THESE DE DOCTORAT DE L'ETABLISSEMENT UNIVERSITE MARIE ET LOUIS PASTEUR

Ecole doctorale n° <numéro de l'Ecole doctorale>

<Nom de l'Ecole doctorale>

Doctorat de Informatique

Par

M. SOTO FORERO Daniel

Adaptation en temps réel d'une séance d'entraînement par intelligence artificielle

Thèse présentée et soutenue à Besançon, le <date>

Composition du Jury :

<Civilité><Nom><Prénom>	<Fonction et établissement d'exercice>	Président
<Civilité><Nom><Prénom>	<Fonction et établissement d'exercice>	Rapporteur
<Civilité><Nom><Prénom>	<Fonction et établissement d'exercice>	Rapporteur
<Civilité><Nom><Prénom>	<Fonction et établissement d'exercice>	Examinateur
<Civilité><Nom><Prénom>	<Fonction et établissement d'exercice>	Examinatrice
<Civilité><Nom><Prénom>	<Fonction et établissement d'exercice>	Directeur de thèse
<Civilité><Nom><Prénom>	<Fonction et établissement d'exercice>	Codirecteur de thèse
<Civilité><Nom><Prénom>	<Fonction et établissement d'exercice>	Invité

REMERCIEMENTS

SOMMAIRE

I

CONTEXTE ET PROBLÉMATIQUES

1

INTRODUCTION

Ce chapitre introductif vise à expliquer les termes du sujet et à introduire la thématique principale abordée. Il présente la problématique de cette thèse, introduit les différentes contributions proposées et annonce le plan du manuscrit.

Comme l'indique [Nkambou et al., 2010], les environnements informatiques pour l'apprentissage humain (EIAH) sont des outils proposant des services devant permettre aux apprenants d'acquérir des connaissances et de développer des compétences dans un domaine spécifique. Pour fournir des services efficaces, le système doit intégrer une représentation des connaissances du domaine et des mécanismes pour utiliser ces connaissances. Il doit également être en mesure de raisonner et de résoudre des problèmes.

Le système *Artificial Intelligence - Virtual Trainer* (AI-VT) est un EIAH générique développé au département d'informatique des systèmes complexes (DISC) de l'institut de recherche FEMTO-ST. Cet outil informatique propose un ensemble d'exercices aux apprenants dans le cadre de séances d'entrainement. AI-VT intègre le fait qu'une séance d'entraînement se situe dans un cycle de plusieurs séances. Les réponses apportées par l'apprenant à chaque exercice sont évaluées numériquement sur une échelle prédéfinie, ce qui permet d'estimer les progrès de l'apprenant et de déduire les sous-domaines dans lesquels il peut avoir des difficultés. Une séance est générée par un système multi-agents associé à un système de raisonnement à partir de cas (RàPC) [Henriet et al., 2017]. Un apprenant choisit le domaine dans lequel il souhaite s'entraîner et AI-VT lui propose un test préliminaire. Les résultats obtenus permettent de placer l'apprenant dans le niveau de maîtrise adéquate. Le système génère ensuite une séance adaptée veillant à l'équilibre entre l'entraînement, l'apprentissage et la découverte de nouvelles connaissances. L'actualisation du niveau de l'apprenant est effectuée à la fin de chaque séance. De cette façon l'apprenant peut avancer dans l'acquisition des connaissances ou s'entraîner sur des connaissances déjà apprises.

Un certