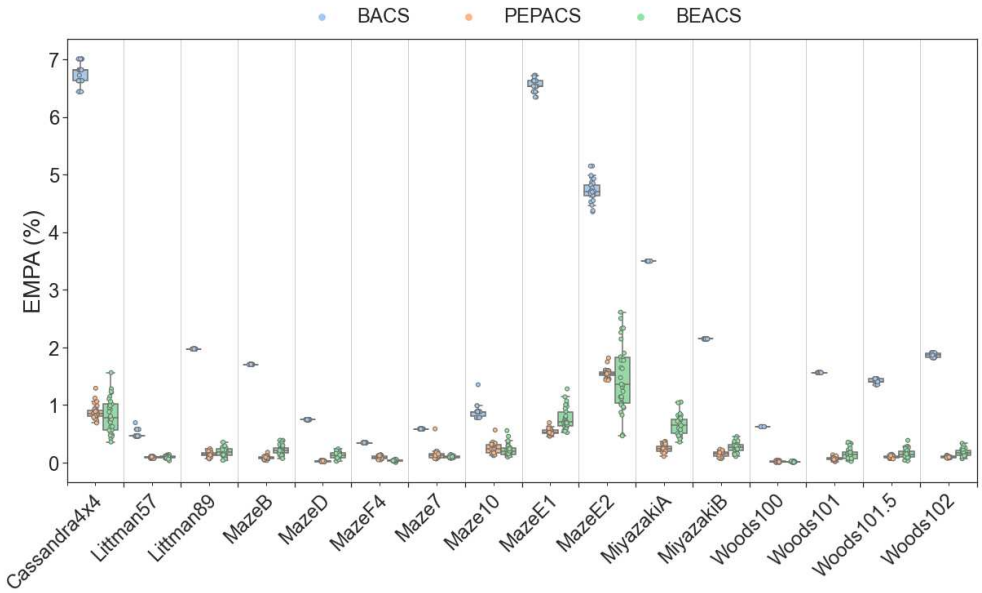


FIGURE 7.8 – Erreurs Moyennes sur la Prédiction de l'Anticipation de BACS, PEPACS et BEACS dans les environnements avec PAI et sans les actions incertaines.

Les Erreurs Moyennes sur la Prédiction de l'Anticipation (EMPA) de BACS

sont les plus élevées dans tous les environnements avec PAI et sans les actions incertaines (cf. figure 7.8). Les EMPA de PEPACS décrivent globalement plus fidèlement les probabilités des transitions environnementales que BEACS où les EMPA de BEACS ne présentent pas de différences significatives avec celles de PEPACS dans 4 environnements, sont plus petites que celles de PEPACS dans 3 environnements et enfin plus grandes dans les 9 environnements restants. L'écart absolu entre entre les EMPA de PEPACS et de BEACS est au plus de 0.4%.

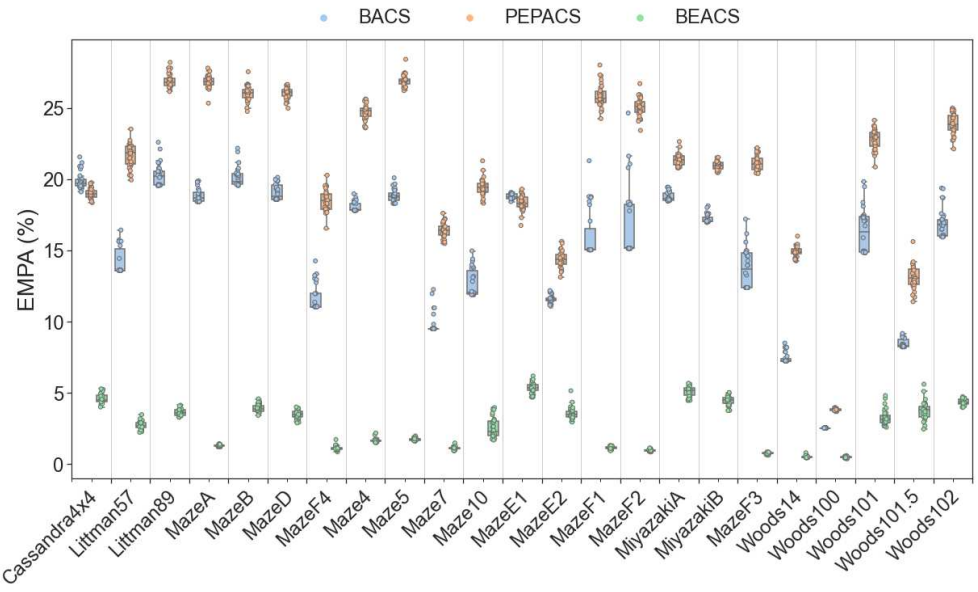


FIGURE 7.9 – Erreurs Moyennes sur la Prédiction de l'Anticipation de BACS, PEPACS et BEACS dans tous les environnements avec les actions incertaines.

En revanche, les EMPA de BEACS sont les plus faibles pour tous les labyrinthes du banc de test avec les actions incertaines (cf. figure 7.9). L'écart absolu entre entre les EMPA de PEPACS et de BEACS est au plus de 25.57%. Les EPE permettent de prédire les probabilités des différentes transitions environnementales indépendamment de l'incertitude environnementale.

Comment le couplage des EPE et des BSeq influencent-elle la taille et la généralité des représentations environnementales dans des environnements incertains ?

Les représentations construites par BACS sont les plus grandes dans 15 des 16 labyrinthes où il n'est pas confronté au PAI et sans les actions incertaines,

étant aussi les plus grandes avec celles de BEACS dans le dernier labyrinthe *Cassandra4x4* (*cf.* figure 7.10). Les représentations construites par PEPACS sont les plus petites dans 15 des 16 labyrinthes où il n'est pas confronté au PAI et sans les actions incertaines, étant aussi les plus petites avec celles de BEACS dans le dernier labyrinthe *MazeF4*. Les tailles des populations de classeurs de BEACS sont ainsi de tailles comprises entre celles de PEPACS et de BACS dans les environnements avec PAI et sans les actions incertaines.

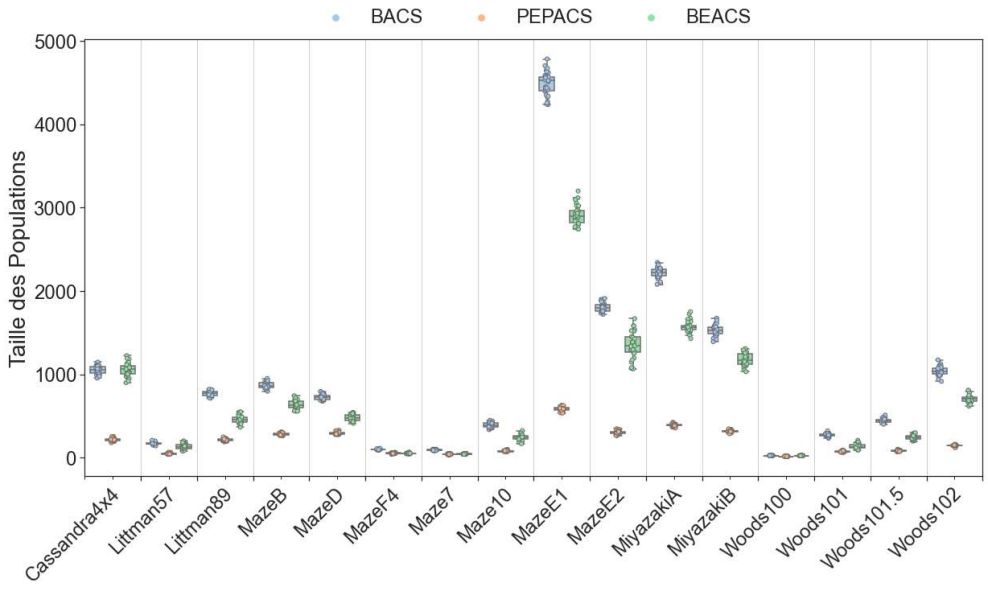


FIGURE 7.10 – Tailles des populations de classeurs de BACS, PEPACS et BEACS dans les environnements avec PAI et sans les actions incertaines.

Dans ces mêmes environnements, les spécificités moyennes des classeurs réguliers fiables de BEACS sont les plus élevées pour 13 labyrinthes des 16 et les plus faibles dans les 3 labyrinthes restants (figure 7.11). Les spécificités moyennes des classeurs comportementaux fiables de BEACS sont plus élevées que celles de BACS pour 5 labyrinthes, équivalentes dans 5 labyrinthes et plus petites dans les 6 labyrinthes restants (*cf.* figure 7.12).

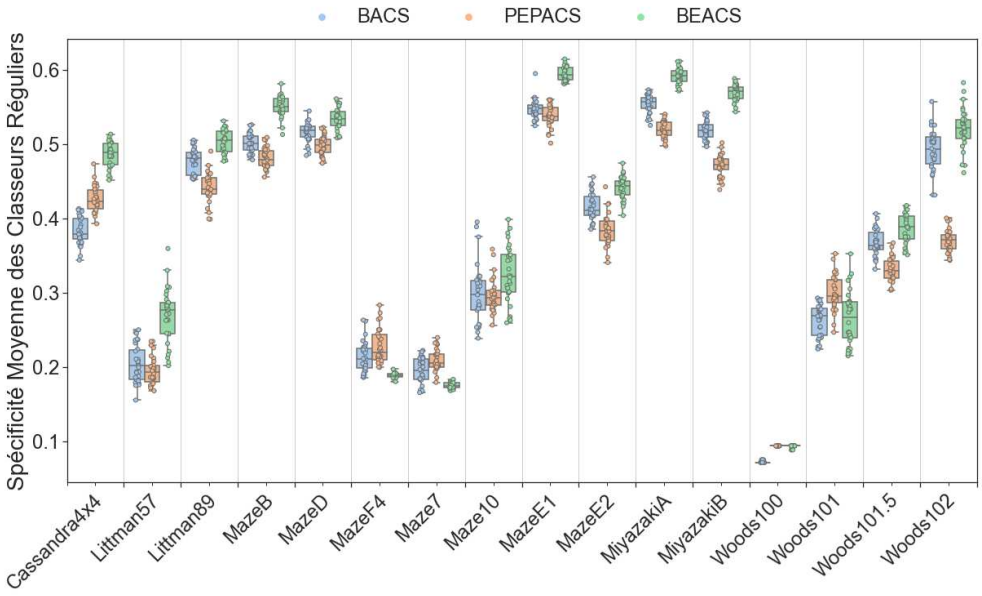


FIGURE 7.11 – Spécificités moyennes des classeurs réguliers fiables de BACS, PEPACS et BEACS dans les environnements avec PAI et sans les actions incertaines.

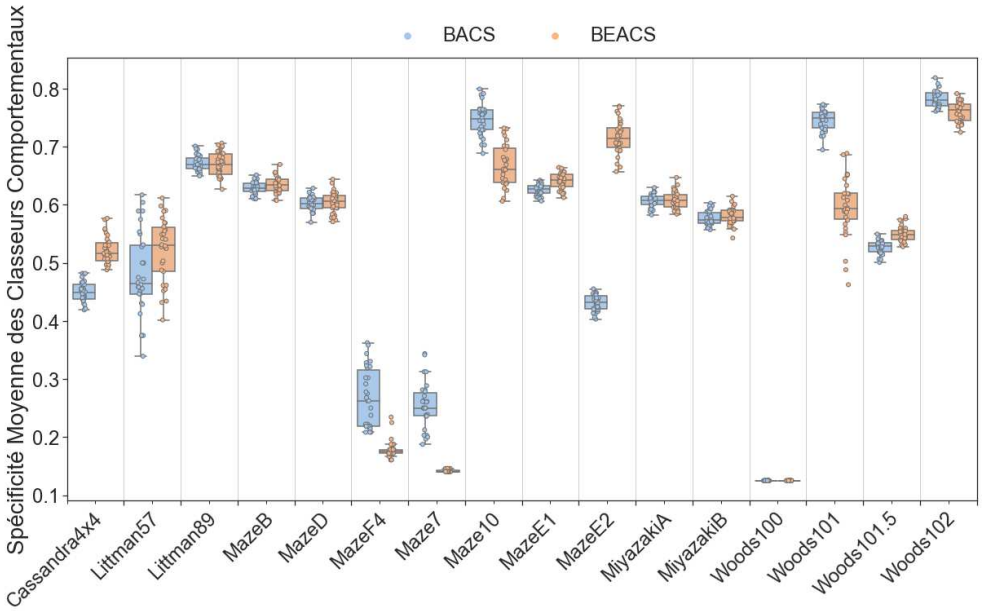


FIGURE 7.12 – Spécificités moyennes des classeurs comportementaux fiables de BACS et BEACS dans les environnements avec PAI et sans les actions incertaines.

Les populations de classeurs de BACS sont les plus grandes dans 21 des 23 labyrinthes où il est confronté aux actions incertaines, celles de BEACS étant les plus grandes dans *MazeE1* (*cf.* figure 7.13). Les populations de classeurs de BEACS sont les plus petites dans 15 des 23 labyrinthes où il est confronté aux actions incertaines, s'agissant autrement de celles de PEPACS. Les tailles des populations de classeurs de BEACS sont globalement de tailles inférieures à celles de BACS et celles de PEPACS dans les environnements avec les actions incertaines.

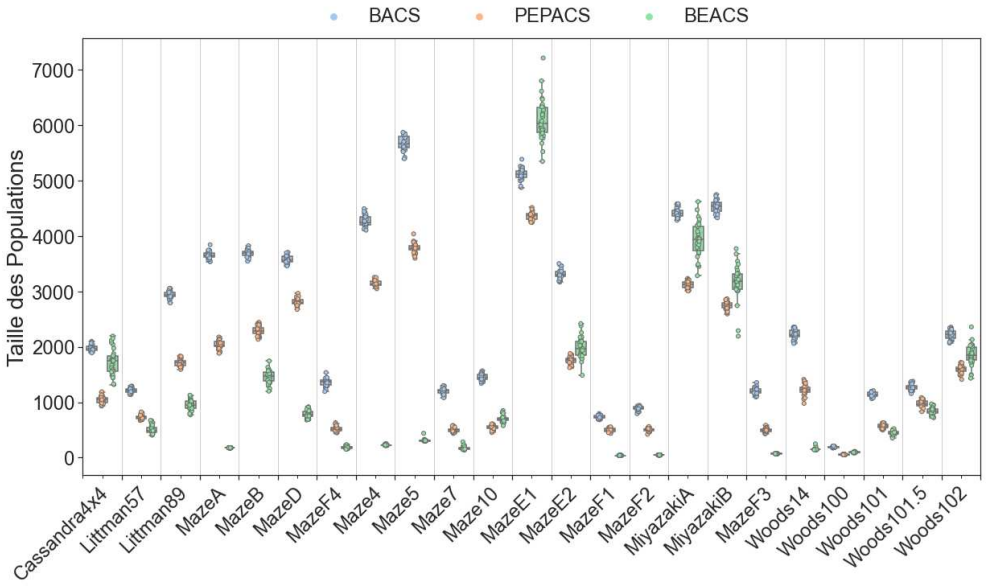


FIGURE 7.13 – Tailles des populations de classeurs de BACS, PEPACS et BEACS dans tous les environnements avec les actions incertaines.

Dans ces mêmes environnements, les spécificités moyennes des classeurs réguliers fiables de BACS sont les plus faibles pour 21 des 23 labyrinthes, celles de BEACS étant les plus faibles dans *Woods14* et *Woods100* (*cf.* figure 7.14). Autrement, cette mesure est la plus élevée pour BEACS dans 11 des 23 labyrinthes ou pour PEPACS dans 12 des 23 labyrinthes. Les spécificités moyennes des classeurs comportementaux fiables de BEACS sont plus élevées que celles de BACS pour 13 labyrinthes, équivalentes dans 6 autres labyrinthes et équivalentes dans les 4 labyrinthes restants (*cf.* figure 7.15).

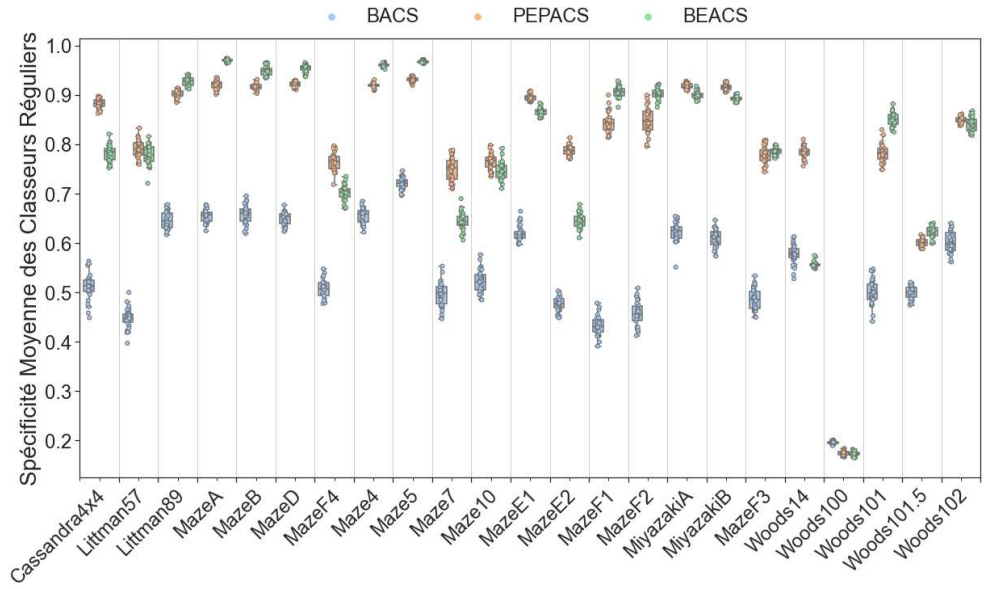


FIGURE 7.14 – Spécificités moyennes des classeurs réguliers fiables de BACS, PEPACS et BEACS dans tous les environnements avec les actions incertaines.

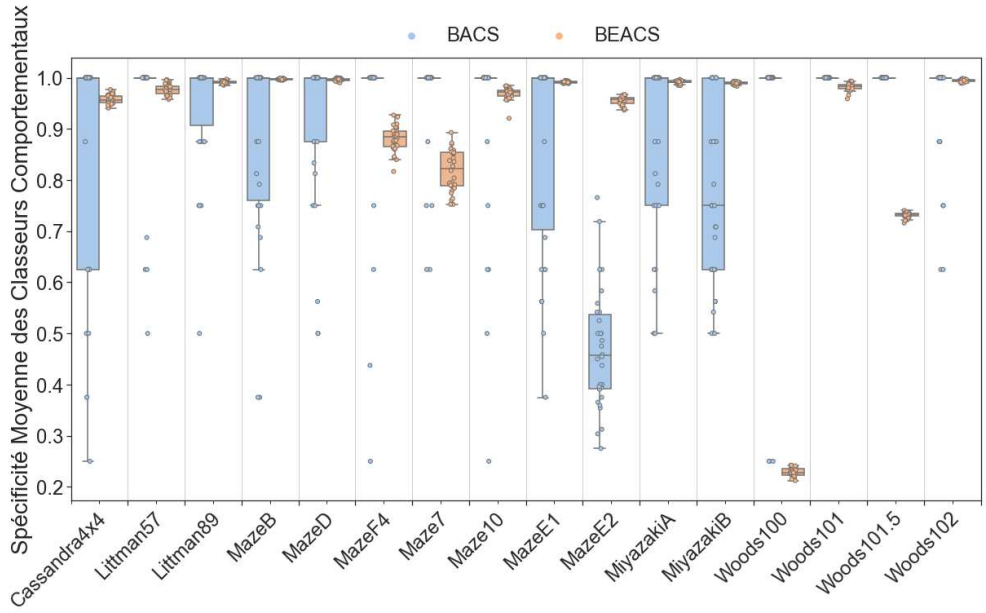


FIGURE 7.15 – Spécificités moyennes des classeurs comportementaux fiables de BACS et BEACS dans les environnements avec PAI et les actions incertaines.

BEACS est-il capable d'éviter de construire des classeurs à EPE surgénéraux ?

Dans 13 des 16 environnements avec PAI et sans les actions incertaines, le nombre de classeurs surgénéraux de BEACS ne présente pas de différences significatives avec le nombre de classeurs surgénéraux de PEPACS avec la copie spécifiée (*cf.* figure 7.16). Pour les 3 environnements restants :

- BEACS construit moins de classeurs surgénéraux que PEPACS sous toutes ses versions dans *Maze10* ;
- BEACS construit moins de classeurs surgénéraux que PEPACS sans la copie spécifiée et plus de classeurs surgénéraux que PEPACS avec la copie spécifiée dans *MazeE2* ;
- BEACS construit plus de classeurs surgénéraux que PEPACS sous toutes ses versions dans *Littman57*.

Dans le pire des cas, BEACS développe 31 classeurs surgénéraux dans *MazeE2* sur une population moyenne de 1351 classeurs (soit un ratio de 0.03%) et il développe en moyenne 12.7 classeurs surgénéraux dans cet environnement (soit un ratio de 0.01%). BEACS arrive alors à prévenir la création de classeurs surgénéraux lors d'un apprentissage.

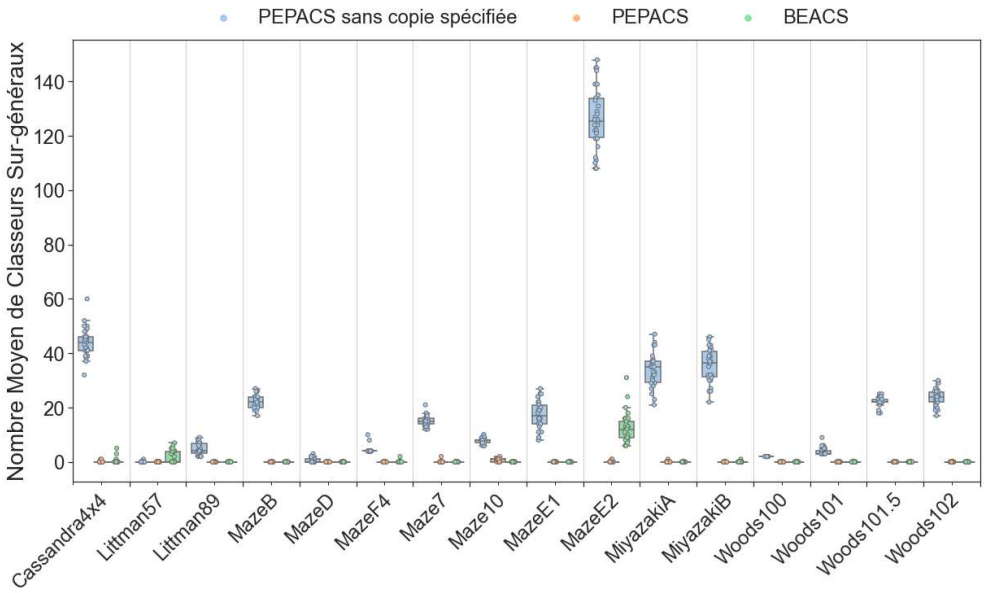


FIGURE 7.16 – Nombres moyens de classeurs surgénéraux à EPE de BEACS et à PEP de PEPACS avec et sans la spécification introduite dans la copie de classeurs, dans les environnements avec PAI et sans les actions incertaines.

Comment le nouvel opérateur de mutation influence-t-il la construction des représentations environnementales ?

BEACS est d'abord comparé à BEACS-IndNomuT afin d'isoler l'effet de l'utilisation de la trace de mutation des classeurs dans l'opérateur de mutation de la généralisation génétique. BEACS est ensuite comparé à BEACS-DmuT pour isoler l'effet d'une généralisation directe des classeurs comportementaux sur les populations de classeurs. Nous comparerons enfin BEACS et BEACS-DNomuT puisque ce dernier utilise un opérateur de mutation où tous les classeurs peuvent être directement généralisés et où les traces de mutation ne sont pas utilisées.

Sans les actions incertaines, les représentations environnementales de BEACS et BEACS-IndNomuT sont globalement équivalentes : les ratios de connaissance sont équivalents dans 22 des 23 environnements ; les EMPA sont équivalentes dans 21 des 23 environnements ; les tailles des populations de classeurs sont équivalentes dans 18 labyrinthes, sinon celles de BEACS sont plus petites ; les spécificités moyennes des classeurs réguliers sont équivalentes dans 19 des 23 environnements et celles des classeurs comportementaux dans 14 des 16 labyrinthes où ces systèmes sont confrontés au PAI ; autant de classeurs à EPE sont construits par BEACS et BEACS-IndNomuT dans 14 des 16 environnements où ces systèmes sont confrontés au PAI. Les politiques décisionnelles développées par BEACS et BEACS-IndNomuT sont elles aussi équivalentes sur tout le banc de test. Toutes ces métriques sont illustrées par des figures fournies dans l'annexe D.4.
Avec les actions incertaines, les représentations environnementales de BEACS sont plus complètes, compactes et cohérentes avec l'environnement que celles de BEACS-IndNomuT, mais elles sont aussi plus spécifiques : les ratios de connaissance sont équivalents dans 15 des 23 environnements et ceux de BEACS sont plus grands dans les 8 labyrinthes restants ; les EMPA de BEACS sont les plus petites dans 21 des 23 environnements ; les tailles des populations de classeurs de BEACS sont les plus petites sur l'ensemble du banc de test ; les spécificités moyennes des classeurs réguliers sont les plus grandes sur l'ensemble du banc de test et celles des classeurs comportementaux aussi dans 15 des 16 labyrinthes où ces systèmes sont confrontés au PAI. Les politiques décisionnelles développées par BEACS et BEACS-IndNomuT sont globalement équivalentes où les mesures ne présentent pas de différences signi-

ficatives pour 14 environnements, celles de BEACS étant plus grandes pour 4 environnements et plus petites pour les 5 derniers labyrinthes. Toutes ces métriques sont illustrées par des figures fournies dans l'annexe D.5.

Sans les actions incertaines, les représentations environnementales de BEACS et BEACS-DmuT sont de complétude et de cohérence équivalentes : les ratios de connaissance sont équivalents dans 19 des 23 environnements ; les EMPA sont équivalentes dans 12 des 23 environnements, celles de BEACS étant plus petites dans 6 autres environnements et plus grandes dans les 5 labyrinthes restants. En revanche, les représentations environnementales construites par BEACS sont les plus compactes, les classeurs réguliers les plus généraux, les classeurs comportementaux les plus spécifiques et ont le moins de classeurs à EPE surgénéraux : les tailles des populations de classeurs de BEACS sont plus petites dans 15 labyrinthes, sinon elles sont équivalentes ; les spécificités moyennes des classeurs réguliers de BEACS sont équivalentes dans 10 des 23 environnements, sinon elles sont les plus faibles ; les spécificités moyennes des classeurs comportementaux de BEACS sont les plus élevées dans 10 des 16 labyrinthes où ces systèmes sont confrontés au PAI, les plus petites dans 4 labyrinthes et équivalentes dans les 2 restants ; autant de classeurs à EPE sont construits par BEACS et BEACS-DmuT dans 6 des 16 environnements où ces systèmes sont confrontés au PAI, sinon BEACS en construit le moins pour 9 environnements. Enfin, BEACS est plus rapide à atteindre la sortie des labyrinthes que BEACS-DmuT dans 12 des 23 labyrinthes, ces deux systèmes étant aussi rapides dans 10 labyrinthes. Toutes ces métriques sont illustrées par des figures fournies dans l'annexe D.6.

Avec les actions incertaines, les représentations environnementales de BEACS sont plus complètes, compactes et cohérentes avec l'environnement que celles de BEACS-IndNomuT, mais elles sont aussi plus spécifiques : les ratios de connaissance sont équivalents dans 13 des 23 environnements et ceux de BEACS sont plus grands dans les 10 labyrinthes restants ; les EMPA de BEACS sont les plus petites dans 17 des 23 environnements, étant sinon équivalentes dans 5 labyrinthes ; les tailles des populations de classeurs de BEACS sont les plus petites pour 16 environnements, étant équivalentes dans les 7 restants ; les spécificités moyennes des classeurs réguliers de BEACS sont les plus grandes pour 10 des 23 labyrinthes ou équivalentes dans 11 autres environnements et celles des classeurs comportementaux aussi dans 15 des

16 labyrinthes où ces systèmes sont confrontés au PAI. Les politiques décisionnelles développées par BEACS et BEACS-IndNomuT sont équivalentes pour 14 environnements où les mesures ne présentent pas de différences significatives, celles de BEACS étant plus rapide à atteindre la sortie pour 8 environnements. Toutes ces métriques sont illustrées par des figures fournies dans l'annexe D.7.

En résumé, l'utilisation des traces de mutation des classeurs ou la généralisation indirecte des classeurs comportementaux par rapport à une généralisation directe de ces classeurs permettent dans des environnements où les actions sont incertaines :

- une construction des représentations environnementales plus complètes ;
- une description des transitions environnementales plus cohérentes avec les environnements d'apprentissage ;
- une construction des représentations environnementales plus compactes ;
- une construction des représentations environnementales plus spécifiques, où tous les classeurs des représentations sont moins généraux, qu'ils contiennent ou non une séquence comportementale.

Les effets de la généralisation indirecte des classeurs comportementaux se retrouvent aussi dans des environnements où le résultat des actions est certain, avec une différence sur la spécificité des classeurs réguliers. La même observation entre BACS et ACS 2 est réalisable entre BEACS et BEACS-DmuT où les classeurs réguliers de BACS, respectivement BEACS, sont plus généraux que ceux d' ACS 2, respectivement BEACS-DmuT.
La généralisation indirecte des classeurs comportementaux a aussi un effet supplémentaire par rapport à une généralisation directe de ces classeurs : elle permet de mettre en place des politiques décisionnelles plus efficaces pour atteindre la sortie des labyrinthes, indépendamment de la présence des actions incertaines.

Une comparaison entre BEACS et BEACS-DNomuT montre que les effets d'une utilisation conjointe des traces de mutation des classeurs et de la généralisation indirecte des classeurs comportementaux se cumulent.

Sans les actions incertaines, les représentations environnementales de BEACS

et BEACS-DNomuT sont globalement de complétude et de cohérence équivalentes : les ratios de connaissance sont équivalents dans 17 des 23 environnements (*cf.* figure 7.17) ; les EMPA sont équivalentes dans 12 des 23 environnements, celles de BEACS étant plus petites dans 6 autres environnements et plus grandes dans les 5 labyrinthes restants (*cf.* figure 7.18). En revanche, les représentations environnementales construites par BEACS sont les plus compactes, les classeurs réguliers les plus généraux, les classeurs comportementaux les plus spécifiques et ont le moins de classeurs à EPE surgénéraux : les tailles des populations de classeurs de BEACS sont plus petites dans 15 labyrinthes, sinon elles sont équivalentes (*cf.* figure 7.19) ; les spécificités moyennes des classeurs réguliers de BEACS sont équivalentes dans 10 des 23 environnements, sinon elles sont les plus faibles dans 12 environnements (*cf.* figure 7.20) ; les spécificités moyennes des classeurs comportementaux de BEACS sont les plus élevées dans 10 des 16 labyrinthes où ces systèmes sont confrontés au PAI, les plus petites dans 3 labyrinthes et équivalentes dans les 3 restants (*cf.* figure 7.21) ; autant de classeurs à EPE sont construits par BEACS et BEACS-DNomuT dans 7 des 16 environnements où ces systèmes sont confrontés au PAI, sinon BEACS en construit le moins pour 9 environnements (*cf.* figure 7.22). Enfin, BEACS est plus rapide à atteindre la sortie des labyrinthes que BEACS-DNomuT dans 12 des 23 labyrinthes, ces deux systèmes étant aussi rapides dans les 11 labyrinthes restants (*cf.* figure 7.23).

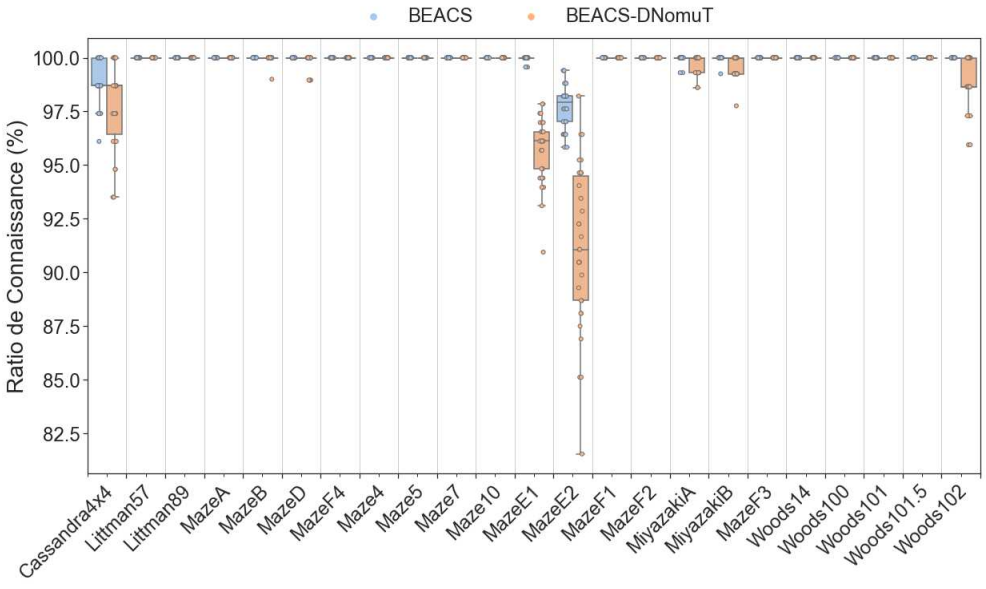


FIGURE 7.17 – Ratios de connaissance de BEACS et BEACS-DNomuT dans tous les environnements sans les actions incertaines.

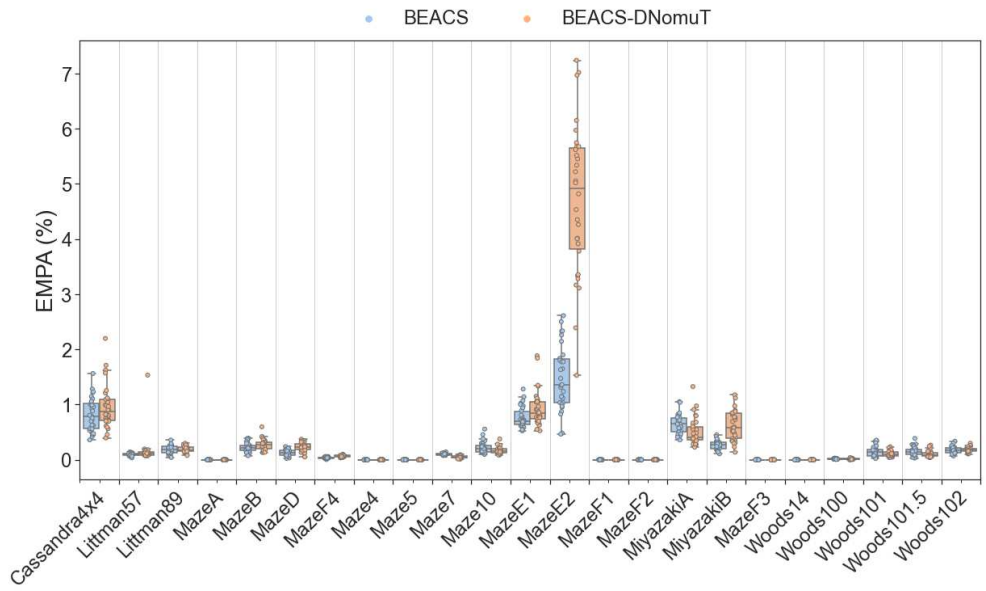


FIGURE 7.18 – Erreurs Moyennes sur la Prédiction de l'Anticipation de BEACS et BEACS-DNomuT dans tous les environnements sans les actions incertaines.

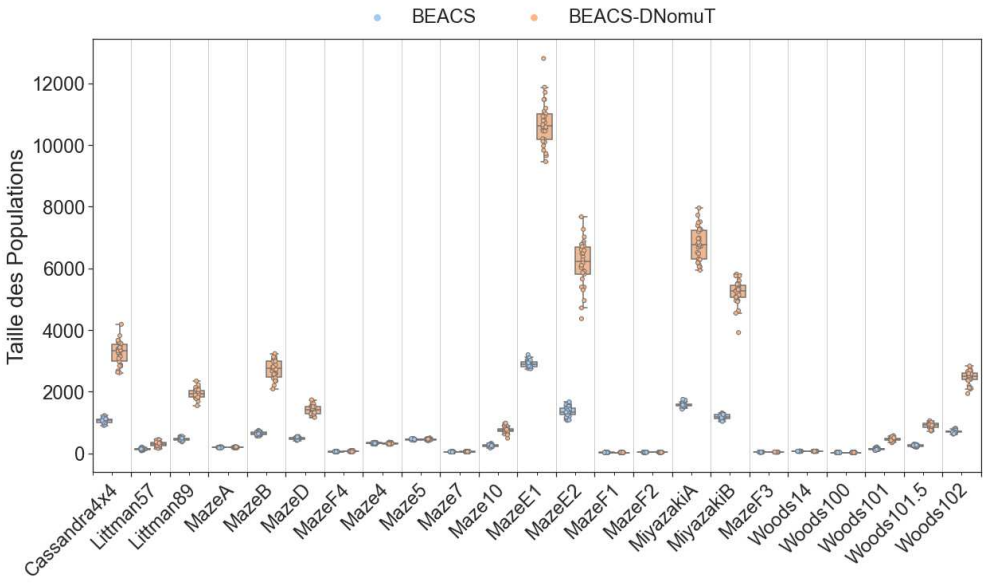


FIGURE 7.19 – Tailles des populations de classeurs de BEACS et BEACS-DNomuT dans tous les environnements sans les actions incertaines.

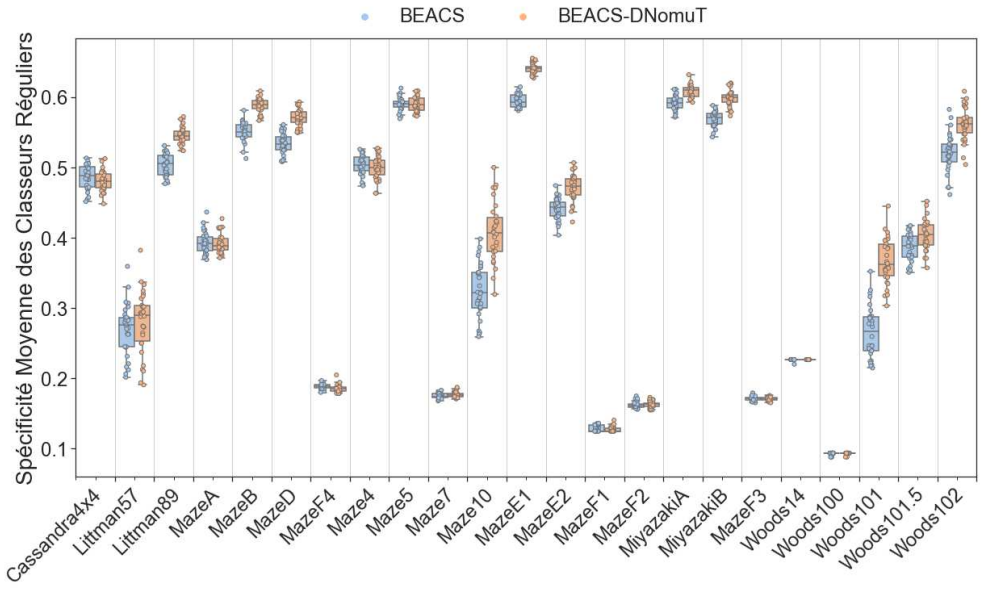


FIGURE 7.20 – Spécificités moyennes des classeurs réguliers fiables de BEACS et BEACS-DNomuT dans tous les environnements sans les actions incertaines.

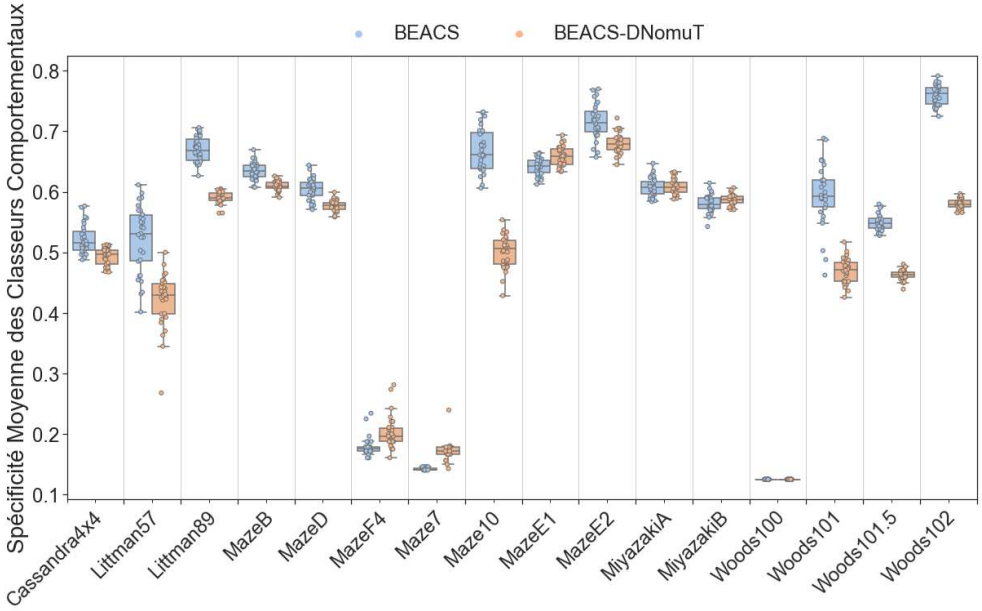


FIGURE 7.21 – Spécificités moyennes des classeurs comportementaux fiables de BEACS et BEACS-DNomuT dans tous les environnements à PAI sans les actions incertaines.

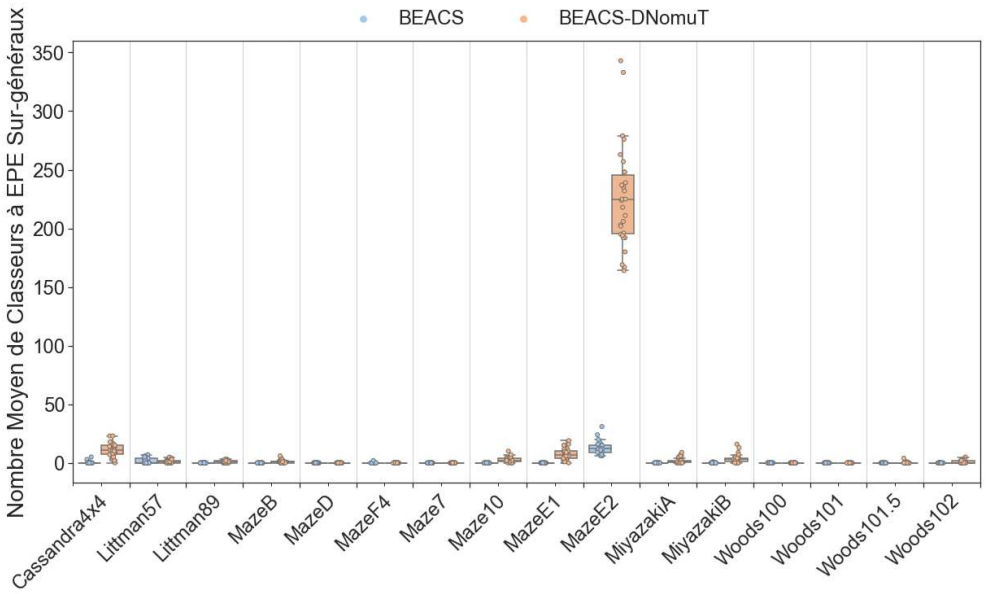


FIGURE 7.22 – Nombres moyens de classeurs surgénéraux à EPE de BEACS et BEACS-DNomuT dans les environnements avec PAI et sans les actions incertaines.

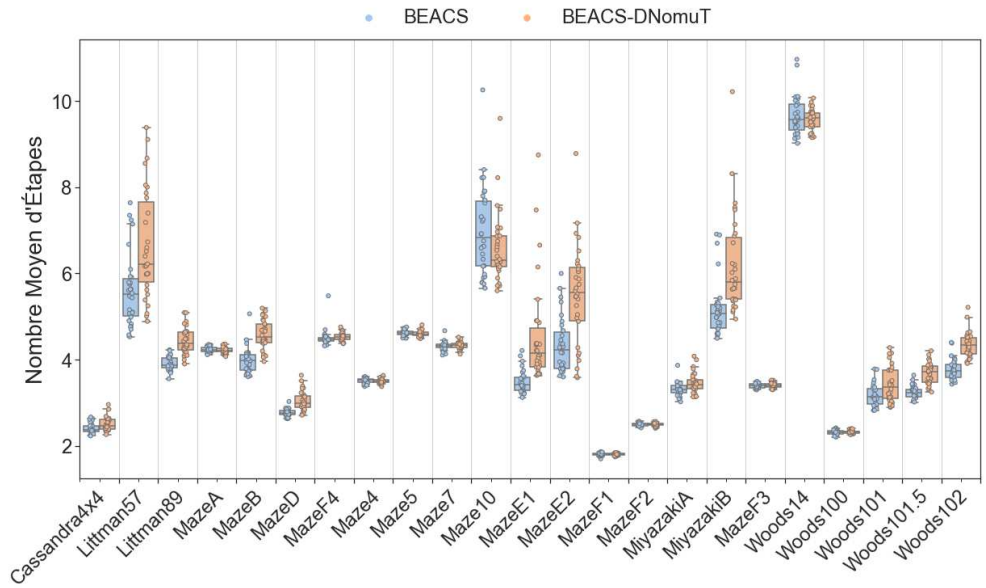


FIGURE 7.23 – Nombres moyens d'étapes de BEACS et BEACS-DNomuT pour atteindre la sortie des labyrinthes sans les actions incertaines.

Avec les actions incertaines, les représentations environnementales de BEACS sont plus complètes, compactes et cohérentes avec l'environnement que celles de BEACS-DNomuT, mais elles sont aussi plus spécifiques : les ratios de connaissance sont équivalents dans 11 des 23 environnements et ceux de BEACS sont plus grands dans les 12 labyrinthes restants (*cf.* figure 7.24) ; les EMPA de BEACS sont les plus petites dans 22 des 23 environnements (*cf.* figure 7.25) ; les tailles des populations de classeurs de BEACS sont les plus petites pour 22 des 23 environnements (*cf.* figure 7.26) ; les spécificités moyennes des classeurs réguliers de BEACS sont les plus grandes pour 22 des 23 labyrinthes (*cf.* figure 7.27) et celles des classeurs comportementaux aussi dans les 16 labyrinthes où ces systèmes sont confrontés au PAI (*cf.* figure 7.28). Les politiques décisionnelles développées par BEACS et BEACS-IndNomuT sont équivalentes pour 9 environnements où les mesures ne présentent pas de différences significatives, celles de BEACS étant plus rapides à atteindre la sortie pour 13 environnements (*cf.* figure 7.29).

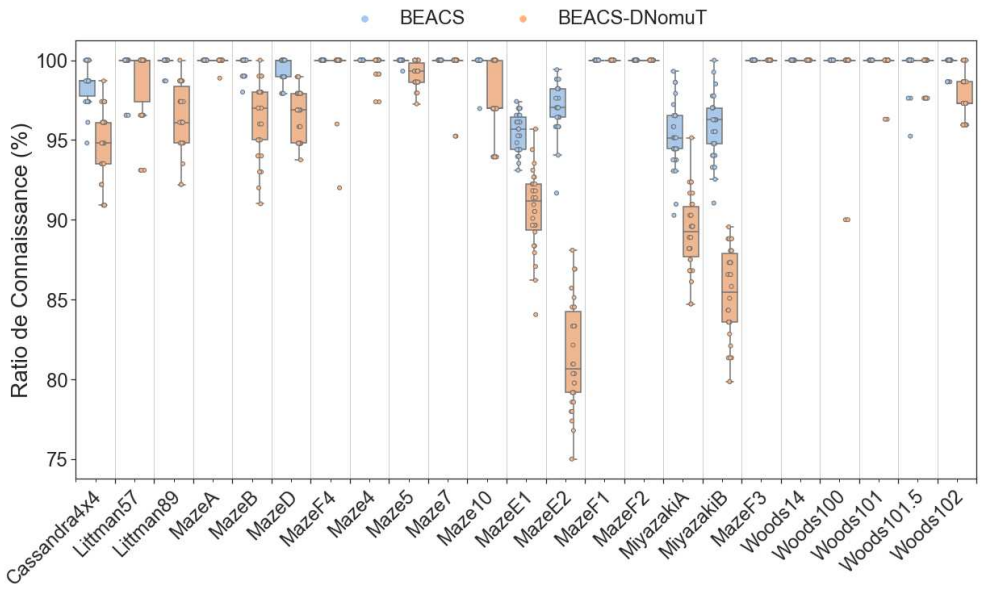


FIGURE 7.24 – Ratios de connaissance de BEACS et BEACS-DNomuT dans tous les environnements avec les actions incertaines.

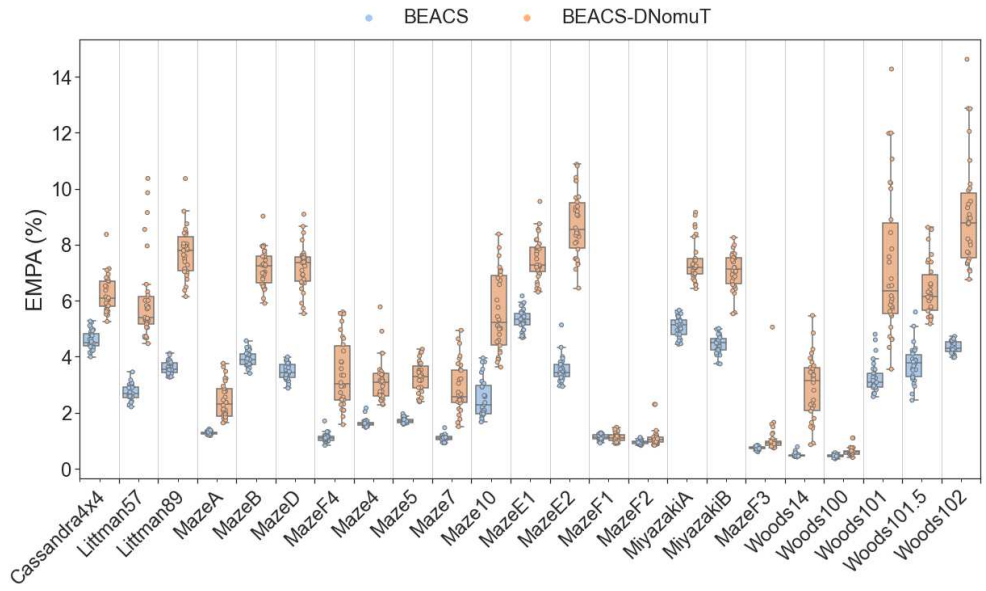


FIGURE 7.25 – Erreurs Moyennes sur la Prédiction de l'Anticipation de BEACS et BEACS-DNomuT dans tous les environnements avec les actions incertaines.

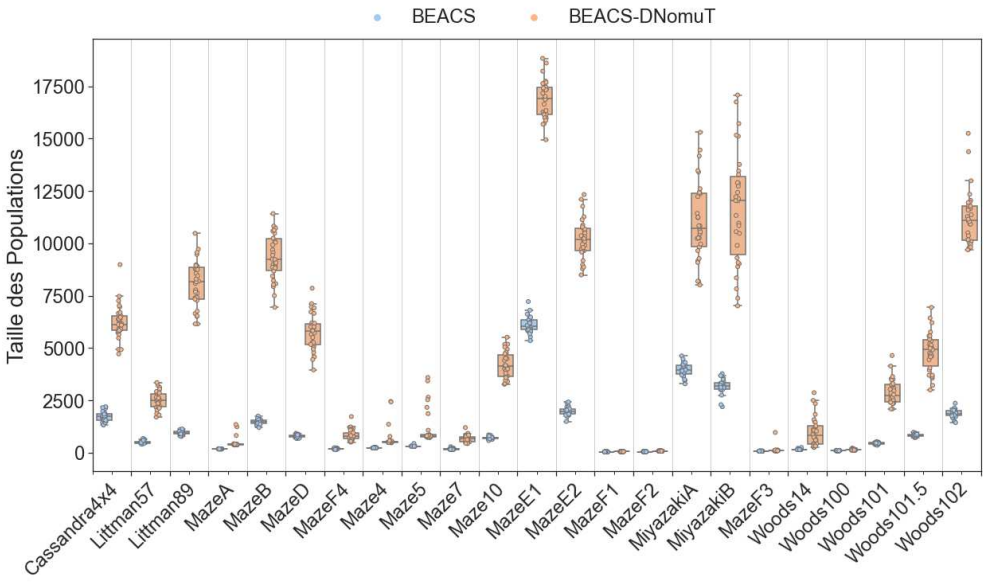


FIGURE 7.26 – Tailles des populations de classeurs de BEACS et BEACS-DNomuT dans tous les environnements avec les actions incertaines.

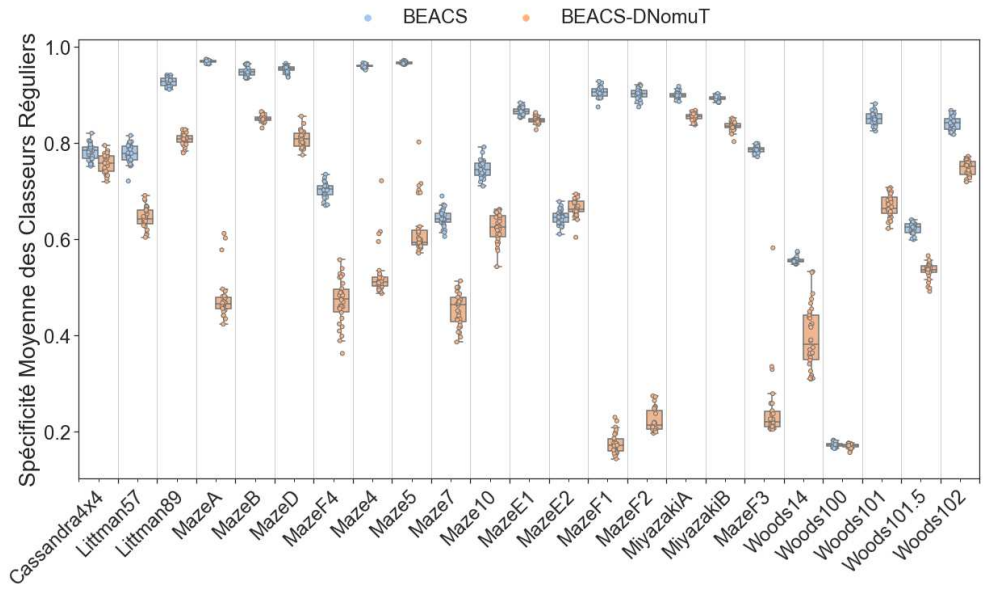


FIGURE 7.27 – Spécificités moyennes des classeurs réguliers fiables de BEACS et BEACS-DNomuT dans tous les environnements avec les actions incertaines.

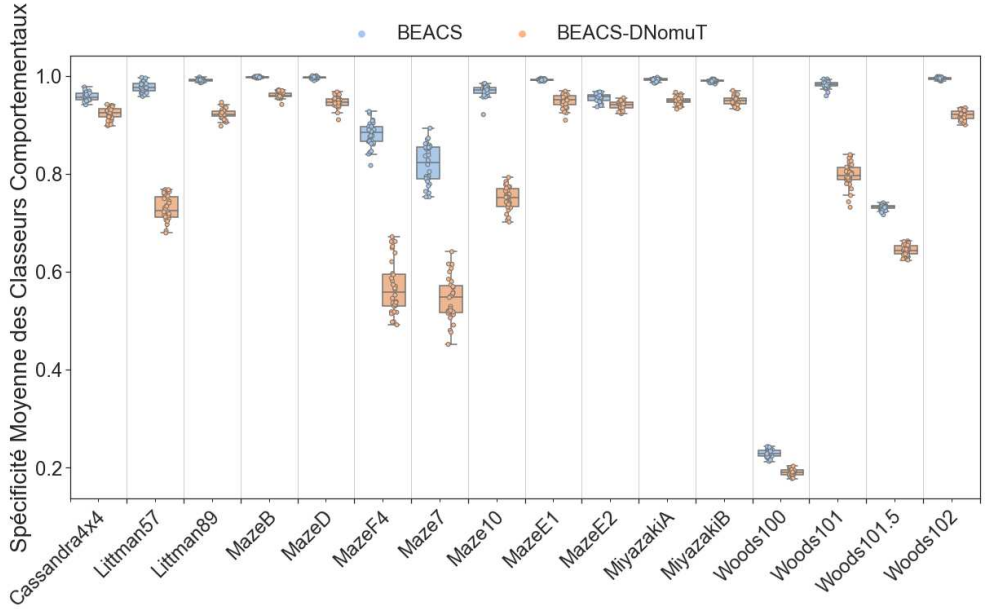


FIGURE 7.28 – Spécificités moyennes des classeurs comportementaux fiables de BEACS et BEACS-DNomuT dans tous les environnements à PAI avec les actions incertaines.

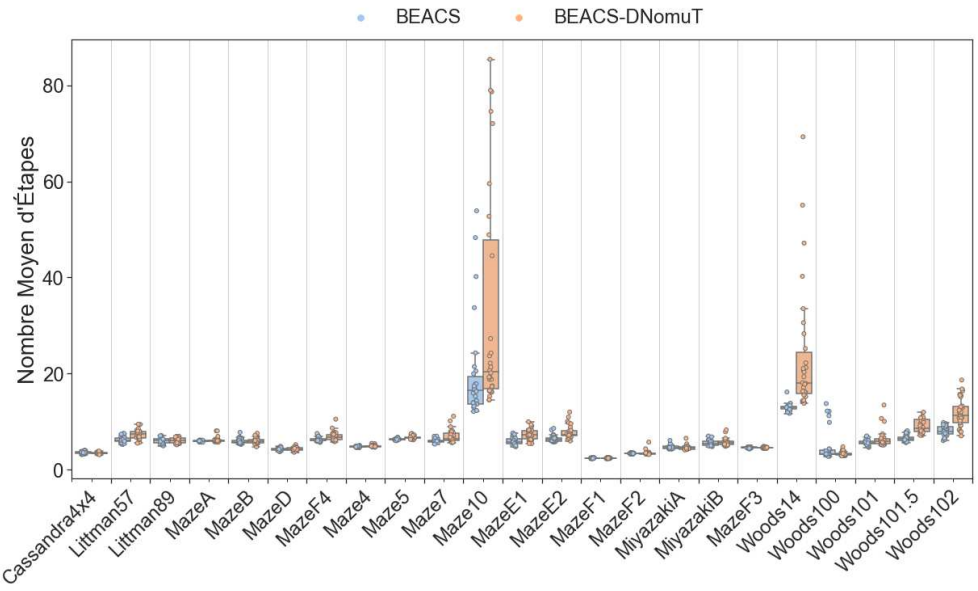


FIGURE 7.29 – Nombres moyens d'étapes de BEACS et BEACS-DNomuT pour atteindre la sortie des labyrinthes avec les actions incertaines.

BEACS est-il capable de détecter les situations où il confronté au problème d'aliasing perceptif quand le résultat des actions est incertain ?

BEACS doit détecter sur l'ensemble des labyrinthes et des apprentissages à réaliser 1590 situations où il est confronté au PAI et 9810 situations qui ne sont pas liées au PAI dans toutes les expériences. BEACS a correctement identifié 1494 des 1590 situations où il est confronté au PAI quand le résultat des actions est incertain, ne détectant pas les 96 situations restantes. Dans le même temps, BEACS a gardé dans sa mémoire des perceptions liées au PAI 26 situations non liées au PAI, considérant ainsi que 9784 situations ne sont effectivement pas liées au PAI quand le résultat des actions est incertain. La *balanced accuracy* du mécanisme de détection du PAI est donc d'environ 96.85%. Sans les actions incertaines, BEACS a correctement identifié 1499 des 1590 situations où il est confronté au PAI, ne détectant pas les 91 situations restantes. Dans le même temps, BEACS a gardé dans sa mémoire des perceptions liées au PAI aucune situation non liée au PAI, considérant ainsi que 9810 situations ne sont effectivement pas liées au PAI. La *balanced accuracy* du mécanisme de détection du PAI sans les actions incertaines est donc d'environ 97.14%.

Comment le couplage des EPE et des BSeq influencent-elle la mise en place de politiques de décision dans des environnements incertains ?

Les figures 7.30 et 7.31 illustrent respectivement les nombres moyens d'étapes nécessaires pour atteindre les sorties des labyrinthes avec PAI sans les actions incertaines et celles de tous les labyrinthes avec les actions incertaines.
Sans les actions incertaines, BEACS est le plus rapide à atteindre la sortie de 11 des 16 labyrinthes dont : 2 où BEACS est aussi rapide que PEPACS et 3 où BEACS est aussi rapide que BACS. PEPACS est le plus rapide à atteindre la sortie des labyrinthes dans les 5 environnements restants, où BEACS est plus rapide que BACS dans 4 de ces 5 labyrinthes ou est aussi rapide que BACS dans le dernier.
Avec les actions incertaines, BEACS est le plus rapide à atteindre la sortie de 15 des 23 labyrinthes dont : 5 où BEACS est aussi rapide que PEPACS, 3 où BEACS est aussi rapide que BACS et un dernier où les trois systèmes sont aussi rapides. PEPACS est le plus rapide à atteindre la sortie des labyrinthes dans les 8 environnements restants où : BEACS est plus rapide que BACS dans 4 de ces 8 labyrinthes ; BEACS est aussi rapide que BACS dans 3 de ces 8 labyrinthes ; BEACS est plus lent que BACS dans le dernier environnement.

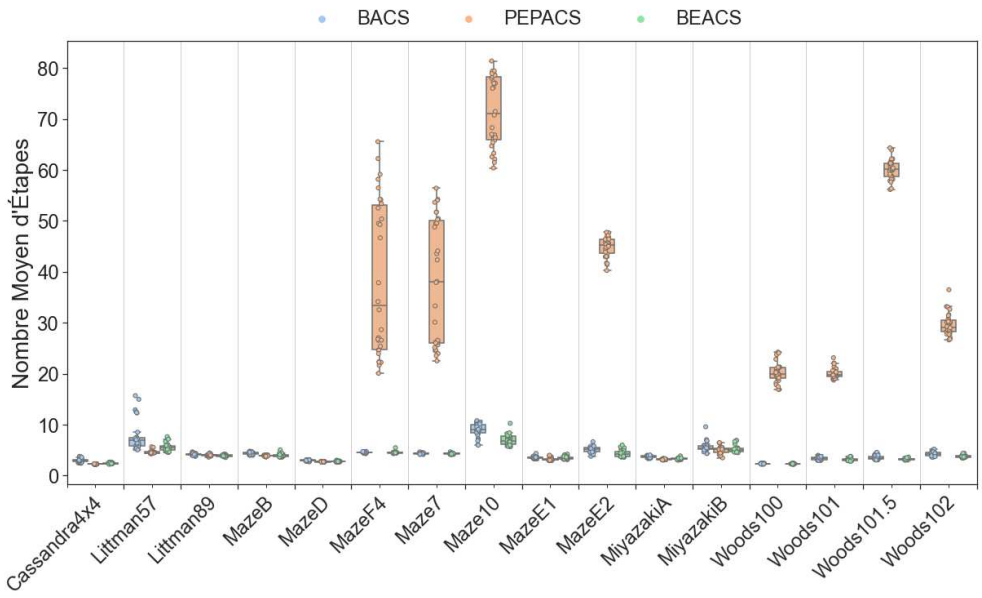


FIGURE 7.30 – Nombres moyens d'étapes de BACS, PEPACS et BEACS pour atteindre la sortie des labyrinthes avec PAI sans les actions incertaines.

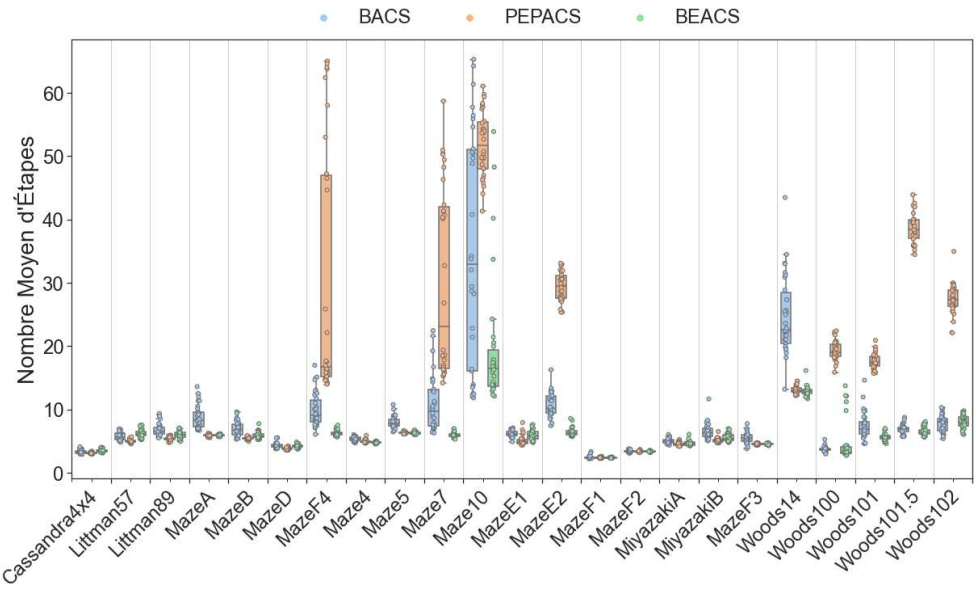


FIGURE 7.31 – Nombres moyens d'étapes de BACS, PEPACS et BEACS pour atteindre la sortie des labyrinthes avec les actions incertaines.

Est-ce que le *Double Q-Learning* permet à BEACS de se doter de politiques décisionnelles plus efficaces ?

Les figures 7.32 et 7.33 illustrent respectivement les nombres moyens d'étapes nécessaires pour atteindre les sorties de tous les labyrinthes sans les actions incertaines et celles de tous les labyrinthes avec les actions incertaines pour BEACS-Q et BEACS. Sans les actions incertaines, BEACS est plus rapide à atteindre la sortie que BEACS-Q dans 7 labyrinthes et aussi rapide que BEACS-Q dans les 16 labyrinthes restants. Avec les actions incertaines, BEACS est plus rapide à atteindre la sortie que BEACS-Q dans 8 labyrinthes, aussi rapide que BEACS-Q dans 14 labyrinthes et plus lent dans le dernier.
Les représentations environnementales ne sont globalement pas impactées par le processus de rétribution de BEACS. Nous fournissons en annexe D.8 les figures illustrant toutes les métriques permettant de comparer les représentations environnementales de BEACS-Q et BEACS.

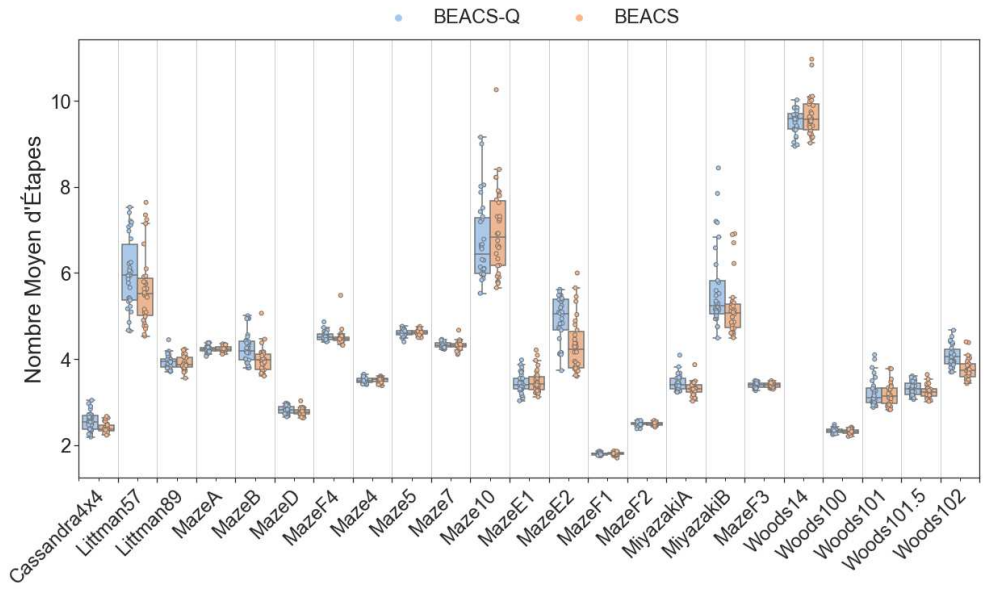


FIGURE 7.32 – Nombres moyens d'étapes de BEACS-Q et BEACS pour atteindre la sortie des labyrinthes sans les actions incertaines.

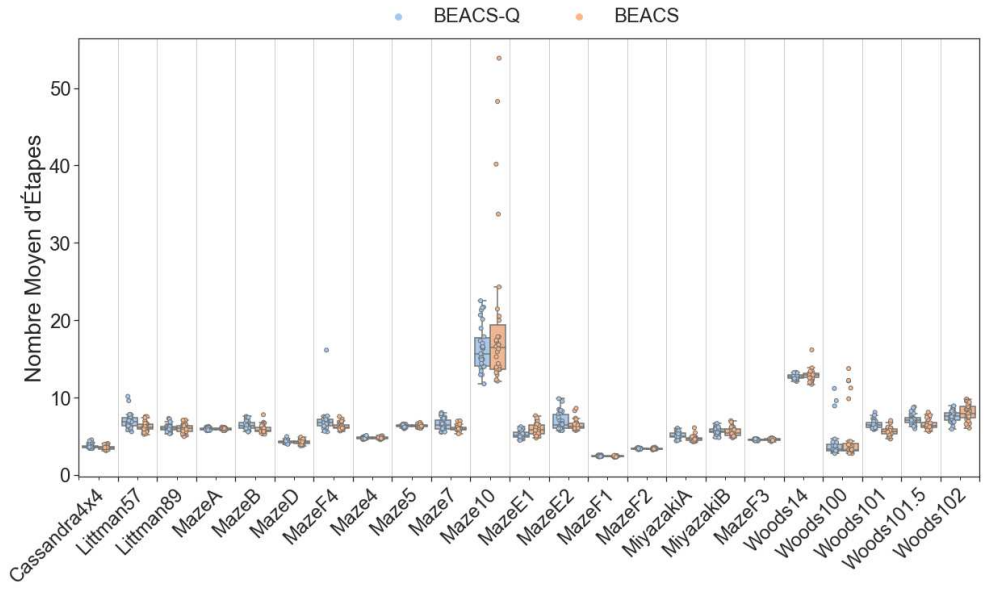


FIGURE 7.33 – Nombres moyens d'étapes de BEACS-Q et BEACS pour atteindre la sortie des labyrinthes avec les actions incertaines.

7.5 Discussion

7.5.1 Une détection incomplète du PAI

BEACS est le premier système de classeurs à anticipation parvenant à détecter le problème d'aliasing perceptif des autres formes d'incertitude environnementale. Cependant, une analyse des situations où le PAI n'a pas été détecté révèle une limite du mécanisme de détection du PAI et aussi du mécanisme de détection d'incertitude : les situations non considérées comme liées au PAI ne sont pas détectées comme étant incertaines. Cela s'explique par le fait que des situations réellement distinctes dans un environnement peuvent être observées avec une même perception et les transitions environnementales depuis ces situations distinctes peuvent également être identiques, rendant impossible leur détection uniquement sur la base des *Anticipations* des classeurs. La figure 7.34 donne un exemple de situations distinctes liées au PAI qui ne sont pas détectées par BEACS (et même par BACS et PEPACS) dans le labyrinthe *Littman57*. Concernant les autres situations liées au PAI qui n'ont pas été détectées, BEACS n'est pas parvenu à construire ou trouver dans sa population les classeurs nécessaires à leur détection : BEACS a donc pu manquer de temps d'apprentissage pour pouvoir pleinement expérimenter toutes les actions en toutes les situations, sachant que toutes les situations environnementales n'ont pas les mêmes probabilités d'être atteintes dans les environnements du banc de test.

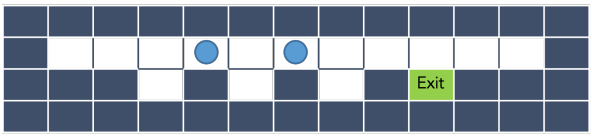


FIGURE 7.34 – Situations environnementales liées au PAI non identifiables pour BEACS, BACS ou PEPACS dans *Littman57*.
Les deux situations liées aux points bleus sont perçues avec une même observation bien qu'elles soient distinctes. À partir de ces situations, les mêmes actions ont les mêmes conséquences sur Littman57 *bien que des situations différentes soient atteintes. BEACS, comme BACS ou PEPACS, n'est pas capable de détecter le PAI en ces deux situations.*

Par ailleurs, le mécanisme de détection du PAI de BEACS a considéré 26 situations comme étant liées au PAI sur l'ensemble du banc de test avec les

actions incertaines, alors qu'il n'en est rien. BEACS fait évoluer ses classeurs afin d'apprendre à anticiper toutes les conséquences de ses actions sur son environnement. Durant son apprentissage, BEACS emploie son processus de généralisation génétique pour essayer d'utiliser des classeurs en de nouvelles situations. Même si BEACS limite la surgénéralisation des classeurs (*cf.* figure 7.16), il reste possible qu'un classeur surgénéral puisse être maintenu dans la population et soit utilisé lors de la détection du PAI : un nombre erroné de situations à anticiper depuis une situation non liée au PAI est alors calculé par BEACS ce qui fausse la détection.

L'objectif premier du mécanisme de détection du PAI étant atteint, l'hypothèse sur laquelle est fondée la détection est donc validée : BEACS est en mesure de s'appuyer sur les situations décrites dans les EPE des classeurs pour identifier les situations liées au PAI. Une analyse plus approfondie du mécanisme pourrait faire apparaître des axes d'amélioration avec par exemple : l'introduction de nouveaux critères pour extraire les classeurs les plus expérimentés dédiés à la détection du PAI ; une analyse détaillée de l'influence du seuil d'application θ_{pai} sur la détection du PAI et sur la construction des représentations environnementales et des politiques décisionnelles.

7.5.2 Des représentations complètes et cohérentes

BEACS est capable de construire des représentations complètes des environnements durant son apprentissage, même s'il atteint des ratios de connaissance légèrement plus faibles que ceux de PEPACS sur des labyrinthes du banc de test (*cf.* figures 7.6 et 7.7). Ces différences entre les ratios de connaissance de BEACS et de PEPACS sont dues à l'emploi des séquences comportementales et à la construction des EPE.

D'une part, les classeurs à BSeq ne permettent pas de décrire directement les transitions environnementales à partir des situations où le PAI est présent ou lorsque le résultat des actions est incertain. Or, la mise en place de classeurs à BSeq nécessite du temps d'apprentissage, qui est donc du temps en moins pour que BEACS, relativement à PEPACS, puisse mettre en place les classeurs à EPE qui décriront avec fiabilité toutes les transitions environnementales.

D'autre part, pour un ensemble de situations à apprendre à anticiper, la représentation des EPE peut induire plus de temps d'apprentissage que la représentation des PEP. Les EPE décrivent complètement chacune des situations anticipées et BEACS doit en conséquence les apprendre individuellement dans les classeurs à EPE. En revanche, les PEP permettent à PEPACS d'apprendre à anticiper plusieurs situations environnementales. Par exemple, à partir de deux situations environnementales où pour chacune deux changements sont observés sur deux attributs perceptifs, les PEP vont décrire 4 situations anticipables dont 2 nouvelles situations potentiellement inexistantes ou qui seraient effectivement à apprendre. La combinaison de plusieurs PEP accélère alors l'acquisition d'un modèle de l'environnement relativement au EPE, mais au prix de l'explicabilité des *Anticipations* puisqu'il n'est pas possible de savoir quelles ont été les situations réellement anticipées à partir des PEP.
Cet avantage des PEP sur les EPE est d'autant plus fort que toutes les transitions environnementales dans les labyrinthes du banc de test n'ont pas les mêmes chances d'être explorées : la combinaison de plusieurs PEP facilite l'acquisition d'une représentation environnementale complète par rapport aux EPE si elle permet d'apprendre par effet de bord à anticiper dans les classeurs à PEP les situations liées aux transitions ayant le moins de chance de se produire.

En plus de construire des représentations complètes des environnements, BEACS est capable de fidèlement apprendre les probabilités d'anticiper chacun des changements perceptifs contrairement à PEPACS (*cf.* figures 7.8 et 7.9), indépendamment de l'incertitude environnementale. BEACS calcule les probabilités des EPE à partir de son expérience avec l'environnement en dénombrant les situations qu'il a correctement anticipées : plus le nombre d'étapes d'apprentissage est grand, plus les probabilités sont précises. Il est possible de vérifier que les probabilités issues des EPE convergent au fur et à mesure de l'apprentissage de BEACS contrairement au PEP de PEPACS avec la même simulation de l'apprentissage d'une transition environnementale de *Woods100* (*cf.* figure 6.7). Nous avions alors observé que les PEP de PEPACS oscillent sans converger quand les actions sont incertaines (*cf.* figure 6.8). La figure 7.35 illustre alors l'évolution de l'EMPA associée à la transition étudiée du labyrinthe *Woods100* avec des EPE : l'EMPA mesurée avec les EPE

décroissent effectivement.

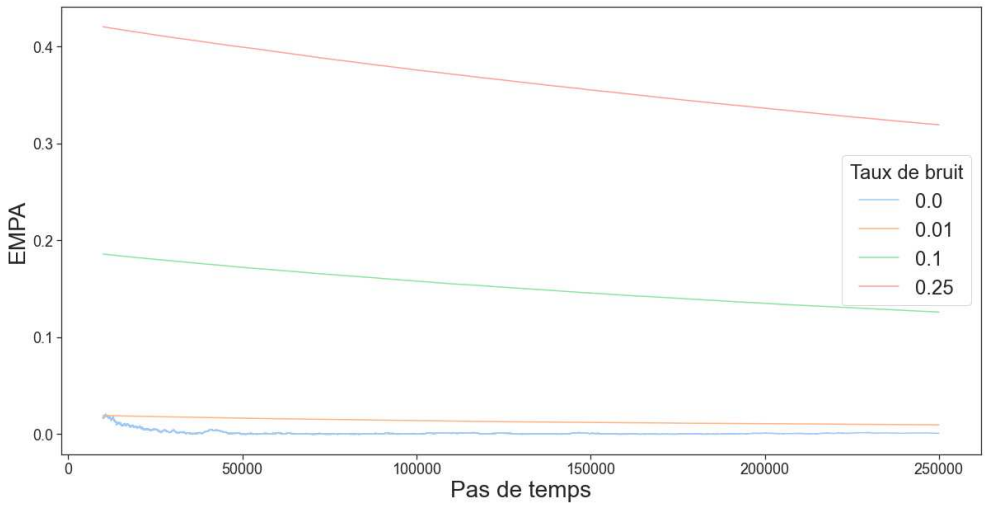


FIGURE 7.35 – Évolution de l'Erreur Moyenne sur la Prédiction de l'Anticipation de la transition environnementale liée au PAI dans le labyrinthe *Woods100* avec des EPE.

7.5.3 Des informations explicatives plus nombreuses

Les classeurs de BEACS fournissent des informations explicatives plus nombreuses. Étant capable de construire des représentations environnementales complètes et cohérentes avec les propriétés des environnements, les *Anticipations* des classeurs de BEACS fournissent donc de nouvelles perspectives sur l'explicabilité des classeurs : les classeurs de BEACS peuvent être chaînés de manière fiable pour retracer les causes possibles d'un événement particulier. Par ailleurs, ils intègrent de nouvelles informations permettant de lier l'utilisation de classeurs avec différentes formes d'incertitude, en plus des informations déjà présentes :

- les *marques PAI* permettent de savoir quelles situations doivent être traversées par l'usage des séquences comportementales car liées au PAI ;
- les *marques incertaines* des classeurs indiquent les situations où BEACS estime que plusieurs situations à anticiper sont à apprendre ;
- la *mémoire PAIMemory* permet de savoir quelles situations sont considérées par BEACS comme liées au problème d'aliasing perceptif ;
- les *marques* permettent de savoir pour quelles situations les classeurs

n'ont pas été capables de correctement anticiper les changements perceptifs suite à la réalisation d'une action décrite dans leur *Conséquent* ;
- les *qualités* indiquent la capacité d'un classeur à réussir ou échouer à anticiper des changements perceptifs ;
- les *récompenses* fournissent une information sur l'adéquation d'une action par rapport à d'autres actions pour résoudre une tâche d'apprentissage ;
- les *récompenses immédiates* des classeurs fournissent une information sur le gain reçu directement de l'environnement en lien avec la réalisation d'une action dans des situations environnementales ;

La présence de toutes ces informations induit un usage plus important de la mémoire d'un ordinateur qui n'a pas été mesuré, la question des performances liées à l'utilisation des processeurs ou de la mémoire n'ayant pas été traitée.

L'incertitude environnementale accroît également les difficultés à construire des représentations environnementales et des politiques décisionnelles qui soient les plus compactes : plus les environnements sont complexes à cause de propriétés environnementales stochastiques ou à cause du fléau de la dimension, plus les populations de classeurs sont volumineuses. Or, il peut être plus facile d'expliquer le comportement d'un système de classeurs si ses populations sont les plus compactes. Chercher à réduire les tailles des populations de classeurs de BEACS peut ainsi être intéressant pour faciliter l'explicabilité du système, quand bien même BEACS est déjà capable de mieux contrôler les représentations qu'il construit par rapport à BACS et PEPACS : les populations de classeurs de BEACS sont globalement plus compactes que celles de BACS et celles de PEPACS dans les environnements avec les actions incertaines alors qu'il manipule des classeurs à EPE et des classeurs à BSeq (*cf.* figure 7.13).

Enfin, la lecture des représentations utilisées par BEACS dans les *Conditions* et les *Anticipations* des classeurs est préservée par le biais de la *trace de mutation* des classeurs. La trace de mutation limite l'ambiguïté dans la lecture des classeurs liée à l'utilisation du symbole générique # et à l'utilisation du mécanisme de généralisation génétique. Cependant, l'emploi de la trace de mutation n'est pas sans conséquence sur la spécificité des représentations environnementales construites par BEACS.

7.5.4 Spécificités des classeurs et généralisation génétique

L'utilisation des traces de mutation dans l'opérateur de mutation de la généralisation génétique induit une pression de spécialisation plus forte que la pression de généralisation dans le but de préserver la façon de lire des classeurs lorsque les actions sont incertaines (*cf.* figures 7.27 et 7.28). Ce résultat est la conséquence des actions incertaines où pour une situation donnée, il y a souvent une action qui n'entraîne pas de changements perceptifs. Puisqu'avec les actions incertaines toutes les actions peuvent être réalisées pour une situation et une action données, la plupart des classeurs doivent apprendre à anticiper une situation inchangée et plusieurs situations où des attributs perceptifs changent. La trace de mutation des classeurs contient alors plusieurs booléens indiquant qu'un attribut perceptif ne peut être généralisé puisque l'*Anticipation* d'un classeur va indiquer que plusieurs attributs perceptifs sont inchangés et changés.

Néanmoins, l'utilisation des traces de mutation semble accélérer la construction des représentations environnementales puisque les représentations ont un meilleur ratio de connaissance (*cf.* figure 7.24), une erreur moyenne sur la prédiction de l'anticipation plus faible (*cf.* figure 7.25) et sont plus compactes (*cf.* figure 7.26). BEACS n'essaie plus alors de généraliser des attributs perceptifs qui selon la trace de mutation ne devraient pas être généralisés, ce qui prévient l'émergence de classeurs pouvant bruiter et ralentir l'acquisition d'un modèle de transition adapté à l'environnement d'apprentissage. Quand les actions sont incertaines, BEACS peut alors plus facilement se concentrer sur les différentes situations à apprendre à anticiper grâce aux traces de mutations.

Nous indiquions avec BEACS et BACS vouloir éviter une surgénéralisation des classeurs comportementaux, indépendamment de l'incertitude environnementale, de telle sorte qu'un classeur contenant une BSeq soit principalement utilisé dans la situation liée au PAI qui a pu déclencher sa construction. Les résultats entre BEACS et BEACS-DmuT, ainsi que BEACS et BEACS-DnomuT dans une moindre mesure, nous montre qu'il est effectivement préférable d'éviter de chercher à généraliser les classeurs à BSeq, sans quoi les politiques décisionnelles perdent en efficacité (*cf.* figures 7.23 et 7.29) et les

représentations environnementales sont de tailles bien plus importantes (jusqu'à un facteur $6\times$ - figures 7.19 et 7.26) et moins adaptées à l'environnement d'apprentissage (figures 7.24 et 7.25). Tenter de généraliser les classeurs à BSeq revient à drastiquement augmenter leur proportion dans les populations de classeurs, d'autant plus si les environnements sont incertains : les classeurs à BSeq sont alors de plus en plus nombreux puisque les séquences comportementales permettent de construire plus de *Conséquents* avec plusieurs actions que de *Conséquents* avec une unique action (n *vs* $n^2 + n^3$ pour $bs_{\max} = 3$). Les classeurs à BSeq occupent alors de plus en plus de ressources calculatoires pour être adaptés à l'environnement d'apprentissage par rapport à des classeurs sans séquences comportementales, ce qui se fait au détriment des représentations environnementales et au détriment des politiques décisionnelles puisque toutes les séquences comportementales ne sont pas utiles pour traiter le PAI.

BEACS parvient alors à construire des classeurs dont la lecture cherche à être la moins ambiguë afin de fournir plus d'éléments explicatifs relatifs au comportement appris, au coût d'une pression de spécification bien plus forte que la pression de généralisation des classeurs. Les résultats obtenus avec les différents opérateurs de mutation de BEACS laissent néanmoins penser qu'il serait possible de réaliser de futurs travaux sur les représentations utilisées dans les classeurs, afin de construire des classeurs intégrant des représentations environnementales et des politiques décisionnelles de qualité au moins équivalentes à celle de BEACS, tout en ayant des représentations aussi générales que celles permises par les différents opérateurs de mutation testés.

7.5.5 Un processus de rétribution perfectible

BEACS avec son processus de rétribution fondé sur le *Double Q-Learning* est capable de se doter de politiques de décision plus efficaces pour résoudre les tâches d'apprentissage que BEACS avec le *Q-Learning* (*cf.* figures 7.32 et 7.33), que BACS et dans une moindre mesure PEPACS (*cf.* figures 7.30 et 7.31). Bien que BEACS soit capable de promouvoir l'utilisation de *Conséquents* avec un nombre réduit d'actions, des efforts peuvent encore être réalisés puisqu'il n'atteint pas les performances de PEPACS dans 8 des 23 environnements lorsque les actions sont incertaines, en particulier dans les environnements

où il est confronté au PAI de type I ou au pseudo-PAI. Plusieurs voies d'amélioration sont possibles pour favoriser l'utilisation des séquences d'actions les plus petites :

- toutes les transitions environnementales ne sont pas uniformément explorées, ce qui peut impacter tant la construction des représentations environnementales que la mise en place des politiques décisionnelles. De nouvelles procédures d'exploration de l'environnement pourraient alors apporter de nouveaux éléments de réflexion ;
- d'autres processus de rétribution s'appuyant sur différents algorithmes de renforcement pourraient être analysés et comparés en détail sur un ensemble varié de problèmes d'apprentissage, ce qui n'a pas encore été réalisé à notre connaissance dans les systèmes de classeurs à anticipation ;
- la construction d'une représentation environnementale pourrait être découplée de la mise en place d'une politique de décision comme pour XACS (Butz and Goldberg, 2003), afin que les politiques de décision ne soient plus conditionnées par les représentations environnementales.

7.6 Synthèse du chapitre

Behavioral Enhanced Anticipatory Classifier System (BEACS) est un ACS 2 dans lequel les prédictions améliorées par les probabilités (PEP) ont d'abord été transformées en des prédictions améliorées par l'expérience (EPE) afin de palier les lacunes des PEP de PEPACS. Les EPE ont ensuite été utilisées pour permettre la détection du PAI qui conditionne l'utilisation des séquences comportementales (BSeq) avec lesquelles les EPE ont été couplées dans BEACS. Les séquences comportementales ont également été retravaillées afin de palier les lacunes qui ont été détectées lors de leur utilisation dans BACS. Ces travaux avaient pour but de coupler dans un unique système de classeurs à anticipation les forces de BACS et les forces de PEPACS, afin de généraliser les résultats atteints avec chacun de ces deux ALCS et d'encore plus accroître l'autonomie et l'explicabilité du système résultant, BEACS.

Les prédictions améliorées par l'expérience permettent à BEACS de construire des classeurs dont les *Anticipations* sont composées de plusieurs situations

anticipées, chacune étant explicitement décrites et dénombrées dans un tableau associatif. Le processus d'apprentissage par anticipation de BEACS a été modifié afin de permettre une adaptation plus souple des classeurs à EPE à leur environnement avec : un nouveau processus de spécification dédiée aux classeurs à EPE ; la suppression du processus de spécification initialement intégré dans la copie des classeurs de PEPACS ; l'adaptation de la spécification des éléments inchangés aux classeurs à EPE ; un nouveau mécanisme de détection de l'incertitude. Le processus de subsomption a également été modifié afin de tenir compte de la nouvelle représentation utilisée dans les *Anticipations* des classeurs de BEACS. BEACS redéfinit aussi de manière plus adaptative ce en quoi consiste un classeur qui anticipe des changements perceptifs en s'appuyant sur les compteurs associés aux situations anticipées décrites dans les EPE.

Les classeurs à séquences comportementales sont conditionnés au tout premier mécanisme de détection du PAI propre aux systèmes de classeurs à anticipation et qui a été développé à partir des EPE. Ce mécanisme est décomposé en un cycle où BEACS planifie si la détection du PAI doit avoir lieu, auquel cas il extrait de sa population les classeurs lui permettant de déterminer si l'observation courante est liée ou non au problème d'aliasing perceptif.

Enfin, les séquences comportementales de BEACS ont également été révisées afin de gérer leur création et leur suppression au gré de la détection du PAI. Contrairement à BACS qui sépare le processus d'apprentissage par anticipation selon qu'un classeur contienne ou non une BSeq, BEACS unifie le processus d'apprentissage par anticipation de tous ses classeurs en un unique processus et laisse le soin à son processus de rétribution de discriminer et adapter l'emploi des classeurs à BSeq à la résolution de la tâche d'apprentissage. Le processus de rétribution de BEACS a également été modifié en s'appuyant sur le *Double Q-Learning* pour promouvoir les séquences comportementales les plus petites. Enfin, le mécanisme de généralisation génétique de BEACS a été retravaillé pour : permettre une généralisation indirecte des classeurs comportementaux sans porter atteinte aux capacités d'apprentissage ; favoriser la mise en compétition de classeurs ; préserver la façon de lire les représentations utilisées dans les *Conditions* et les *Anticipations* des classeurs.

Les résultats montrent que BEACS est le système de classeurs à anticipation qui construit les représentations environnementales les plus complètes et cohérentes avec les environnements incertains dans lesquels il réalise son apprentissage, qui décrit plus précisément les situations anticipées par les classeurs ainsi que leurs probabilités d'être anticipées, qui construit les politiques de décision les plus efficaces pour résoudre les tâches d'apprentissage et qui fournit le plus d'éléments explicatifs sur les politiques de décision créées et les représentations environnementales.

Des pistes d'améliorations ont alors été dégagées où :

- le mécanisme de détection du PAI pourrait être amélioré afin de détecter les situations liées au PAI qu'il n'a pas été en mesure de détecter ;
- les paramètres du mécanisme de détection du PAI et les différents critères entrants dans l'extraction des classeurs nécessaires à la détection pourraient être analysés plus en détail ;
- l'utilisation de la mémoire ou des ressources calculatoires d'un ordinateur pourraient être analysées afin d'optimiser l'exécution de BEACS ou le stockage des éléments explicatifs construits par BEACS ;
- les pressions de généralisation de BEACS pourraient être renforcées sans que les représentations environnementales ou les politiques décisionnelles en pâtissent ;
- différents processus de rétribution ou d'exploration de l'environnement pourraient être comparés afin d'étudier leurs conséquences sur les représentations environnementales et d'accroître les capacités de BEACS à résoudre différentes tâches d'apprentissage.

Une dernière voie d'amélioration serait de parvenir à extraire les connaissances des populations de classeurs avant d'exploiter celles-ci pour résoudre une tâche d'apprentissage. Les techniques d'extraction de connaissance peuvent promouvoir l'explicabilité de BEACS et des systèmes de classeurs à anticipation, en réduisant le nombre de classeurs dans les populations construites par ces systèmes : il peut alors être plus aisé d'identifier les éléments explicatifs en lien avec le comportement du système. Le chapitre suivant propose la première technique d'extraction de connaissance conçue et utilisée pour des systèmes de classeurs à anticipation.

CHAPITRE 8

Extraire les connaissances des systèmes de classeurs à anticipation

Le contrôle du nombre de règles dans les systèmes de classeurs à anticipation est une problématique importante pour prévenir les difficultés à expliquer le comportement de ces modèles. Après avoir dressé un portrait des différentes techniques d'extraction de connaissance des systèmes de classeurs, nous proposons une nouvelle technique d'extraction générique à tous les ALCS. Nous montrerons que celle-ci est capable d'exposer des représentations environnementales compactes et sans pertes des environnements, tout en minimisant l'impact de cette technique sur les politiques de décisions développées pour résoudre les tâches d'apprentissage.

Sommaire

8.1 Compresser pour extraire les connaissances

Trois techniques permettant d'extraire les connaissances des populations de classeurs ont été introduites dans la section 2.2.9 : la compression, la conden-

sation et la construction de nouvelles règles. Ces techniques s'inscrivent sous deux approches antonymes selon qu'elles sont employées à partir des données d'apprentissage (« *data-driven* ») ou non (« *rule-driven* »).

8.1.1 Extraire selon les données d'apprentissage ou selon les règles apprises

Quand les environnements sont incertains, il semble délicat d'utiliser des approches s'appuyant sur les données issues de ces environnements pour extraire des connaissances. Le caractère volatil des environnements incertains implique que les données d'apprentissage ne peuvent pas représenter complètement les tâches d'apprentissage. En conséquence, l'extraction des connaissances est biaisée par les données lors de la création d'une solution se voulant idéale (complète, correcte et compacte).

(Kharbat et al., 2007) montrent que les approches « *rule-driven* » mettent en évidence les classeurs qui décrivent efficacement le problème, quand les approches « *data-driven* » souffrent de surapprentissage. Afin de créer une représentation correspondant au mieux aux données d'apprentissage, les approches « *data-driven* » peuvent utiliser des classeurs trop spécifiques ou ignorer des connaissances déjà acquises.

C'est pourquoi l'algorithme d'extraction de connaissance recherché doit *s'appuyer sur les classeurs* et par extension, sur les représentations environnementales construites par les systèmes de classeurs.

8.1.2 Une condensation non adaptée

La condensation a été premièrement proposée par (Wilson, 1995) : elle supprime les classeurs les moins adéquats au problème à partir des données d'apprentissage, tout en bloquant la création de nouveaux classeurs par les processus de découverte de nouvelles règles. (Kovacs, 1998) a ensuite démontré que cette technique permettait aux systèmes de classeurs de construire des représentations de fonctions booléennes avec un nombre minimal de classeurs, le moins de recouvrement possible, tout en décrivant complètement le

problème. (Lanzi et al., 2001) ont appliqué la condensation pour un système de classeurs dédié à l'exploration de données : la condensation a permis de drastiquement réduire des populations de classeurs d'au moins 94% selon les problèmes étudiés, avec de très faibles impacts sur la résolution des problèmes. (Butz et al., 2008) a enfin adapté cette technique afin de permettre à un système de classeurs de résoudre des problèmes d'approximation de fonctions.

Pour que la condensation soit efficace, elle nécessite que le système de classeurs se soit déjà doté d'une représentation complète de ses données d'apprentissage, sinon la condensation ne pourrait faire converger la population vers l'ensemble le plus compact de classeurs (Kovacs, 1998). Elle pose aussi le problème de savoir à quel moment la condensation doit être appliquée : autrement dit, à quel moment une représentation complète des données d'apprentissage existe au sein d'une population de classeurs. Or, en pratique, il est difficile d'obtenir ou de savoir si la représentation construite par les systèmes de classeurs est complète pour des environnements incertains. Cela supposerait que ces données d'apprentissage décrivent parfaitement de tels environnements qui sont changeants. L'usage de la condensation est ainsi contraignant et aussi non adapté, puisqu'elle s'appuie sur les données d'apprentissage.

8.1.3 Des pertes de connaissances avec la construction de nouvelles règles

La construction de nouvelles règles a été introduite par (Kharbat et al., 2007) où les classeurs sont regroupés en fonction de leur similarité, afin de générer de nouveaux classeurs représentant les caractéristiques des groupes ainsi formés. Cette technique dépend d'une mesure d'une distance entre deux classeurs (plus cette distance est petite, plus les classeurs sont similaires) et d'un algorithme de partitionnement qui emploie cette distance. Elle a également été employée par (Urbanowicz et al., 2012b) qui combine des outils de visualisation à des outils statistiques afin de guider la découverte de connaissances en identifiant les attributs conditionnels discriminants des classeurs.

En cherchant à anticiper tous les changements perçus dans leur environne-

ment, les systèmes de classeurs à anticipation construisent des classeurs qui décrivent des situations environnementales par une combinaison d'éléments perceptifs pouvant changer ou rester inchangés. Cette technique, non appliquée à ce jour aux ALCS, pourrait alors fournir des éléments explicatifs sur l'usage des classeurs pour réaliser la tâche d'apprentissage, ainsi que sur les représentations des situations environnementales construites par les ALCS.

Ces éléments explicatifs seraient fournis par les caractéristiques des groupes de classeurs similaires, construits en s'appuyant par exemple sur les structures conditionnelles des classeurs et leurs récompenses. Par exemple, le fait d'anticiper la sortie d'un labyrinthique serait alors plus significatif que la présence d'un mur ou d'un chemin, pour un agent dont le but est d'atteindre la sortie. Plus généralement, les attributs perceptifs permettant d'atteindre la sortie des labyrinthes pourraient alors être mis en avant, pour cette tâche d'apprentissage.

En revanche, la construction de nouvelles règles semble moins adaptée à la construction d'une représentation environnementale compacte et sans pertes de connaissance. Afin d'éviter les pertes de connaissance, le calcul de la distance entre deux classeurs peut privilégier les classeurs correspondant aux mêmes situations environnementales et correspondant à un même *Conséquent*, ce qui revient à chercher à appliquer les algorithmes de partitionnement aux différents sous-ensembles de classeurs $[A]$ ou $[C]$ (comme définis en section 2.2.3). Or, les algorithmes génétiques des ALCS opèrent déjà à ce niveau pour généraliser les classeurs appartenant aux sous-ensembles $[A]$, permettant *de facto* de construire de nouveaux classeurs plus généraux. Par ailleurs, un partitionnement de classeurs ne prenant pas en compte la structure des classeurs des ALCS (*i.e. Condition*, *Conséquent* **et** *Anticipation*) peut amener à confondre des situations environnementales distinctes.

8.1.4 S'inspirer de la compression

La compression de populations de classeurs a été proposée pour la première fois par (Wilson, 2001) :

1. Les classeurs sont d'abord triés selon une propriété permettant de les classer selon leur pertinence pour résoudre une tâche d'apprentissage,

telle que leur numérosité, leur *fitness* ou leur expérience.

2. Un sous-ensemble de classeurs est ensuite formé à partir des classeurs triés, de telle sorte à ce que les performances sur un ensemble d'apprentissage soient maximales : le classeur ayant la meilleure adéquation forme un sous-ensemble candidat et si les performances ne sont pas maximales, le classeur ayant la meilleure seconde adéquation est ajouté au sous-ensemble, *etc.*
3. Les classeurs qui ont été ajoutés à ce sous-ensemble sans en améliorer les performances sont ensuite supprimés.
4. Les classeurs dont les structures conditionnelles se recouvrent sont enfin supprimés du sous-ensemble précédemment formé, en préférant ceux qui correspondent au plus grand nombre d'instances de l'ensemble d'apprentissage.

Des variations de l'algorithme de (Wilson, 2001) ont été proposées où :

— (Fu and Davis, 2002) modifient la quatrième étape en préférant les classeurs les plus généraux ;
— (Dixon et al., 2002) modifient la seconde étape en s'appuyant sur le produit de la numérosité par la récompense des classeurs ;
— (Tan et al., 2013) s'appuient sur les changements apportés par (Dixon et al., 2002) et proposent d'utiliser le produit de numérosité, de la récompense et de la généralité des classeurs lors de la seconde étape ;
— (Tan et al., 2013) proposent de s'appuyer sur la *fitness* des classeurs pour trier les classeurs, supprime la seconde étape et la troisième étape, et modifie la quatrième étape en préférant les classeurs ayant la plus grande *fitness* ;
— (Gao et al., 2007) modifient les trois premières étapes en s'appuyant sur l'expérience et les prédictions de récompense de classeurs pour former un sous-ensemble réduit de classeurs.

Cependant, ces algorithmes s'appuient sur les données d'apprentissage et se situent donc sous l'approche *« data-driven »* qui n'est pas souhaitée.

Parmi les techniques de compression de populations de classeurs, deux propositions ne s'appuient ni sur les données d'apprentissage, ni sur l'algorithme de (Wilson, 2001). (Tan et al., 2013) proposent un filtrage adapté à des systèmes de classeurs réalisant des tâches de classification : des classeurs sont supprimés de la population si leur prédiction n'est pas meilleure que le

hasard, ou si ces classeurs ont surappris les données d'entraînement. (Butz et al., 2008) introduisent un algorithme de compression pour un système de classeurs dont la *fitness* des classeurs est basée sur la précision. Il propose de supprimer les classeurs dont les situations décrites par leur structure conditionnelle sont incluses dans un classeur dont l'erreur sur la prédiction est plus faible.

Même si ces techniques de compression n'ont pas été employées avec les systèmes de classeurs à anticipation, il est possible de s'en inspirer pour proposer une nouvelle technique de compression pour les ALCS. Des modifications de ces deux propositions sont alors à prévoir afin de les adapter aux ALCS, puisqu'ils ne sont pas initialement conçus pour des tâches de classification et que la *fitness* des ALCS est basée sur la force.

L'idée d'un filtrage est intéressante pour contrôler l'accroissement de la population de classeurs des ALCS dû à l'usage des séquences comportementales. De nombreux classeurs possédant de telles séquences peuvent être créés, sans qu'ils aient été éprouvés ou qu'ils prennent part à la politique de décision du système : l'expérience de ces classeurs est alors inférieure au seuil θ_{exp} paramétré par l'utilisateur et permettant de les considérer comme suffisamment expérimentés. Dans le même temps, ces classeurs peuvent inclure des séquences d'actions dont l'anticipation n'amène à aucun changement de situations environnementales, à l'image d'un agent réalisant un pas en avant, puis un pas en arrière. Or, des changements environnementaux sont attendus quand le but des séquences comportementales est de traverser les états souffrant du problème d'aliasing perceptif.
Identifier et filtrer ces classeurs à séquences comportementales permettrait de réduire le nombre de classeurs de ces populations, tout en facilitant l'identification des classeurs à séquences comportementales d'intérêt pour l'étude de la population et pour la mise en place de la politique de décision.

Enfin, l'idée d'un algorithme de compression identifiant les classeurs qui se recouvrent permettrait de réduire la taille des populations de classeurs et de faciliter l'étude des populations. L'usage d'un tel algorithme nécessite de définir comment un classeur peut en inclure un autre, tout en prenant en compte la structure des classeurs (l'usage des séquences comportementales ou la présence de plusieurs anticipations avec les PEP ou les EPE).

8.2 Un algorithme de compression générique aux systèmes de classeurs à anticipation

L'algorithme de compression que nous proposons est constitué de deux étapes où, pour chaque classeur *cl* de la population :

- *cl* est comparé à chaque autre classeur de la population pour déterminer s'il peut être subsumé et donc, supprimé de la population ;
- si *cl* ne peut être subsumé par un autre classeur de la population, *cl* passe par un filtre qui supprime les classeurs à séquences comportementales si leur anticipation n'anticipe aucun changement ou s'ils ne sont pas suffisamment expérimentés en accord avec le paramètre θ_{exp} des ALCS.

Algorithme 8.2.1 Compression d'une population de classeurs d'ALCS

```
1 : function CompactPopulation(Cls,L,θexp,θr)
2 :     for each cl1 ∈ Cls do
3 :         keepCl1 ← True
4 :         for each cl2 ∈ Cls do
5 :             if IsSubsumed(cl1,cl2,θexp,θr) == True then
6 :                 keepCl1 ← False
7 :                 break
8 :             end if
9 :         end for
10 :        if keepCl1 == True and
              length(Acl1) > 1 and
              (expcl1 < θexp or Ecl1 = {#}^L) then
11 :            keepCl1 ← False
12 :        end if
13 :        if keepCl1 == False then
14 :            Discard cl1 from Cls
15 :        end if
16 :    end for
```

Le pseudo-code est fourni par l'algorithme 8.2.1 où l'opération permettant de déterminer si un classeur en subsume un autre est donnée par l'algorithme 8.2.2. La compression de la population de classeurs est réalisée une seule fois en amont de l'exploitation de la population. Elle prend en argument la population de classeurs *Cls*, les seuils utilisateur θ_{exp} et θ_{r} déterminant chacun si un classeur a été suffisamment éprouvé par le système ou si un classeur

est fiable, et enfin la longueur L caractérisant les perceptions reçues par le système.

L'algorithme 8.2.2 prend en argument deux classeurs cl_1, cl_2, ainsi que les seuils utilisateur θ_{exp} et θ_{r}. Un classeur cl_2 subsume cl_1 si :

- l'ensemble des situations environnementales décrites par la condition de cl_1, noté C_{cl_1}, est inclus dans celui de cl_2, noté C_{cl_2}.
- les *Conséquents* des classeurs sont identiques. Autrement dit, si les actions A_{cl_1} de cl_1 et les actions A_{cl_2} de cl_2 sont strictement les mêmes.
- l'ensemble des changements environnementaux décrits par l'anticipation de cl_1, noté E_{cl_1}, est inclus dans celui de cl_2, noté E_{cl_2}.
- le classeur cl_2 a été suffisamment éprouvé par le système de classeurs.
- le classeur cl_2 est fiable.
- le classeur cl_2 n'a pas été marqué par une situation environnementale ou que sa marque M_{cl_2} correspond strictement à celle de cl_1, notée M_{cl_1}.

Algorithme 8.2.2 Subsomption de classeurs d'ALCS

```
1 : function IsSubsumed(cl1,cl2,θexp,θr)
2 :     if Ccl1 ⊆ Ccl2 and
           Acl1 == Acl2 and
           Ecl1 ⊆ Ecl2 and
           expcl2 > θexp and qcl2 > θr then
3 :         if Mcl2 == ∅ then
4 :             return True
5 :         end if
6 :         if Mcl2 == Mcl1 then
7 :             return True
8 :         end if
9 :     end if
10 :    return False
```

En s'assurant que les conditions et anticipations de cl_1 sont bien incluses dans cl_2 pour une même série d'actions, toute transition environnementale décrite par un classeur à supprimer de la population est alors nécessairement décrite par un autre classeur. Ces comparaisons sur les transitions environnementales permettent aussi de prendre en compte les différentes structures des classeurs des ALCS, qui peuvent intégrer des séquences comportementales (BSeq), des prédictions améliorées par l'expérience (EPE) ou encore des

prédictions améliorées par les probabilités (PEP).

Les seuils θ_{exp} et θ_{r} sont employés pour ne pas supprimer de classeurs dont les transitions environnementales sont décrites par un classeur pour lequel on ne peut s'assurer de sa qualité.

S'appuyer sur des classeurs non marqués par une situation environnementale indique que leurs anticipations ont toujours été correctes. Si les classeurs sont marqués, vérifier que les marques des deux classeurs correspondent permet de savoir qu'ils ont été employés dans les mêmes situations environnementales. Utiliser les marques des classeurs permet ainsi de supprimer les classeurs qui ne sont que des produits intermédiaires d'autres classeurs, tout en limitant une altération de la représentation environnementale construite par les ALCS.

L'algorithme de compression réalise d'abord la subsomption entre des classeurs (ligne 4 à 9 de l'algorithme 8.2.1), avant de réaliser le filtrage proposé pour les classeurs à séquences comportementales (ligne 10 à 12 de l'algorithme 8.2.1). Le temps d'exécution de cet algorithme, bien que quadratique selon la taille de la population de classeurs, reste négligeable par rapport au temps pris par les ALCS pour réaliser l'apprentissage (Butz et al., 2008). Néanmoins, réaliser le filtrage avant la comparaison entre classeurs permettrait de réduire le nombre de classeurs présents dans la population avant de réaliser les comparaisons, et potentiellement d'accélérer la compression.

8.3 Efficacité de l'algorithme de compression

8.3.1 Méthodologie

Les métriques d'évaluation et le protocole expérimental détaillés dans le chapitre 4 permettent de mesurer l'efficacité de l'algorithme de compression en répondant aux questions suivantes :

1. Comment l'algorithme de compression affecte-t-il les représentations environnementales construites par les systèmes de classeurs à anticipation ?

2. A quel point les populations de classeurs des ALCS sont-elles réduites par cet algorithme ?

3. En quelle mesure l'algorithme de compression impacte la capacité des systèmes de classeurs à anticipation à réaliser leur tâche d'apprentissage ?

L'algorithme de compression est étudié avec BACS, PEPACS et BEACS, afin d'en démontrer la généricité. Une copie profonde des populations de classeurs est réalisée à la fin de l'apprentissage pour chaque environnement pour étudier l'impact de la compression proposée sur les copies, quand les populations originales servent de témoin. Les paramètres propres à BACS, PEPACS et BEACS utilisés lors de l'apprentissage sont fixés à leurs valeurs par défaut, définies dans leur chapitre respectif.

Pour déterminer comment la compression modifie les représentations environnementales, les ratios de connaissances, les erreurs moyennes sur la prédiction de l'anticipation (EMPA) ainsi que les spécificités moyennes des classeurs fiables sont utilisés. Les spécificités moyennes des classeurs fiables permettent de calculer le *taux de généralisation* selon :

$$1 - \frac{Spécificité\ moyenne\ des\ classeurs\ fiables\ compressés}{Spécificité\ moyenne\ des\ classeurs\ fiables\ témoin}$$

Le *taux de généralisation* évalue l'augmentation ou la diminution de la généralisation des classeurs fiables entre les populations témoins et celles compressées.

Les tailles des populations de classeurs et le nombre d'étapes nécessaires pour atteindre la sortie des labyrinthes permettent respectivement de répondre aux deux dernières interrogations. Les tailles des populations de classeurs permettent en particulier de calculer le *taux de compression* entre les populations témoins et celles compressées selon :

$$1 - \frac{Taille\ de\ la\ population\ compressée}{Taille\ de\ la\ population\ témoin}$$

Le *taux de compression* quantifie la réduction du nombre de classeurs entre les populations étudiées.

Les résultats obtenus sur chacune des métriques d'évaluation de notre algorithme de compression sont présentés dans la section 8.3.2. Ils sont ensuite discutés dans la section 8.4.

8.3.2 Résultats

Comment l'algorithme de compression affecte-t-il les représentations environnementales construites par les systèmes de classeurs à anticipation ?

Les ratios de connaissances des populations de classeurs témoins sont strictement identiques aux ratios de connaissances des populations compressées, pour tous les environnements et pour BACS, PEPACS et BEACS. L'algorithme de compression n'a pas supprimé de classeurs qui auraient induits des pertes de connaissances dans les représentations environnementales construites par les ALCS. Ainsi, toute transition environnementale décrite par un classeur d'une population témoin est retrouvée dans un classeur de la population compressée correspondante.

Les Erreurs Moyennes sur la Prédiction de l'Anticipation permettent alors de vérifier si les populations compressées des ALCS sont capables de correctement indiquer les probabilités d'anticiper les prochaines situations, pour chaque transition environnementale. Les EMPA des populations témoins et compressées de BEACS n'ont aucune différence significative, quel que soit l'environnement ou les actions incertaines. Les EMPA des populations témoins et compressées de BACS n'ont de différences significatives que dans deux cas : il s'agit des environnements *Woods101.5* et *Woods102* sans les actions incertaines, où les EMPA des populations compressées sont 1.5% plus faibles que celles des populations témoins. Les EMPA des populations témoins et compressées de PEPACS n'ont aucune différence significative, quel que soit le labyrinthe, quand les actions ne sont pas bruitées. Si les actions sont incertaines, les EMPA des populations compressées sont 2% plus grandes que celles des populations témoins pour 11 des 23 environnements, et n'ont pas de différences significatives pour les 12 environnements restants.

Les spécificités moyennes des classeurs fiables des populations compressées

de BEACS sont plus élevées pour 2 environnements (*Maze7* et *MazeF4*), n'ont pas de différences significatives pour 9 environnements et sont plus faibles que celles des populations témoins pour les 12 environnements restants, quand les actions sont incertaines. S'il n'y a pas ces actions incertaines, elles sont alors plus élevées pour *Maze7* et *MazeF4*, identiques pour 4 labyrinthes et plus faibles pour les 17 labyrinthes restants. Les spécificités moyennes des classeurs fiables des populations compressées de BACS sont plus faibles que celles des populations témoins, pour tous les environnements, indépendamment des actions incertaines. Les spécificités moyennes des classeurs fiables des populations compressées de PEPACS sont plus faibles que celles des populations témoins à l'exception de deux cas : elles sont identiques pour *MazeF2* si les actions sont bruitées et pour *Woods100* si les actions ne sont pas bruitées. Le détail des différentes spécificités moyennes des populations de classeurs de BEACS, BACS et PEPACS est fourni dans l'annexe E.2.

Les figures 8.1 et 8.2 présentent les taux de généralisation atteints par les populations témoins et compressées des trois ALCS sans les actions incertaines et avec les actions incertaines.

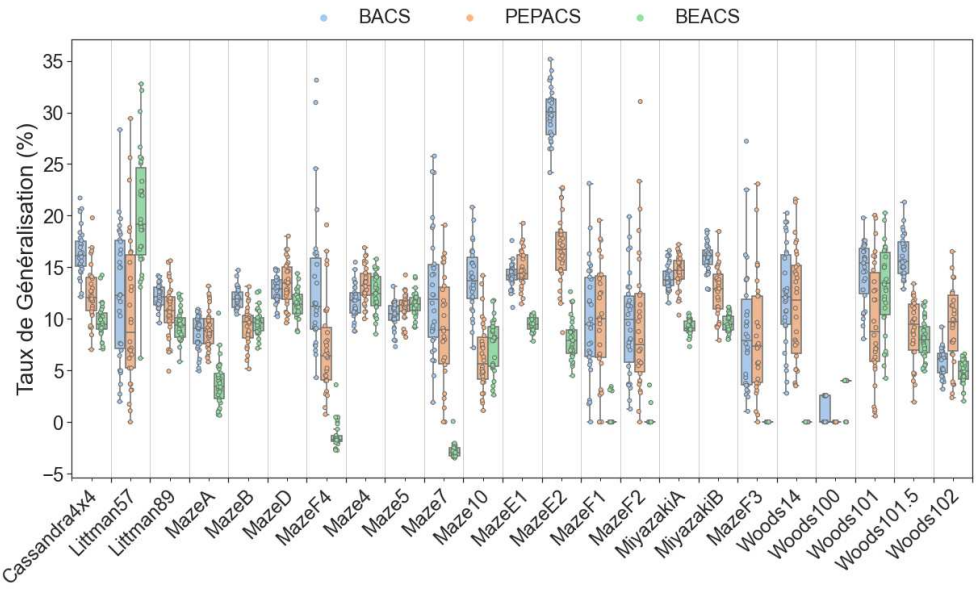


Figure 8.1 – Taux de généralisation pour chaque environnement du banc de test et pour chaque système de classeurs, sans les actions incertaines.

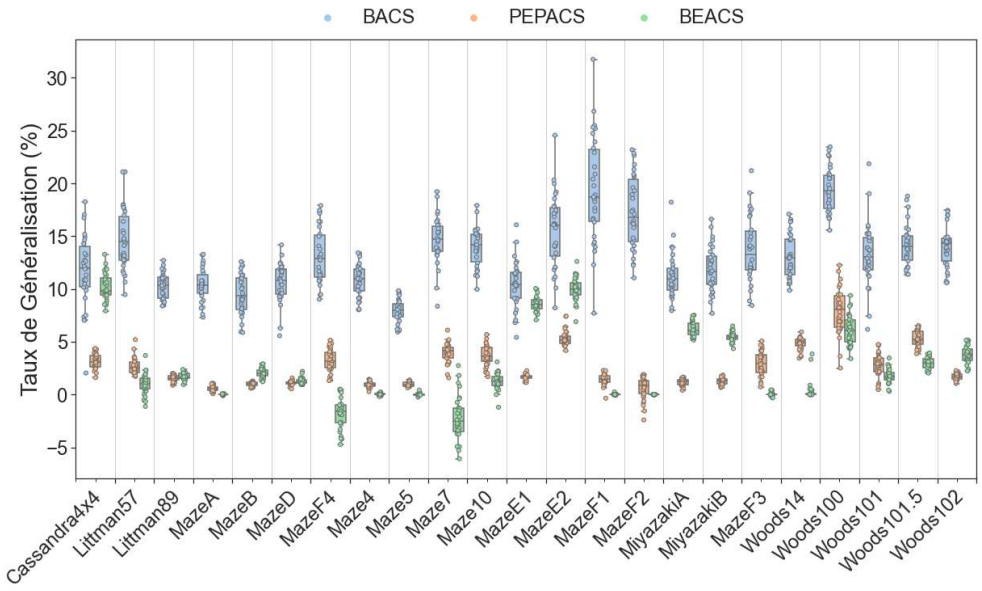


FIGURE 8.2 – Taux de généralisation pour chaque environnement du banc de test et pour chaque système de classeurs, avec les actions incertaines.

L'algorithme de compression accroît la généralisation des sous-populations de classeurs fiables de BEACS jusqu'à 10% pour 12 environnements avec les actions incertaines, quand il accroît la généralisation jusqu'à 20% pour 17 environnements si cette forme d'incertitude n'est pas présente. À l'inverse, pour *Maze7* et *MazeF4*, cet algorithme spécialise les sous-populations de classeurs fiables d'environ 3%, qu'il y ait ou non les actions incertaines. L'algorithme accroît la généralisation des classeurs fiables de BACS jusqu'à 17.5% si les actions sont bruitées, sinon jusqu'à 30%. Enfin, l'algorithme accroît la généralisation des classeurs fiables de PEPACS jusqu'à 7.5% si les actions sont bruitées, sinon jusqu'à 17%.

A quel point les populations de classeurs des ALCS sont-elles réduites par cet algorithme ?

La figure 8.3 présente les taux de compression suite à l'utilisation de l'algorithme de compression pour chacun des environnements et pour BEACS, BACS et PEPACS, lorsque les actions ne sont pas bruitées. L'algorithme de compression permet de réduire significativement les populations de classeurs construites par BEACS pour 19 des 23 labyrinthes, où le taux de compression moyen peut monter jusqu'à 52%. Les populations construites pour

MazeF1, *MazeF2*, *MazeF3* et *Woods14* ne sont pas modifiées par l'algorithme. Les populations de classeurs de BACS sont réduites pour tous les environnements, avec une réduction moyenne pouvant aller jusqu'à 50%. Enfin, les populations de PEPACS sont également significativement réduites pour 22 des 23 environnements, à une hauteur maximale de 22%. Les populations ne présentent pas de différences pour *Woods100*.

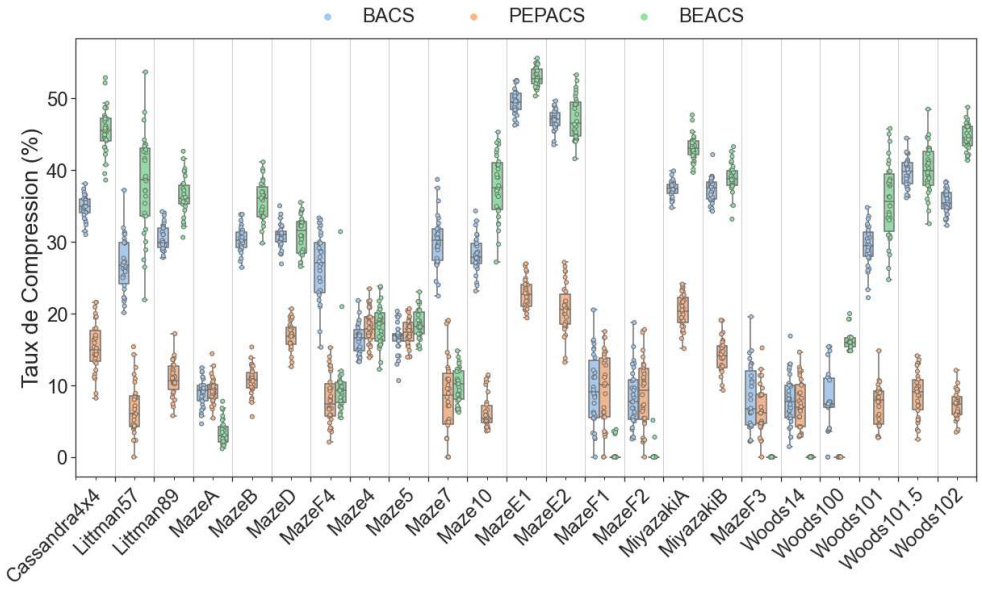


Figure 8.3 – Taux de compression pour chaque environnement du banc de test et pour chaque système de classeurs, sans les actions incertaines.

La figure 8.4 présente les taux de compression suite à l'utilisation de l'algorithme de compression pour chacun des environnements et pour BEACS, BACS et PEPACS, lorsque les actions sont bruitées. L'algorithme de compression permet également de réduire significativement les populations de classeurs construites par BEACS pour 19 des 23 labyrinthes, où le taux de compression moyen peut monter jusqu'à 80%. Les populations construites pour *MazeF1*, *MazeF2*, *MazeF3* et *Woods14* ne sont pas modifiées par l'algorithme. Les populations de classeurs de BACS sont réduites pour tous les environnements, avec une réduction moyenne pouvant aller jusqu'à 55%. Enfin, les populations de PEPACS sont également significativement réduites l'ensemble des environnements, à une hauteur maximale de 28%. Le détail des différentes tailles des populations de classeurs de BEACS, BACS et PEPACS est fourni dans l'annexe E.1.

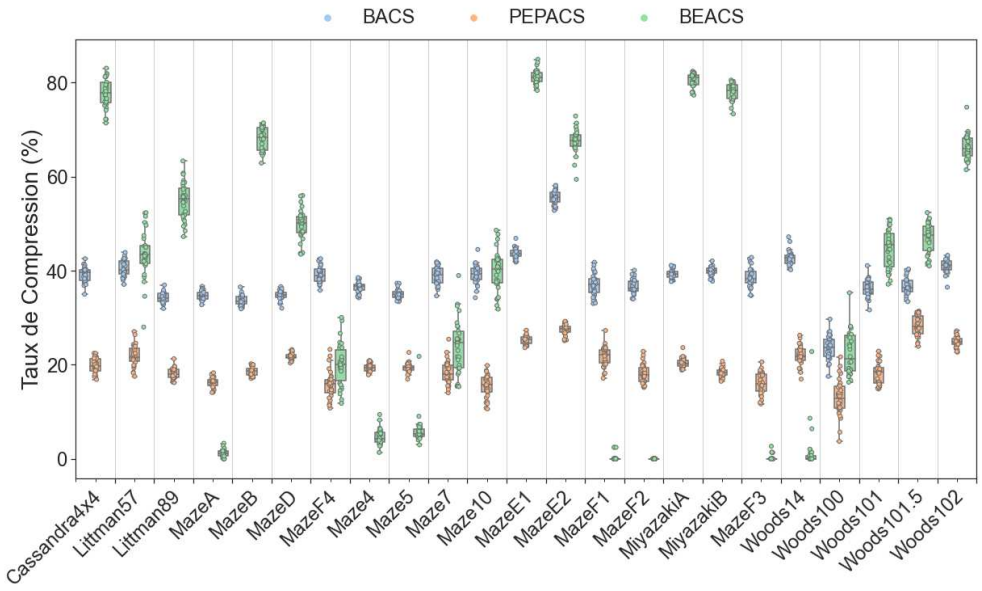


FIGURE 8.4 – Taux de compression pour chaque environnement du banc de test et pour chaque système de classeurs, avec les actions incertaines.

En quelle mesure l'algorithme de compression impacte la capacité des systèmes de classeurs à anticipation à réaliser leur tâche d'apprentissage ?

Les figures 8.5 et 8.6 donnent le nombre moyen d'étapes nécessaires à BEACS pour atteindre la sortie des labyrinthes, lorsque les actions ne sont pas bruitées et lorsqu'elles sont bruitées. Les figures 8.7 et 8.8, respectivement 8.9 et 8.10, fournissent cette même information pour BACS, respectivement PEPACS.

L'algorithme de compression ne modifie pas la capacité de BEACS à atteindre la sortie des labyrinthes pour 22 des 23 environnements, si les actions ne sont pas bruitées (figure 8.5). Cette capacité est modifiée pour le labyrinthe *Littman89* où l'algorithme augmente le nombre moyen d'étapes nécessaires à hauteur de 0.1 étape supplémentaire (soit une augmentation relative de 2.5%). L'algorithme de compression ne modifie pas la capacité de BEACS à atteindre la sortie des labyrinthes pour l'ensemble des environnements, si les actions sont bruitées (figure 8.6).

L'algorithme de compression ne modifie pas la capacité de BACS à atteindre la sortie des labyrinthes pour 22 des 23 environnements, que les actions ne soient pas bruitées (figure 8.7) ou qu'elles le soient (figure 8.8). Cette capacité est, indépendamment des actions incertaines, modifiée pour le labyrinthe

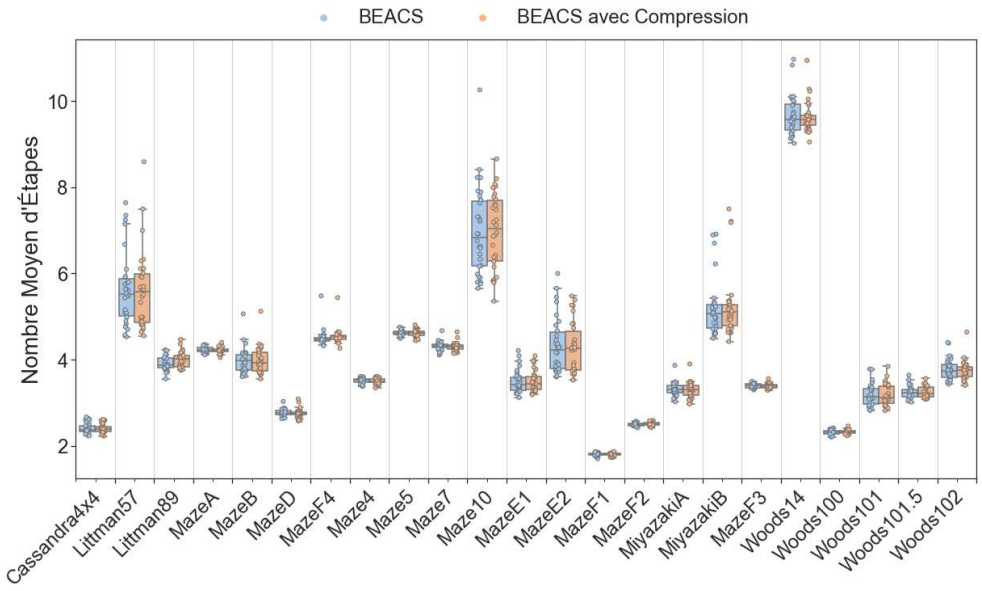


Figure 8.5 – Nombre d'étapes nécessaires à BEACS pour atteindre la sortie de chaque labyrinthe sans les actions incertaines, avant et après compression.

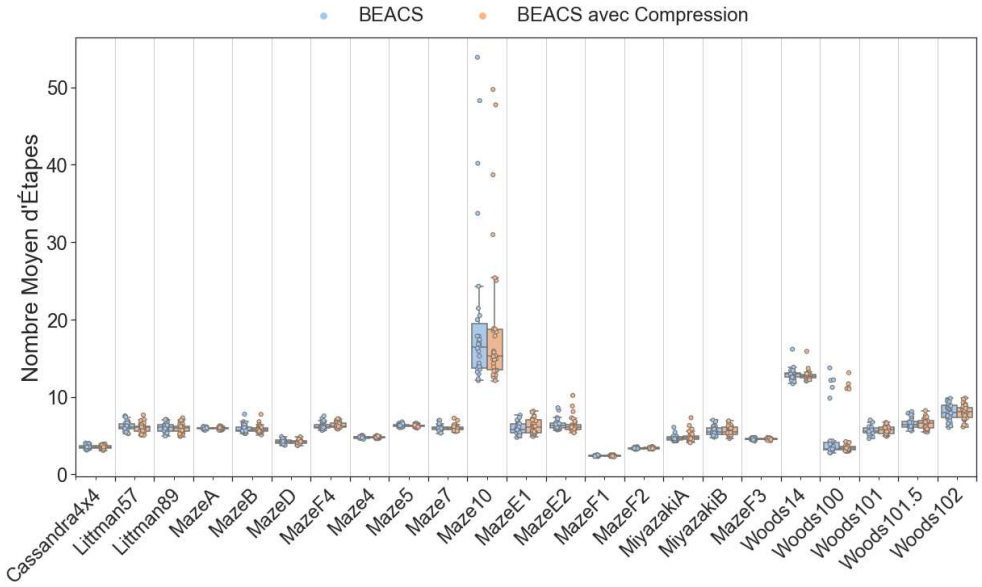


Figure 8.6 – Nombre d'étapes nécessaires à BEACS pour atteindre la sortie de chaque labyrinthe avec les actions incertaines, avant et après compression.

MazeE2 où l'algorithme augmente le nombre moyen d'étapes nécessaire à hauteur de 0.33 étape supplémentaire si les actions ne sont pas bruitées (respectivement 1.75 si les actions sont bruitées), soit une augmentation relative de 6% (respectivement 14%).

L'algorithme de compression ne modifie pas la capacité de PEPACS à atteindre

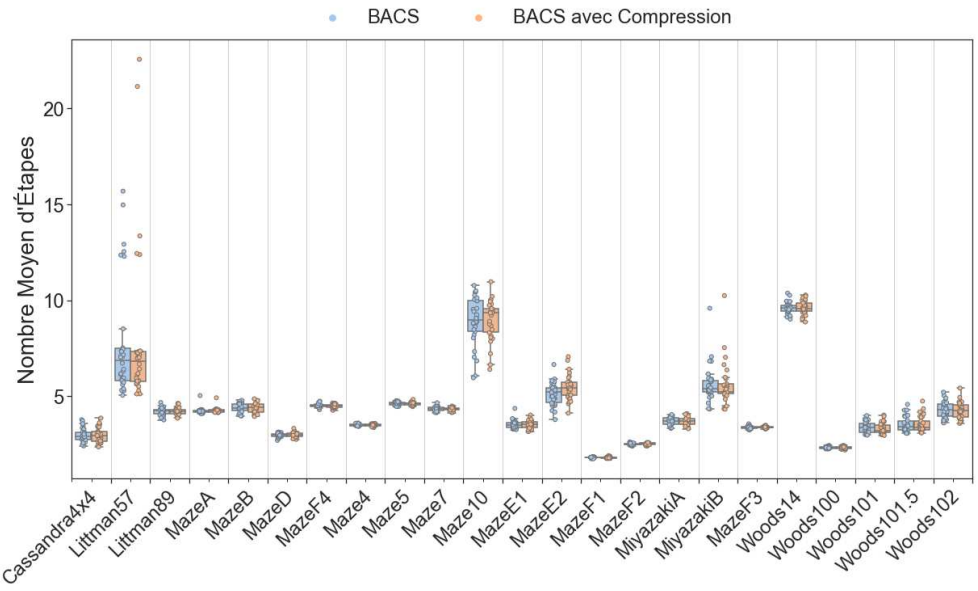


FIGURE 8.7 – Nombre d'étapes nécessaires à BACS pour atteindre la sortie de chaque labyrinthe sans les actions incertaines, avant et après compression.

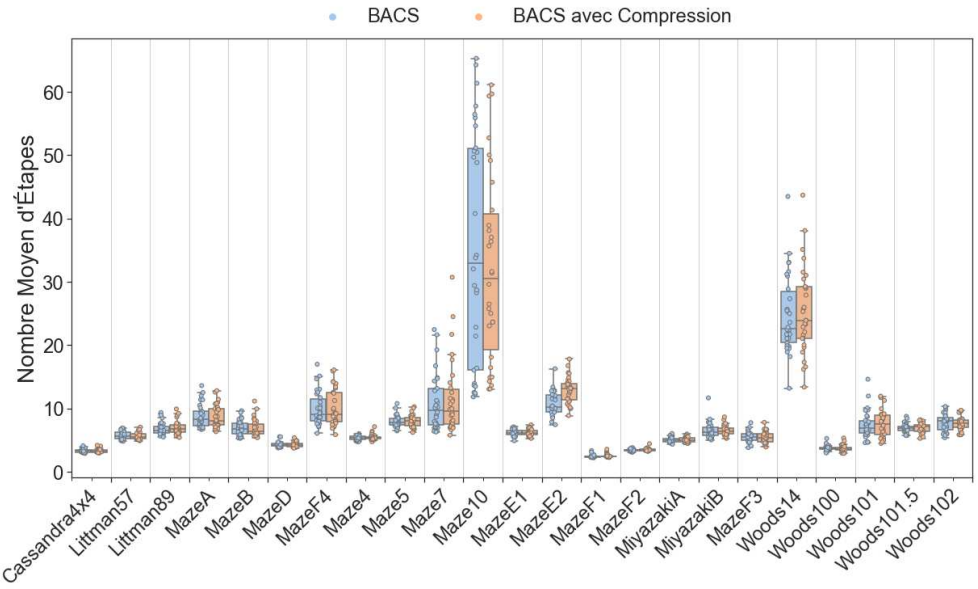


FIGURE 8.8 – Nombre d'étapes nécessaires à BACS pour atteindre la sortie de chaque labyrinthe avec les actions incertaines, avant et après compression.

la sortie des labyrinthes pour l'ensemble des environnements, si les actions ne sont pas bruitées (figure 8.9). L'algorithme de compression ne modifie pas la capacité de PEPACS à atteindre la sortie des labyrinthes pour 21 des 23 environnements, si les actions sont bruitées (figure 8.10). L'algorithme augmente le nombre moyen d'étapes pour le labyrinthe *MiyazakiA* à hauteur

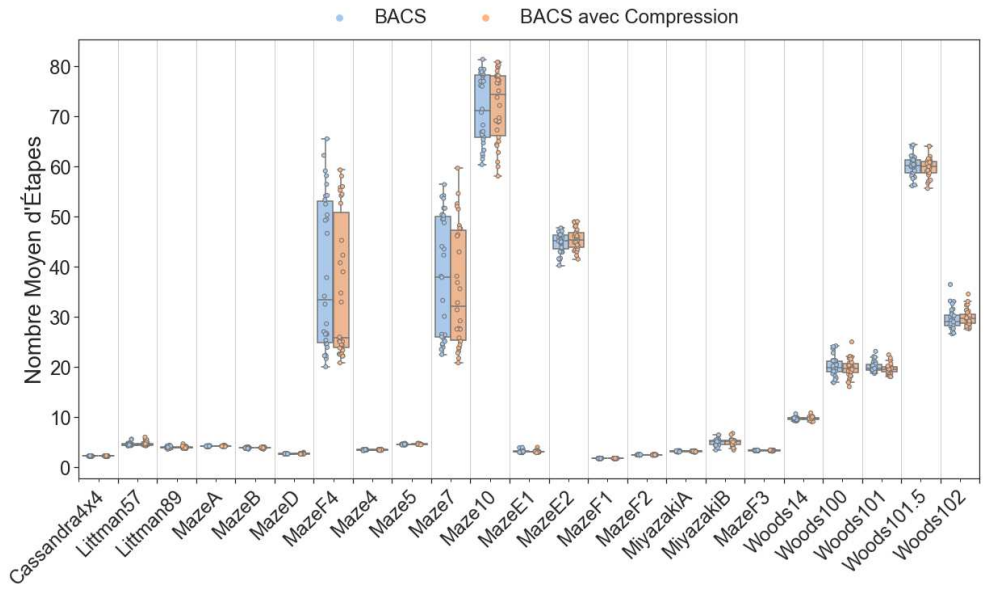


Figure 8.9 – Nombre d'étapes nécessaires à PEPACS pour atteindre la sortie de chaque labyrinthe sans les actions incertaines, avant et après compression.

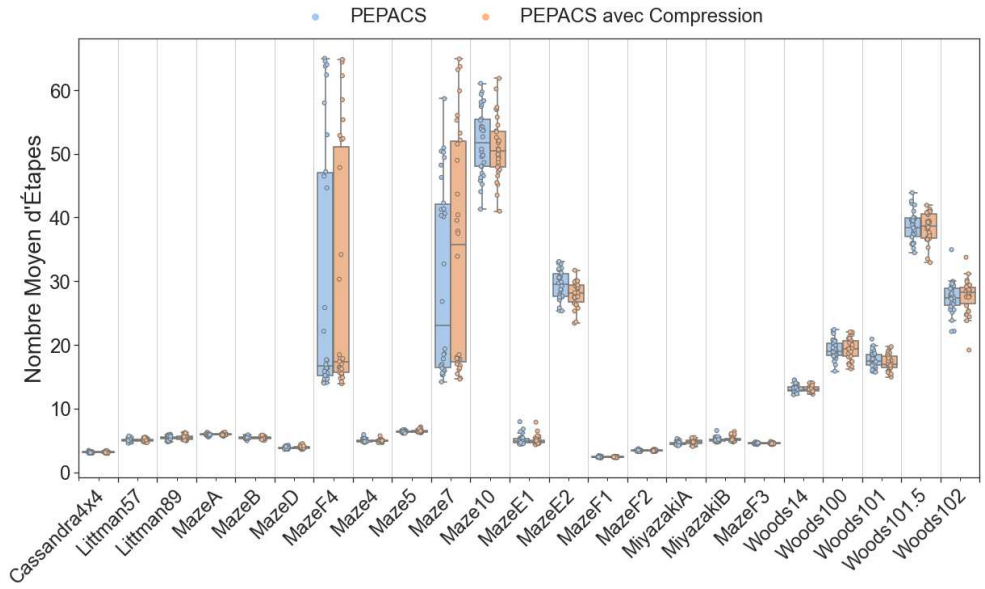


Figure 8.10 – Nombre d'étapes nécessaires à PEPACS pour atteindre la sortie de chaque labyrinthe avec les actions incertaines, avant et après compression.

de 0.2 étape supplémentaire (soit une augmentation relative de 4.6%). Enfin, l'algorithme diminue le nombre moyen d'étapes pour le labyrinthe *MazeE2* à hauteur de 1.4 étape supplémentaire (soit une diminution relative de 4.8%).

8.4 Discussion

8.4.1 Des représentations environnementales préservées

Les résultats montrent que les représentations environnementales construites par les trois systèmes de classeurs à anticipation ne sont pas détériorées par l'algorithme de compression, peu importe l'incertitude environnementale. Il n'y a donc pas de pertes quant à la capacité à expliquer les représentations environnementales développées ou les décisions prises par les systèmes de classeurs à anticipation.

En effet, les ratios de connaissances montrent qu'aucune transition environnementale n'est perdue. Les descriptions de ces transitions restent globalement de qualité, en particulier pour BEACS où elles sont inchangées (les EMPA n'ont aucune différence significative). Cet élément est particulièrement d'importance pour BEACS, car ce système a été conçu afin de permettre de telles descriptions, ce que BACS et même PEPACS ne peuvent réaliser. Les faibles écarts relatifs des EMPA de BACS et PEPACS qui sont d'au plus de 2% sont ainsi négligeables.

Les spécificités moyennes des classeurs fiables sont elles aussi inchangées ou plus faibles pour les trois systèmes. L'algorithme de compression permet ainsi aux représentations environnementales de s'appuyer sur les classeurs les plus généraux des populations : les taux de généralisation des figures 8.1 et 8.2 chiffrent cet effet. Ce résultat reste valable, en particulier pour BEACS, pour les deux environnements *Maze7* et *MazeF4* où ces spécificités augmentent. Cette augmentation n'est pas due à la suppression des classeurs fiables les plus généraux, et donne en réalité une preuve supplémentaire du bon fonctionnement de l'algorithme de compression. La figure 8.11 illustre la perte de spécificité observée dans ces environnements, où des classeurs dont la spécificité est inférieure à la spécificité moyenne d'un sous-ensemble entraînent une hausse quand ils sont supprimés.

	Spécificité d'un classeur					Spécificité moyenne
	Cl_1	Cl_2	Cl_3	Cl_4	Cl_5	
Ensemble de départ E	4	4	3	3	2	3.2
Ensemble compressé Ec_1		4		3	2	3
Ensemble compressé Ec_2	4	4			2	3.33

Figure 8.11 – Illustration du gain en spécificité moyenne des classeurs fiables observé dans *Maze7* et *MazeF4* avec BEACS.
Pour un ensemble de classeurs E, si l'algorithme de compression supprime les classeurs Cl_1 et Cl_3, alors l'ensemble de classeurs compressée Ec_1 a une spécificité moyenne plus faible. Inversement, si l'algorithme supprime Cl_3 et Cl_4, alors l'ensemble de classeurs compressée Ec_2 a une spécificité moyenne plus élevée.

La présence de stochasticité dans la réalisation des actions n'impacte pas les ratios de connaissance et les Erreurs Moyennes sur la Prédiction de l'Anticipation lorsque l'algorithme de compression est utilisé. En revanche, sa présence semble diminuer les taux de généralisation de BEACS et PEPACS atteints avec l'algorithme, quand ceux de BACS semblent augmenter. Ce qui est mis ici en lumière est une conséquence de la présence de cette stochasticité et non pas une limite de l'algorithme proposé. Les actions incertaines impliquent que plusieurs situations environnementales sont à anticiper par les ALCS, pour une action donnée et une situation donnée. Ainsi, plusieurs changements sont à anticiper entraînant *de facto* une spécialisation plus poussée des classeurs afin de décrire ces changements. Les classeurs des ALCS sont ainsi moins généralisables, d'où les taux de généralisation plus faibles.

Cependant, ce raisonnement est incomplet avec BACS et ne permet pas d'expliquer totalement la hausse des taux de généralisation. La stochasticité des actions entraîne dans BACS la création de nombreux classeurs à séquence comportementale, puisque toutes les situations environnementales déclenchent leur création même s'ils ne sont pas nécessaires. De plus, ces classeurs à séquences seront de plus en plus spécifiques, puisque les erreurs associées à leur usage seront plus nombreuses, le nombre d'actions à réaliser étant plus grand. L'algorithme de compression a donc un rôle plus important, puisqu'il agit directement sur les classeurs à séquence comportementale *via* le filtrage et la subsomption. Le filtrage supprime un plus grand nombre de classeurs à séquence comportementale qui n'auront pas été éprouvés par BACS et dont la spécificité est plus grande à cause des erreurs liées à leur usage. La sub-

somption favorise alors les classeurs à séquence comportementale qui auront été créés avant que la surspécificité susmentionnée n'intervienne.

BEACS, qui emploie aussi des séquences comportementales, n'a pas cet écueil, puisque leur création est conditionnée par des mécaniques qui cherchent justement à déterminer leur nécessité.

8.4.2 Une compression plus forte selon l'incertitude

Les populations de classeurs sont d'autant plus compressées que les environnements sont incertains. Les comparaisons statistiques des tailles de populations de classeurs et les taux de compression illustrés figure 8.3 et 8.4 montrent une compression significative des populations de classeurs. Cette réduction du nombre de classeurs s'opère quand les mécanismes de contrôle des populations des ALCS n'ont pas fait converger ces populations vers un ensemble de classeurs n'évoluant plus, comme cela est le cas avec BEACS pour 4 environnements.

En particulier, les taux de compression sont en moyenne plus élevés quand les actions sont incertaines dans les environnements. Cette hausse s'explique par l'usage d'une structure permettant d'anticiper plusieurs situations environnementales dans le cas de BEACS et PEPACS, et par l'usage non contrôlé des séquences comportementales dans BACS.

Avec BEACS et PEPACS, la construction de classeurs capables de correctement anticiper l'ensemble des situations environnementales pour une situation et une action donnée implique un plus grand nombre de classeurs intermédiaires décrivant une partie des situations à anticiper. Ces classeurs intermédiaires peuvent être encore présents dans la population à la fin de l'apprentissage, et sont alors effectivement supprimés par l'algorithme de compression.

Avec BACS, la discussion est similaire à celle tenue autour de la hausse des taux de généralisation. De nombreux classeurs à séquence comportementale sont créés par BACS, sans que le système ne puisse réellement les éprouver. Le filtrage introduit dans l'algorithme de compression occupe ainsi une place plus importante que la subsomption pour réduire la taille des populations de

classeurs.

8.4.3 Des politiques décisionnelles préservées

Les résultats montrent que l'algorithme de compression n'impacte pas la capacité des systèmes de classeurs à anticipation à atteindre le plus rapidement possible la sortie des labyrinthes, malgré quelques écarts relevés (figure 8.5 - *Littman89* avec BEACS, figures 8.7 et 8.8 *MazeE2* avec BACS, figure 8.10 *MiyazakiA* et *MazeE2* avec PEPACS).

Ces écarts peuvent découler du fait que l'algorithme de compression ne prend pas en compte les récompenses des classeurs des ALCS et aussi, de caractéristiques propres à certains environnements. Les politiques de décision établies par les ALCS durant l'exploration des environnements peuvent être modifiées par l'application de la compression, au risque de favoriser des politiques de décision sous-optimales. Le protocole expérimental cherche à permettre aux ALCS de se doter ensuite d'une politique de décision la plus performante possible pour la réalisation de la tâche. Ce protocole est commun à l'ensemble des environnements : les mêmes ressources sont allouées aux ALCS quand ils cherchent à résoudre leur tâche d'apprentissage, indépendamment des propriétés environnementales. Ainsi, les politiques de décision après la compression et l'exploitation des populations peuvent rester sous-optimales.

Une autre possibilité pour expliquer ces écarts relevés, en particulier pour BEACS et BACS, réside dans le filtrage réalisé sur les classeurs à séquence comportementale. Ce filtrage supprime les classeurs à séquence comportementale qui n'ont pas été suffisamment éprouvés par les ALCS. Or, il est possible que les séquences comportementales les plus adéquates n'aient pas été assez utilisées durant l'exploration des environnements et se retrouvent alors supprimées. Leur suppression favoriserait alors la mise en place de politiques de décision moins performantes par rapport à des populations où elles seraient encore présentes.

Bien que ces écarts nécessiteraient des analyses plus approfondies pour déterminer plus précisément leur origine, ils restent marginaux au regard des performances globales de l'algorithme de compression.

8.5 Synthèse

Pour extraire les connaissances des populations de classeurs des ALCS, un algorithme de compression s'appuyant exclusivement sur les connaissances contenues dans les classeurs est proposé dans ce chapitre.

Cet algorithme de compression cherche à réduire les tailles des populations de classeurs, tout en préservant complètement les représentations environnementales et les politiques de décision construites par les ALCS associées aux tâches d'apprentissage.

La mise en place de l'algorithme de compression s'appuie deux idées principales :

- la subsomption, où les structures conditionnelles des classeurs sont soigneusement comparées pour déterminer quels classeurs peuvent être supprimés sans affecter les connaissances contenues dans les populations ;
- un filtrage adapté aux classeurs à séquence comportementale afin de contrôler l'accroissement des populations de classeurs dû à leur usage, et de faciliter l'identification des séquences comportementales d'intérêt pour l'étude des populations.

Cette compression se veut aussi générique à tous les systèmes de classeurs à anticipation en prenant en compte les différentes représentations utilisées dans leurs classeurs, telles que les prédictions améliorées par l'expérience (EPE), les prédictions améliorées par les probabilités (PEP) ou les séquences comportementales (BSeq). Si de nouvelles représentations sont utilisées dans les classeurs des ALCS, il suffit alors de redéfinir les relations d'inclusion des *Conditions*, des *Conséquents* et des *Anticipations* des classeurs pour que la compression reste applicable. Elle est ainsi adaptable à toute nouvelle structure conditionnelle qui serait utilisée dans les systèmes de classeurs à anticipation.

Les résultats montrent que :

- la compression proposée ne détériore pas les représentations environnementales construites par les ALCS, indépendamment de l'incertitude environnementale ;

- les populations de classeurs des ALCS sont d'autant plus compressées que les environnements sont incertains ;
- les politiques de décision élaborées par les ALCS pour résoudre leur tâche d'apprentissage ne sont pas impactées par cette compression.

Cette technique de compression générique à tous les ALCS est ainsi capable de renforcer leur explicabilité, sans impacter leurs capacités d'apprentissage et leur autonomie.

L'étude des différentes techniques d'extraction des connaissances des systèmes de classeurs ouvre la voie vers de futurs travaux sur les populations des ALCS. De nombreuses techniques appliquées à d'autres systèmes de classeurs n'ont pas été appliquées ou adaptées aux ALCS. C'est le cas en particulier des techniques basées sur la création de nouveaux classeurs, qui pourraient guider les mécaniques de généralisation des classeurs des ALCS, ou qui pourraient pondérer les changements perceptifs contenus dans les classeurs des ALCS afin de fournir de nouveaux éléments explicatifs quant à leur usage.

L'algorithme proposé pourrait aussi être amélioré dans le but d'accroître la compression des représentations environnementales. Quand bien même la compression est d'autant plus importante que les environnements sont incertains, la présence de stochasticité dans la réalisation des actions impacte l'ordre de grandeur des tailles des populations de classeurs d'un facteur 10 : si les actions ne sont pas bruitées, les populations de classeurs sont de l'ordre de la centaine ; sinon, elles sont de l'ordre des milliers quand les actions sont bruitées.

Conclusions

Nous nous sommes intéressés au développement d'une intelligence artificielle autonome et explicable pour des environnements incertains. Les tâches à résoudre par les intelligences artificielles, et plus particulièrement par les algorithmes d'apprentissage automatique, sont de plus en plus complexes : ils doivent pouvoir s'adapter et co-évoluer en autonomie avec des environnements complexes, changeants et incertains qui sont à l'image de nos environnements quotidiens. Dans le même temps, il est de plus en plus nécessaire de pouvoir expliquer les comportements de ces algorithmes, ceux-ci pouvant être amenés à prendre des décisions critiques dans des situations qui peuvent profondément impacter la vie d'un individu.

Pour répondre à ce double objectif, les concepts d'autonomie et d'explicabilité des intelligences artificielles, ainsi que d'incertitude environnementale, ont été cadrés afin de guider nos choix de conception d'une intelligence artificielle d'abord vers un algorithme d'apprentissage automatique intrinsèquement explicable comme les approches à base de règles, puis vers les systèmes de classeurs à anticipation. D'une part, une approche intrinsèquement explicable permet de garantir la fidélité des éléments explicatifs au comportement des algorithmes. D'autre part, les approches à base de règles utilisent une même représentation pour résoudre une tâche d'apprentissage et pour expliquer le comportement de l'algorithme, trouvant leurs inspirations dans le comportement des êtres humains, ce qui facilite le transfert d'information inhérent à l'explicabilité. Enfin, les systèmes de classeurs à anticipation sont des algorithmes à base de règles dédiés à l'apprentissage des interactions entre ces algorithmes et leurs environnements, où le mécanisme d'anticipation permet notamment d'accéder aux effets des décisions prises durant leur apprentissage. Les systèmes de classeurs à anticipation sont aussi adaptés à la gestion des différentes formes d'incertitude environnementale, ceux-ci étant par conception : capables d'évaluer par eux-mêmes les décisions prises pour résoudre une tâche d'apprentissage (ambiguïté) ; capables de détecter

et de s'adapter par eux-mêmes à différentes formes de non-déterminisme (stochasticité) ; robustes aux changements environnementaux (volatilité).

Nous avons alors mis en place de nouveaux systèmes de classeurs à anticipation dont le but était de renforcer leurs capacités à évoluer en autonomie et de manière explicable dans des environnements incertains. Les capacités de chacun de ces systèmes de classeurs à anticipation ont été évaluées au travers d'un protocole d'évaluation expérimental que nous avons conçu. Ce protocole expérimental nous a notamment permis d'agir sur l'incertitude des environnements pour mettre en avant les capacités des systèmes de classeurs à anticipation que nous avons développés. Ainsi, nous avons effectivement réussi à accroître l'autonomie et l'explicabilité des systèmes de classeurs à anticipation avec :

- *Behavioral Anticipatory Classifier System* (BACS) qui est capable de construire des séquences comportementales (des chaînes d'actions) pour traverser des situations environnementales incertaines dans le but de résoudre de nouvelles tâches d'apprentissage ;
- *Probability-Enhanced Predictions in Anticipatory Classifier System* (PEPACS) qui est capable d'anticiper toutes les conséquences possibles d'une action dans une situation environnementale incertaine *via* des anticipations améliorées, permettant de fournir plus d'éléments explicatifs sur les interactions de PEPACS avec un environnement et aussi de résoudre de nouvelles tâches d'apprentissage ;
- *Behavioral Enhanced Anticipatory Classifier System* (BEACS) qui couple et améliore les capacités de BACS et celles de PEPACS dans un unique système de classeurs à anticipation afin de pousser plus loin l'autonomie et l'explicabilité du système résultant dans des environnements incertains.

Nous avons également conçu un algorithme d'extraction des connaissances propre aux systèmes de classeurs à anticipation capable de renforcer l'explicabilité inhérente à ces systèmes, sans détériorer leurs capacités d'apprentissage et leur autonomie. Parvenir à réduire le nombre de règles développées par ces approches permet effectivement de mieux mettre en évidence les éléments explicatifs associés au comportement de ces systèmes.

Les travaux présentés dans ce manuscrit sont loin d'être exempts de défauts et plusieurs perspectives de travail sont à envisager, tant sur les algorithmes

sous-jacents aux systèmes de classeurs à anticipation que sur les concepts sur lesquels nous nous sommes appuyés pour proposer une intelligence artificielle autonome et explicable pour des environnements incertains.

Un premier ensemble de perspectives concerne les systèmes de classeurs à anticipation et plus particulièrement BEACS, qui est le système résultant de l'ensemble des travaux et réflexions réalisés dans le cadre de cette thèse.
Les systèmes de classeurs à anticipation possèdent de nombreux paramètres pour lesquels peu d'études ont été réalisées à notre connaissance. BEACS introduit aussi de nouveaux paramètres dont l'influence sur l'apprentissage et les synergies avec les autres paramètres n'ont pas été étudiées en détail. Une première perspective serait de réaliser des études de ces paramètres afin de renforcer notre compréhension du fonctionnement de ces modèles : de nouvelles propositions pourraient en découler permettant par exemple d'automatiser leur configuration ou de pouvoir les adapter plus efficacement aux tâches d'apprentissage à résoudre.
Le cadre des systèmes de classeurs à anticipation aborde différents champs de recherche que nous avons pu laisser de côté tels que le développement d'algorithmes inspirés par des mécanismes cognitifs ou les problématiques liées au dilemme exploration-exploitation classique à l'apprentissage par renforcement. Ces thématiques ont pu être abordées au travers d'autres systèmes de classeurs, qu'ils soient à anticipation ou non. Une deuxième perspective consisterait à aborder ces autres thématiques au travers de BEACS afin de pouvoir renforcer ses capacités d'apprentissage et d'autonomie. Par exemple, il serait intéressant que BEACS soit de lui-même en mesure de basculer d'un mode exploratoire vers un mode lié à l'exploitation s'il détecte qu'il n'acquiert plus de nouvelles connaissances dans sa population de classeurs (si par exemple il n'y a plus de variations des qualités des classeurs) et inversement s'il détecte qu'il ne parvient pas à résoudre sa tâche d'apprentissage. BEACS serait alors en mesure d'adapter sans intervention extérieure ses politiques de sélection de classeurs en fonction de ce qui pourrait être le plus bénéfique à la résolution d'une tâche d'apprentissage (acquérir des connaissances ou bien exploiter les connaissances déjà acquises). Dans cette même veine, le processus de rétribution des systèmes de classeurs à anticipation s'appuie principalement sur la force des estimations de récompenses de classeurs, celle-ci étant pondérée par la qualité des anticipations. Utili-

ser un processus de rétribution s'appuyant sur la précision des estimations de récompenses pourrait fournir des informations en lien avec la résolution d'une tâche d'apprentissage et donc aider à mettre en place cette bascule automatique entre exploration et exploitation.
Enfin, une troisième perspective serait de permettre aux systèmes de classeurs à anticipation et à BEACS de gérer des données d'apprentissage à valeurs continues et de réduire l'impact du fléau de la dimension sur ces systèmes. Il s'agirait alors de travailler sur la représentation des données d'apprentissage pour renforcer les capacités d'apprentissage de ces systèmes et leur permettre de résoudre de nouvelles tâches. Les systèmes de classeurs à anticipation s'appuient initialement sur des représentations symboliques pour réaliser leur apprentissage et interagir avec leurs environnements. Si les données d'apprentissage contiennent des valeurs continues (réelles par exemple), seule une discrétisation de ces valeurs leur permet de réaliser un apprentissage, limitant de fait l'utilisation de toutes les informations contenues dans ces données. Une gestion directe des valeurs continues par ces systèmes n'est pas privilégiée puisqu'elle suppose de pouvoir détailler pour toutes les valeurs possibles d'une donnée continue les changements perceptifs associés dans les anticipations des classeurs. En revanche, il nous semble intéressant de nous appuyer sur des modèles à même d'extraire les informations des données continues et d'appliquer une réduction de la dimensionnalité des données, modèles qui seraient à coupler à l'interface des systèmes de classeurs à anticipation avec leur environnement. L'utilisation d'auto-encodeurs nous parait être en ce sens une piste prometteuse, ces modèles ayant les caractéristiques recherchées et ayant déjà été couplés avec d'autres systèmes de classeurs.

Un deuxième ensemble de perspectives concerne l'autonomie et l'étude des différentes formes d'incertitude environnementale.
Nos travaux s'appuient sur une définition de l'autonomie des intelligences artificielles au travers de laquelle nous avons étudié différentes sources d'incertitude environnementale afin de réaliser nos travaux. Cependant, il existe d'autres définitions de l'autonomie qui pourraient être utilisées pour proposer d'autres intelligences artificielles autonomes. Il pourrait alors être intéressant de se pencher sur la notion d'autonomie au travers d'autres disciplines scientifiques et des sciences humaines pour mettre en avant d'autres

facettes de cette notion, ce qui permettrait d'établir de nouvelles propositions et aussi d'aborder les travaux que nous menés sous un angle différent. Il en est de même pour les différentes sources d'incertitude de l'environnement, où le fait de croiser nos travaux avec ceux d'autres disciplines scientifiques (comme la psychologie cognitive) permettrait d'approfondir les concepts abordés avec les sources identifiées ou bien de détecter de nouvelles sources d'incertitude.

Nos travaux sur la gestion des différentes sources d'incertitude par les systèmes de classeurs à anticipation se sont principalement concentrés sur la stochasticité et l'ambiguïté des environnements incertains. Si les systèmes de classeurs à anticipation sont par conception capables de gérer la volatilité des environnements grâce à leurs mécanismes de découvertes de nouvelles règles et leur mécanisme de recouvrement, peu d'études ont été réalisées permettant de savoir jusqu'à quel point ces systèmes peuvent s'adapter aux nouveautés de leur environnement durant un apprentissage et durant la réalisation de leur tâche d'apprentissage. Il serait alors intéressant de concevoir différents protocoles d'expérimentation afin de pouvoir caractériser leur gestion de la volatilité et de proposer de nouveaux mécanismes afin de renforcer leur autonomie. Dans le même temps, les capacités des systèmes de classeurs à anticipation et de BEACS à gérer la stochasticité et l'ambiguïté des environnements peuvent être renforcées. Par exemple, nos perspectives permettant à BEACS d'adapter ses politiques de sélection de classeurs à la résolution d'une tâche d'apprentissage, ou nos perspectives en lien avec son processus de rétribution, viseraient aussi à améliorer la gestion de l'ambiguïté des environnements incertains. Nos travaux se sont par ailleurs concentrés sur différentes formes de stochasticité et il serait intéressant d'en étudier de nouvelles, telles que le bruitage des récompenses, afin de pousser plus loin l'autonomie de BEACS et par extension, l'autonomie des systèmes de classeurs à anticipation dans les environnements incertains.

Les environnements incertains étant ambigus, stochastiques et volatils, une intelligence artificielle autonome doit être capable de gérer simultanément ces trois sources d'incertitude et d'adapter son apprentissage d'une tâche en conséquence. En particulier, ces trois sources d'incertitude peuvent mutuellement s'influencer, complexifiant leur identification et leur gestion par une intelligence artificielle. Par exemple, une volatilité importante implique de nombreux changements dans l'environnement auxquels un système doit pou-

voir apprendre à s'adapter pour savoir quelles décisions lui permettent d'atteindre un objectif (autrement dit, une forte ambiguïté). Aussi, il peut être difficile de savoir si une nouvelle situation observée est due à un environnement volatil ou bien à un environnement stochastique où cette nouvelle observation résulterait d'une décision suivant une faible probabilité. Une nouvelle perspective serait de permettre à BEACS et aux systèmes de classeurs à anticipation d'identifier distinctement ces différentes sources d'incertitude afin qu'ils puissent adapter leur apprentissage en conséquence, au lieu de chercher à se concentrer sur des formes spécifiques d'incertitude.
Enfin, il est possible d'élargir notre propos à l'apprentissage de différentes tâches, ce qui peut survenir dans des environnements volatils si une tâche d'apprentissage venait à être modifiée. Il est par exemple possible d'associer des ensembles de règles à la résolution d'une tâche. Une dernière perspective consisterait à permettre à BEACS et aux systèmes de classeurs à anticipation de gérer différents ensembles de règles au sein d'une même population et d'adapter chacun de ces ensembles de règles à une tâche à résoudre. Cette dernière perspective remet en particulier en question la notion de généralité d'une règle : une règle serait d'autant plus générale que celle-ci appartiendrait à différents ensembles. Autrement dit, la généralité d'une règle n'est plus uniquement désignée par le nombre d'attributs perceptifs spécifiés dans sa condition, mais aussi par la réutilisation de cette règle dans différentes tâches. Cette perspective met alors en avant plusieurs notions qui seraient intéressantes à étudier telles que le transfert de règles d'une tâche à une autre, la gestion de plusieurs ensembles de règles allant de leur création à leur maintien et leur suppression ou encore, la gestion des sources d'incertitudes *via* ces différents ensembles.

Un troisième et dernier ensemble de perspectives s'intéresse à l'explicabilité. Si nos travaux sur l'autonomie des intelligences artificielles peuvent se focaliser à l'échelle des systèmes, l'explicabilité des intelligences artificielles se situe à différents niveaux, à savoir : celui des systèmes dont le comportement et les décisions sont à expliquer, celui du public auquel les explications sont destinées et celui d'une interface permettant de réaliser la transmission des explications. Nos travaux se sont alors concentrés sur la génération d'éléments explicatifs des décisions et du comportement de nos modèles dans le but de favoriser par la suite la transmission de ces éléments selon le public

cible. Dès lors, deux premières limites et perspectives de travaux émergent. D'une part, la qualité des éléments explicatifs qui sont construits par BEACS, et par les systèmes de classeurs à anticipation, n'a pas été évaluée. L'approche adoptée avec BEACS consistait à enrichir les informations existantes au sein des populations de classeurs et à en apporter de nouvelles afin de pouvoir mieux saisir quels sont les comportements appris. Dans le même temps, une multiplicité des informations peut obscurcir les informations pertinentes à l'explication des décisions prises : c'est pourquoi nous avons également fait plusieurs propositions afin que BEACS puisse efficacement contrôler ses populations de classeurs en favorisant par exemple la compacité et la complétude des populations. Cependant, tous ces éléments *nous* facilitent l'explication du comportement appris par BEACS, mais qu'en est-il du *public* auquel les explications sont destinées ? Une réduction du nombre de classeurs dans les populations et un accroissement du nombre d'éléments explicatifs sont-ils réellement suffisants pour que n'importe quelle personne puisse « déplier » l'ensemble des éléments ayant mené aux décisions ? Une réponse pourrait être apportée à ces questions en élargissant le cadre de nos travaux au travers de disciplines scientifiques autres que l'informatique telles que la philosophie ou les sciences cognitives.

D'autre part, nos travaux n'ont pas abordé la transmission des éléments explicatifs qui intervient dans le processus d'explicabilité. L'élargissement proposé du cadre de nos travaux permettrait alors d'évaluer la transmissibilité de ces éléments. Autrement dit, il s'agirait d'évaluer l'interprétabilité des explications fournies, en tant que la capacité de fournir aux explications un sens compréhensible. L'interprétabilité pouvant être spécifique par exemple à un domaine (où chaque domaine peut utiliser de mêmes mots mais avec des sens différents), voire à une personne, l'interface réalisant la transmission des explications doit être capable de s'adapter tant à la source des éléments explicatifs qu'à leurs destinataires. En adoptant une approche transdisciplinaire, de multiples formes de cette interface pourraient être conçues et comparées afin de globalement promouvoir l'explicabilité du système : il est par exemple d'ores et déjà possible de s'inspirer de l'interface existante des systèmes experts ou encore des ontologies informatiques pour travailler l'interprétabilité de BEACS et des systèmes de classeurs à anticipation.

Par ailleurs, BEACS, comme tous les autres systèmes de classeurs à anticipation, considèrent qu'un changement observé est le fruit d'une interaction du

système avec son environnement. Or, un changement observé peut avoir une origine différente de l'action réalisée par le système, impactant de fait son explicabilité. Ce point est d'autant plus vrai dans un environnement où plusieurs agents peuvent être en interaction, chacun réalisant des actions pouvant modifier la perception de l'environnement des autres agents. L'explicabilité de BEACS et des systèmes de classeurs à anticipation peut donc être abordée sous l'angle des approches multi-agents afin de permettre à ces systèmes d'identifier l'origine des changements observés. En particulier, de nouveaux mécanismes réalisant une comparaison des différentes populations de règles peuvent être intéressants pour successivement éliminer de possibles causes liées aux changements observés. Les approches multi-agents permettraient aussi d'aborder de nouvelles thématiques telles que celles associées aux partages de connaissances lors d'un apprentissage et donc de potentiellement travailler sous un nouvel angle l'autonomie de ces systèmes.

Enfin, il est possible de pousser plus loin les travaux sur l'explicabilité en se demandant si tous les éléments à destination des explications que nous avons conçus avec BEACS sont réellement utiles pour construire une explication. Il ne s'agit pas ici d'établir quelle serait la qualité de ces éléments, mais plutôt de pousser plus loin la réflexion sur ce en quoi consiste une explication et quels raisonnements sous-jacents permettent d'expliquer. Cette perspective nous amènerait à nous questionner sur les différentes formes de raisonnement, sur la notion de causalité qui est fortement intriquée aux systèmes de classeurs à anticipation ou encore sur l'élaboration des règles puisqu'elles sont, avec notre approche, l'interface commune entre l'explicabilité et l'autonomie. Pour conclure, cette dernière perspective nous conduirait à positionner nos travaux dans un cadre bien plus large que celui abordé dans ce manuscrit de thèse : des réponses à l'ensemble de ces questions et aux autres perspectives présentées précédemment nous permettraient de concevoir par exemple de nouvelles intelligences artificielles dignes de confiance ou éthiques.

Bibliographie

Ahluwalia, M., Bull, L., Banzhaf, W., et al. (1999). A genetic programming-based classifier system. In *GECCO*, pages 11–18. 21

Aliprandi, D., Mancastroppa, A., and Matteucci, M. (2006). A bayesian approach to learning classifier systems in uncertain environments. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 1537–1544. 21

Arai, S. and Sycara, K. (2001). Credit assignment method for learning effective stochastic policies in uncertain domains. In *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*, pages 815–822. 136

Arrieta, A. B., Diaz-Rodriguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-López, S., Molina, D., Benjamins, R., et al. (2020). Explainable artificial intelligence (xai) : Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information fusion*, 58 :82–115. 12, 18, 20

Bacardit, J. and Krasnogor, N. (2009). A mixed discrete-continuous attribute list representation for large scale classification domains. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 1155–1162. 34

Bagnall, A. J. and Zatuchna, Z. V. (2005). On the classification of maze problems. In *Foundations of Learning Classifier Systems*, pages 305–316. Springer. xxxi, 125, 127, 129, 130, 135, 136

Bastani, O., Pu, Y., and Solar-Lezama, A. (2018). Verifiable reinforcement learning via policy extraction. *Advances in neural information processing systems*, 31. 26

Bellman, R. (1957). A markovian decision process. *Indiana University Mathematics Journal*, 6 :679–684. 104

Berger-Tal, O., Nathan, J., Meron, E., and Saltz, D. (2014). The exploration-exploitation dilemma : a multidisciplinary framework. *PloS one*, 9(4) :e95693. 119

Bernadó, E., Llora, X., and Garrell, J. M. (2001). Xcs and gale : A comparative study of two learning classifier systems on data mining. In *International Workshop on Learning Classifier Systems*, pages 115–132. Springer. 21

Bernadó-Mansilla, E. and Garrell-Guiu, J. M. (2003). Accuracy-based learning classifier systems : models, analysis and applications to classification tasks. *Evolutionary computation*, 11(3) :209–238. 41

Bishop, J. T., Gallagher, M., and Browne, W. N. (2022). Pittsburgh learning classifier systems for explainable reinforcement learning : comparing with xcs. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 323–331. 31

Boissier, O., Carabelea, C., and Florea, A. M. (2003). Autonomie dans les systèmes multi-agents. essai de classification. In *Revue des Sciences et Technologies de l'Information-Série TSI : Technique et Science Informatiques*, volume 22, pages pp–191. Hermes Lavoisier. 13

Breiman, L. (2001). Random forests. *Machine learning*, 45(1) :5–32. 26

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016a). Openai gym. 103

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016b). Openai gym. 135

Buchanan, B. G. and Shortliffe, E. H. (1984). *Rule-based expert systems : the MYCIN experiments of the Stanford Heuristic Programming Project*. CUMINCAD. 24

Buche, C., Septseault, C., and De Loor, P. (2006). Les systèmes de classeurs : Une présentation générale. *TSI. Technique et science informatiques*, 25(8-9) :963–990. 40, 41

Bull, L. and O'Hara, T. (2002). Accuracy-based neuro and neuro-fuzzy classifier systems. In *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*, pages 905–911. Citeseer. 32

Burkart, N. and Huber, M. F. (2021). A survey on the explainability of supervised machine learning. *Journal of Artificial Intelligence Research*, 70 :245–317. 20

Büttner, J. and Von Mammen, S. (2021). Training a reinforcement learning agent based on xcs in a competitive snake environment. In *2021 IEEE Conference on Games (CoG)*, pages 1–5. IEEE. 50

Butz, A. M. V., Goldberg, B. D. E., and Stolzmann, C. W. (2002). The anticipatory classifier system and genetic generalization. *Natural Computing*, 1(4) :427–467. 79, 111, 114, 116, 136

Butz, M. V. (2001). Biasing exploration in an anticipatory learning classifier system. In *International Workshop on Learning Classifier Systems*, pages 3–22. Springer. 28, 73, 90, 93, 114, 116, 117, 136

Butz, M. V. (2005). Kernel-based, ellipsoidal conditions in the real-valued xcs classifier system. In *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, pages 1835–1842. 33

Butz, M. V. (2015). Learning classifier systems. In *Springer Handbook of Computational Intelligence*, pages 961–981. Springer. 4, 27, 51, 52

Butz, M. V. and Goldberg, D. E. (2003). Generalized state values in an anticipatory learning classifier system. In *Anticipatory behavior in adaptive learning systems*, pages 282–301. Springer. 106, 119, 262

Butz, M. V., Goldberg, D. E., and Stolzmann, W. (2000). Probability-enhanced predictions in the anticipatory classifier system. In *International Workshop on Learning Classifier Systems*, pages 37–51. Springer. 15, 94, 136, 189

Butz, M. V. and Hoffmann, J. (2002). Anticipations control behavior : Animal behavior in an anticipatory learning classifier system. *Adaptive Behavior*. 100

Butz, M. V., Lanzi, P. L., and Wilson, S. W. (2008). Function approximation with xcs : Hyperellipsoidal conditions, recursive least squares, and compaction. *IEEE Transactions on Evolutionary Computation*, 12(3) :355–376. 267, 270, 273

Butz, M. V. and Pelikan, M. (2001). Analyzing the evolutionary pressures in xcs. In *Proceedings of the Third Genetic and Evolutionary Computation Conference (GECCO-2001)*, volume 935, page 942. Citeseer. 44

Butz, M. V., Sigaud, O., and Gérard, P. (2003). Anticipatory behavior : Exploiting knowledge about the future to improve current behavior. In *Anticipatory behavior in adaptive learning systems*, pages 1–10. Springer. 119

Butz, M. V. and Stolzmann, W. (2001). An algorithmic description of acs2. In *International Workshop on Learning Classifier Systems*, pages 211–229. Springer. 38, 39, 42, 47, 72, 90, 103, 137, 138

Butz, M. V. and Wilson, S. W. (2000). An algorithmic description of xcs. In *International Workshop on Learning Classifier Systems*, pages 253–272. Springer. 38, 42

Capozzoli, A., Cerquitelli, T., and Piscitelli, M. (2016). Enhancing energy efficiency in buildings through innovative data analytics technologies. *Next Generation Platforms for Intelligent Data Collection ; Dobre, C., Xhafa, F., Eds*, pages 353–389. 27

Casillas, J., Carse, B., and Bull, L. (2007). Fuzzy-xcs : A michigan genetic fuzzy system. *IEEE Transactions on Fuzzy Systems*, 15(4) :536–550. 32

Charbuty, B. and Abdulazeez, A. (2021). Classification based on decision tree algorithm for machine learning. *Journal of Applied Science and Technology Trends*, 2(01) :20–28. 28

Charniak, E. (1991). Bayesian networks without tears. *AI magazine*, 12(4) :50–50. 19

Chen, H., Wang, C., Huang, J., and Gong, J. (2021). Efficient use of heuristics for accelerating xcs-based policy learning in markov games. *Swarm and Evolutionary Computation*, 65 :100914. 49

Chi, H.-M., Chen, L.-Y., and Hsiao, T.-C. (2021). Extraction of psychological symptoms and instantaneous respiratory frequency as indicators of internet addiction using rule-based machine learning. *Advances in Science, Technology and Engineering Systems Journal*, 6(5) :203–212. 50

Coats, P. K. (1988). Why expert systems fail. *Financial Management*, pages 77–86. 24

Collins, A. and Koechlin, E. (2012). Reasoning, learning, and creativity : frontal lobe function and human decision-making. *PLoS biology*, 10(3) :e1001293. 16

Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1) :21–27. 18

Dam, H. H., Abbass, H. A., and Lokan, C. (2005). Be real ! xcs with continuous-valued inputs. In *Proceedings of the 7th annual workshop on Genetic and evolutionary computation*, pages 85–87. 33

Dam, H. H., Abbass, H. A., Lokan, C., and Yao, X. (2007). Neural-based learning classifier systems. *IEEE Transactions on Knowledge and Data Engineering*, 20(1) :26–39. 32

De Jong, K. A., Spears, W. M., and Gordon, D. F. (1993). Using genetic algorithms for concept learning. *Machine learning*, 13(2-3) :161–188. 30

DeJong, K. A. and Spears, W. M. (1990). Learning concept classification rules using genetic algorithms. Technical report, GEORGE MASON UNIV FAIRFAX VA. 32

DeTore, A. W. (1989). An introduction to expert systems. *Journal of insurance Medicine*, 21(4) :233–236. 24

Dixon, P. W., Corne, D. W., and Oates, M. J. (2002). A ruleset reduction algorithm for the xcs learning classifier system. In *International workshop on learning classifier systems*, pages 20–29. Springer. 269

Dorigo, M. (1993). Genetic and non-genetic operators in alecsys. *Evolutionary Computation*, 1(2) :151–164. 108, 119

Dorigo, M. and Bersini, H. (1994). A comparison of q-learning and classifier systems. *From animals to animats*, 3 :248–255. 41

Du, M., Liu, N., and Hu, X. (2019). Techniques for interpretable machine learning. *Communications of the ACM*, 63(1) :68–77. 17

Fagerland, M. W. and Sandvik, L. (2009). Performance of five two-sample location tests for skewed distributions with unequal variances. *Contemporary Clinical Trials*, pages 490 – 496. 138

Faraut, M. (2015). *Apprendre à apprendre dans un environnement incertain, et dynamique des réseaux corticaux pour la flexibilité comportementale*. PhD thesis, Université Claude Bernard-Lyon I. 4, 14, 15, 16

Feigenbaum, E. A. (1981). Expert systems in the 1980s. *State of the art report on machine intelligence. Maidenhead : Pergamon-Infotech*. 23

Fu, C. and Davis, L. (2002). A modified classifier system compaction algorithm. In *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*, pages 920–925. Citeseer. 269

Gao, Y., Huang, J. Z., and Wu, L. (2007). Learning classifier system ensemble and compact rule set. *Connection Science*, 19(4) :321–337. 47, 269

Gérard, P., Meyer, J.-A., and Sigaud, O. (2005). Combining latent learning with dynamic programming in the modular anticipatory classifier system. *European Journal of Operational Research*, 160(3) :614–637. 105

Gerard, P. and Sigaud, O. (2001). Adding a generalization mechanism to yacs. In *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*, pages 951–957. 104

Gérard, P. and Sigaud, O. (2001). Généralisation et apprentissage latent dans les systèmes de classeurs. *Extraction des connaissances et apprentissage : Apprentissage automatique et évolution artificielle*, 3(1) :87–114. 105

Gérard, P. and Sigaud, O. (2003). Designing efficient exploration with macs : Modules and function approximation. In *Genetic and Evolutionary Computation Conference*, pages 1882–1893. Springer. 105, 119

Gerard, P., Stolzmann, W., and Sigaud, O. (2002). Yacs : a new learning classifier system using anticipation. *Soft Computing*, 6(3) :216–228. 103, 104

Ghavamzadeh, M., Mannor, S., Pineau, J., Tamar, A., et al. (2015). Bayesian reinforcement learning : A survey. *Foundations and Trends® in Machine Learning*, 8(5-6) :359–483. 19

Gilpin, L. H., Bau, D., Yuan, B. Z., Bajwa, A., Specter, M., and Kagal, L. (2018). Explaining explanations : An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 80–89. 12, 55

Goldberg, D. E. and Holland, J. H. (1988). Genetic algorithms and machine learning. *Mach. Learn.* 39

Hahsler, M. (2015). A probabilistic comparison of commonly used interest measures for association rules. *United States. Southern Methodist University.* 27

Hansmeier, T. and Platzner, M. (2021). An experimental comparison of explore/exploit strategies for the learning classifier system xcs. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1639–1647. 50

Hasselt, H. (2010). Double q-learning. *Advances in neural information processing systems*, 23. 226

Hastie, T. J. and Tibshirani, R. J. (2017). *Generalized additive models.* Routledge. 19

Hayes-Roth, F., Waterman, D. A., and Lenat, D. B. (1983). *Building expert systems.* Addison-Wesley Longman Publishing Co., Inc. 24

Heider, M., Stegherr, H., Wurth, J., Sraj, R., and Hähner, J. (2022). Separating rule discovery and global solution composition in a learning classifier system. *arXiv preprint arXiv :2202.01677.* 47

Hoffmann, J. (1993). *Vorhersage und erkenntnis.* Hogrefe. 53

Holland, J. and Reitman, J. (1978). Cognitive systems based on adaptive agents. *Pattern-directed inference systems.* 30, 51

Holland, J. H. (1985). Properties of the bucket brigade. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pages 1–7. 41, 66

Holland, J. H., Booker, L. B., Colombetti, M., Dorigo, M., Goldberg, D. E., Forrest, S., Riolo, R. L., Smith, R. E., Lanzi, P. L., Stolzmann, W., et al. (1999). What is a learning classifier system ? In *International Workshop on Learning Classifier Systems*, pages 3–32. Springer. 27

Holland, J. H. et al. (1992). *Adaptation in natural and artificial systems : an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press. 39

Holland, J. H., Holyoak, K. J., Nisbett, R. E., and Thagard, P. R. (1989). *Induction : Processes of inference, learning, and discovery*. MIT press. 27

Holley, J. C., Pipe, A. G., and Carse, B. (2004). Oneiric processing utilising the anticipatory classifier system. In *International Conference on Parallel Problem Solving from Nature*, pages 1103–1112. Springer. 100

IEEE Spectrum (2021). How boston dynamics taught its robots to dance. https://spectrum.ieee.org/automaton/robotics/humanoids/how-boston-dynamics-taught-its-robots-to-dance. Consulté le 1er août 2022. 1

Irfan, M., Jiangbin, Z., Iqbal, M., Masood, Z., and Arif, M. H. (2022). Knowledge extraction and retention based continual learning by using convolutional autoencoder-based learning classifier system. *Information Sciences*. 48, 49

Jackson, P. (1986). *Introduction to expert systems*. Addison-Wesley Pub. Co., Reading, MA. 23

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Tunyasuvunakool, K., Ronneberger, O., Bates, R., Žídek, A., Bridgland, A., et al. (2020). High accuracy protein structure prediction using deep learning. *Fourteenth Critical Assessment of Techniques for Protein Structure Prediction (Abstract Book)*, 22 :24. 2

Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). Reinforcement learning : A survey. *Journal of artificial intelligence research*, 4 :237–285. 15

Kharbat, F., Odeh, M., and Bull, L. (2007). New approach for extracting knowledge from the xcs learning classifier system. *International Journal of Hybrid Intelligent Systems*, 4(2) :49–62. 45, 266, 267

Kovacs, T. (1998). Xcs classifier system reliably evolves accurate, complete, and minimal representations for boolean functions. In *Soft computing in engineering design and manufacturing*, pages 59–68. Springer. 45, 266, 267

Kovacs, T. (1999). Strength or accuracy ? fitness calculation in learning classifier systems. In *International Workshop on Learning Classifier Systems*, pages 143–160. Springer. 40

Kozlowski, N. and Unold, O. (2018). Integrating anticipatory classifier systems with openai gym. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1410–1417. 103, 111, 113, 114, 135

Kozlowski, N. and Unold, O. (2019). Preliminary tests of a real-valued anticipatory classifier system. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1289–1294. 33, 34, 101, 111, 112

Kozlowski, N. and Unold, O. (2020). Investigating exploration techniques for acs in discretized real-valued environments. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, pages 1765–1773. 102, 111, 115

Kozłowski, N. and Unold, O. (2021). Anticipatory classifier system with average reward criterion in discretized multi-step environments. *Applied Sciences*, 11(3) :1098. 49, 102, 112, 115, 119

Kozlowski, N. and Unold, O. (2022). Internalizing knowledge for anticipatory classifier systems in discretized real-valued environments. *IEEE Access*, 10 :33816–33828. 103, 111, 115, 116

Kristen, M. (2011). *Vers la notion d'agent énactif virtuel : Application à l'approche dynamique évolutionnaire*. PhD thesis, Université Européenne de Bretagne. 13

Landau, S., Sigaud, O., and Schoenauer, M. (2005). Atnosferes revisited. In *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, pages 1867–1874. 32

Lanzi, P. L. (1998). Adding memory to xcs. In *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98TH8360)*, pages 609–614. IEEE. 30

Lanzi, P. L. and Colombetti, M. (1999). An extension to the xcs classifier system for stochastic environments. In *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 1*, pages 353–360. 15

Lanzi, P. L. et al. (2001). Mining interesting knowledge from data with the xcs classifier system. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 958–965. Morgan Kaufmann San Francisco, CA 94104, USA. 267

Lanzi, P. L. and Loiacono, D. (2007). Classifier systems that compute action mappings. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1822–1829. 32

Lanzi, P. L. and Riolo, R. L. (2003). Recent trends in learning classifier systems research. In *Advances in Evolutionary Computing*, pages 955–988. Springer. 32

Lanzi, P. L. and Wilson, S. W. (2006). Using convex hulls to represent classifier conditions. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 1481–1488. 33

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553) :436–444. 1

Ledford, H. (2019). Millions of black people affected by racial bias in healthcare algorithms. *Nature*, 574(7780) :608–610. 3

legifrance.gouv.fr (2019). Loi n° 2019-222 du 23 mars 2019 de programmation 2018-2022 et de réforme pour la justice. https://www.legifrance.gouv.fr/jorf/article_jo/JORFARTI000038261732?r=GeFRV20ZDn. Consulté le 1er août 2022. 3

Lindsay, R. K., Buchanan, B. G., Feigenbaum, E. A., and Lederberg, J. (1993). Dendral : a case study of the first expert system for scientific hypothesis formation. *Artificial intelligence*, 61(2) :209–261. 24

Liu, Y. (2021). *Learning Classifier Systems for Understanding Patterns in Data*. PhD thesis, Open Access Victoria University of Wellington| Te Herenga Waka. 48, 49

Liu, Y., Browne, W. N., and Xue, B. (2018). Adapting bagging and boosting to learning classifier systems. In *International conference on the applications of evolutionary computation*, pages 405–420. Springer. 47

Liu, Y., Browne, W. N., and Xue, B. (2021). Visualizations for rule-based machine learning. *Natural Computing*, pages 1–22. 45

Loch, J. and Singh, S. P. (1998). Using eligibility traces to find the best memoryless policy in partially observable markov decision processes. In *ICML*, pages 323–331. 136

Lorenz, K. (1935). Der kumpan in der umwelt des vogels. der artgenosse als auslösendes moment sozialer verhaltungsweisen. *Journal für Ornithologie. Beiblatt.(Leipzig)*. 54

Makery (2016). Comment l'ia de sony flow machines se prend pour les beatles. https://www.makery.info/2016/10/14/comment-lia-de-sony-flow-machines-se-prend-pour-les-beatles/. Consulté le 1er août 2022. 1

Martin H, J. A., Lope, J. d., and Maravall, D. (2009). The knn-td reinforcement learning algorithm. In *International Work-Conference on the Interplay Between Natural and Artificial Computation*, pages 305–314. Springer. 18

Mellor, D. (2005). A first order logic classifier system. In *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, pages 1819–1826. 32

Métivier, M. and Lattaud, C. (2002). Anticipatory classifier system using behavioral sequences in non-markov environments. In *International Workshop on Learning Classifier Systems*, pages 143–162. Springer. 96, 118, 136, 143, 149, 162, 174

Midi Libre (2019). Une intelligence artificielle jugée "dangereuse" par ses créateurs. https://www.midilibre.fr/amp/2019/02/17/une-intelligence-artificielle-jugee-dangereuse-par-ses-createurs,8020560.php. Consulté le 1er août 2022. 1

Miller, B. L., Goldberg, D. E., et al. (1995). Genetic algorithms, tournament selection, and the effects of noise. *Complex systems*, 9(3) :193–212. 83

Miller, T. (2019). Explanation in artificial intelligence : Insights from the social sciences. *Artificial Intelligence*, 267 :1–38. 12, 20

Mingers, J. (1989). An empirical comparison of pruning methods for decision tree induction. *Machine learning*, 4(2) :227–243. 28

Miyazaki, K. and Kobayashi, S. (1999). Proposal for an algorithm to improve a rational policy in pomdps. In *IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 99CH37028)*, volume 5, pages 492–497. IEEE. 136

Molnar, C., Casalicchio, G., and Bischl, B. (2020). Interpretable machine learning–a brief history, state-of-the-art and challenges. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 417–431. Springer. 55

Nakamura, Y., Horiuchi, M., and Nakata, M. (2021). Convergence analysis of rule-generality on the xcs classifier system. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 332–339. 50

Nelder, J. A. and Wedderburn, R. W. (1972). Generalized linear models. *Journal of the Royal Statistical Society : Series A (General)*, 135(3) :370–384. 19

Oliver, J. J., Dowe, D. L., and Wallace, C. (1992). Inferring decision graphs using the minimum message length principle. In *Proceedings of the 5th Australian Joint Conference on Artificial Intelligence*, pages 361–367. World Scientific. 26

Orhand, R., Jeannin-Girardon, A., Parrend, P., and Collet, P. (2020a). Bacs : a thorough study of using behavioral sequences in acs2. In *International Conference on Parallel Problem Solving from Nature*, pages 524–538. Springer. 141

Orhand, R., Jeannin-Girardon, A., Parrend, P., and Collet, P. (2020b). Bacs : integrating behavioral sequences to acs2. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, pages 147–148. 141

Orhand, R., Jeannin-Girardon, A., Parrend, P., and Collet, P. (2020c). Pepacs : integrating probability-enhanced predictions to acs2. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, pages 1774–1781. 173

Orhand, R., Jeannin-Girardon, A., Parrend, P., and Collet, P. (2021). Explainability and performance of anticipatory learning classifier systems in non-deterministic environments. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 163–164. 199

Orhand, R., Jeannin-Girardon, A., Parrend, P., and Collet, P. (2022). Accurate and interpretable representations of environments with anticipatory learning classifier systems. In *European Conference on Genetic Programming (Part of EvoStar)*, pages 245–261. Springer. 199

Ouest France (2021). Un jeu vidéo vient d'être totalement créé par une intelligence artificielle. https://www.ouest-france.fr/high-tech/un-jeu-video-vient-d-etre-totalement-cree-par-une-intelligence-artificielle-7322923. Consulté le 1er août 2022. 1

Patrice Bertail, David Bounie, S. C. and Waelbroeck, P. (2019). Algorithmes : biais, discrimination et équité. https://www.telecom-paris.fr/wp-content-EvDsK19/uploads/2019/02/Algorithmes-Biais-discrimination-equite.pdf. Consulté le 1er août 2022. 3

Pätzel, D., Heider, M., and Wagner, A. R. (2021). An overview of lcs research from 2020 to 2021. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1648–1656. 32, 47, 50, 51

Pätzel, D., Stein, A., and Nakata, M. (2020). An overview of lcs research from iwlcs 2019 to 2020. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, pages 1782–1788. 47, 50, 121

Payzan-LeNestour, E. and Bossaerts, P. (2011). Risk, unexpected uncertainty, and estimation uncertainty : Bayesian learning in unstable settings. *PLoS computational biology*, 7(1) :e1001048. 13

Preen, R. J. and Bull, L. (2021). Deep learning with a classifier system : Initial results. *arXiv preprint arXiv :2103.01118*. 49

Preen, R. J., Wilson, S. W., and Bull, L. (2021). Autoencoding with a classifier system. *IEEE Transactions on Evolutionary Computation*, 25(6) :1079–1090. 49

Rakotomalala, R. (2005). Arbres de décision. *Revue Modulad*, 33 :163–187. 26, 28

Rosenbauer, L., Pätzel, D., Stein, A., and Hähner, J. (2021). An organic computing system for automated testing. In *International Conference on Architecture of Computing Systems*, pages 135–149. Springer. 50

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5) :206–215. 16, 17

Russell, S. and Norvig, P. (2003). *Artificial intelligence : a modern approach*. Prentice Hall. 1

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 3(3) :210–229. 1

Schwartz, A. (1993). A reinforcement learning method for maximizing undiscounted rewards. In *Proceedings of the tenth international conference on machine learning*, volume 298, pages 298–305. 102

Shillingford, B., Assael, Y., Hoffman, M. W., Paine, T., Hughes, C., Prabhu, U., Liao, H., Sak, H., Rao, K., Bennett, L., et al. (2018). Large-scale visual speech recognition. *arXiv preprint arXiv :1807.05162*. 2

Shiraishi, H., Tadokoro, M., Hayamizu, Y., Fukumoto, Y., Sato, H., and Takadama, K. (2021). Increasing accuracy and interpretability of high-dimensional rules for learning classifier system. In *2021 IEEE Congress on Evolutionary Computation (CEC)*, pages 311–318. IEEE. 49

Siddique, A., Browne, W. N., and Grimshaw, G. M. (2021). Frames-of-reference based learning : Overcoming perceptual aliasing in multi-step decision making tasks. *IEEE Transactions on Evolutionary Computation*. 48, 54, 55

Sigaud, O. and Wilson, S. W. (2007). Learning classifier systems : a survey. *Soft Computing*, 11(11) :1065–1078. 27, 39, 51, 53, 54

Stokes, J. M., Yang, K., Swanson, K., Jin, W., Cubillos-Ruiz, A., Donghia, N. M., MacNair, C. R., French, S., Carfrae, L. A., Bloom-Ackermann, Z., et al.

(2020). A deep learning approach to antibiotic discovery. *Cell*, 180(4) :688–702. 1

Stolzmann, W. (1999). An introduction to anticipatory classifier systems. In *International Workshop on Learning Classifier Systems*, pages 175–194. Springer. 30, 38, 39, 42, 57, 66, 67, 68, 69, 95, 96, 97, 136, 143, 162, 174

Stolzmann, W. and Butz, M. (1999). Latent learning and action planning in robots with anticipatory classifier systems. In *International Workshop on Learning Classifier Systems*, pages 301–317. Springer. 97, 98, 114, 116

Stolzmann, W., Butz, M. V., and Goldberg, D. E. (2000). First cognitive capabilities in the anticipatory classi er system. *et al.[228]*, pages 287–296. 98, 99

Stolzmann, W. et al. (1998). Anticipatory classifier systems. *Genetic Programming*, 98 :658–664. 52, 53, 63, 116

Stone, C. and Bull, L. (2003). For real ! xcs with continuous-valued inputs. *Evolutionary Computation*, 11(3) :299–336. 33

Stone, C. and Bull, L. (2005). Comparing xcs and zcs on noisy continuous-valued environments. *ETechnical Report. UWE*. 15

Subasi, A. (2020). Chapter 3 - machine learning techniques. In Subasi, A., editor, *Practical Machine Learning for Data Analysis Using Python*, pages 91–202. Academic Press. 27

Sutton, R. S., Barto, A. G., et al. (1999). Reinforcement learning. *Journal of Cognitive Neuroscience*, 11(1) :126–134. 14

Tadokoro, M., Sato, H., and Takadama, K. (2021). Xcs with weight-based matching in vae latent space and additional learning of high-dimensional data. In *2021 IEEE Congress on Evolutionary Computation (CEC)*, pages 304–310. IEEE. 48, 49

Tan, J., Moore, J., and Urbanowicz, R. (2013). Rapid rule compaction strategies for global knowledge discovery in a supervised learning classifier system. In *ECAL 2013 : The Twelfth European Conference on Artificial Life*, pages 110–117. MIT Press. 29, 45, 269

Tavana, M. and Hajipour, V. (2019). A practical review and taxonomy of fuzzy expert systems : methods and applications. *Benchmarking : An International Journal*. 25

The AlphaStar team (2019). Alphastar : Grandmaster level in starcraft ii using multi-agent reinforcement learning. https://www.deepmind.com/blog/alphastar-grandmaster-level-in-starcraft-ii-using-multi-agent-reinforcement-learning. Consulté le 1er août 2022. 17

The Conversation (2021). Comment fonctionne l'algorithme de recommandation de youtube ? ou comment j'ai découvert le rap hardcore. https://theconversation.com/comment-fonctionne-lalgorithme-de-recommandation-de-youtube-ou-comment-jai-decouvert-le-rap-hardcore-152189. Consulté le 1er août 2022. 1

The New York Times (2018). He drove a tesla on autopilot from the passenger seat. the court was not amused. https://www.nytimes.com/2018/04/29/world/europe/uk-autopilot-driver-no-hands.html. Consulté le 1er août 2022. 3

Tomlinson, A. and Bull, L. (2001). Cxcs : Improvements and corporate generalization. In *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*, pages 966–973. 30

Tufts, P. (1995). Dynamic classifiers : genetic programming and classifier systems. In *Proceedings of the Genetic Programming. Papers from the 1995 AAAI Fall Symposium*, pages 114–119. 32

UCI (1987). Congressional Voting Records. UCI Machine Learning Repository. 110

UCI (1992). MONK's Problems. UCI Machine Learning Repository. 109

Unold, O., Rogula, E., and Kozłowski, N. (2019). Introducing action planning to the anticipatory classifier system acs2. In *International Conference on Computer Recognition Systems*, pages 264–275. Springer. 100, 114, 119

Unold, O. and Tuszyński, K. (2008). Mining knowledge from data using anticipatory classifier system. *Knowledge-Based Systems*, 21(5) :363–370. 110, 111

Urbanowicz, R., Granizo-Mackenzie, A., and Moore, J. (2012a). Instance-linked attribute tracking and feedback for michigan-style supervised learning classifier systems. In *Proceedings of the 14th annual conference on Genetic and evolutionary computation*, pages 927–934. 39

Urbanowicz, R. J., Granizo-Mackenzie, A., and Moore, J. H. (2012b). An analysis pipeline with statistical and visualization-guided knowledge discovery for michigan-style learning classifier systems. *IEEE computational intelligence magazine*, 7(4) :35–45. 45, 46, 267

Urbanowicz, R. J. and Moore, J. H. (2009). Learning classifier systems : a complete introduction, review, and roadmap. *Journal of Artificial Evolution and Applications*, 2009. 28, 31, 32, 38, 40, 41, 44, 51

Urbanowicz, R. J. and Moore, J. H. (2015). Exstracs 2.0 : description and evaluation of a scalable learning classifier system. *Evolutionary intelligence*, 8(2) :89–116. 29, 34, 47

Venturini, G. (1994). Adaptation in dynamic environments through a minimal probability of exploration. In *Proceedings of the third international conference on Simulation of adaptive behavior : from animals to animats 3 : from animals to animats 3*, pages 371–379. 75

Villani, C., Bonnet, Y., Berthet, C., Levin, F., Schoenauer, M., Cornut, A. C., and Rondepierre, B. (2018). *Donner un sens à l'intelligence artificielle : pour une stratégie nationale et européenne*. Conseil national du numérique. 1

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., et al. (2019). Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782) :350–354. 2, 16

Wagner, A. R. and Stein, A. (2021). On the effects of absumption for xcs with continuous-valued inputs. In *International Conference on the Applications of Evolutionary Computation (Part of EvoStar)*, pages 697–713. Springer. 49

Watkins, C. J. and Dayan, P. (1992). Q-learning. *Machine learning*, 8(3-4) :279–292. 41, 51, 87, 131

Wilson, S. W. (1994). Zcs : A zeroth level classifier system. *Evolutionary computation*, 2(1) :1–18. 47, 51

Wilson, S. W. (1995). Classifier fitness based on accuracy. *Evolutionary computation*, 3(2) :149–175. 32, 38, 41, 51, 52, 266

Wilson, S. W. (1999). Get real ! xcs with continuous-valued inputs. In *International Workshop on Learning Classifier Systems*, pages 209–219. Springer. 21, 33, 34

Wilson, S. W. (2000). Mining oblique data with xcs. In *International Workshop on Learning Classifier Systems*, pages 158–174. Springer. 33, 34, 35

Wilson, S. W. (2001). Compact rulesets from xcsi. In *International Workshop on Learning Classifier Systems*, pages 197–208. Springer. 268, 269

Witten, I. H., Frank, E., Hall, M. A., and Pal, C. J. (2017). Chapter 3 - output : Knowledge representation. In Witten, I. H., Frank, E., Hall, M. A., and Pal, C. J., editors, *Data Mining (Fourth Edition)*, pages 67–89. Morgan Kaufmann, fourth edition edition. 27

Wolberg, W. (1992). Breast Cancer Wisconsin (Original). UCI Machine Learning Repository. 110

Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G. J., Ng, A., Liu, B., Philip, S. Y., et al. (2008). Top 10 algorithms in data mining. *Knowledge and information systems*, 14(1) :1–37. 26

Zadeh, L. (1988). Fuzzy logic. *Computer*, 21(4) :83–93. 25

Zang, Z., Li, D., and Wang, J. (2015). Learning classifier systems with memory condition to solve non-markov problems. *Soft Computing*, 19(6) :1679–1699. 136

Zatuchna, Z. V. (2004). Agentp model : Learning classifier system with associative perception. In *International Conference on Parallel Problem Solving from Nature*, pages 1172–1181. Springer. 54, 55

ZDNet (2020). Qu'est-ce que gpt-3 ? https://www.zdnet.fr/pratique/qu-est-ce-que-gpt-3-tout-ce-que-votre-entreprise-doit-savoir-sur-le-programme-de-langage-d-ia-d-openai-39908563.htm. Consulté le 1er août 2022. 3

Zhang, R., Stolzenberg-Solomon, R., Lynch, S. M., and Urbanowicz, R. J. (2021). Lcs-dive : An automated rule-based machine learning visualization pipeline for characterizing complex associations in classification. *arXiv preprint arXiv :2104.12844*. 45, 46, 50

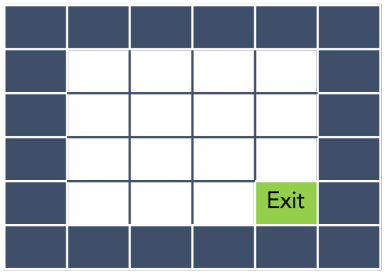
Zhou, Z.-H. (2021). Ensemble learning. In *Machine Learning*, pages 181–210. Springer. 47

Annexes

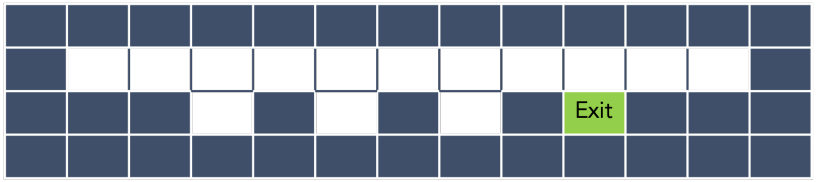
ANNEXE A

Banc de test de labyrinthes

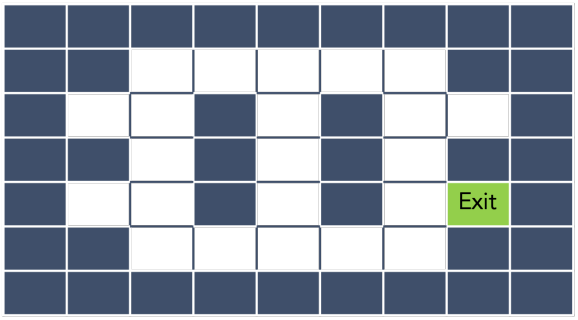
A.1 Illustrations des 23 labyrinthes



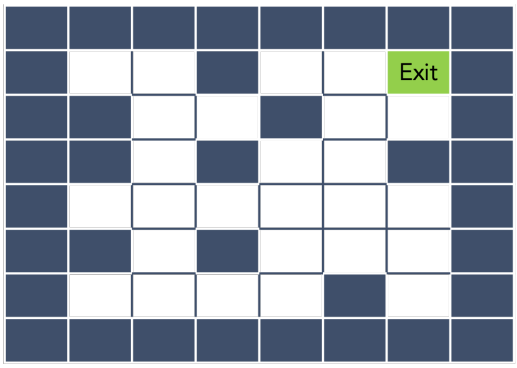
Cassandra4x4



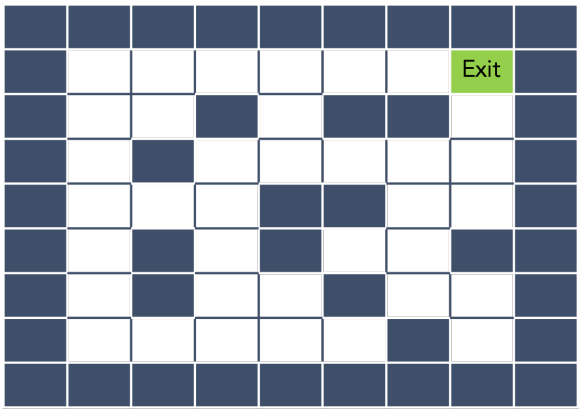
Littman57



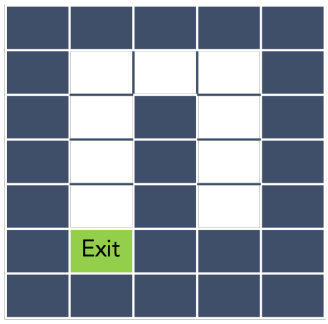
Littman89



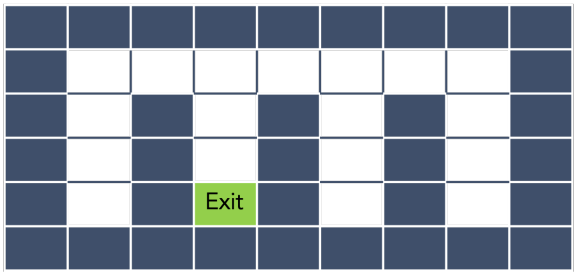
Maze4



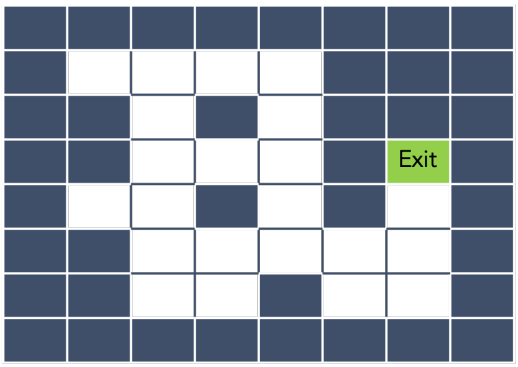
Maze5



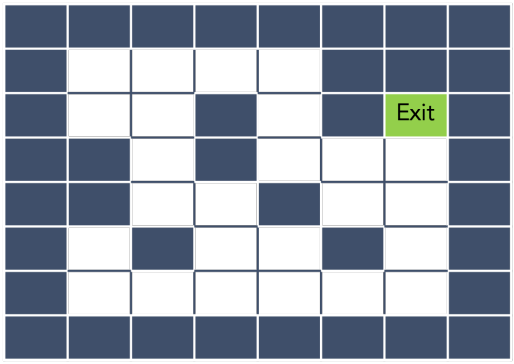
Maze7



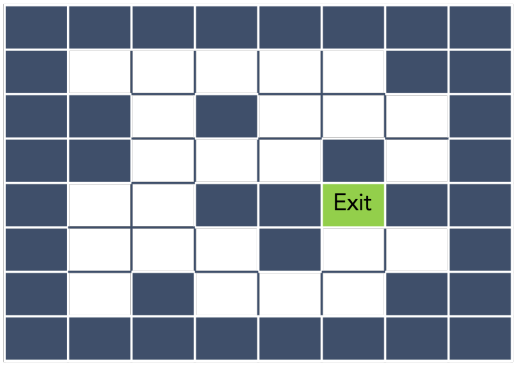
Maze10



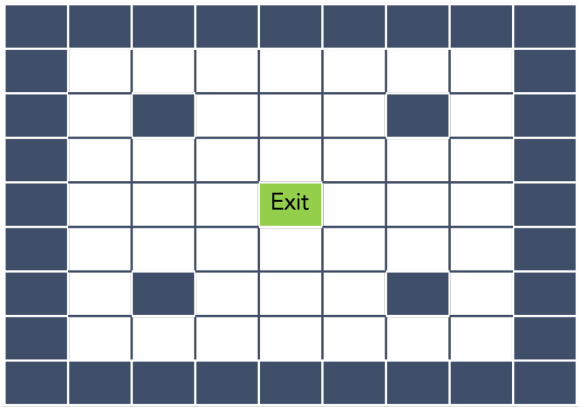
MazeA



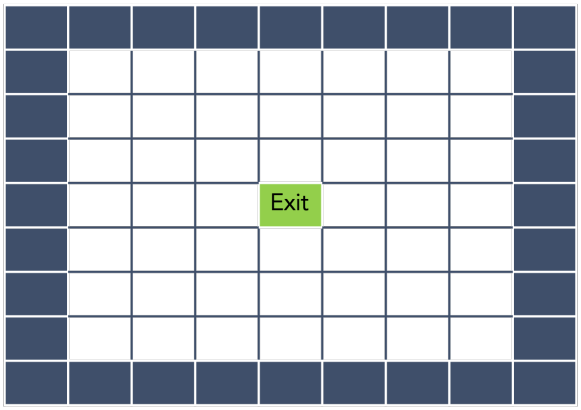
MazeB



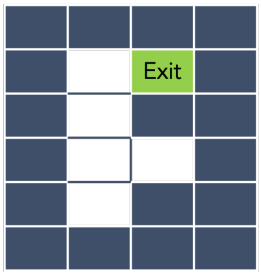
MazeD



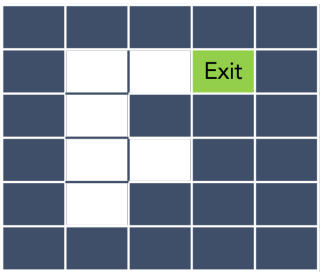
MazeE1



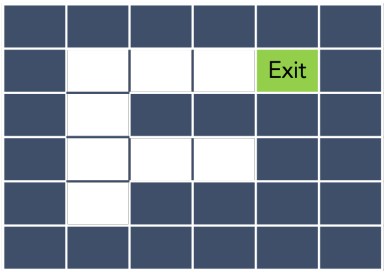
MazeE2



MazeF1

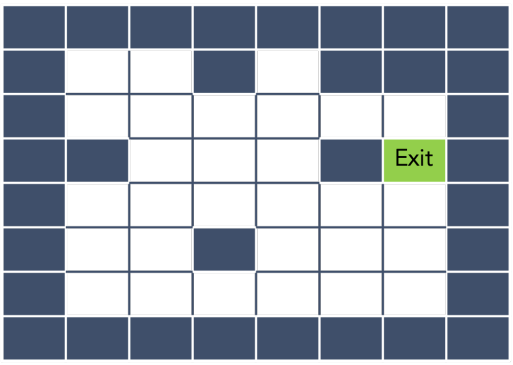


MazeF2

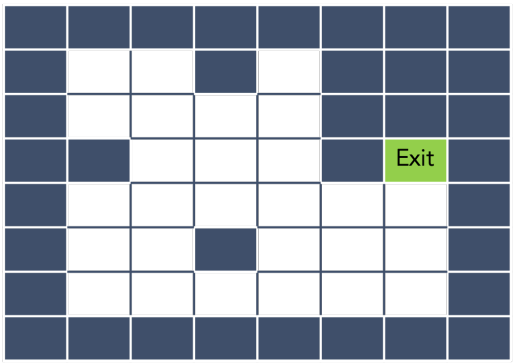


MazeF3

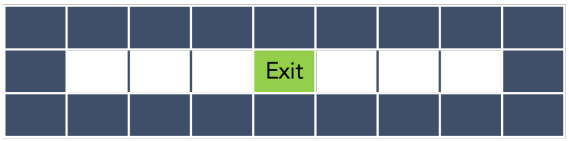
MazeF4



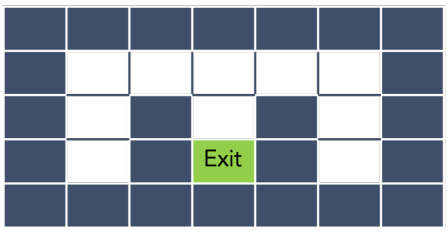
MiyazakiA



MiyazakiB



Woods100



Woods101

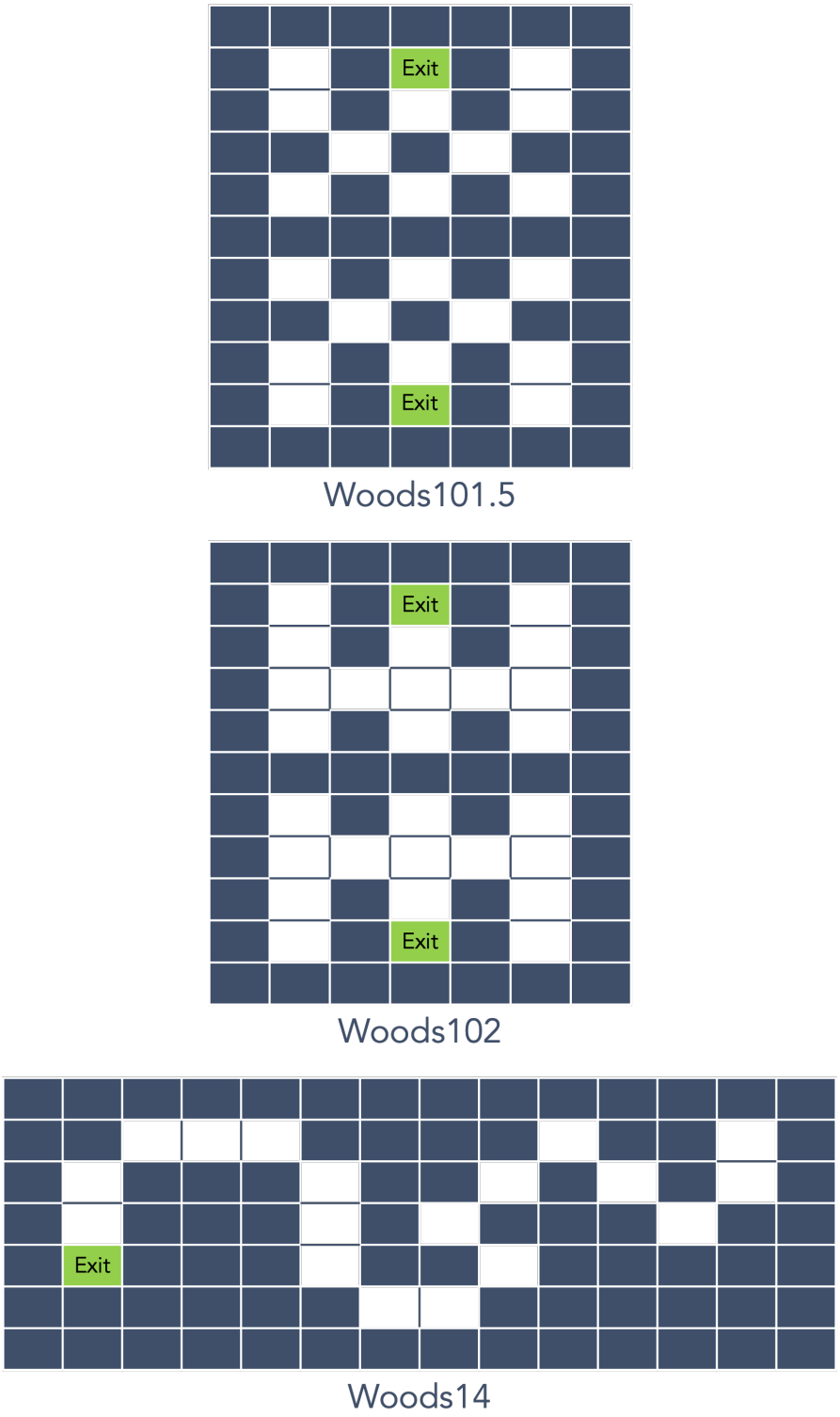


FIGURE A.1 – Illustrations des 23 labyrinthes du banc de test, par ordre alphabétique.
Les cellules bleues grises désignent les murs des labyrinthes. Les cellules vertes désignent la sortie des labyrinthes. Les cellules blanches désignent les chemins.

A.2 Caractéristiques des 23 environnements labyrinthiques

Labyrinthe	Distance à la sortie	Taille	Densité	Complexité	Type de PAI
Cassandra4x4	2.27	36	0.56	1	Pseudo
Littman57	3.71	52	0.71	1	I
Littman89	3.77	63	0.63	< 10	I
Maze4	3.5	64	0.58	1	∅
Maze5	4.61	81	0.59	1	∅
Maze7	4.11	35	0.71	> 100	II
Maze10	4.56	54	0.65	> 150	III
MazeA	4.23	64	0.64	1	∅
MazeB	3.5	64	0.59	< 10	I
MazeD	2.75	64	0.61	< 10	I
MazeF1	1.8	24	0.75	1	∅
MazeF2	2.5	30	0.77	1	∅
MazeF3	3.38	36	0.75	1	∅
MazeF4	4.3	42	0.74	> 125	II
MazeE1	2.45	81	0.44	> 100	III
MazeE2	2.27	81	0.40	251	III
MiyazakiA	3.05	64	0.53	< 10	I
MiyazakiB	3.35	64	0.56	< 10	II
Woods100	2	27	0.74	> 150	III
Woods101	2.7	35	0.69	> 150	III
Woods101.5	2.8	77	0.71	> 225	III
Woods102	2.77	77	0.64	> 150	III
Woods14	9.5	98	0.81	1	∅

Table A.1 – Caractéristiques détaillées des environnements labyrinthiques, par ordre alphabétique.

ANNEXE B

Ressources supplémentaires aux capacités d'apprentissage de BACS

B.1 Équivalence des capacités d'apprentissage de BACS et d'ACS 2 sans incertitude environnementale

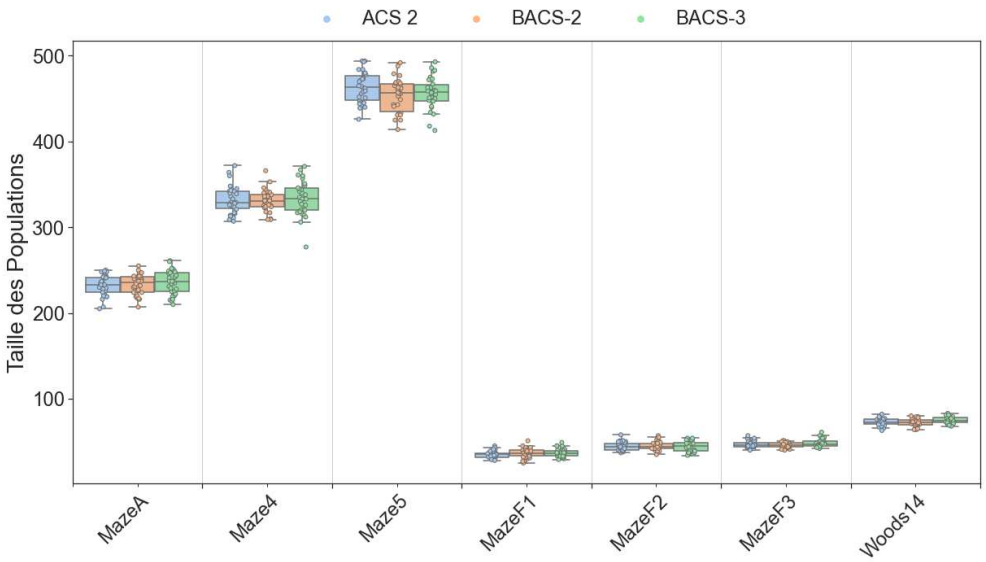


Figure B.1 – Tailles des populations de classeurs de BACS et ACS 2 dans les environnements sans incertitude environnementale.

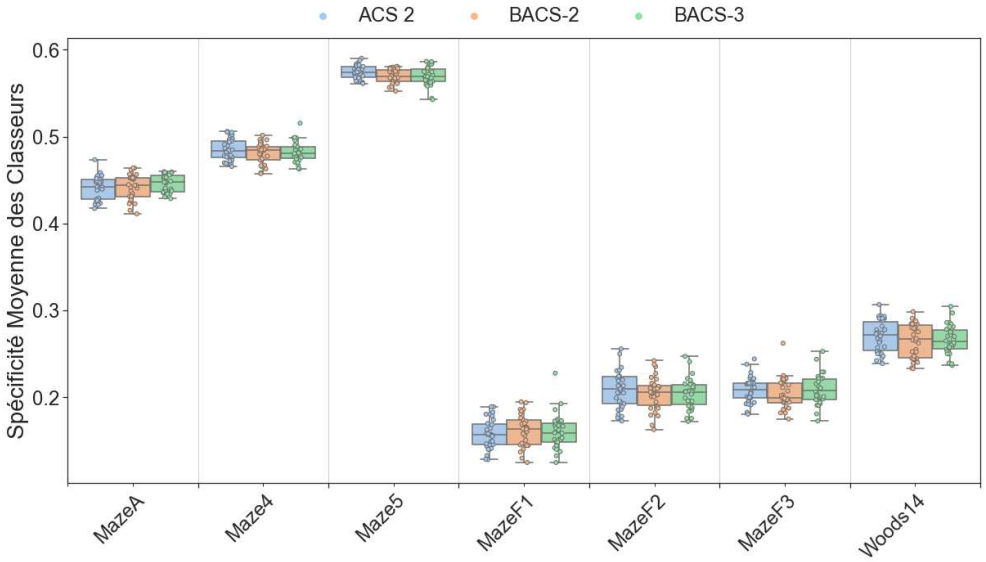


FIGURE B.2 – Spécificités des populations de classeurs de BACS et ACS 2 dans les environnements sans incertitude environnementale.

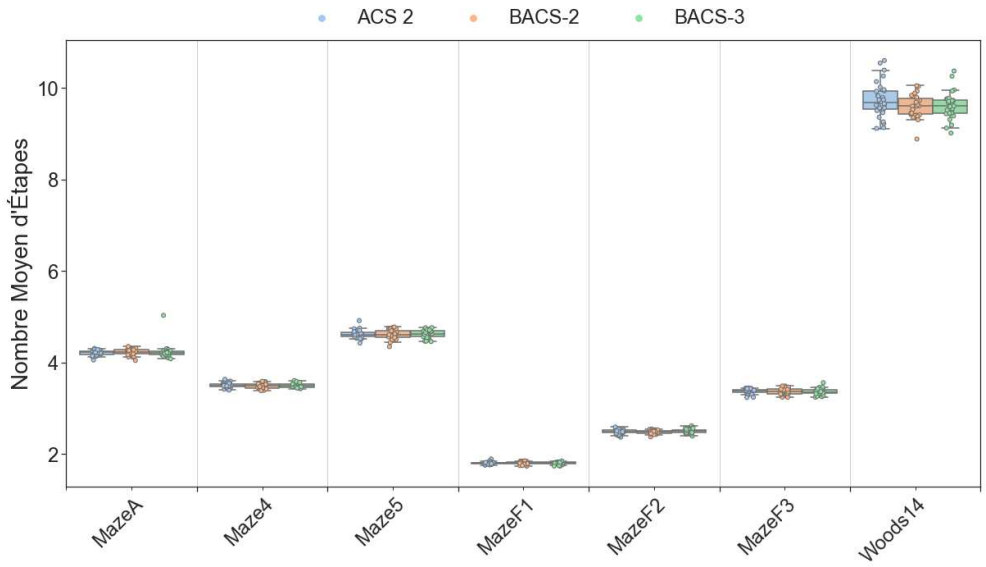


FIGURE B.3 – Nombres moyens d'étapes de BACS et ACS 2 pour atteindre la sortie des environnements sans incertitude environnementale.

B.2 Métriques environnementales de BACS et ACS 2 avec PAI

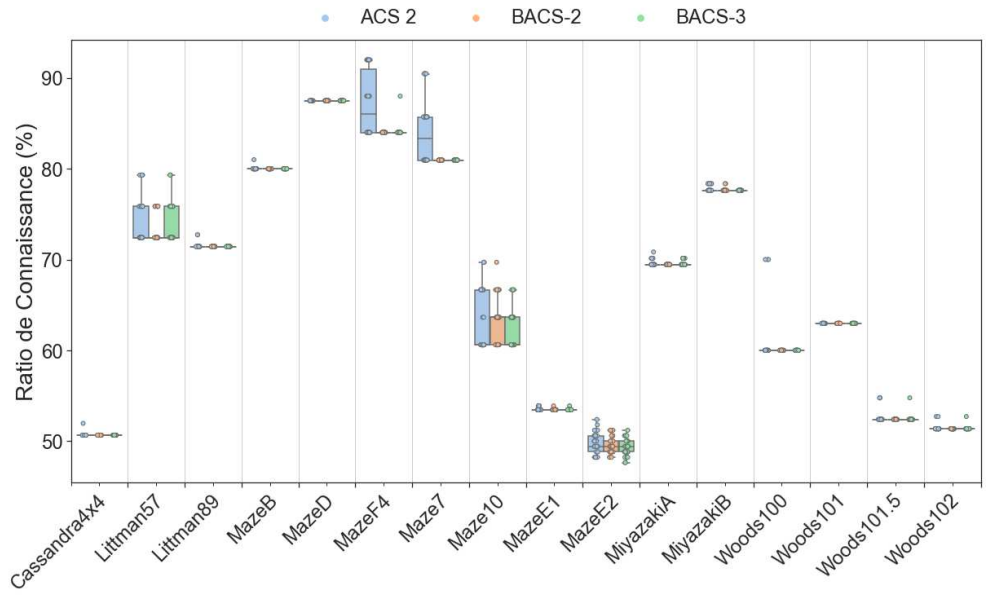


FIGURE B.4 – Ratios de connaissance des populations de classeurs de BACS et ACS 2 dans les environnements avec PAI.

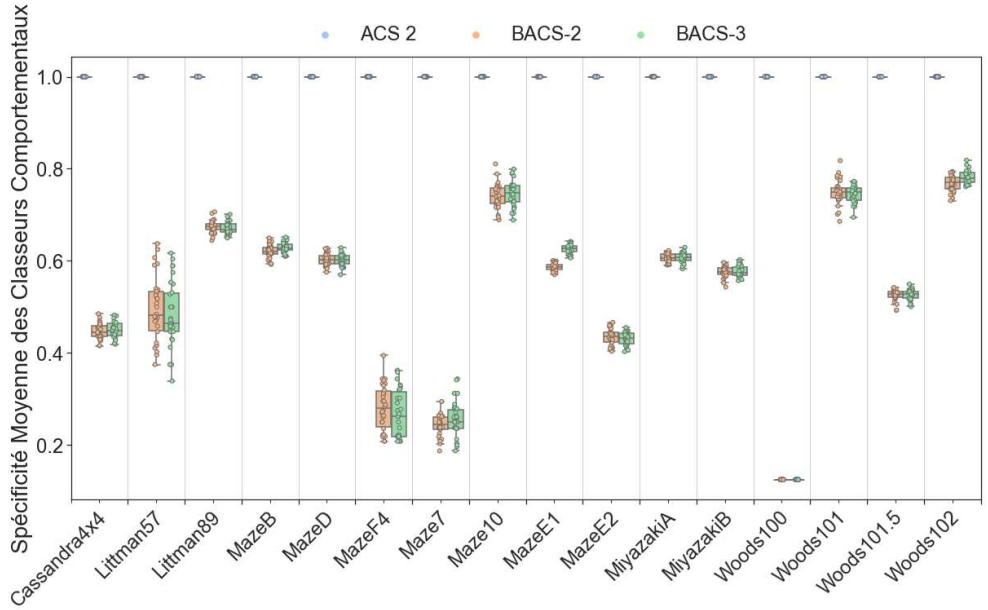


FIGURE B.5 – Spécificités moyennes des classeurs comportementaux de BACS dans les environnements avec PAI.
Puisqu'ACS 2 ne construit aucun classeur comportemental, il n'a pas de spécificité moyenne : elle est ici mise par défaut à 1.

B.3 Métriques environnementales de BACS avec et sans les actions incertaines

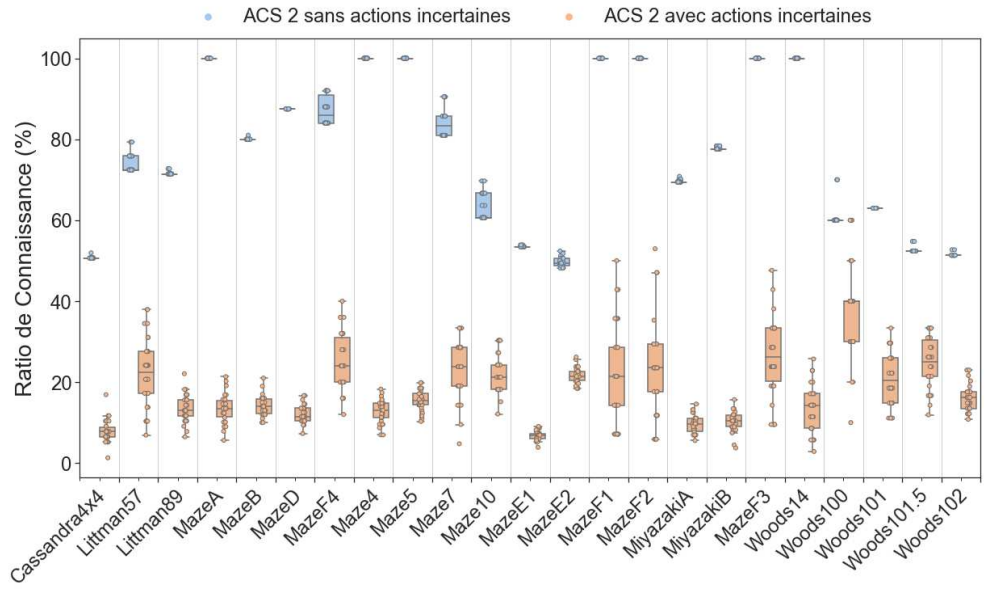


FIGURE B.6 – Ratios de connaissance atteints par ACS 2 avec et sans l'incertitude liée à la réalisation des actions.

FIGURE B.7 – Tailles moyennes des populations de classeurs de ACS 2 avec et sans l'incertitude liée à la réalisation des actions.

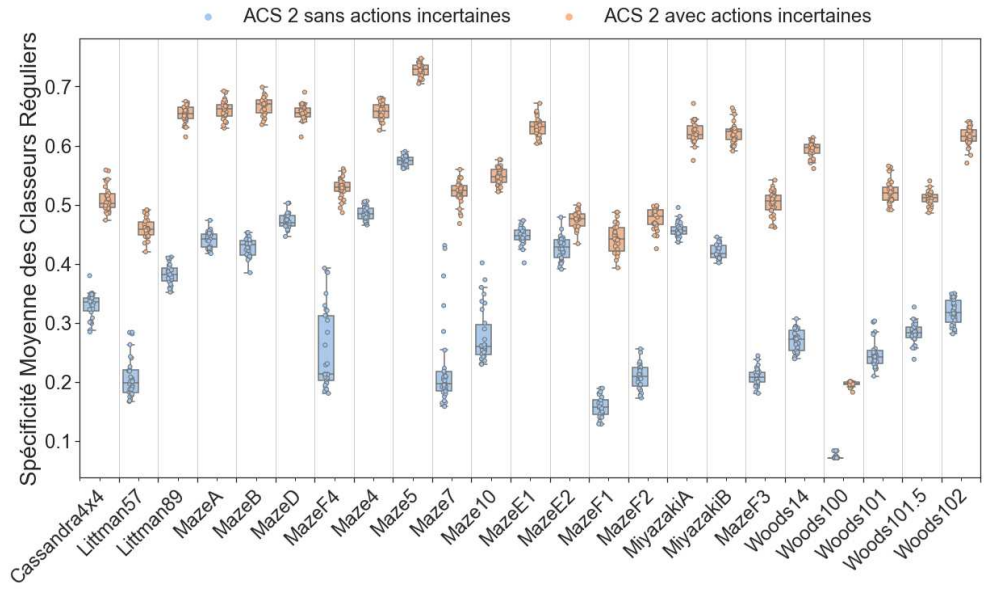


FIGURE B.8 – Spécificités moyennes de classeurs fiables de ACS 2 avec et sans l'incertitude liée à la réalisation des actions.

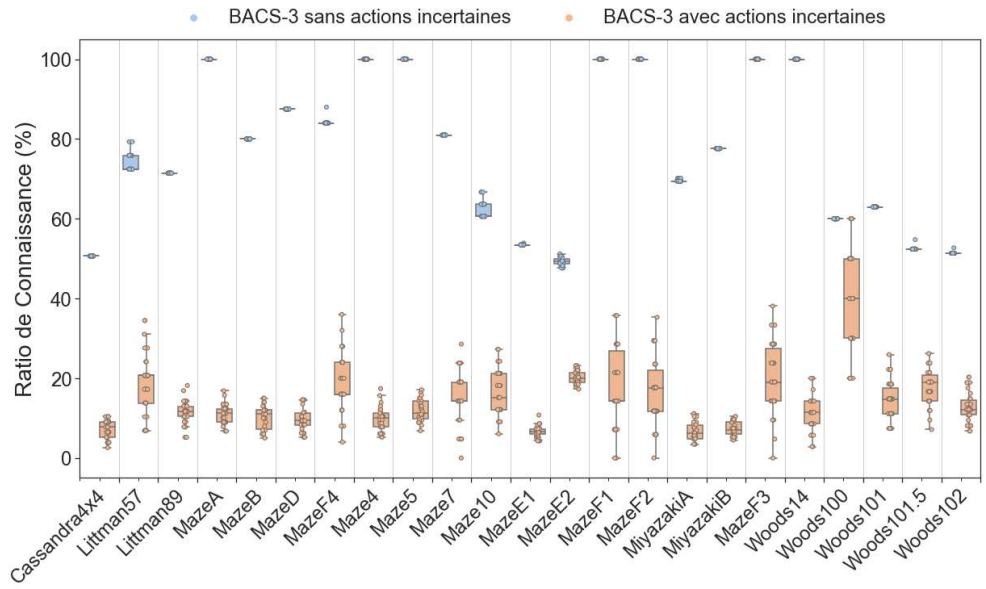


FIGURE B.9 – Ratios de connaissance atteints par BACS-3 avec et sans l'incertitude liée à la réalisation des actions.

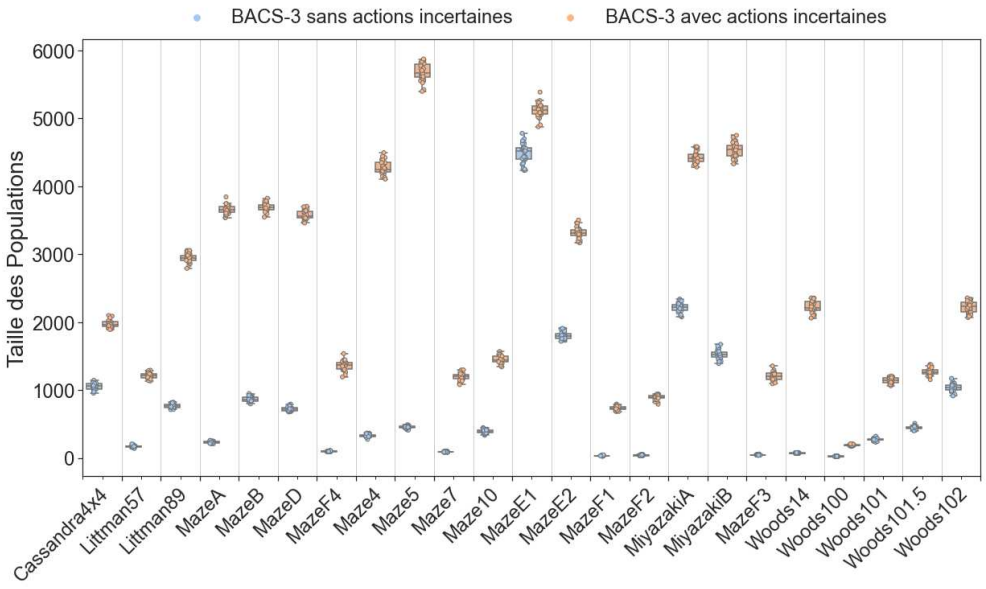


FIGURE B.10 – Tailles moyennes des populations de classeurs de BACS-3 avec et sans l'incertitude liée à la réalisation des actions.

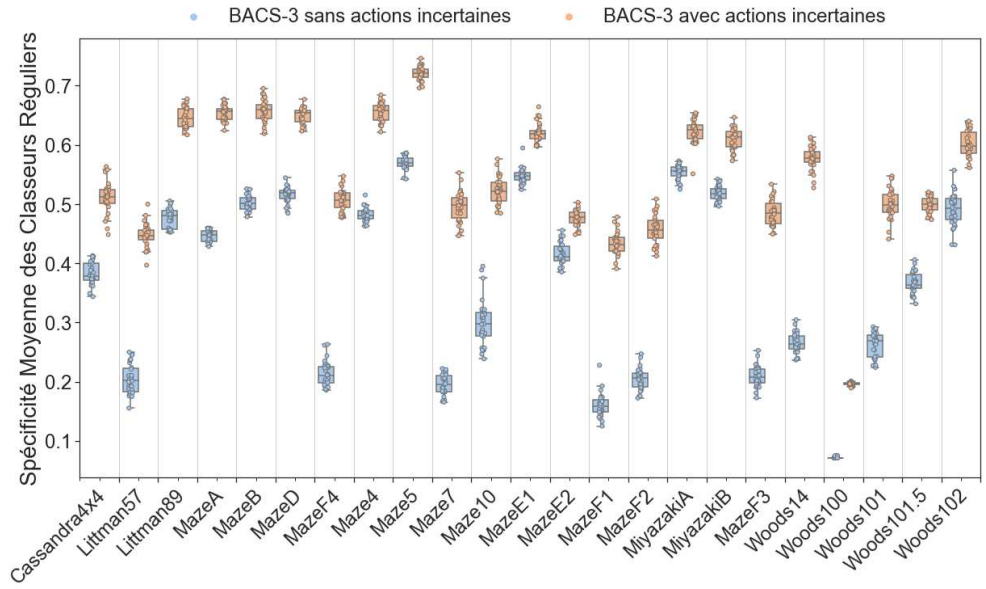


FIGURE B.11 – Spécificités moyennes de classeurs fiables de BACS-3 avec et sans l'incertitude liée à la réalisation des actions.

B.4 Nombres moyens d'étapes de BACS avec et sans les actions incertaines

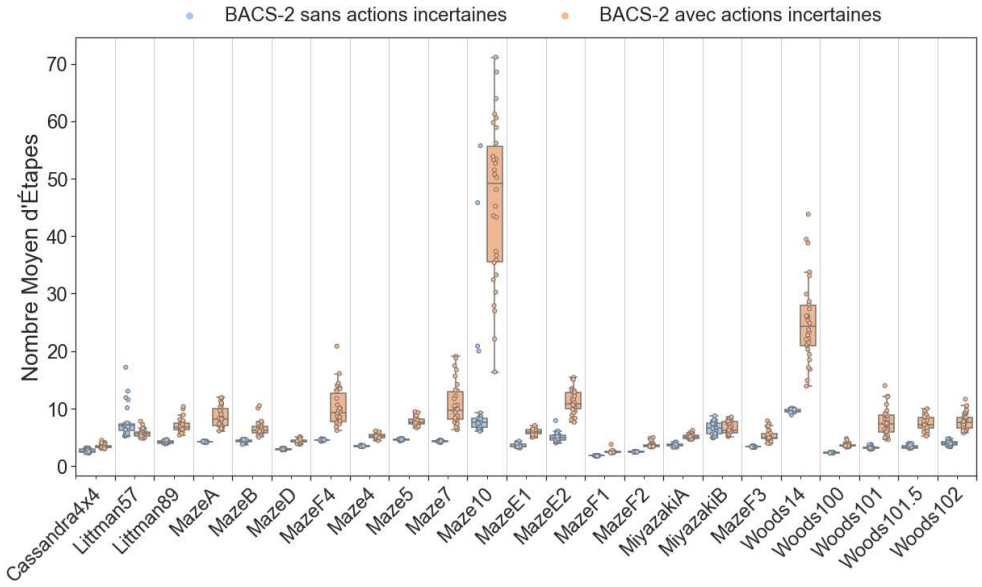


FIGURE B.12 – Nombres moyens d'étapes de BACS-2 pour atteindre la sortie des labyrinthes, avec et sans l'incertitude liée à la réalisation des actions.

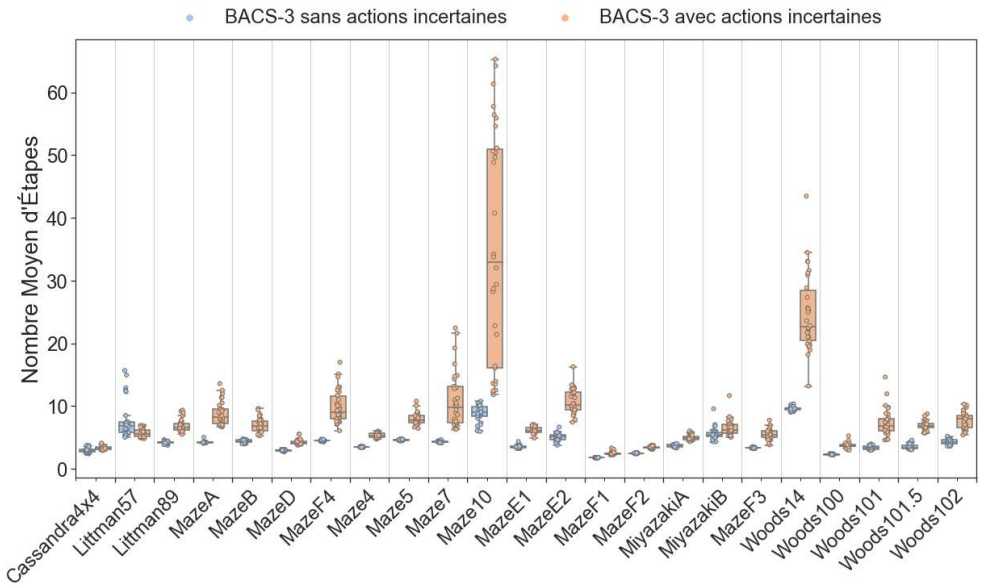


FIGURE B.13 – Nombres moyens d'étapes de BACS-3 pour atteindre la sortie des labyrinthes, avec et sans l'incertitude liée à la réalisation des actions.

B.5 Tailles et spécificités des populations de classeurs d'ACS 2 et de BACS avec les actions incertaines

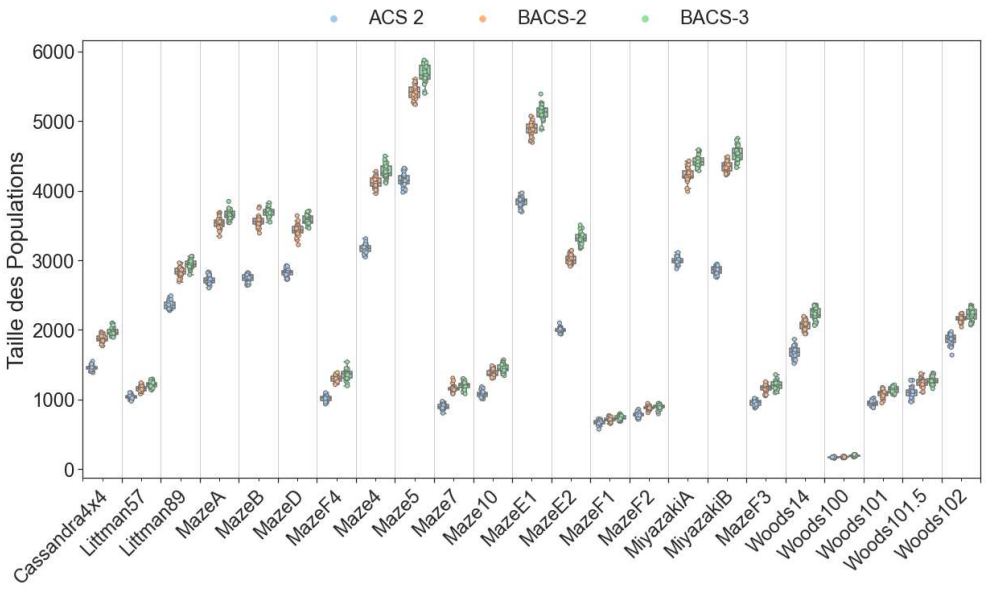


FIGURE B.14 – Tailles moyennes des populations de classeurs d'ACS 2 et de BACS avec l'incertitude liée à la réalisation des actions.

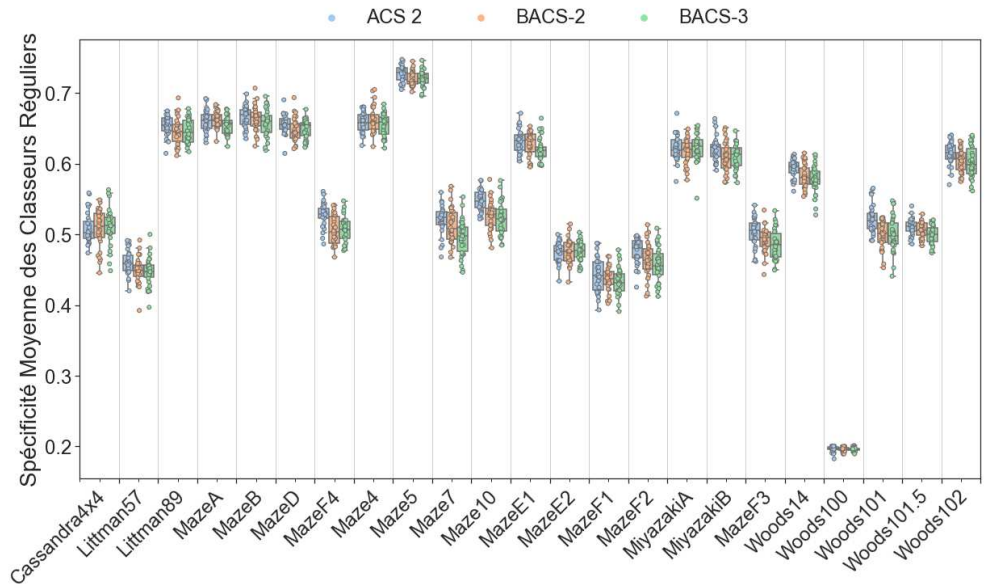


FIGURE B.15 – Spécificités moyennes de classeurs fiables d'ACS 2 et de BACS avec l'incertitude liée à la réalisation des actions.

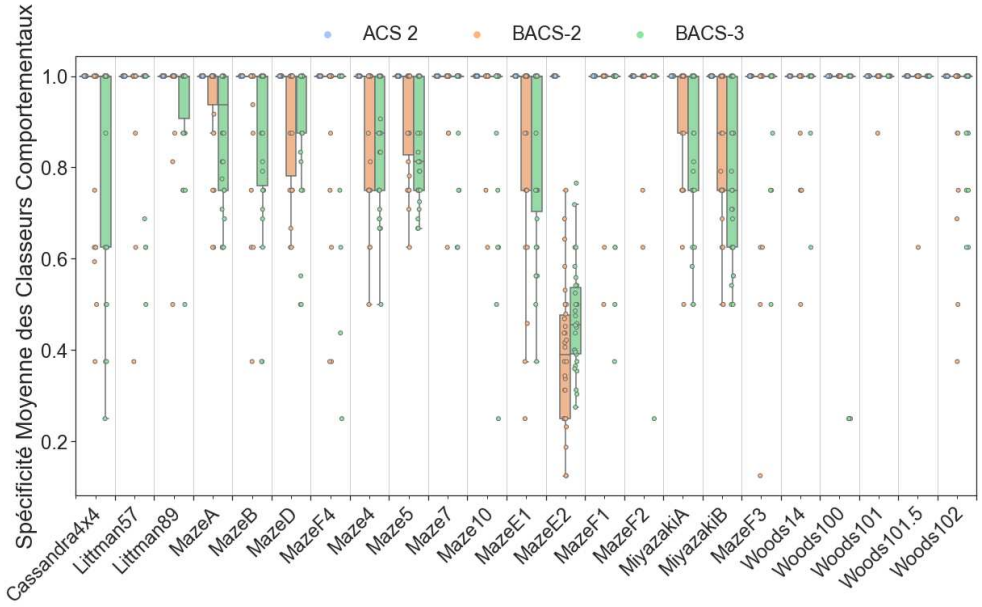


FIGURE B.16 – Spécificités moyennes de classeurs comportementaux de BACS avec l'incertitude liée à la réalisation des actions.
Puisqu'ACS 2 ne construit aucun classeur comportemental, il n'a pas de spécificité moyenne : elle est ici mise par défaut à 1.

ANNEXE C

Ressources supplémentaires aux capacités d'apprentissage de PEPACS

C.1 Équivalence des capacités d'apprentissage de PEPACS et d'ACS 2 sans incertitude environnementale

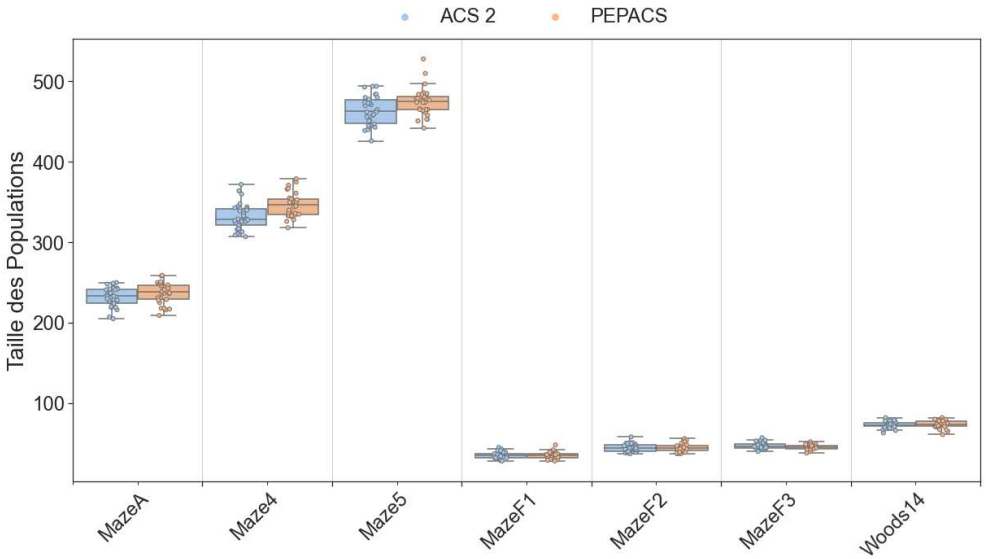


Figure C.1 – Tailles des populations de classeurs de PEPACS et ACS 2 dans les environnements sans incertitude environnementale.

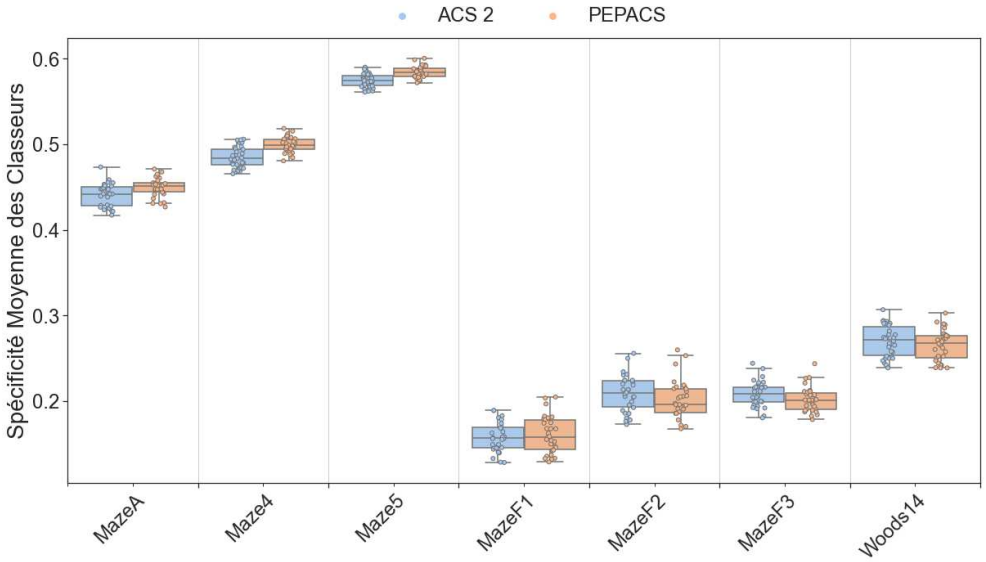


FIGURE C.2 – Spécificités des populations de classeurs de PEPACS et ACS 2 dans les environnements sans incertitude environnementale.

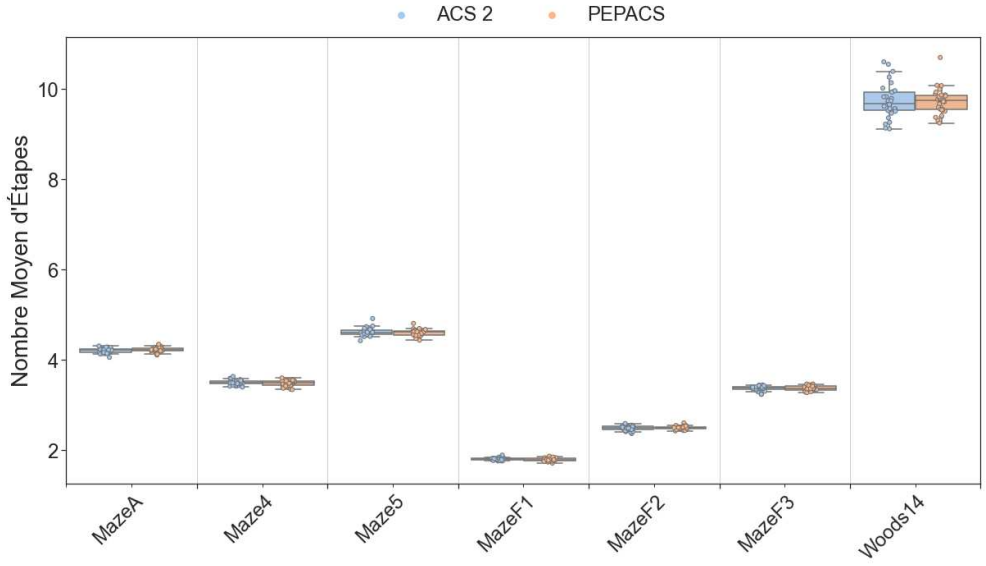


FIGURE C.3 – Nombres moyens d'étapes de PEPACS et ACS 2 pour atteindre la sortie les environnements sans incertitude environnementale.

C.2 Métriques environnementales de PEPACS et d'ACS 2 sans les actions incertaines

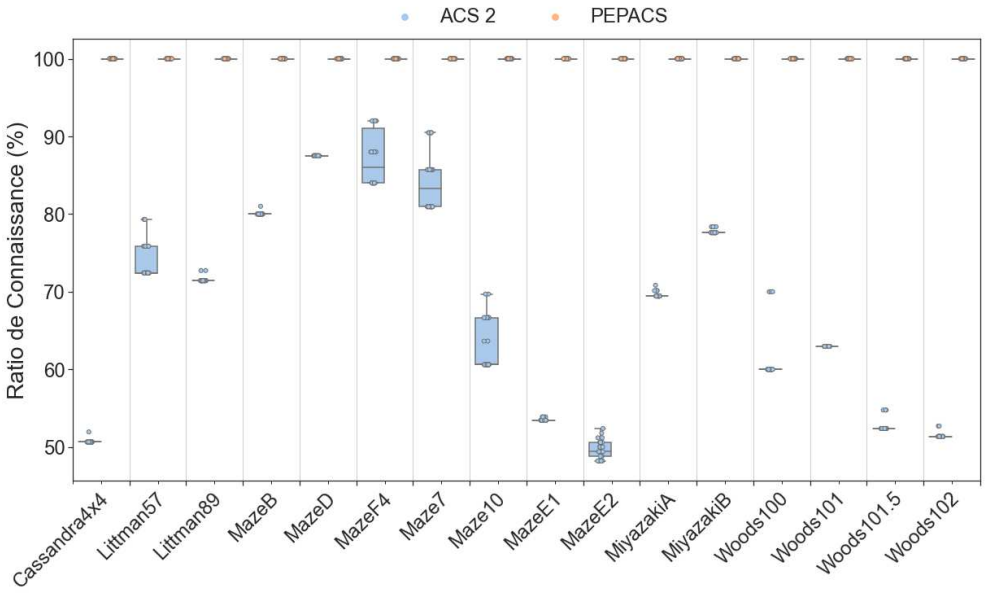


FIGURE C.4 – Ratios de connaissance des populations de classeurs de PEPACS et ACS 2 dans les environnements avec PAI et sans les actions incertaines.

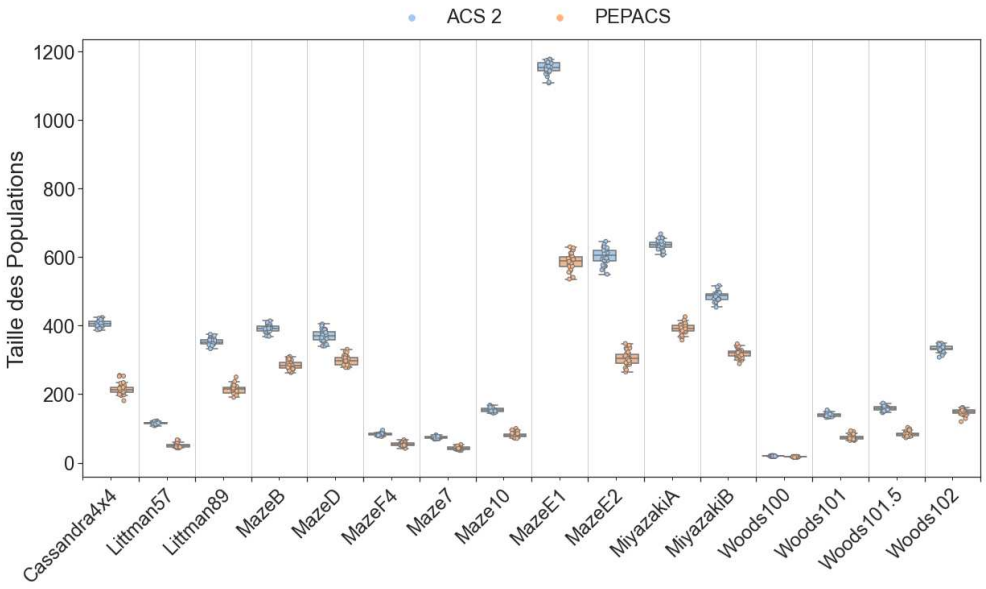


FIGURE C.5 – Tailles des populations de classeurs de PEPACS et ACS 2 dans les environnements avec PAI et sans les actions incertaines.

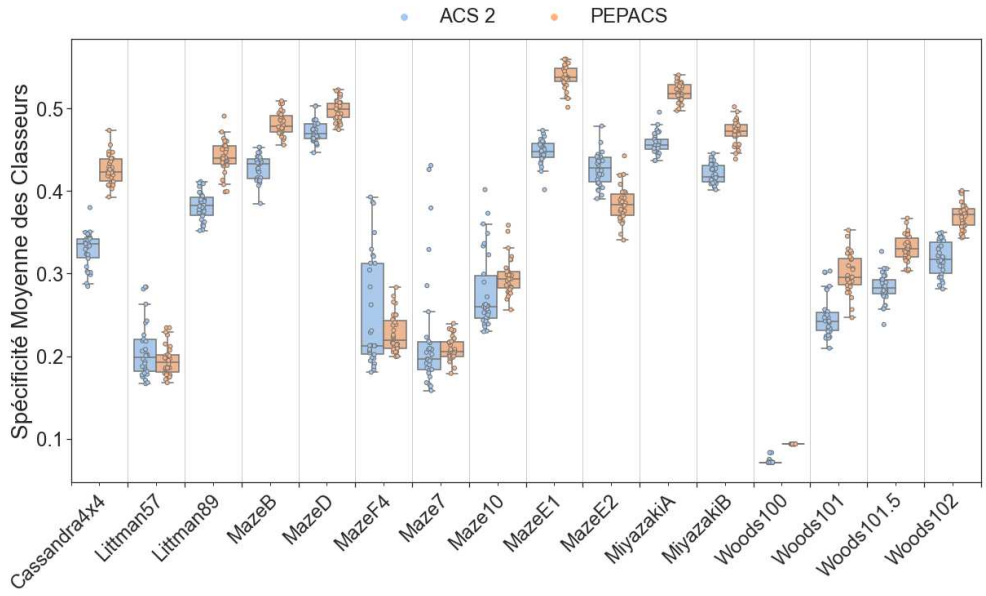


FIGURE C.6 – Spécificités moyennes des classeurs fiables de PEPACS et ACS 2 dans les environnements avec PAI et sans les actions incertaines.

C.3 Métriques environnementales de PEPACS selon la copie spécifiée

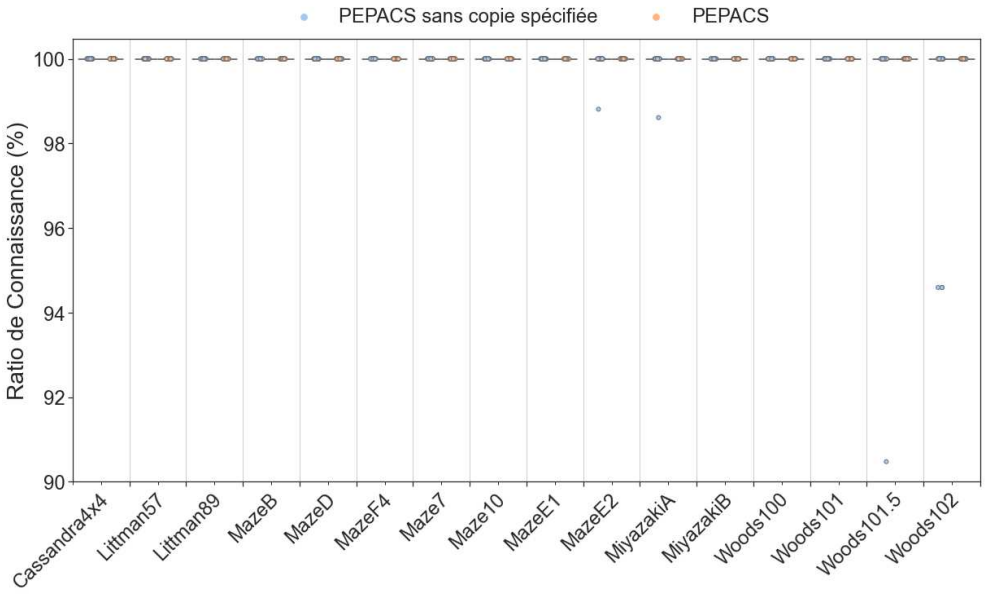


FIGURE C.7 – Ratios de connaissances atteints par PEPACS dans les environnements avec PAI et sans les actions incertaines.

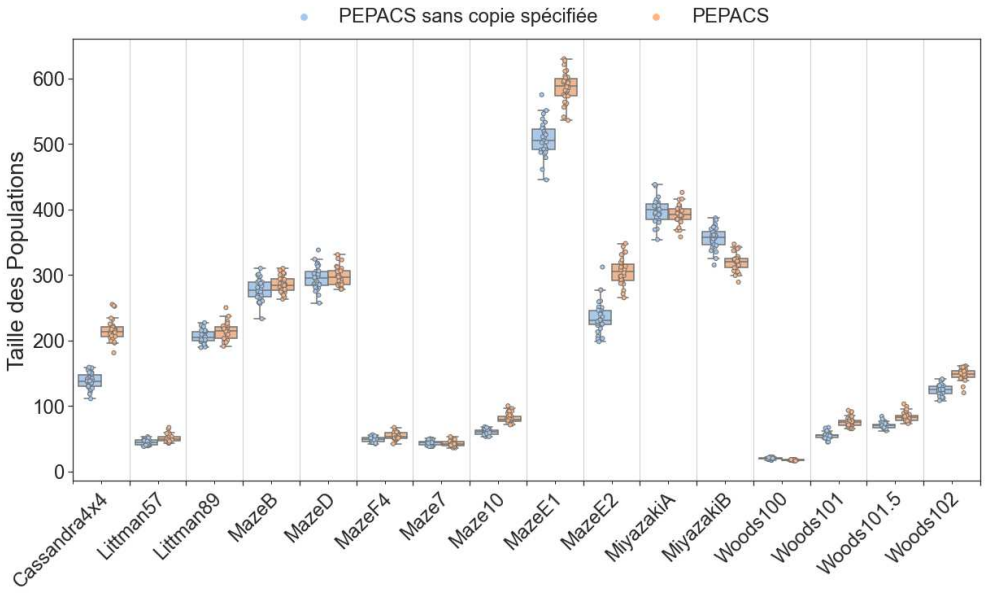


FIGURE C.8 – Tailles des populations de classeurs de PEPACS dans les environnements avec PAI et sans les actions incertaines.

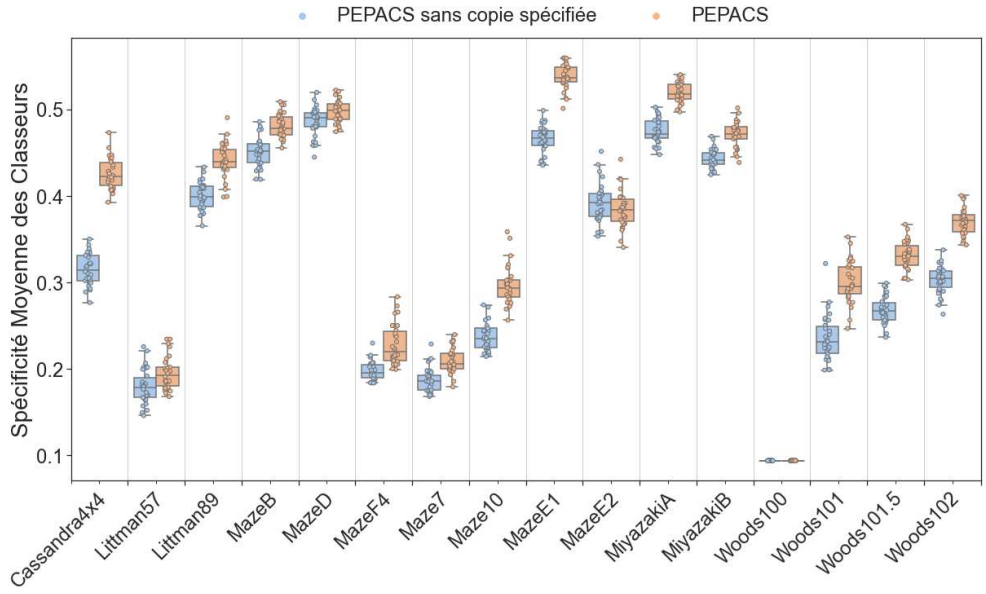


FIGURE C.9 – Spécificités moyennes des classeurs fiables de PEPACS dans les environnements avec PAI et sans les actions incertaines.

C.4 Métriques environnementales de PEPACS comparées avec BACS et ACS 2

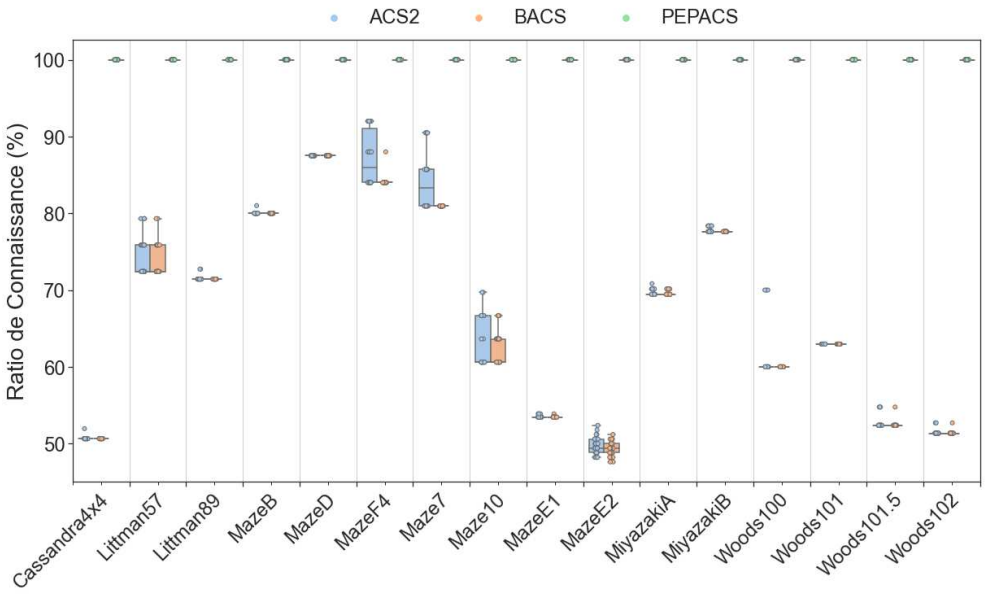


FIGURE C.10 – Ratios de connaissances atteints par PEPACS, BACS et ACS 2 dans les environnements avec PAI et sans les actions incertaines.

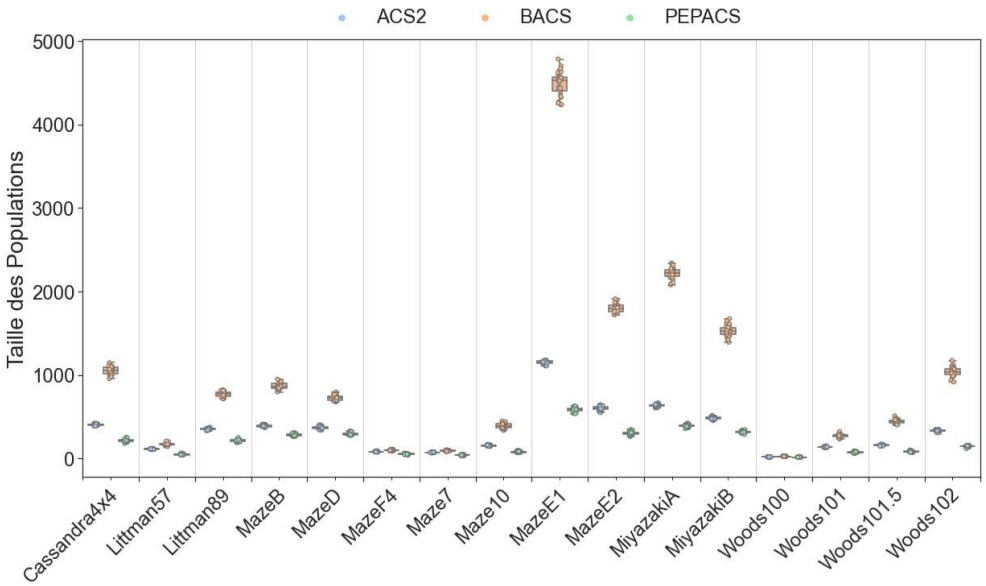


FIGURE C.11 – Tailles des populations de classeurs de PEPACS, BACS et ACS 2 dans les environnements avec PAI et sans les actions incertaines.

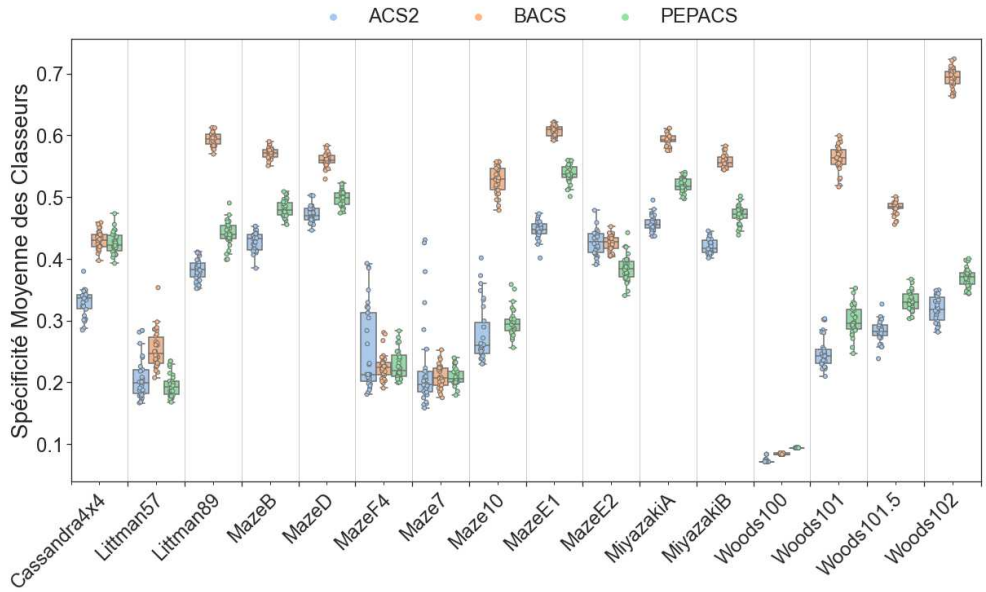


FIGURE C.12 – Spécificités moyennes des classeurs fiables de PEPACS, BACS et ACS 2 dans les environnements avec PAI et sans les actions incertaines.

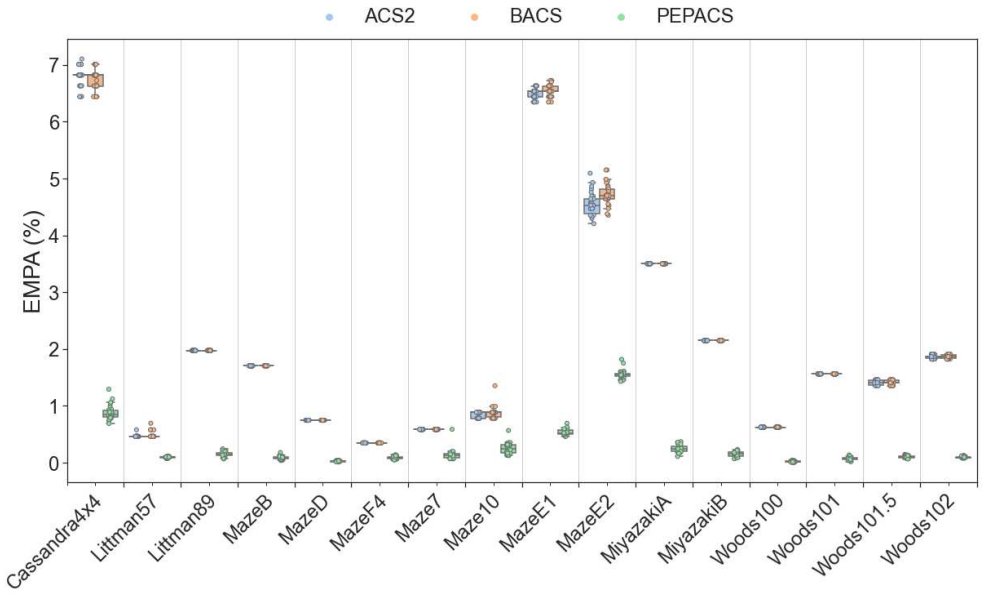


FIGURE C.13 – Erreurs moyennes sur la Prédiction de l'Anticipation de PEPACS, BACS et ACS 2 dans les environnements avec PAI et sans les actions incertaines.

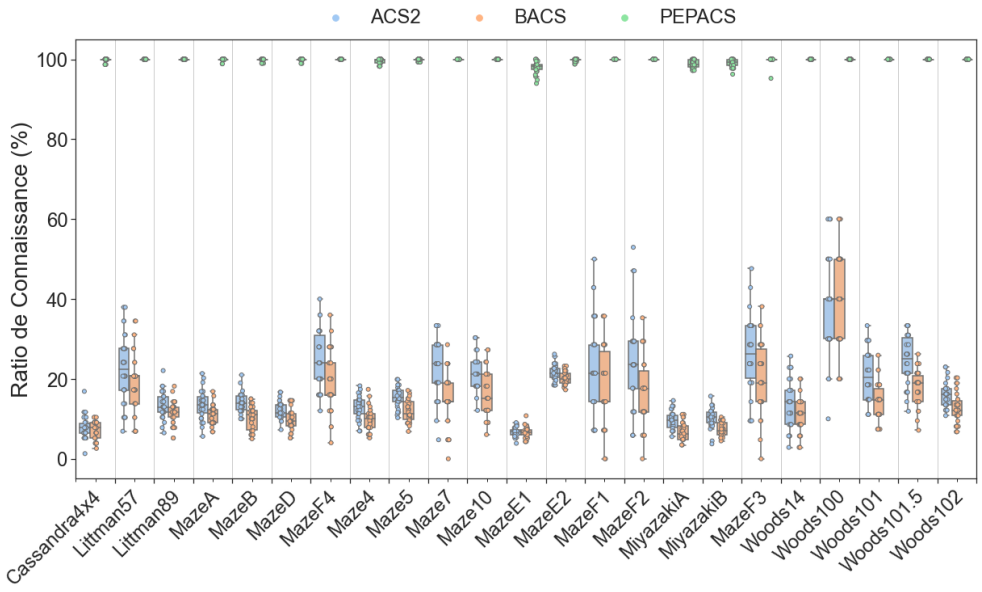


FIGURE C.14 – Ratios de connaissances atteints par PEPACS, BACS et ACS 2 dans tous les environnements avec les actions incertaines.

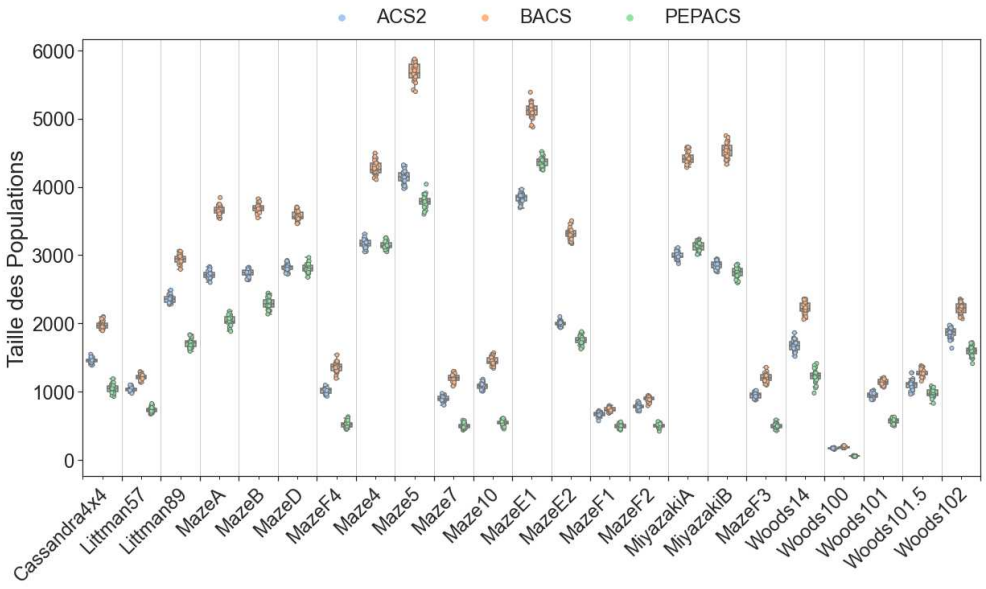


FIGURE C.15 – Tailles des populations de classeurs de PEPACS, BACS et ACS 2 dans tous les environnements avec les actions incertaines.

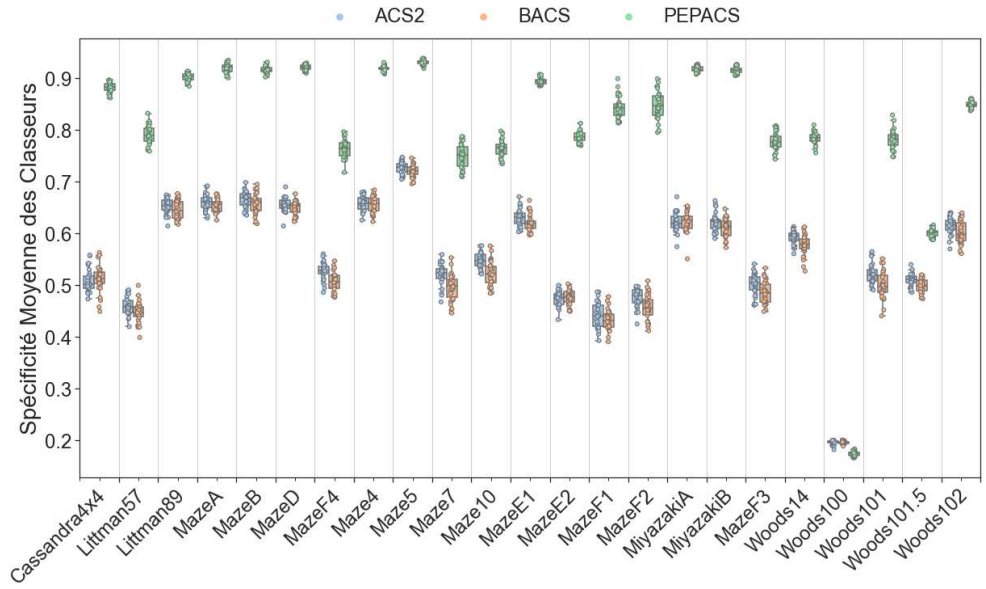


FIGURE C.16 – Spécificités moyennes des classeurs fiables de PEPACS, BACS et ACS 2 dans tous les environnements avec les actions incertaines.

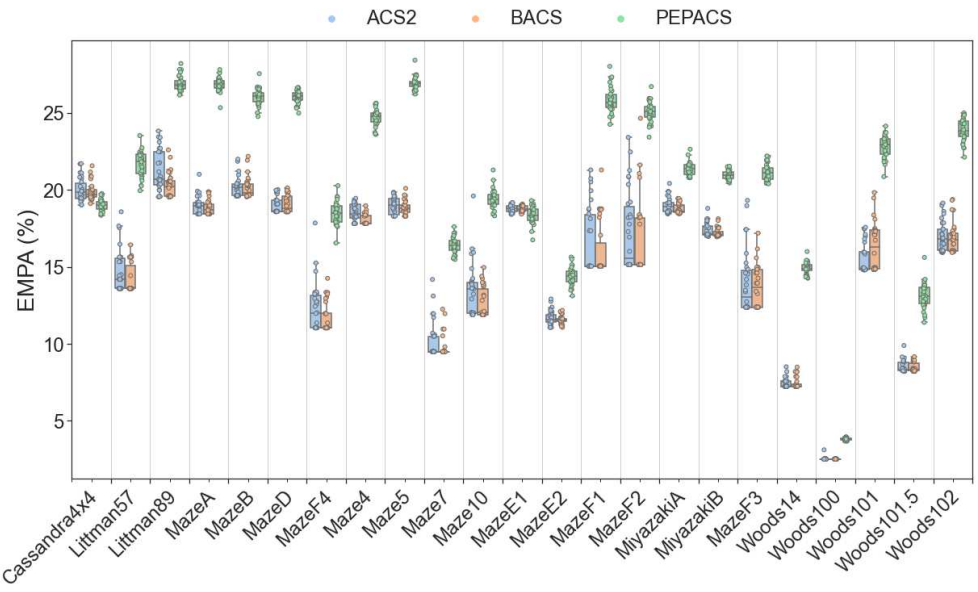


FIGURE C.17 – Erreurs moyennes sur la Prédiction de l'Anticipation de PEPACS, BACS et ACS 2 dans tous les environnements avec les actions incertaines.

ANNEXE D

Ressources supplémentaires aux capacités d'apprentissage de BEACS

D.1 Capacités d'apprentissage de BEACS et d'ACS 2 face à l'incertitude environnementale

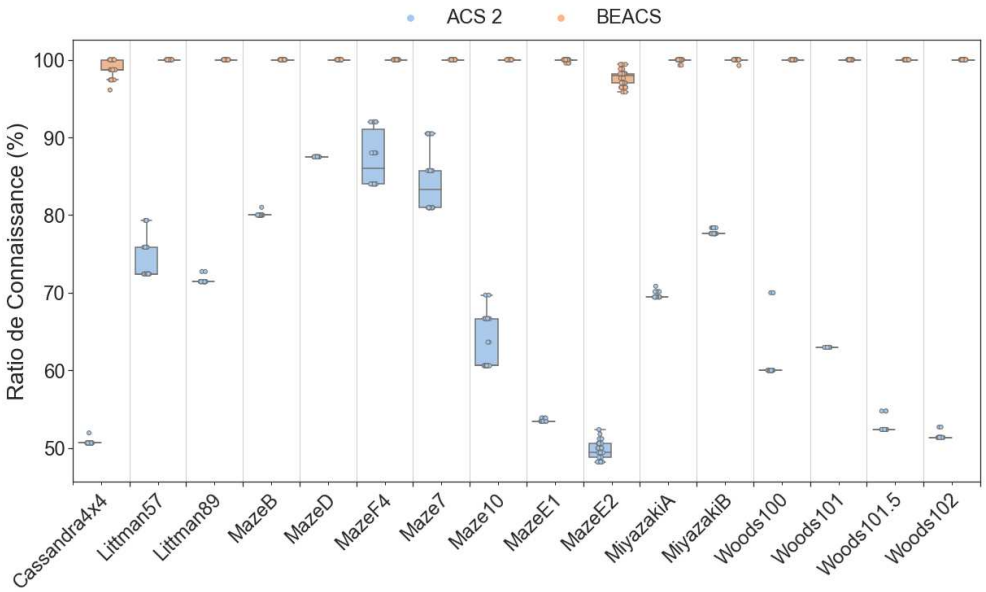


Figure D.1 – Ratios de connaissance de BEACS et ACS 2 dans les environnements avec PAI et sans les actions incertaines.

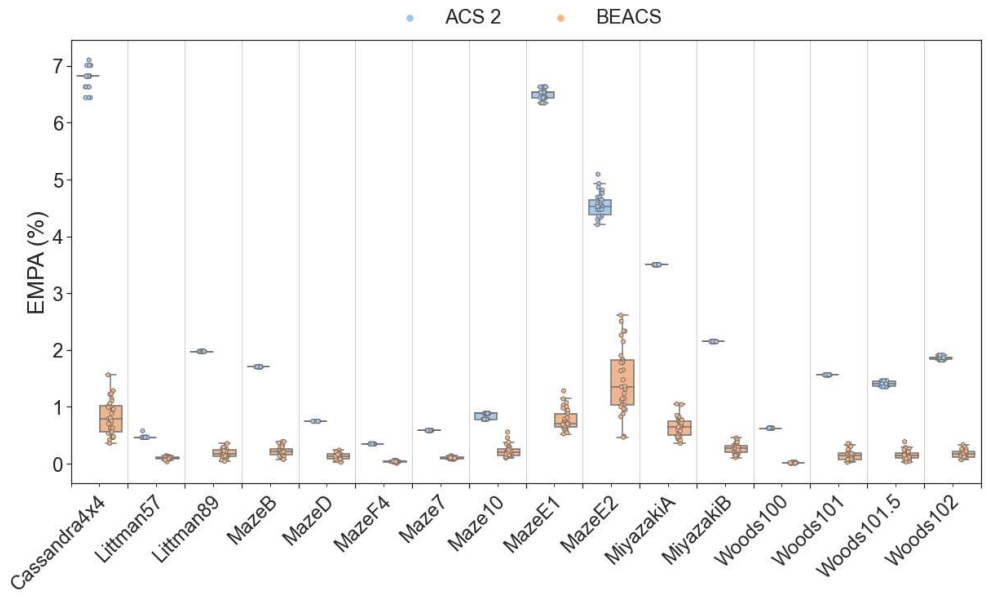


FIGURE D.2 – Erreurs Moyennes sur la Prédiction de l'Anticipation de BEACS et ACS 2 dans les environnements avec PAI et sans les actions incertaines.

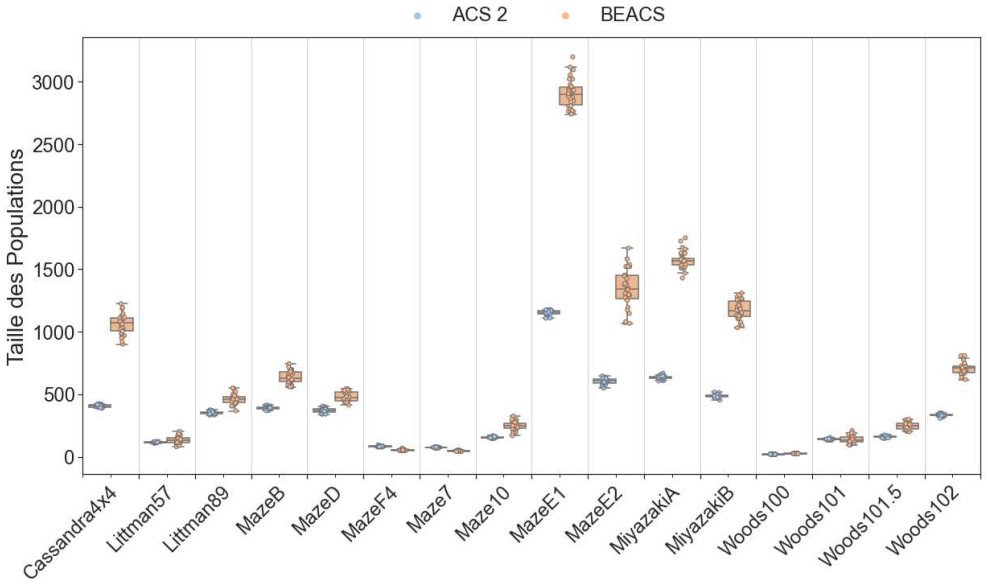


FIGURE D.3 – Tailles des populations de classeurs de BEACS et ACS 2 dans les environnements avec PAI et sans les actions incertaines.

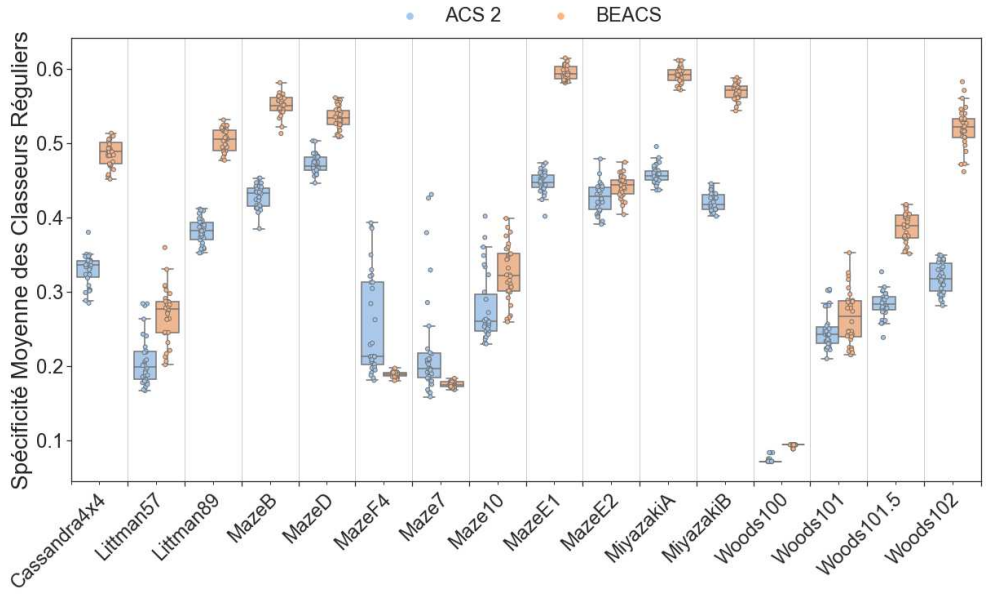


FIGURE D.4 – Spécificités moyennes des classeurs réguliers fiables de BEACS et ACS 2 dans les environnements avec PAI et sans les actions incertaines.

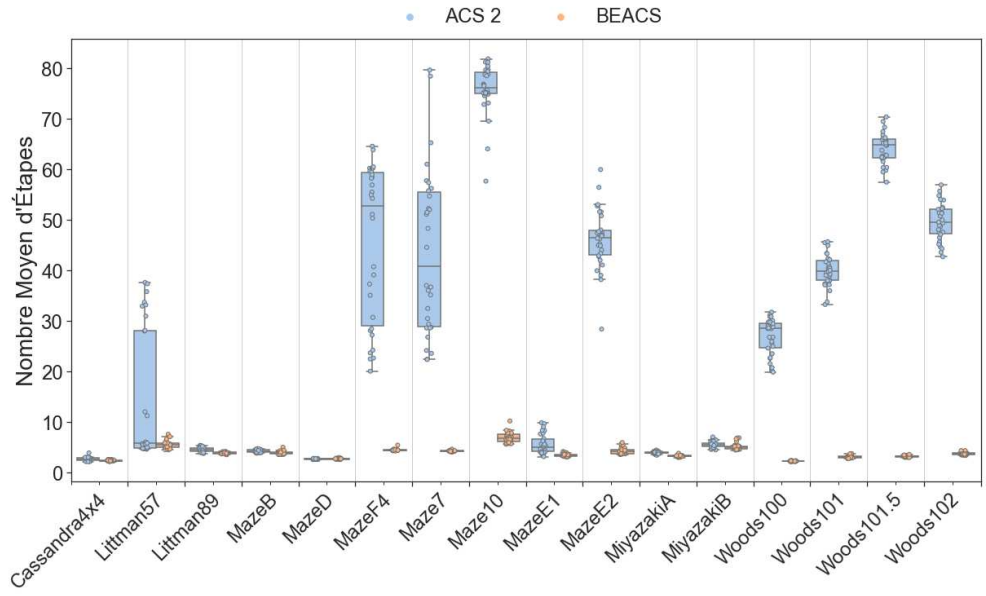


FIGURE D.5 – Nombres moyens d'étapes de BEACS et ACS 2 pour atteindre la sortie les environnements avec PAI et sans les actions incertaines.

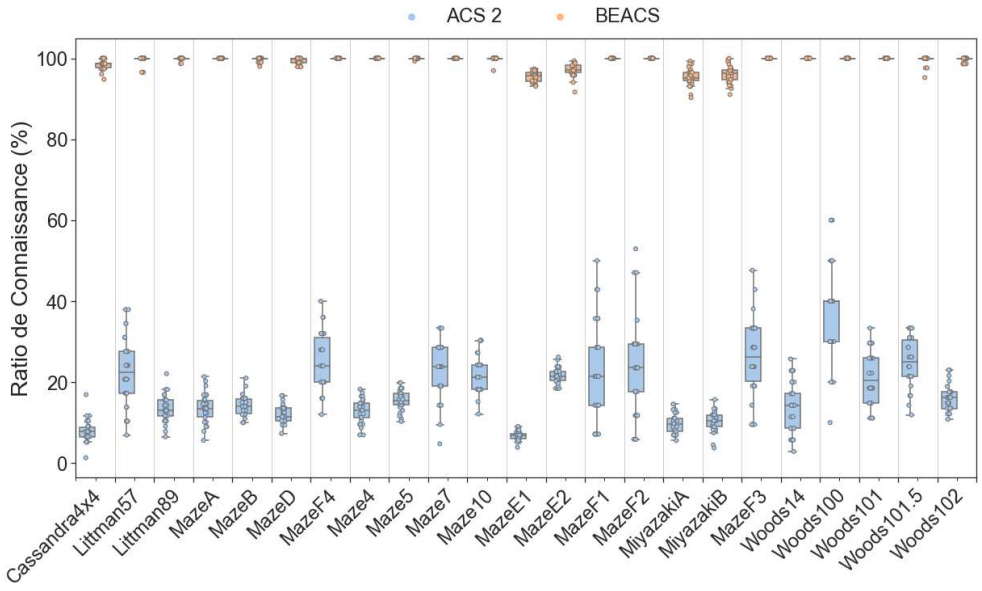


FIGURE D.6 – Ratios de connaissance de BEACS et ACS 2 dans tous les environnements avec les actions incertaines.

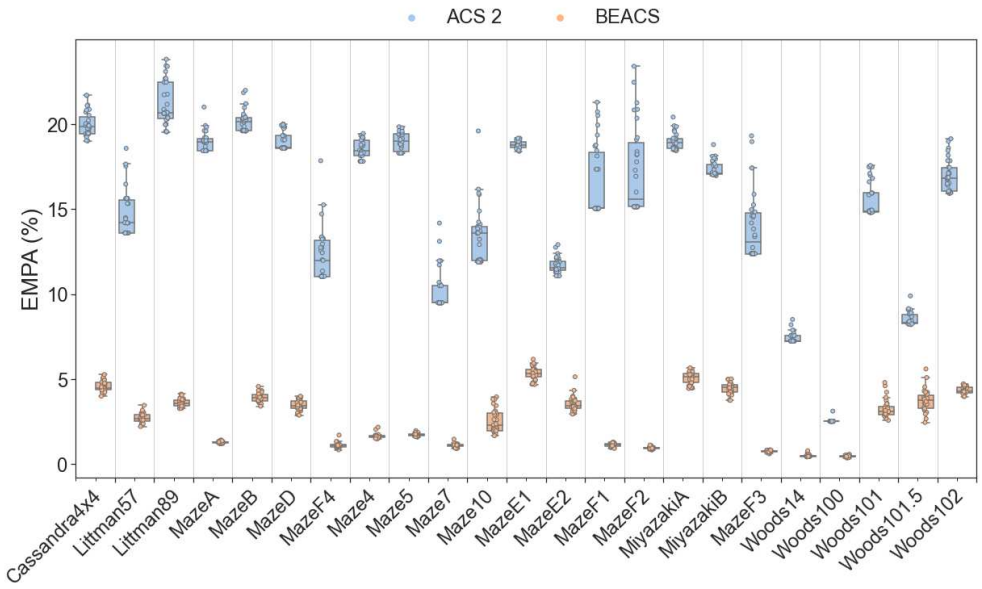


FIGURE D.7 – Erreurs Moyennes sur la Prédiction de l'Anticipation de BEACS et ACS 2 dans tous les environnements avec les actions incertaines.

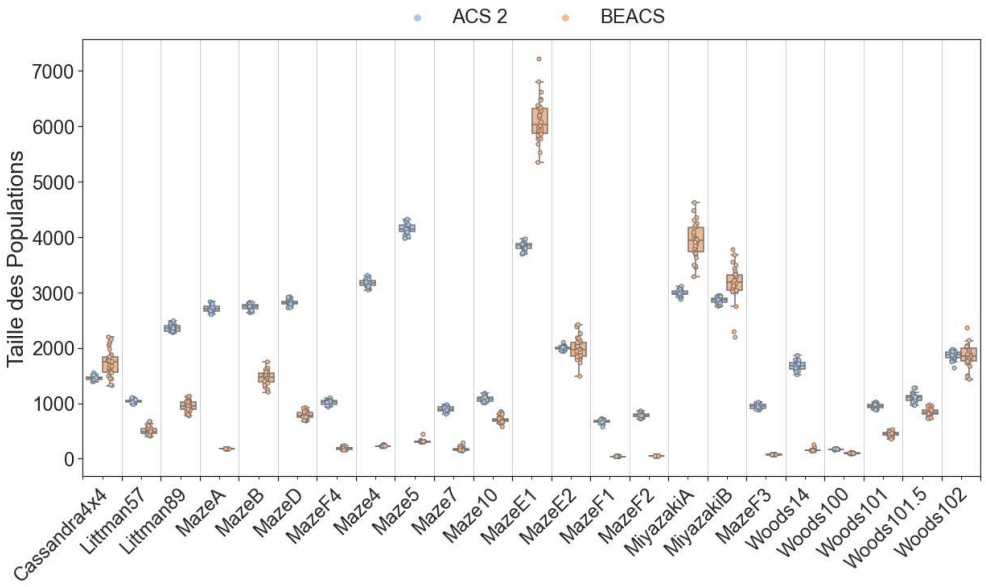


FIGURE D.8 – Tailles des populations de classeurs de BEACS et ACS 2 dans tous les environnements avec les actions incertaines.

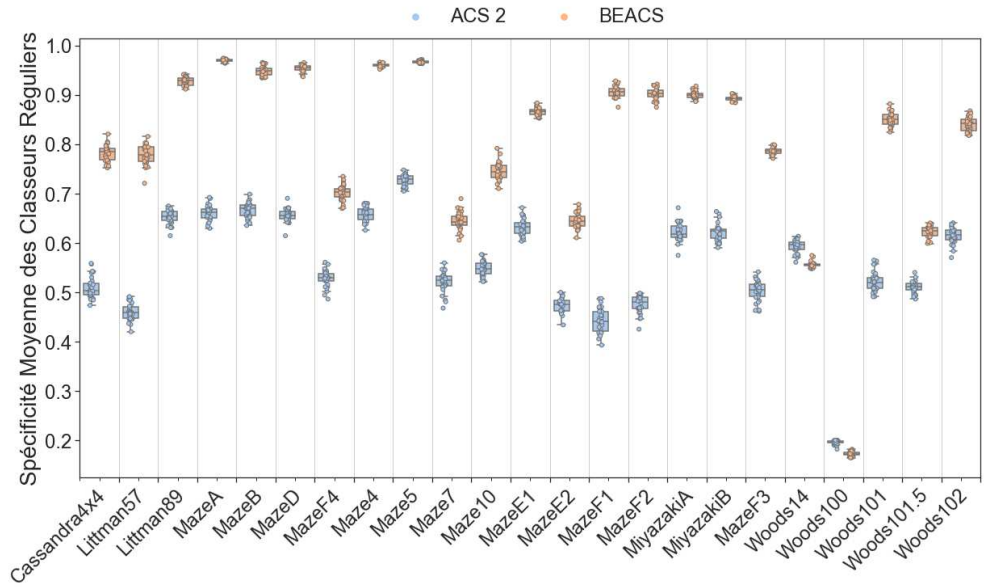


FIGURE D.9 – Spécificités moyennes des classeurs réguliers fiables de BEACS et ACS 2 dans tous les environnements avec les actions incertaines.

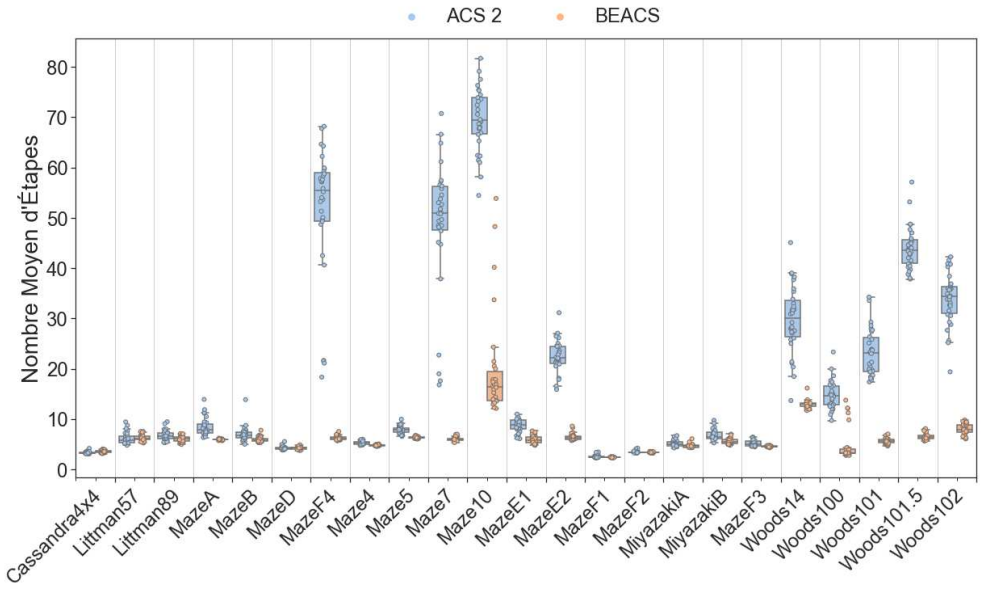


FIGURE D.10 – Nombres moyens d'étapes de BEACS et ACS 2 pour atteindre la sortie de tous les environnements avec les actions incertaines.

D.2 Opérateurs de mutation de BEACS

Algorithme D.2.1 Mutation des classeurs de BEACS

```
1 : function MUTATE(cl_1, cl_2, μ, L)
2 :     for i ← 0 to L − 1 do
3 :         if C_cl1[i] ≠ # and C_cl2[i] == # then
4 :             if random number in [0, 1[ < μ and μT_cl1 == True then
5 :                 C_cl1[i] ← #
6 :             end if
7 :             continue
8 :         end if
9 :         if C_cl1[i] == # and C_cl2[i] ≠ # then
10 :            if random number in [0, 1[ < μ and μT_cl2 == True then
11 :                C_cl2[i] ← #
12 :            end if
13 :            continue
14 :        end if
15 :        if cl_1 is not a behavioral classifier and C_cl1[i] ≠ # and
              random number in [0, 1[ < μ and μT_cl1 == True then
16 :            C_cl1[i] ← #
17 :        end if
18 :        if cl_2 is not a behavioral classifier and C_cl2[i] ≠ # and
              random number in [0, 1[ < μ and μT_cl2 == True then
19 :            C_cl2[i] ← #
20 :        end if
21 :    end for
```

Algorithme D.2.2 Mutation des classeurs de BEACS-DmuT

```
1 : function MUTATE(cl_1, cl_2, μ, L)
2 :     for i ← 0 to L − 1 do
3 :         if C_cl1[i] ≠ # and random number in [0, 1[ < μ and μT_cl1 == True then
4 :             C_cl1[i] ← #
5 :         end if
6 :         if C_cl2[i] ≠ # and random number in [0, 1[ < μ and μT_cl2 == True then
7 :             C_cl2[i] ← #
8 :         end if
9 :     end for
```

Algorithme D.2.3 Mutation des classeurs de BEACS-IndNomut

```
1 : function MUTATE(cl_1, cl_2, μ, L)
2 :     for i ← 0 to L − 1 do
3 :         if C_cl1[i] ≠ # and C_cl2[i] == # then
4 :             if random number in [0, 1[ < μ then
5 :                 C_cl1[i] ← #
6 :             end if
7 :             continue
8 :         end if
9 :         if C_cl1[i] == # and C_cl2[i] ≠ # then
10 :            if random number in [0, 1[ < μ then
11 :                C_cl2[i] ← #
12 :            end if
13 :            continue
14 :        end if
15 :        if cl_1 is not a behavioral classifier and
              C_cl1[i] ≠ # and random number in [0, 1[ < μ then
16 :            C_cl1[i] ← #
17 :        end if
18 :        if cl_2 is not a behavioral classifier and
              C_cl2[i] ≠ # and random number in [0, 1[ < μ then
19 :            C_cl2[i] ← #
20 :        end if
21 :    end for
```

Algorithme D.2.4 Mutation des classeurs de BEACS-DNomut

```
1 : function MUTATE(cl_1, cl_2, μ, L)
2 :     for i ← 0 to L − 1 do
3 :         if C_cl1[i] ≠ # and random number in [0, 1[ < μ then
4 :             C_cl1[i] ← #
5 :         end if
6 :         if C_cl2[i] ≠ # and random number in [0, 1[ < μ then
7 :             C_cl2[i] ← #
8 :         end if
9 :     end for
```

D.3 Capacités d'apprentissage de BEACS et d'ACS 2 sans incertitude environnementale

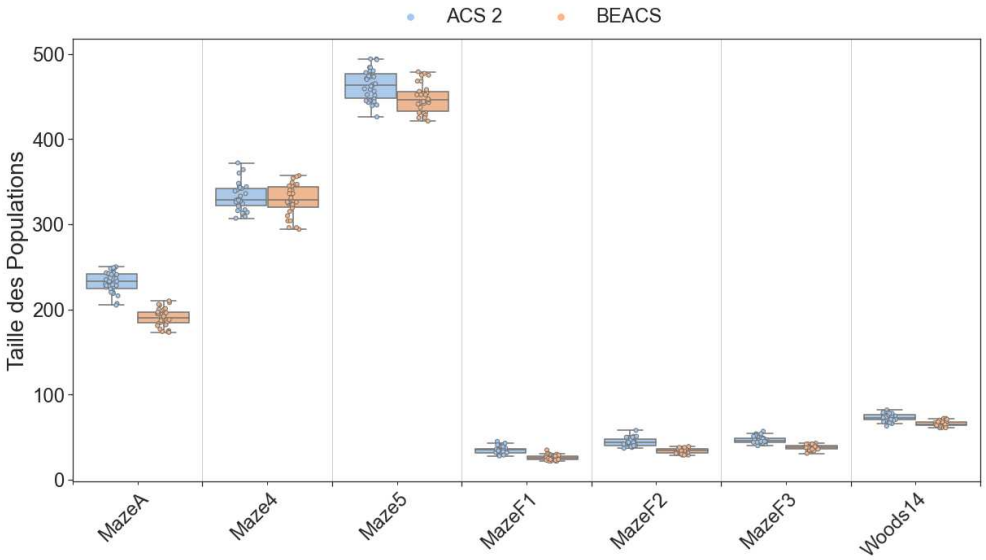


FIGURE D.11 – Tailles des populations de classeurs de BEACS et ACS 2 dans les environnements sans incertitude environnementale.

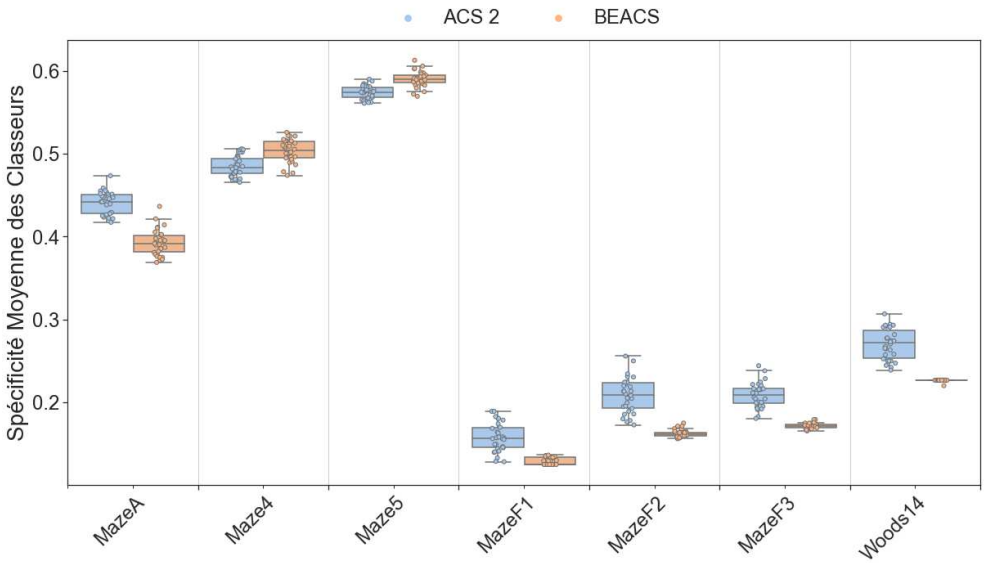


FIGURE D.12 – Spécificités moyennes des classeurs réguliers fiables de BEACS et ACS 2 dans les environnements sans incertitude environnementale.

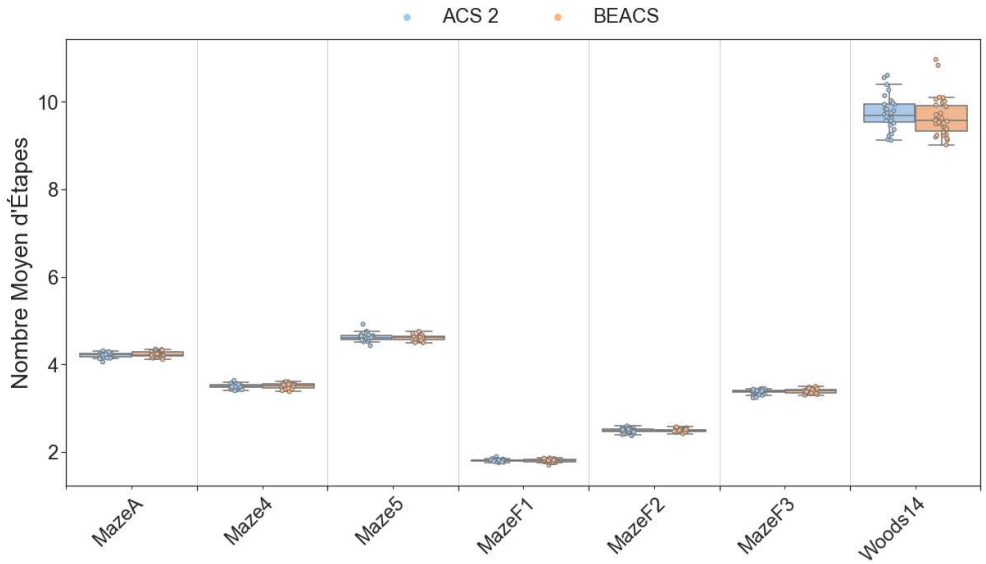


FIGURE D.13 – Nombres moyens d'étapes de BEACS et ACS 2 pour atteindre la sortie les environnements sans incertitude environnementale.

D.4 Capacités d'apprentissage de BEACS et BEACS-IndNomuT sans les actions incertaines

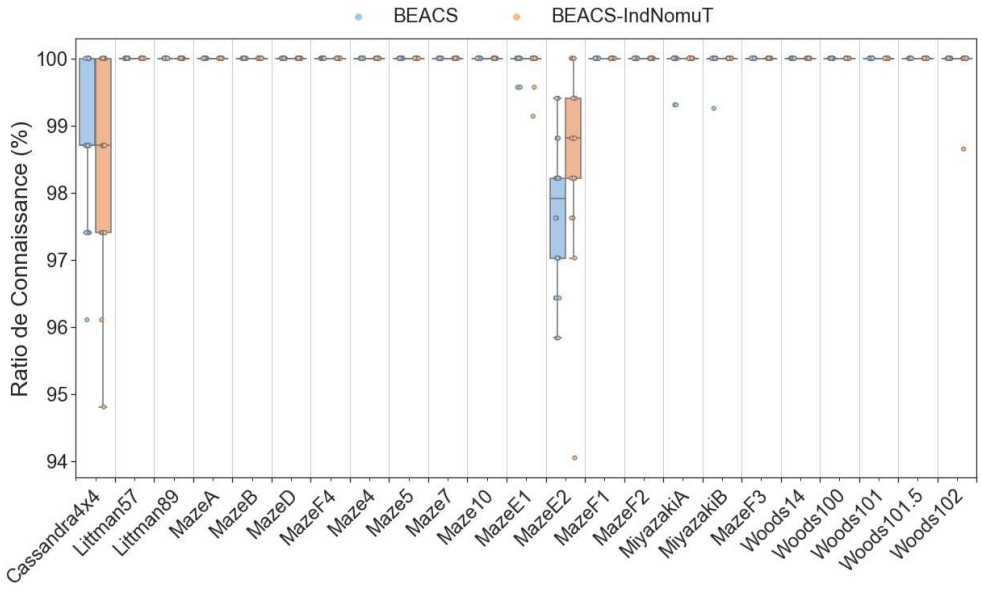


FIGURE D.14 – Ratios de connaissance de BEACS et BEACS-IndNomuT dans tous les environnements sans les actions incertaines.

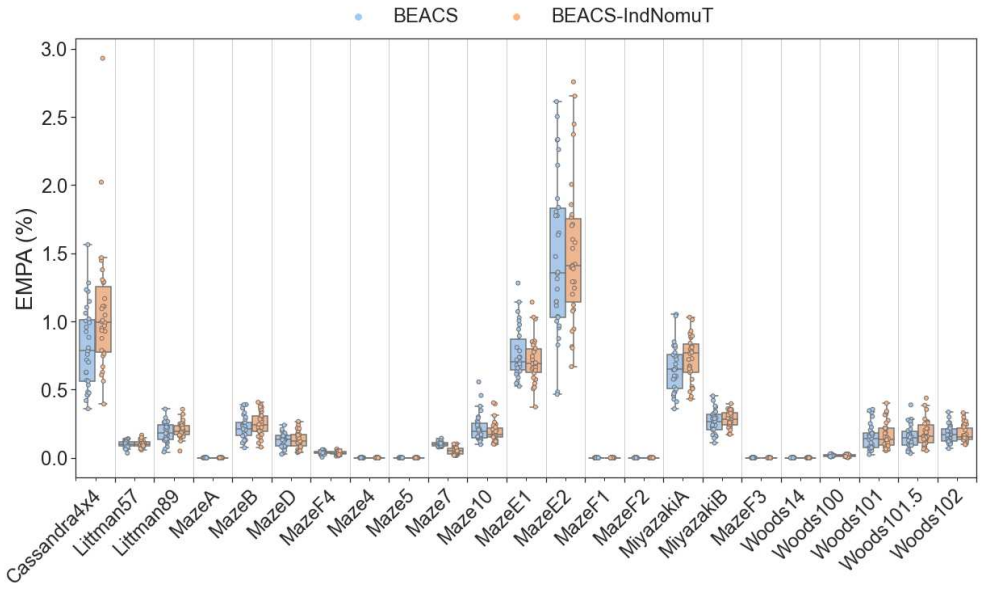


FIGURE D.15 – Erreurs Moyennes sur la Prédiction de l'Anticipation de BEACS et BEACS-IndNomuT dans tous les environnements sans les actions incertaines.

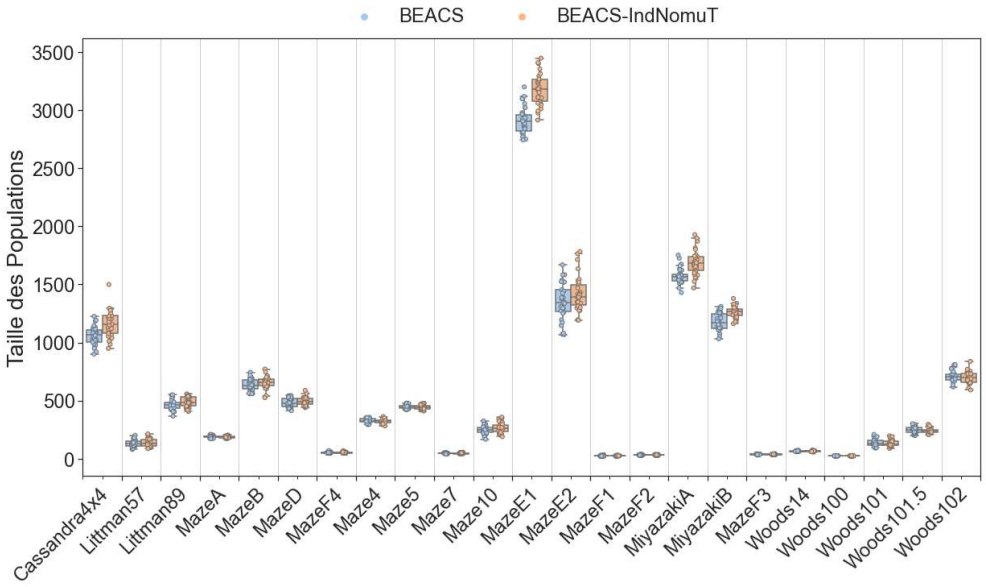


FIGURE D.16 – Tailles des populations de classeurs de BEACS et BEACS-IndNomuT dans tous les environnements sans les actions incertaines.

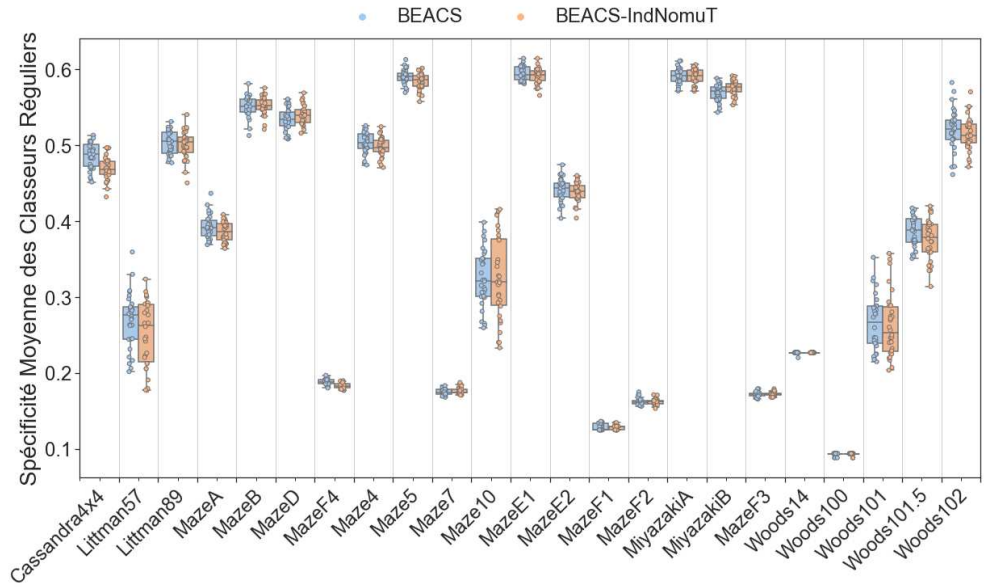


FIGURE D.17 – Spécificités moyennes des classeurs réguliers fiables de BEACS et BEACS-IndNomuT dans tous les environnements sans les actions incertaines.

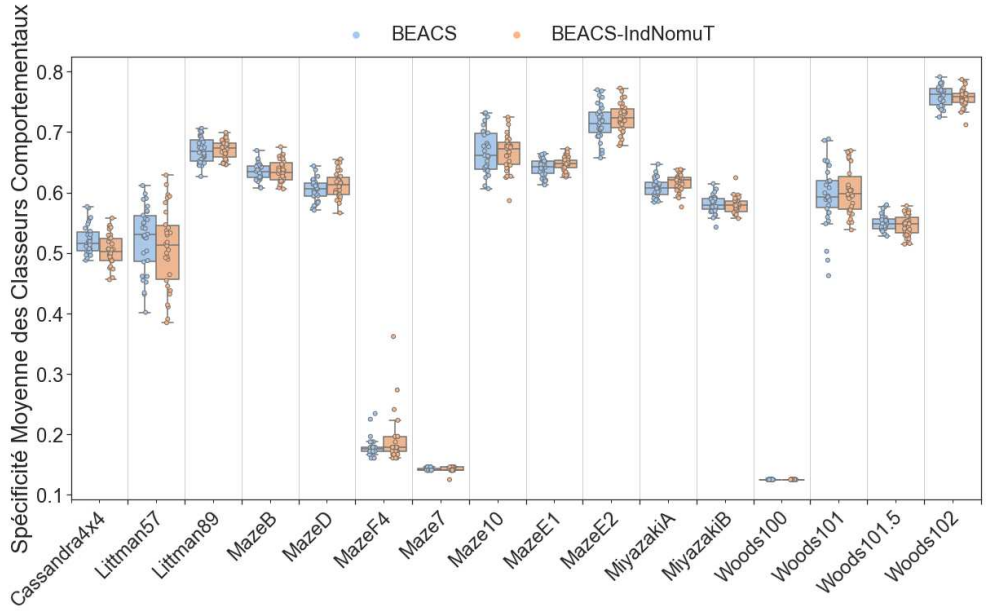


FIGURE D.18 – Spécificités moyennes des classeurs comportementaux fiables de BEACS et BEACS-IndNomuT dans tous les environnements à PAI sans les actions incertaines.

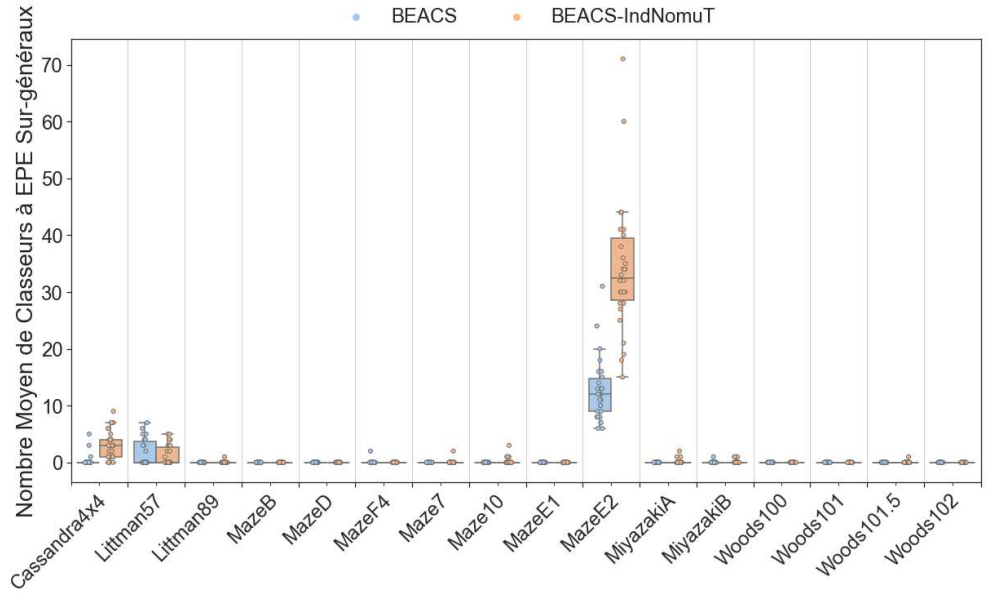


FIGURE D.19 – Nombres moyens de classeurs sur-généraux à EPE de BEACS et BEACS-IndNomuT dans les environnements avec PAI et sans les actions incertaines.

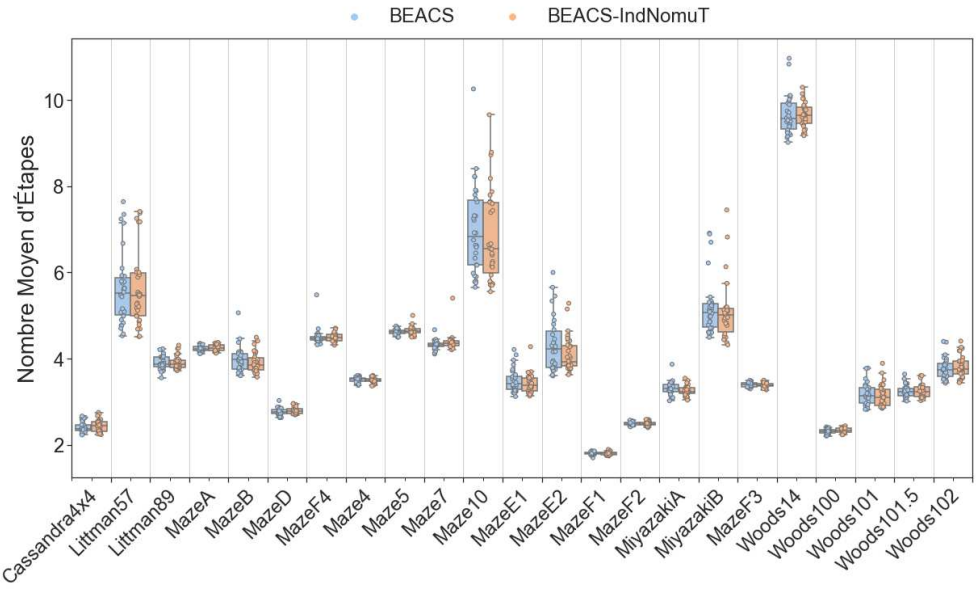


Figure D.20 – Nombres moyens d'étapes de BEACS et BEACS-IndNomuT pour atteindre la sortie des labyrinthes sans les actions incertaines.

D.5 Capacités d'apprentissage de BEACS et BEACS-IndNomuT avec les actions incertaines

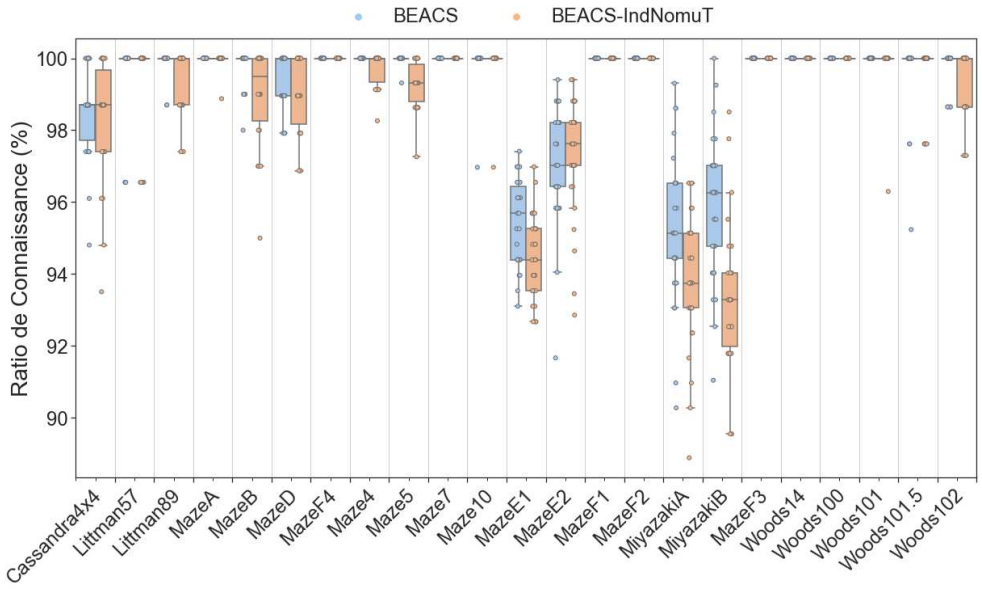


FIGURE D.21 – Ratios de connaissance de BEACS et BEACS-IndNomuT dans tous les environnements avec les actions incertaines.

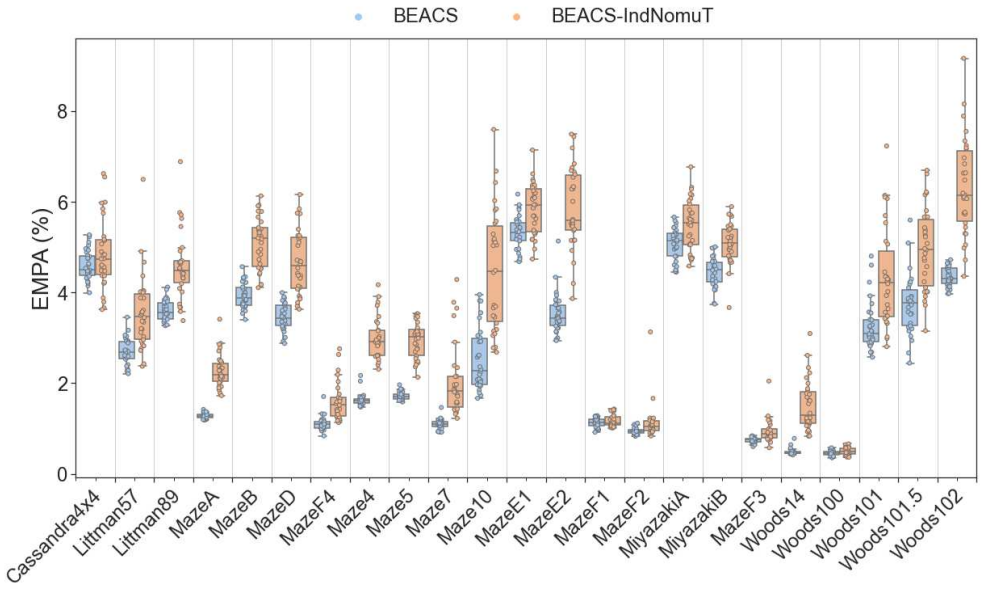


FIGURE D.22 – Erreurs Moyennes sur la Prédiction de l'Anticipation de BEACS et BEACS-IndNomuT dans tous les environnements avec les actions incertaines.

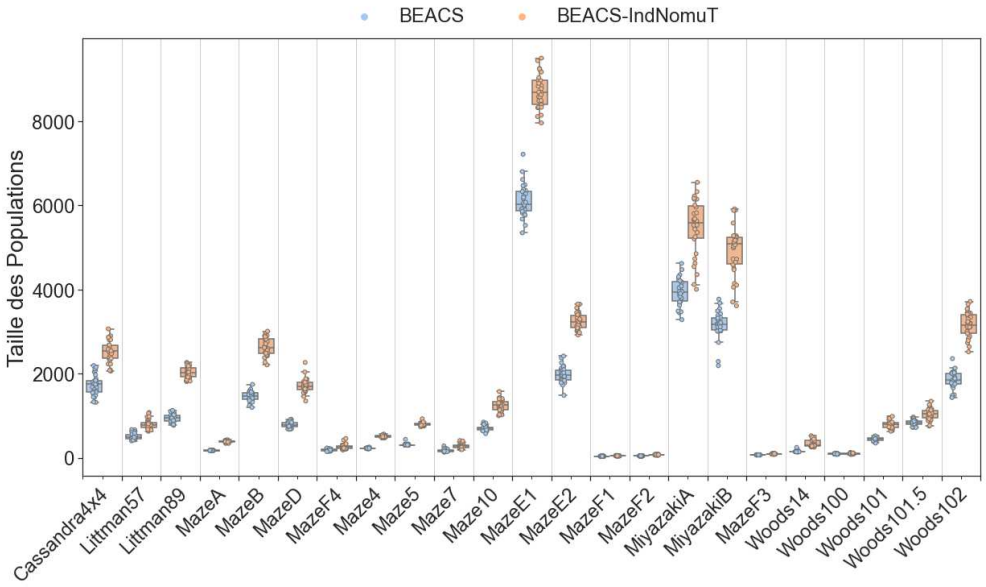


FIGURE D.23 – Tailles des populations de classeurs de BEACS et BEACS-IndNomuT dans tous les environnements avec les actions incertaines.

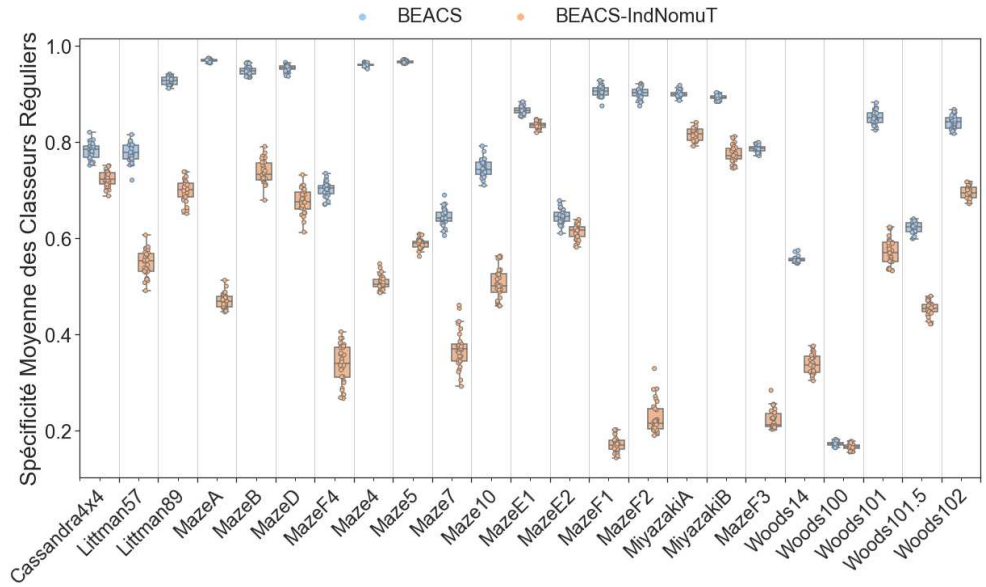


FIGURE D.24 – Spécificités moyennes des classeurs réguliers fiables de BEACS et BEACS-IndNomuT dans tous les environnements avec les actions incertaines.

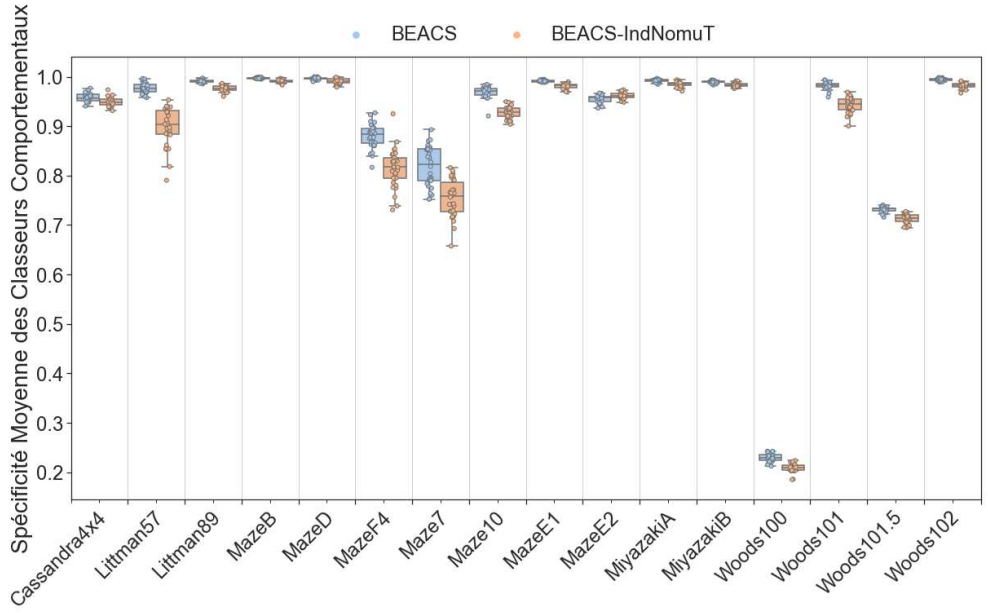


FIGURE D.25 – Spécificités moyennes des classeurs comportementaux fiables de BEACS et BEACS-IndNomuT dans tous les environnements à PAI avec les actions incertaines.

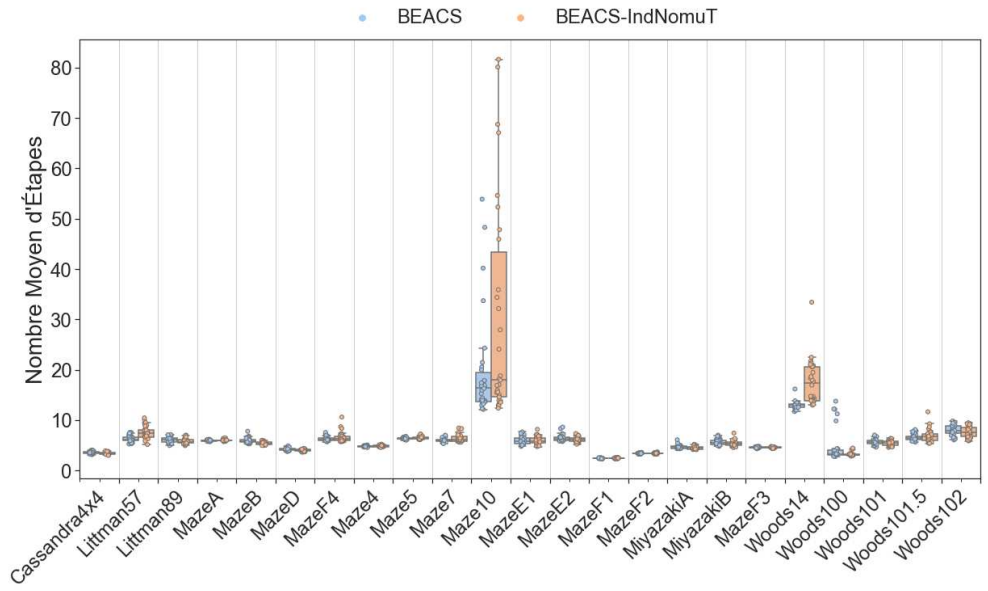


FIGURE D.26 – Nombres moyens d'étapes de BEACS et BEACS-IndNomuT pour atteindre la sortie des labyrinthes avec les actions incertaines.

D.6 Capacités d'apprentissage de BEACS et BEACS-DmuT sans les actions incertaines

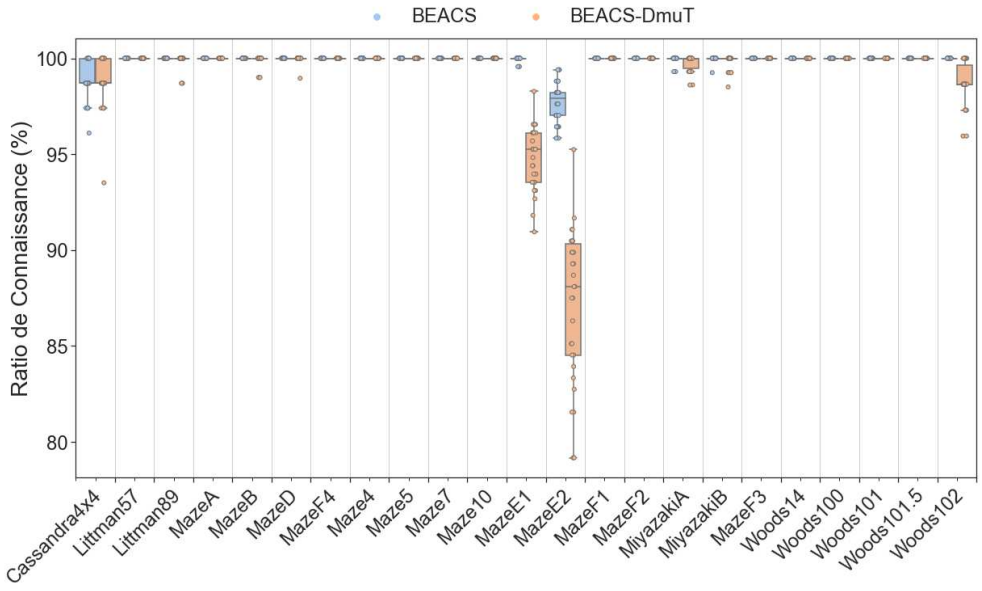


FIGURE D.27 – Ratios de connaissance de BEACS et BEACS-DmuT dans tous les environnements sans les actions incertaines.

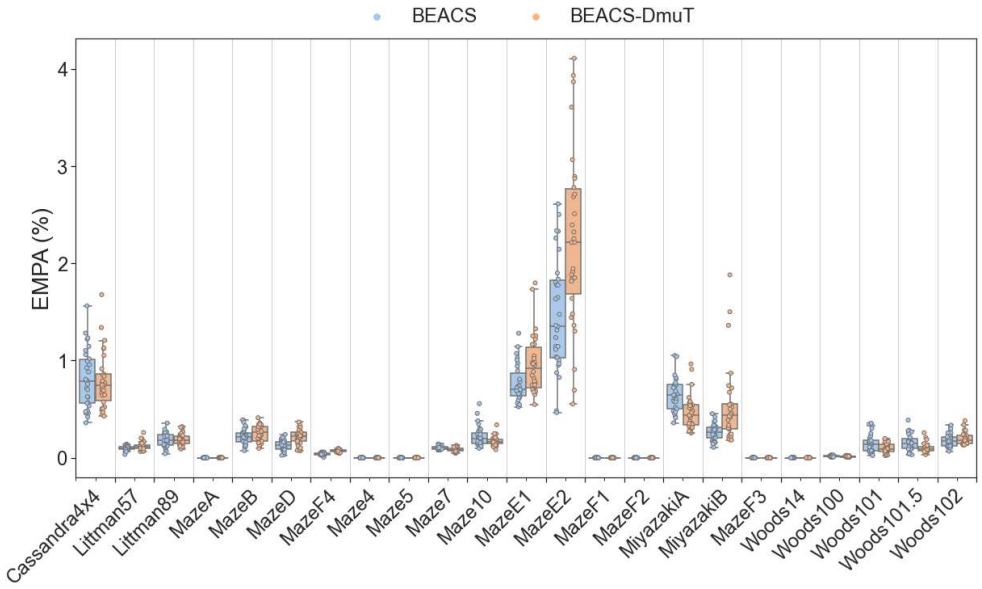


FIGURE D.28 – Erreurs Moyennes sur la Prédiction de l'Anticipation de BEACS et BEACS-DmuT dans tous les environnements sans les actions incertaines.

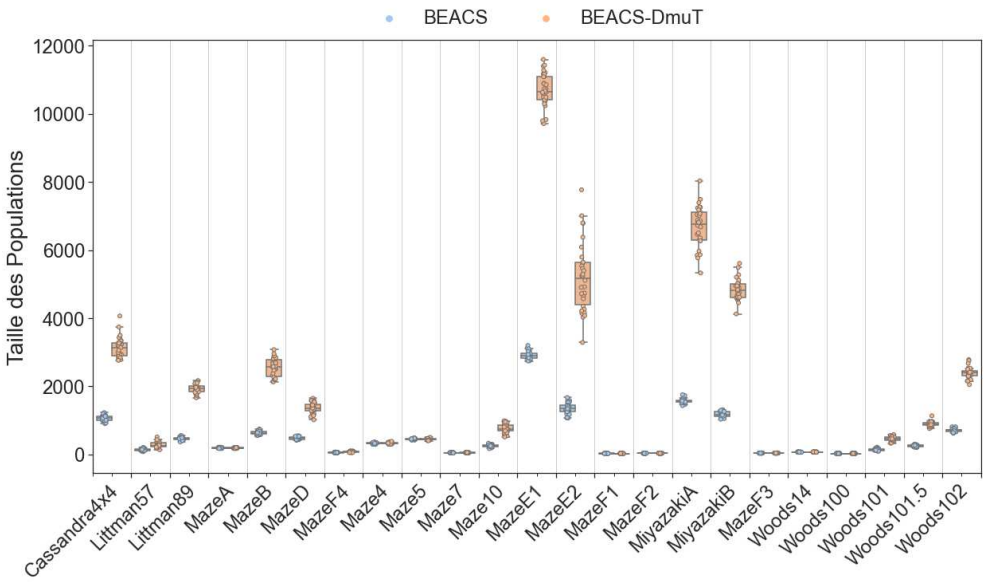


FIGURE D.29 – Tailles des populations de classeurs de BEACS et BEACS-DmuT dans tous les environnements sans les actions incertaines.

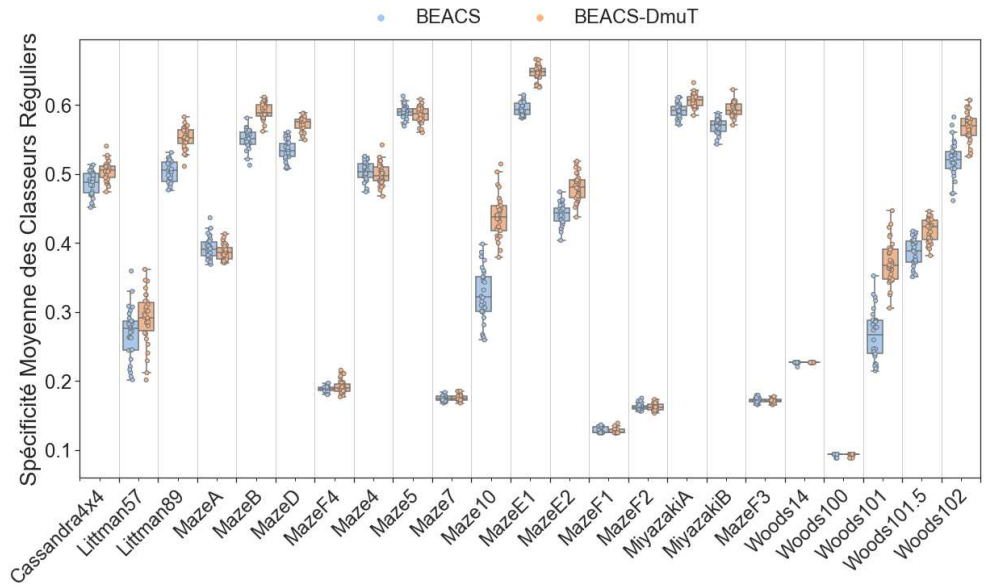


FIGURE D.30 – Spécificités moyennes des classeurs réguliers fiables de BEACS et BEACS-DmuT dans tous les environnements sans les actions incertaines.

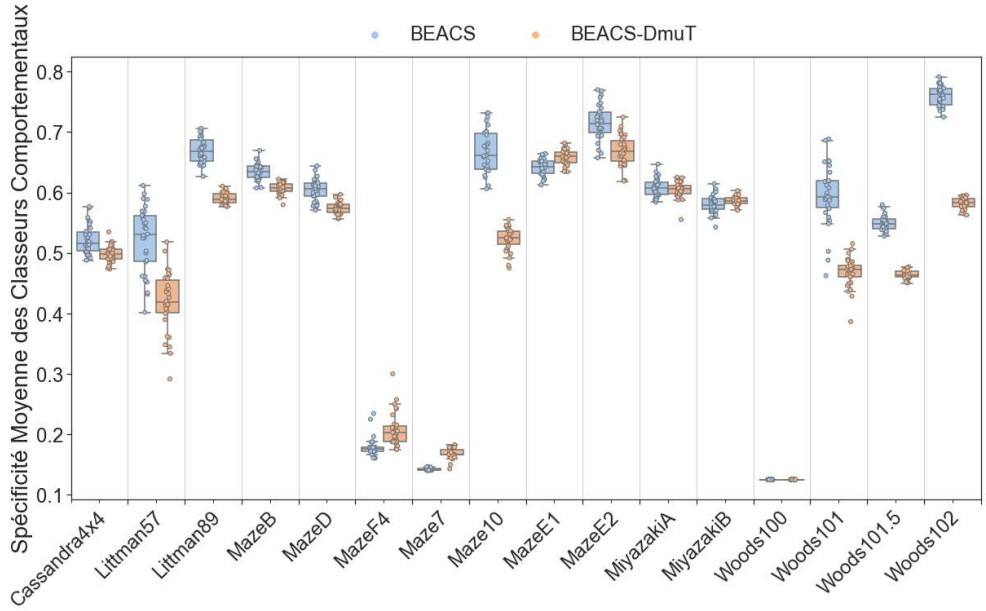


FIGURE D.31 – Spécificités moyennes des classeurs comportementaux fiables de BEACS et BEACS-DmuT dans tous les environnements à PAI sans les actions incertaines.

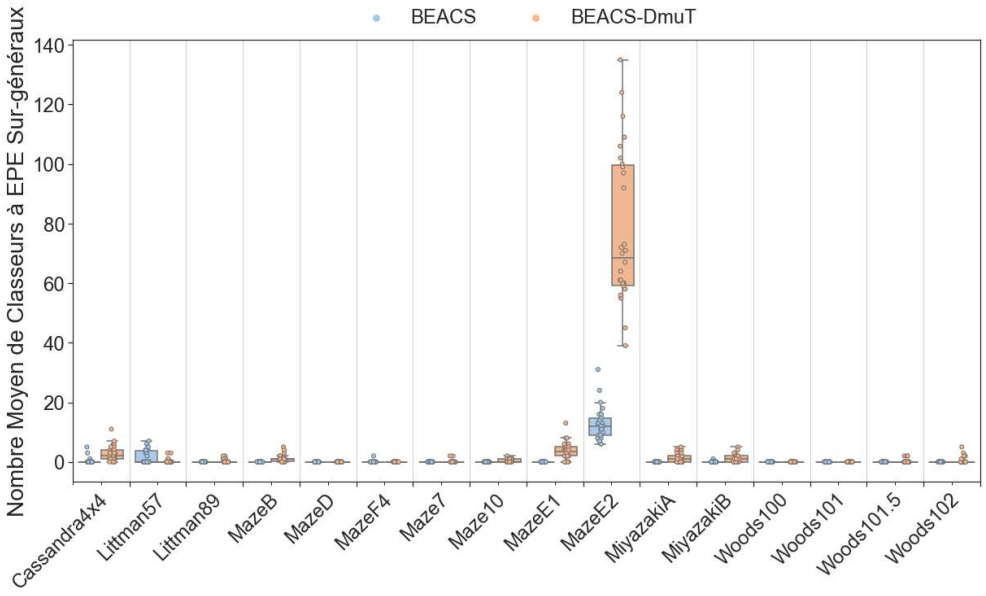


FIGURE D.32 – Nombres moyens de classeurs sur-généraux à EPE de BEACS et BEACS-DmuT dans les environnements avec PAI et sans les actions incertaines.

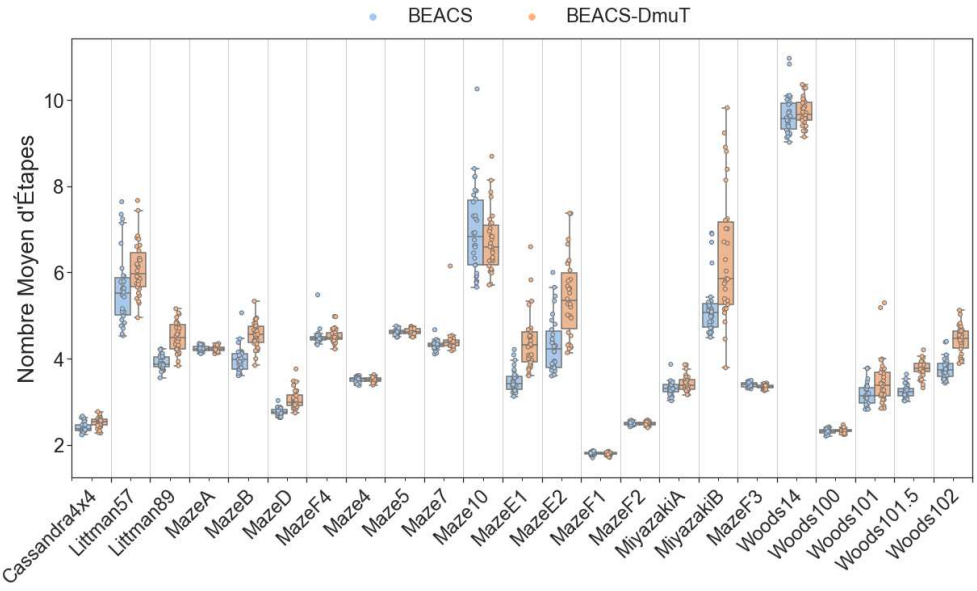


FIGURE D.33 – Nombres moyens d'étapes de BEACS et BEACS-DmuT pour atteindre la sortie des labyrinthes sans les actions incertaines.

D.7 Capacités d'apprentissage de BEACS et BEACS-DmuT avec les actions incertaines

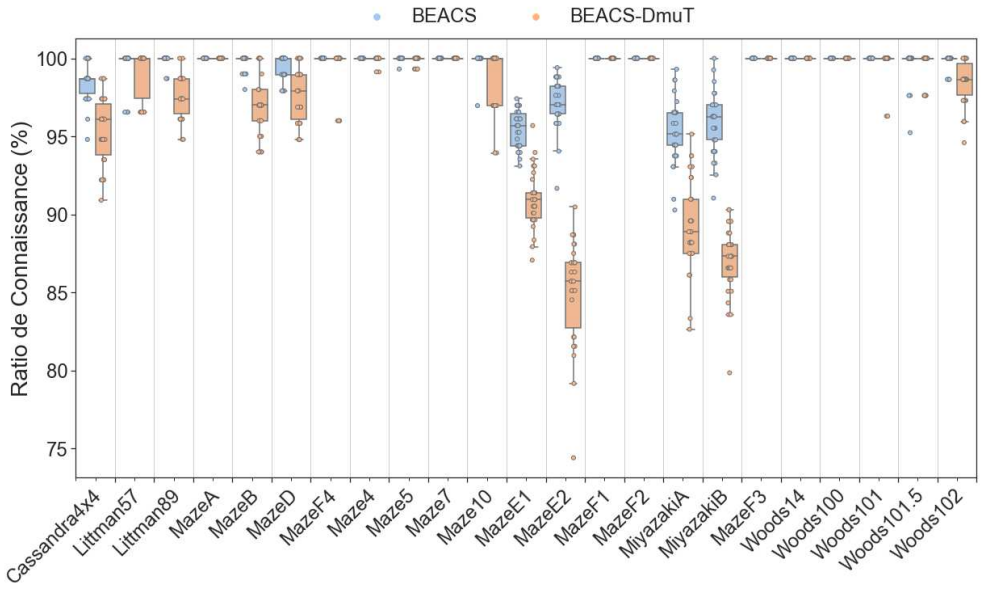


FIGURE D.34 – Ratios de connaissance de BEACS et BEACS-DmuT dans tous les environnements avec les actions incertaines.

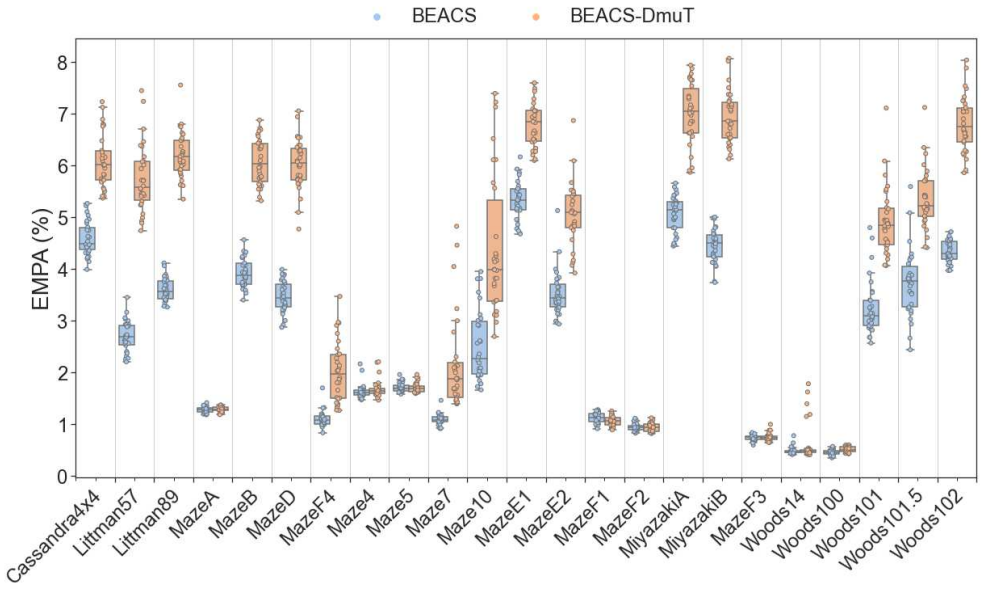


FIGURE D.35 – Erreurs Moyennes sur la Prédiction de l'Anticipation de BEACS et BEACS-DmuT dans tous les environnements avec les actions incertaines.

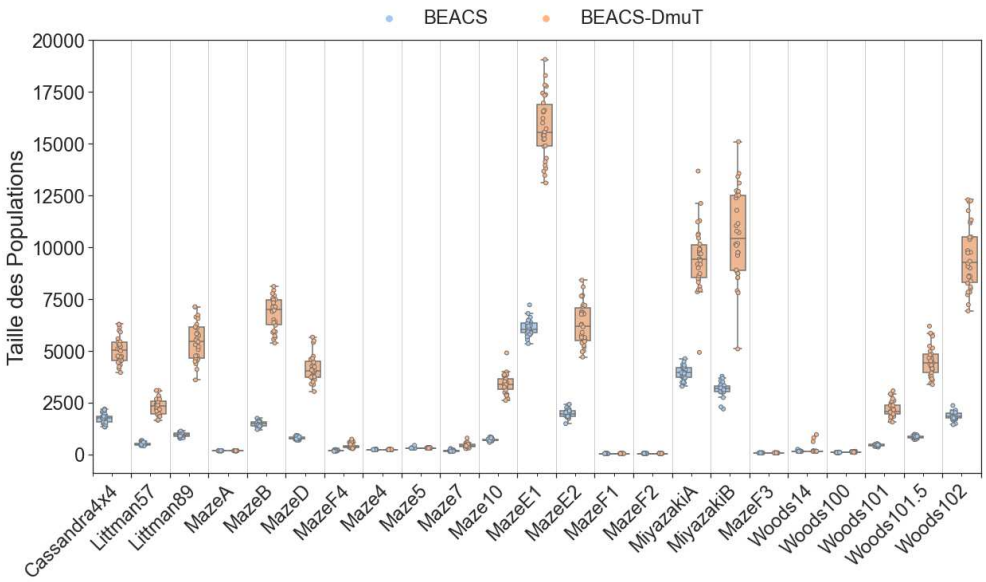


FIGURE D.36 – Tailles des populations de classeurs de BEACS et BEACS-DmuT dans tous les environnements avec les actions incertaines.

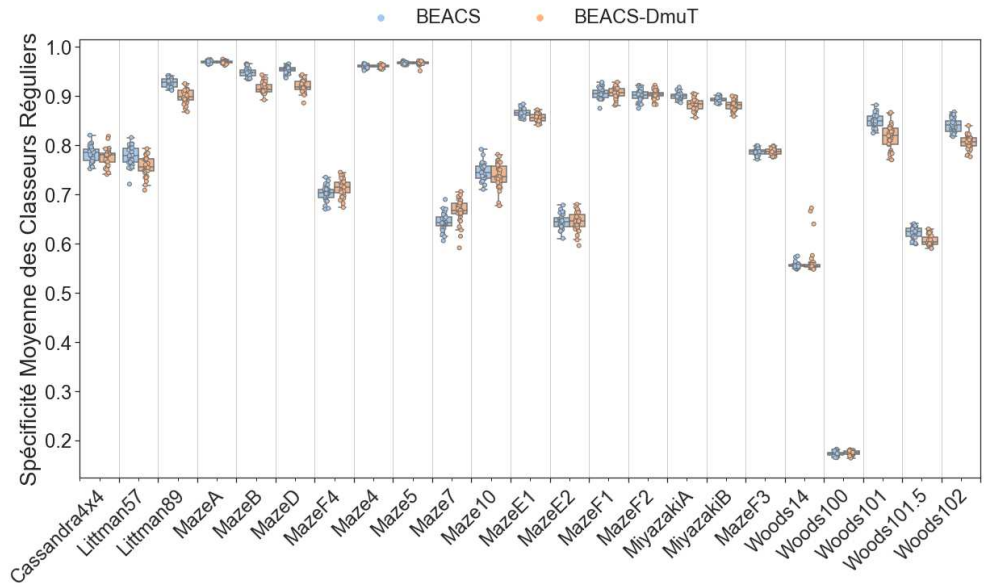


FIGURE D.37 – Spécificités moyennes des classeurs réguliers fiables de BEACS et BEACS-DmuT dans tous les environnements avec les actions incertaines.

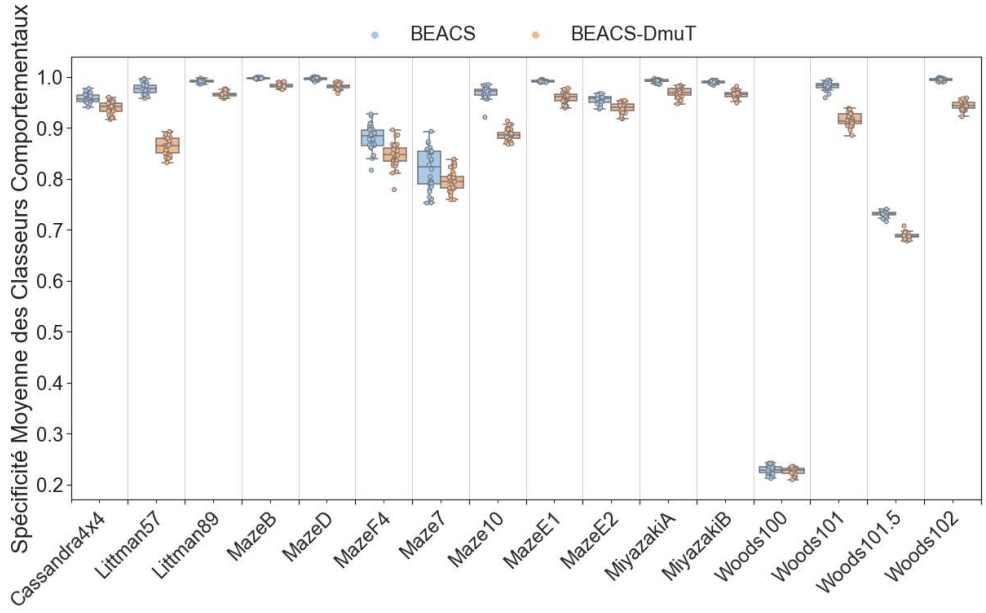


FIGURE D.38 – Spécificités moyennes des classeurs comportementaux fiables de BEACS et BEACS-DmuT dans tous les environnements à PAI avec les actions incertaines.

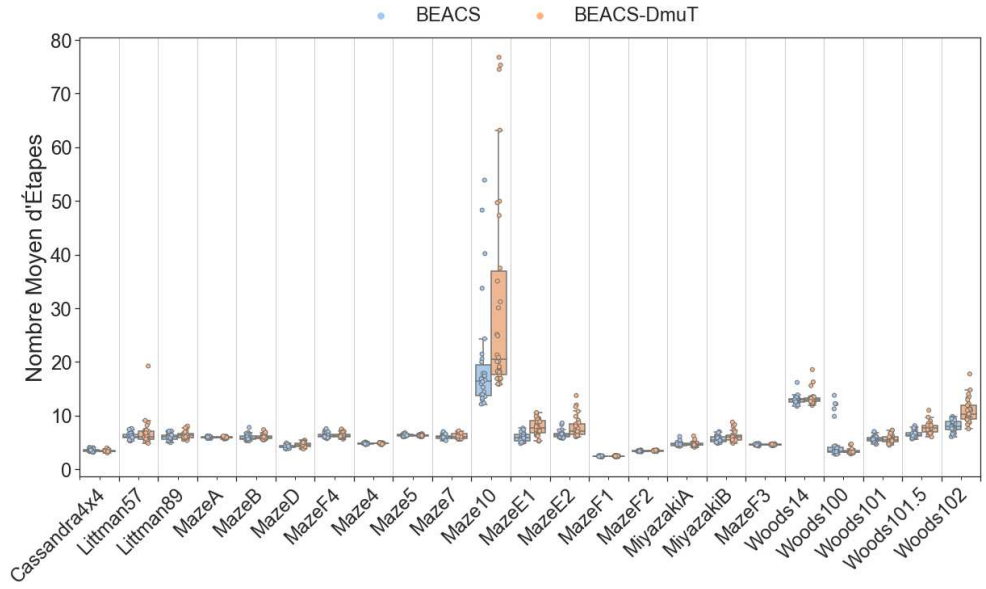


FIGURE D.39 – Nombres moyens d'étapes de BEACS et BEACS-DmuT pour atteindre la sortie des labyrinthes avec les actions incertaines.

D.8 Comparaison des représentations environnementales de BEACS-Q et BEACS

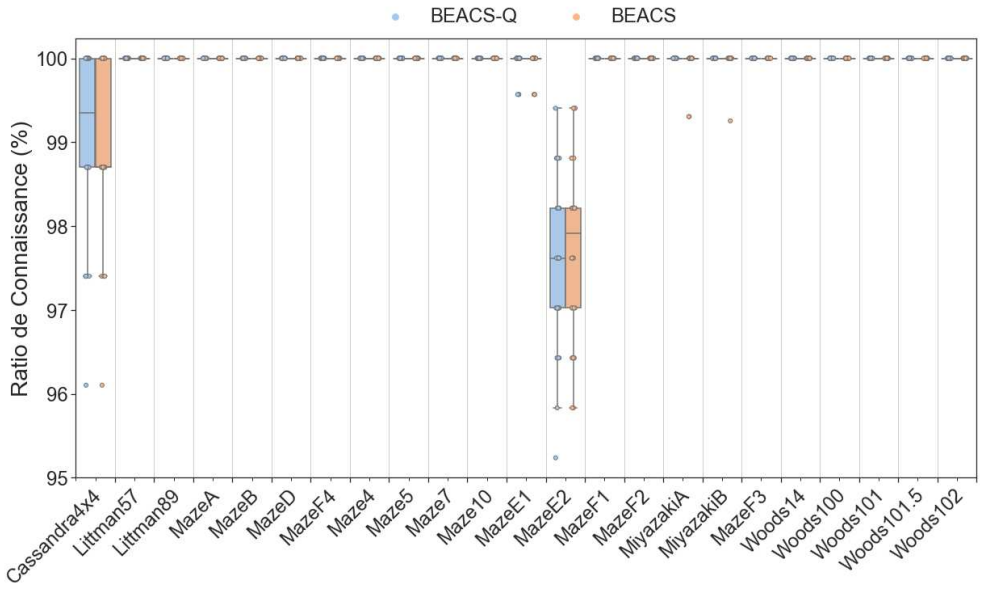


FIGURE D.40 – Ratios de connaissance de BEACS-Q et BEACS dans tous les environnements sans les actions incertaines.

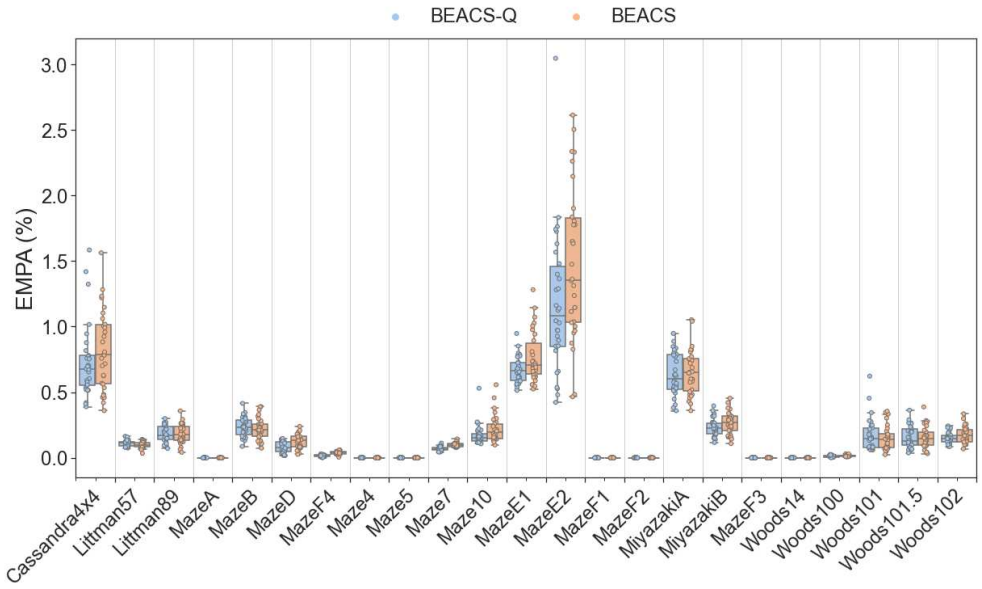


FIGURE D.41 – Erreurs Moyennes sur la Prédiction de l'Anticipation de BEACS-Q et BEACS dans tous les environnements sans les actions incertaines.

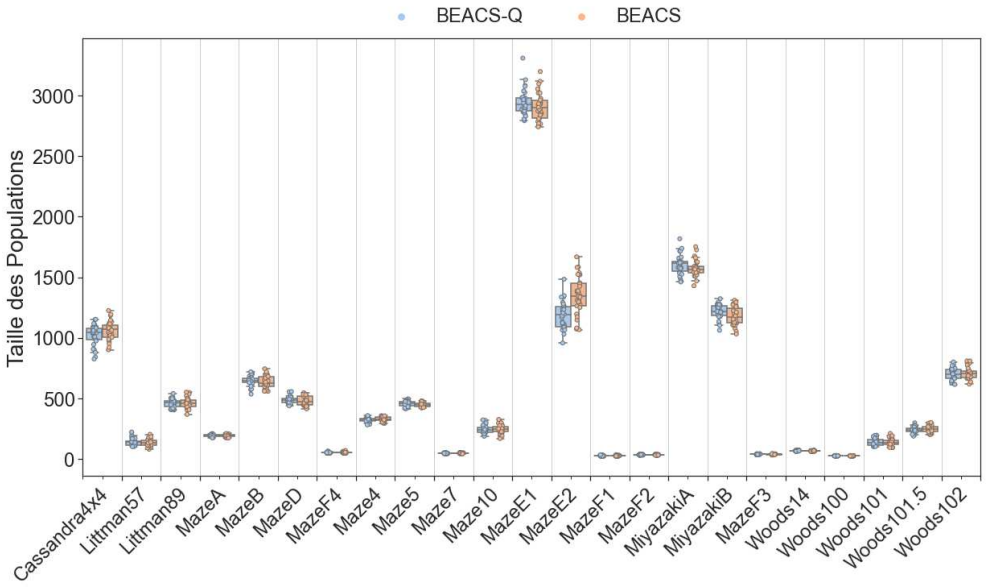


FIGURE D.42 – Tailles des populations de classeurs de BEACS-Q et BEACS dans tous les environnements sans les actions incertaines.

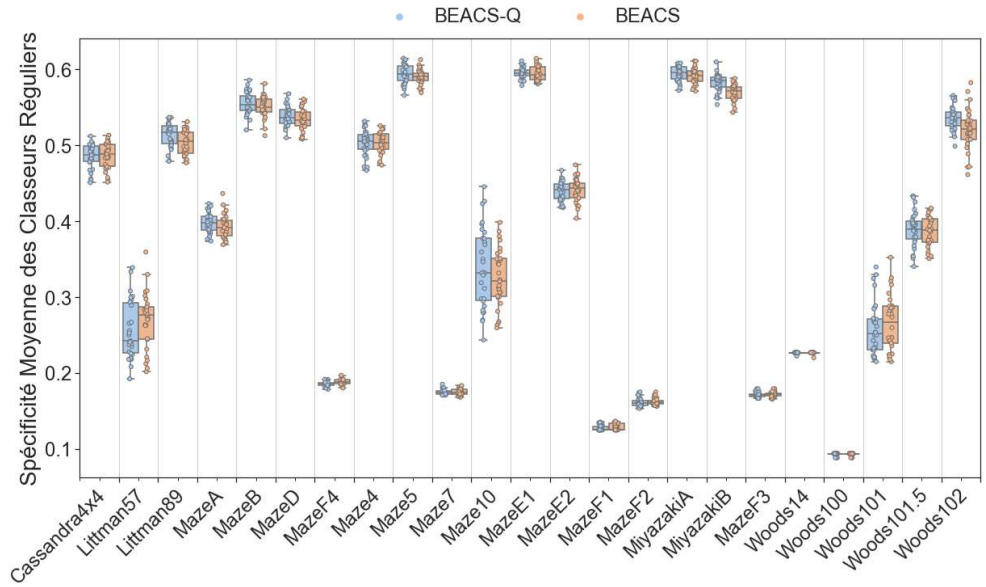


FIGURE D.43 – Spécificités moyennes des classeurs réguliers fiables de BEACS-Q et BEACS dans tous les environnements sans les actions incertaines.

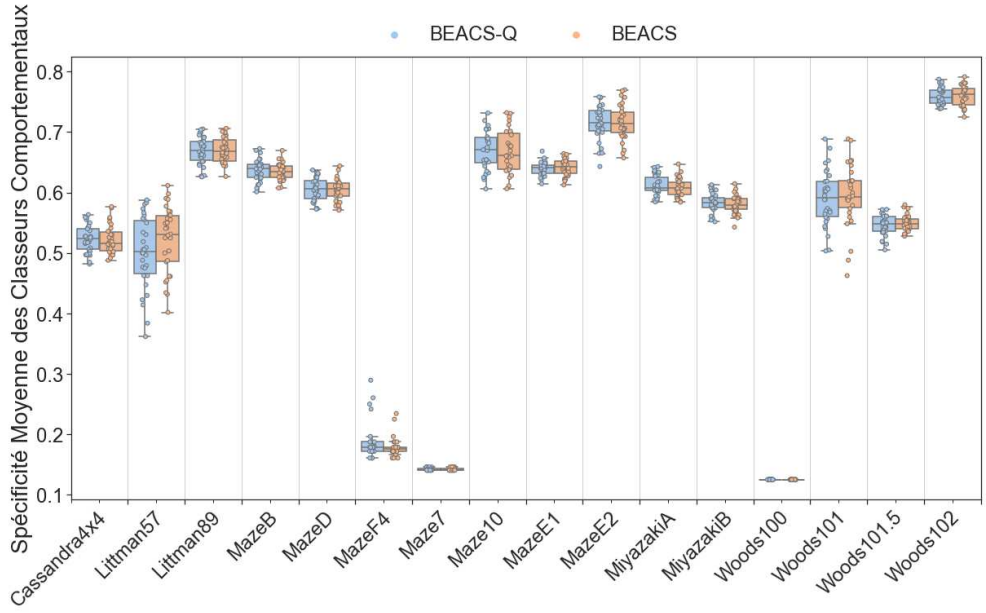


FIGURE D.44 – Spécificités moyennes des classeurs comportementaux fiables de BEACS-Q et BEACS dans tous les environnements à PAI sans les actions incertaines.

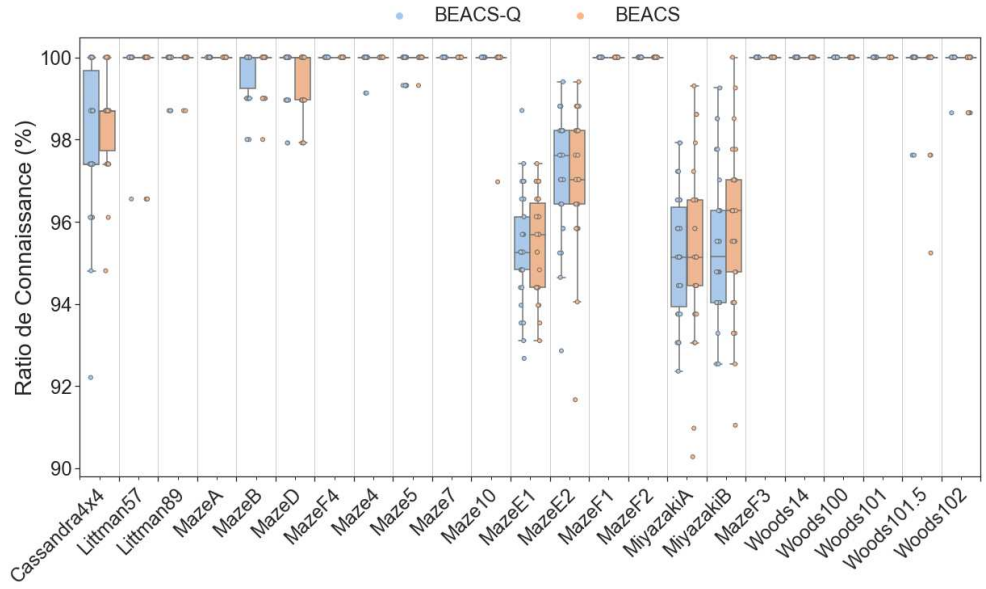


FIGURE D.45 – Ratios de connaissance de BEACS-Q et BEACS dans tous les environnements avec les actions incertaines.

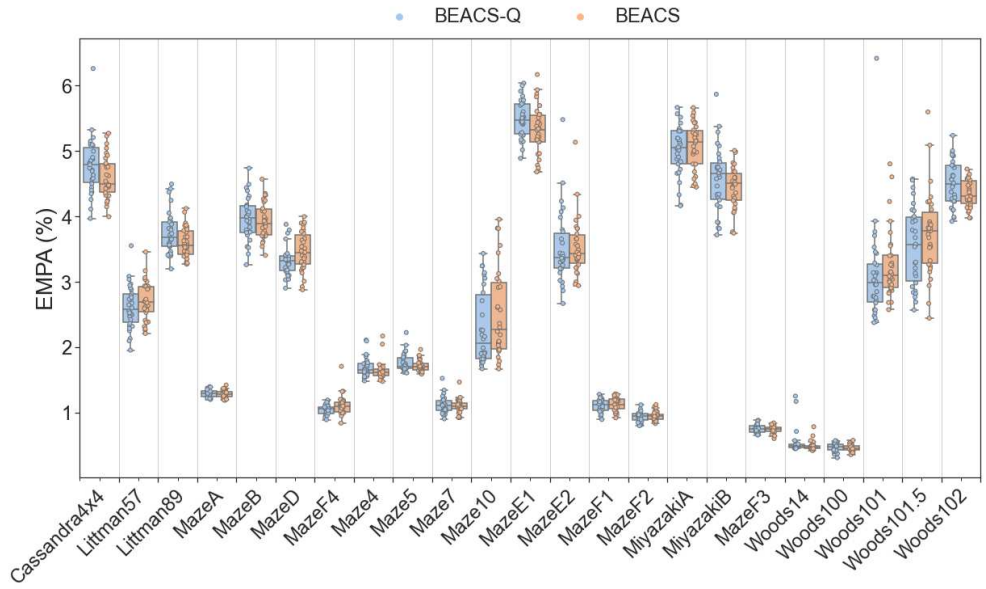


FIGURE D.46 – Erreurs Moyennes sur la Prédiction de l'Anticipation de BEACS-Q et BEACS dans tous les environnements avec les actions incertaines.

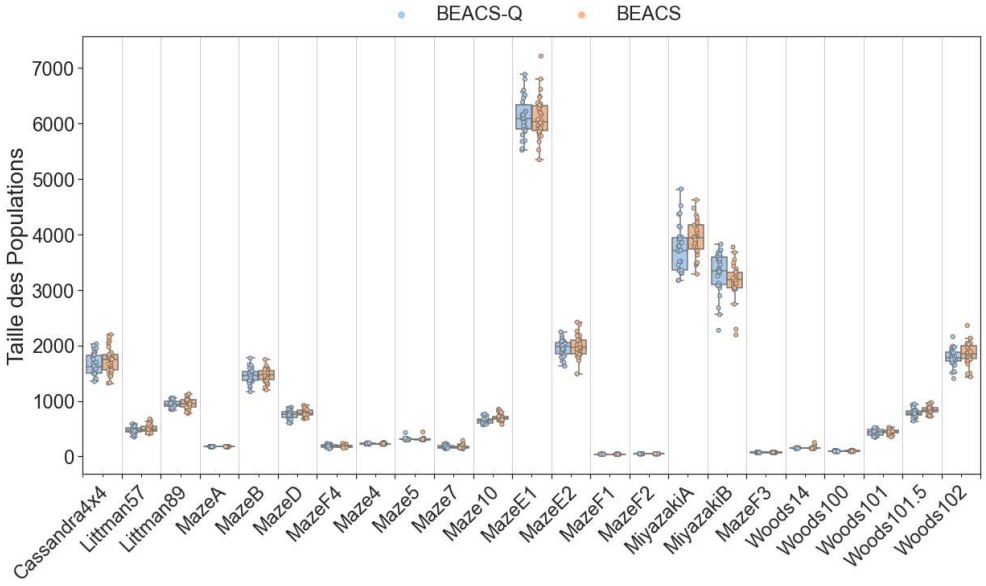


FIGURE D.47 – Tailles des populations de classeurs de BEACS-Q et BEACS dans tous les environnements avec les actions incertaines.

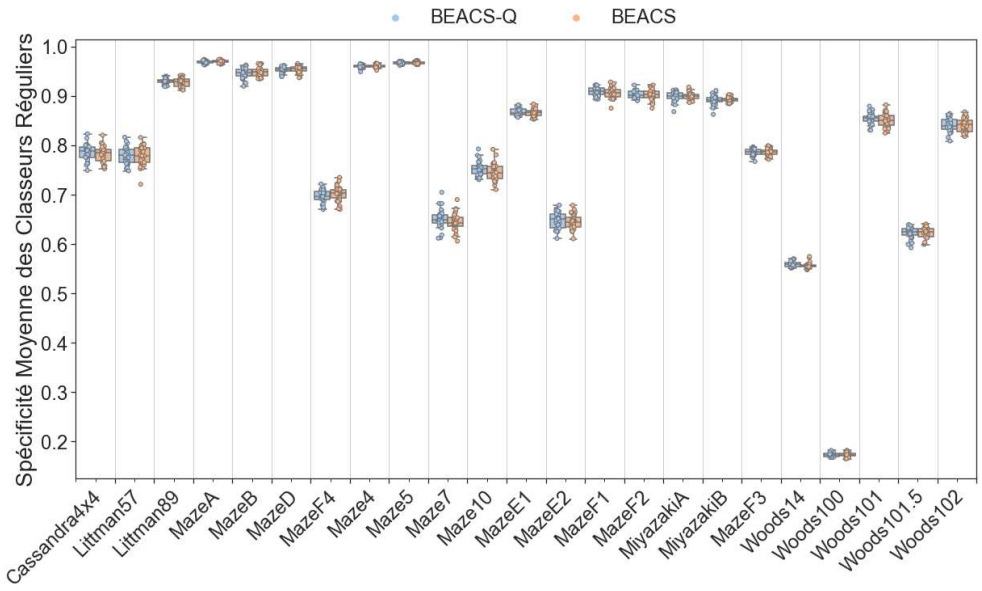


FIGURE D.48 – Spécificités moyennes des classeurs réguliers fiables de BEACS-Q et BEACS dans tous les environnements avec les actions incertaines.

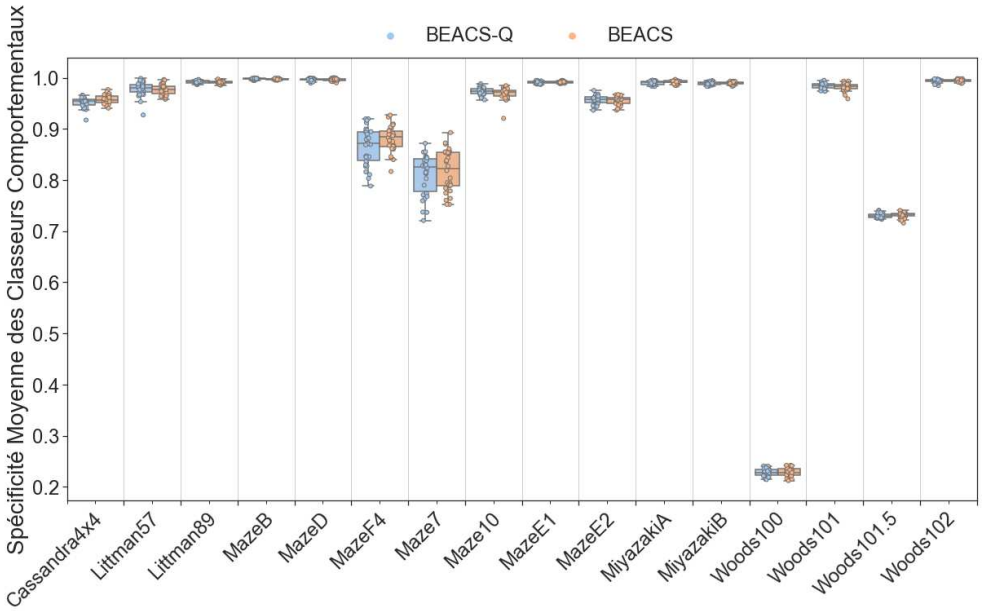


FIGURE D.49 – Spécificités moyennes des classeurs comportementaux fiables de BEACS-Q et BEACS dans tous les environnements à PAI avec les actions incertaines.

ANNEXE E

Ressources supplémentaires à l'extraction des connaissances des systèmes de classeurs à anticipations

E.1 Tailles des populations de classeurs avant et après compression

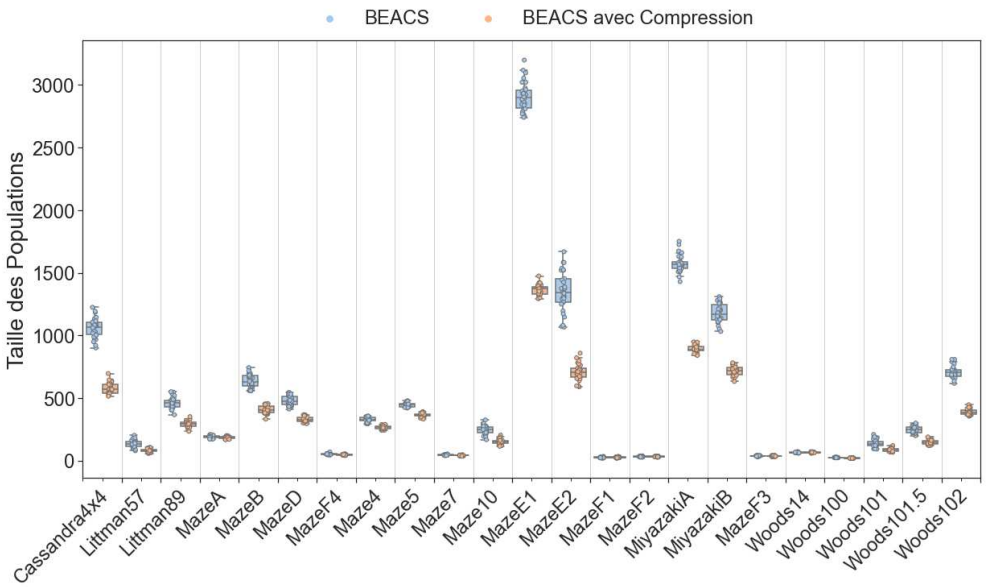


Figure E.1 – Tailles des populations de classeurs de BEACS avant et après compression pour chaque labyrinthe, sans bruitage des actions.

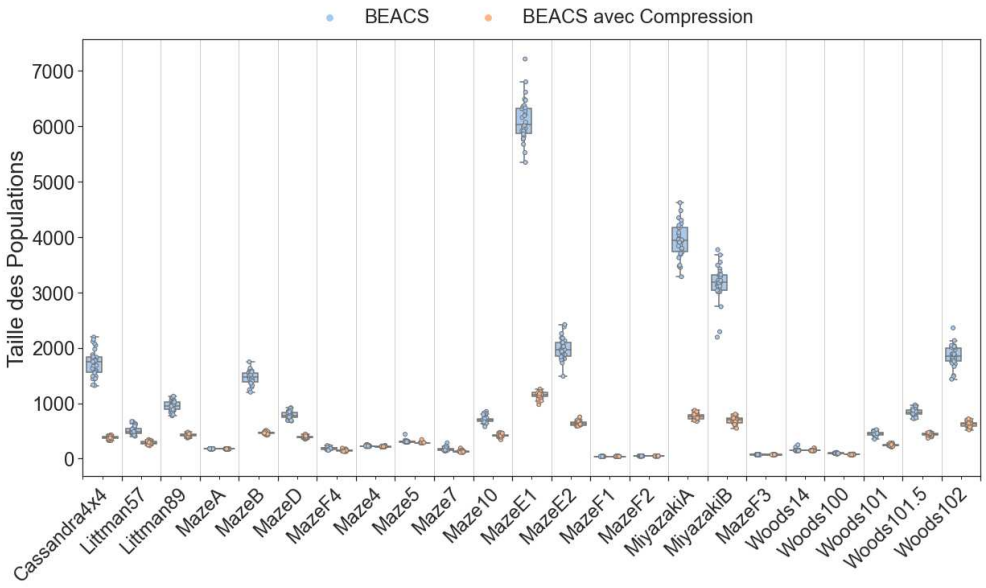


Figure E.2 – Tailles des populations de classeurs de BEACS avant et après compression pour chaque labyrinthe, avec bruitage des actions.

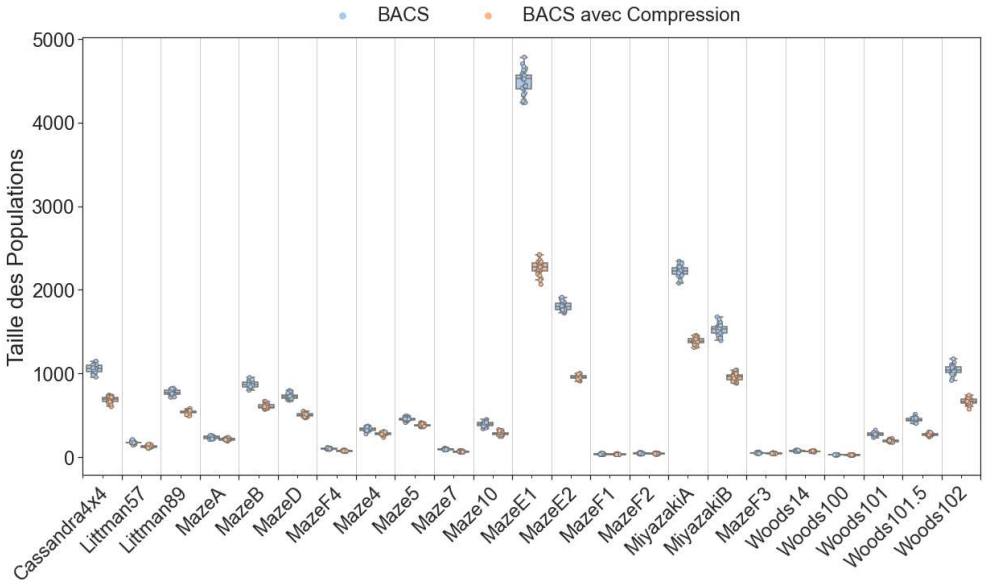


Figure E.3 – Tailles des populations de classeurs de BACS avant et après compression pour chaque labyrinthe, sans bruitage des actions.

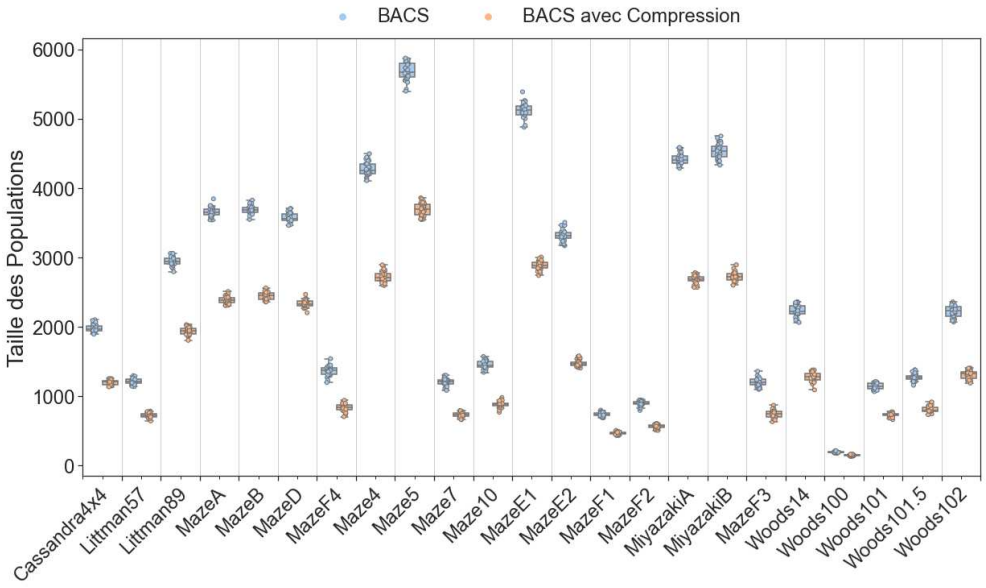


FIGURE E.4 – Tailles des populations de classeurs de BACS avant et après compression pour chaque labyrinthe, avec bruitage des actions.

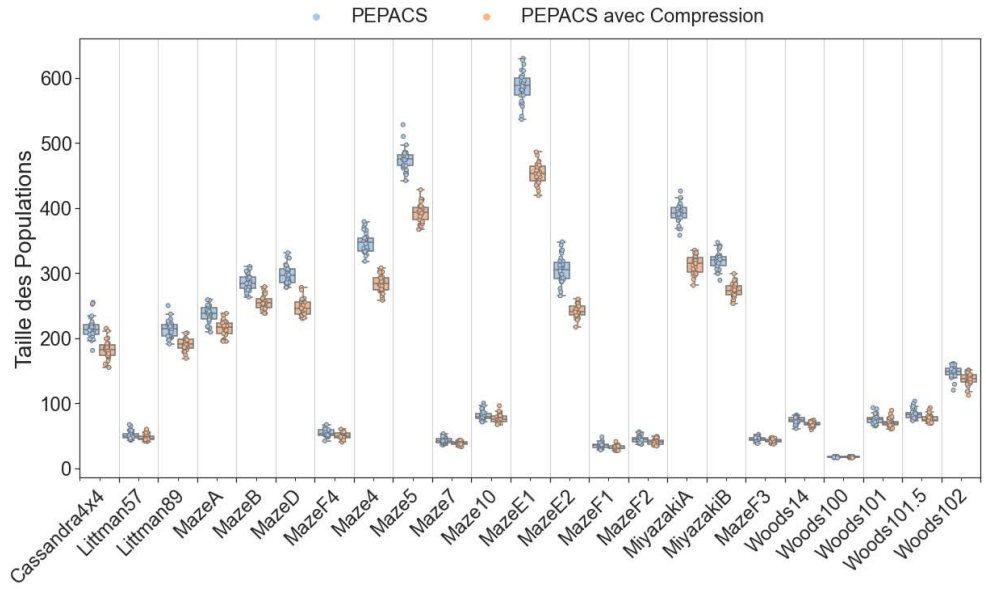


FIGURE E.5 – Tailles des populations de classeurs de PEPACS avant et après compression pour chaque labyrinthe, sans bruitage des actions.

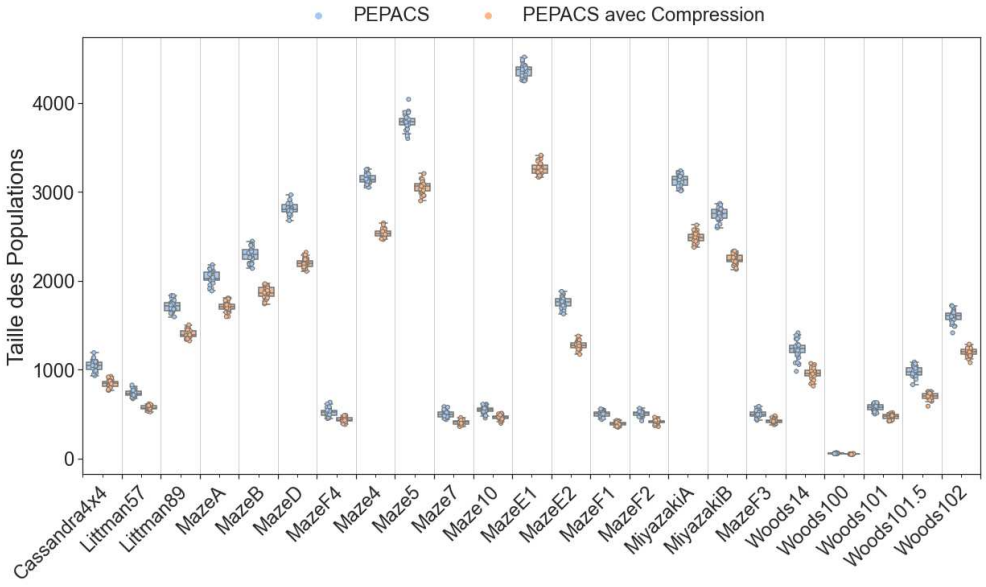


Figure E.6 – Tailles des populations de classeurs de PEPACS avant et après compression pour chaque labyrinthe, sans bruitage des actions.

E.2 Spécificités moyennes des classeurs fiables avant et après compression

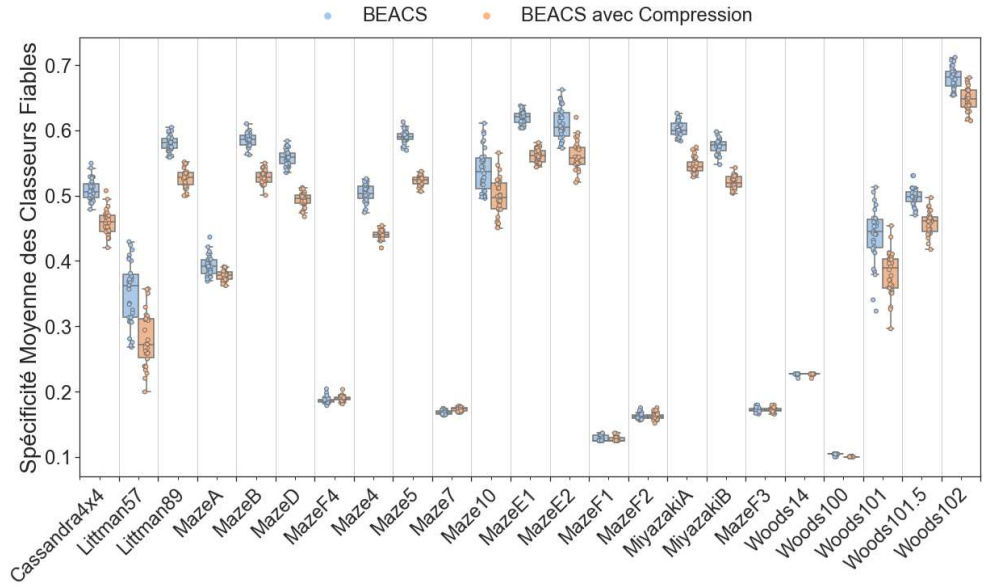


Figure E.7 – Spécificités moyennes des classeurs fiables de BEACS avant et après compression pour chaque labyrinthe, sans bruitage des actions.

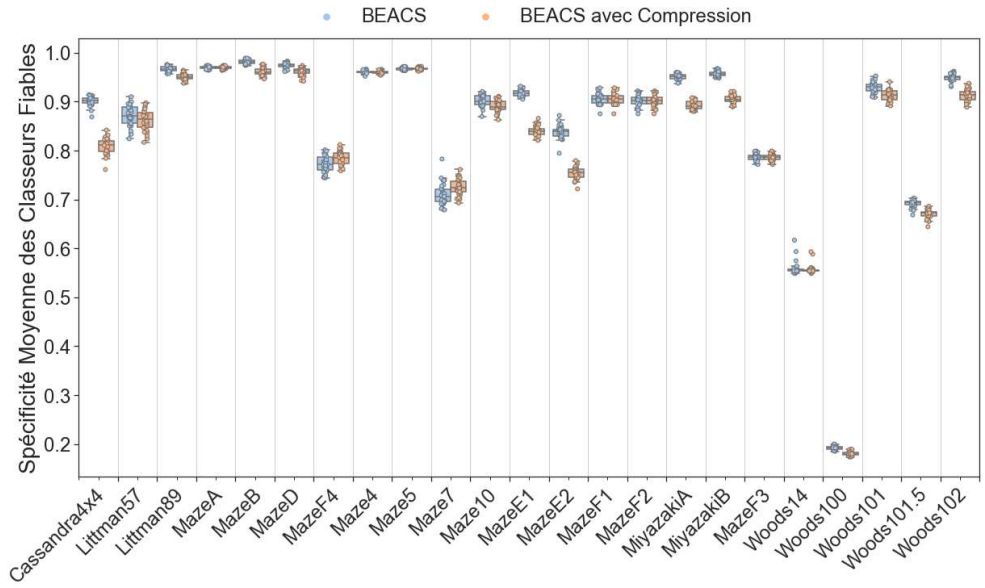


FIGURE E.8 – Spécificités moyennes des classeurs fiables de BEACS avant et après compression pour chaque labyrinthe, avec bruitage des actions.

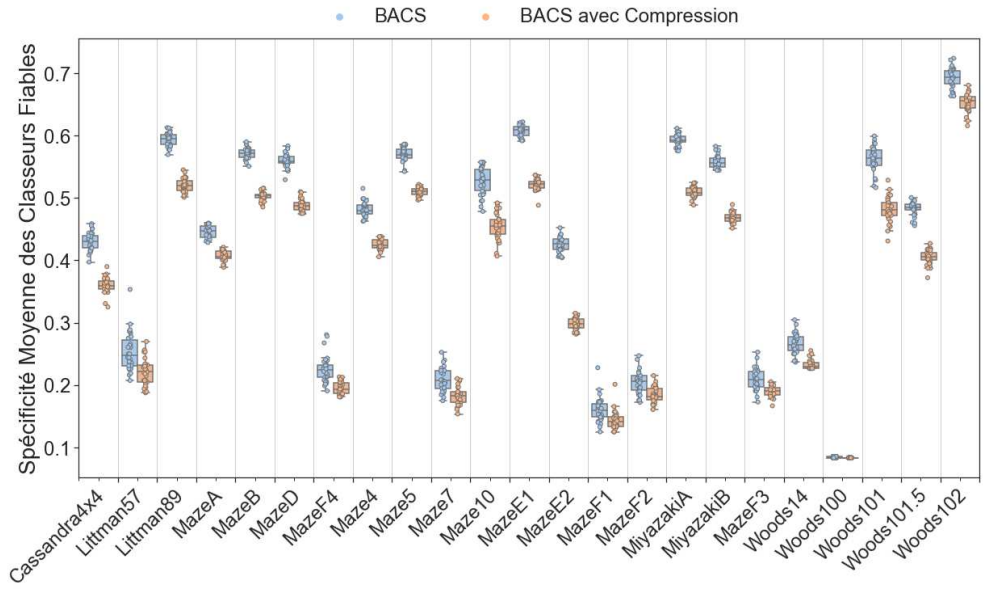


FIGURE E.9 – Spécificités moyennes des classeurs fiables de BACS avant et après compression pour chaque labyrinthe, sans bruitage des actions.

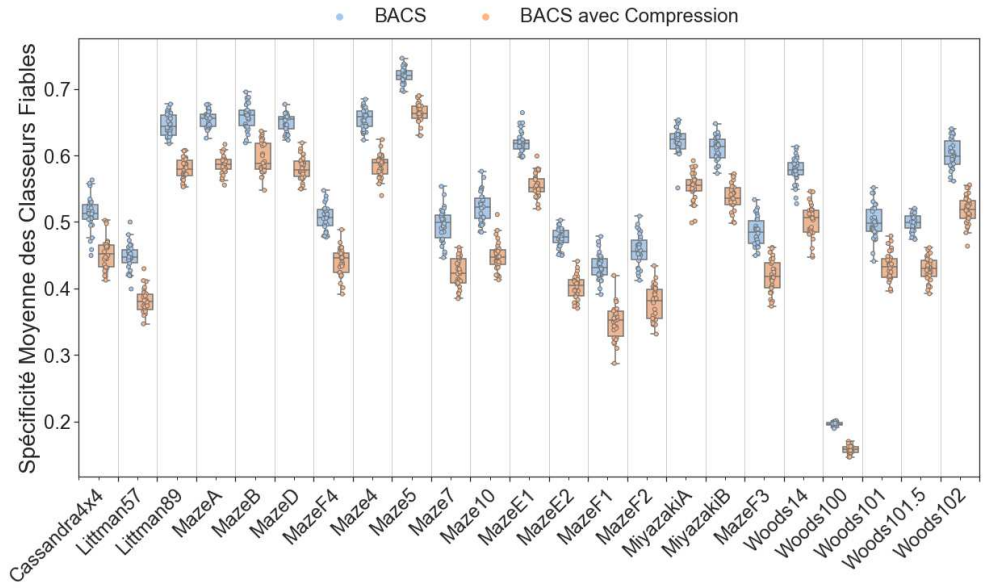


Figure E.10 – Spécificités moyennes des classeurs fiables de BACS avant et après compression pour chaque labyrinthe, avec bruitage des actions.

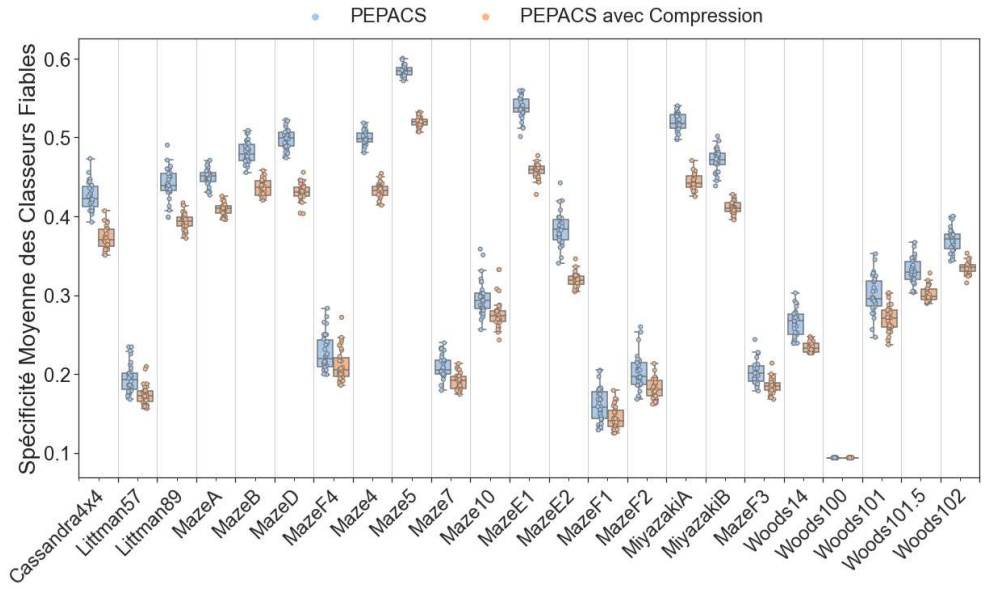


Figure E.11 – Spécificités moyennes des classeurs fiables de PEPACS avant et après compression pour chaque labyrinthe, sans bruitage des actions.

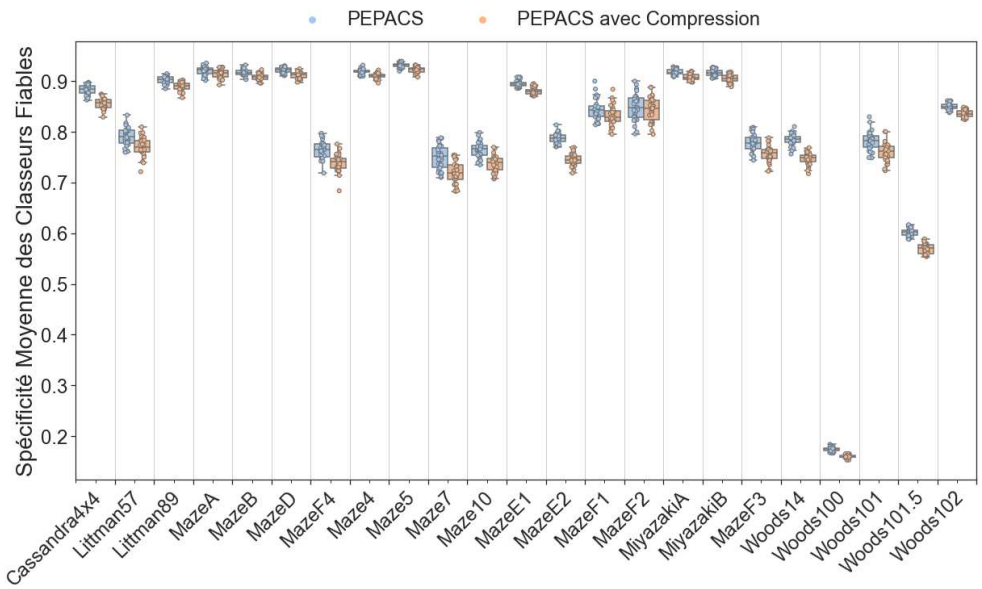


FIGURE E.12 – Spécificités moyennes des classeurs fiables de PEPACS avant et après compression pour chaque labyrinthe, avec bruitage des actions.

Résumé

Nous nous sommes intéressés au développement d'une intelligence artificielle autonome et explicable pour des environnements incertains. Les tâches à résoudre par les intelligences artificielles, et plus particulièrement par les algorithmes d'apprentissage automatique, sont de plus en plus complexes : ils doivent pouvoir s'adapter et co-évoluer en autonomie avec des environnements complexes, changeants et incertains qui sont à l'image de nos environnements quotidiens. Dans le même temps, il est de plus en plus nécessaire de pouvoir expliquer les comportements de ces algorithmes, ceux-ci pouvant être amenés à prendre des décisions critiques dans des situations qui peuvent profondément impacter la vie d'un individu. Pour répondre à ce double objectif, les concepts d'autonomie et d'explicabilité des intelligences artificielles, ainsi que d'incertitude environnementale, ont été cadrés afin de guider nos choix de conception d'une intelligence artificielle d'abord vers un algorithme d'apprentissage automatique intrinsèquement explicable comme les approches à base de règles, puis vers les systèmes de classeurs à anticipation. Nous avons alors mis en place de nouveaux systèmes de classeurs à anticipation dont le but était de renforcer leurs capacités à évoluer en autonomie et de manière explicable dans des environnements incertains. Les capacités de chacun de ces systèmes de classeurs à anticipation ont été évaluées au travers d'un protocole d'évaluation expérimental que nous avons conçu. Ce protocole expérimental nous a notamment permis d'agir sur l'incertitude des environnements pour mettre en avant les capacités des systèmes de classeurs à anticipation que nous avons développés. Nous avons également conçu un algorithme d'extraction des connaissances propre aux systèmes de classeurs à anticipation capable de renforcer l'explicabilité inhérente à ces systèmes, sans détériorer leurs capacités d'apprentissage et leur autonomie.
Mots-clés : Autonomie, Explicabilité, Incertitude, Systèmes de classeurs à anticipation

Abstract

We are interested in the development of autonomous and explainable artificial intelligence for uncertain environments. The tasks to be solved by artificial intelligences, and more particularly by machine learning algorithms, are increasingly complex : they must be able to adapt and co-evolve autonomously with complex, changing and uncertain environments that reflect our daily environment. Meanwhile, it is increasingly necessary to be able to explain the behavior of these algorithms, as they could make critical decisions that could have a major impact on an individual's life. To meet this double objective, the concepts of autonomy and explainability of artificial intelligences, as well as environmental uncertainty, have been framed in order to guide our design choices towards an intrinsically explainable machine learning algorithm such as rule-based approaches, and then towards anticipatory learning classifier systems. We then developed new anticipatory learning classifier systems in order to strengthen their ability to evolve autonomously and in an explainable way in uncertain environments. The capacities of each of these anticipatory learning classifier systems were evaluated through a carefully designed experimental evaluation protocol. This protocol enabled us to control the uncertainty of the environments to highlight the capabilities of the anticipatory learning classifier systems we devised. We also designed an algorithm dedicated to the extraction of knowledge that is specific to anticipatory learning classifier systems, capable of reinforcing the inherent explainability of these systems, without degrading their learning capacities and autonomy.
Keywords : Autonomy, Explainability, Uncertainty, Anticipatory learning classifier systems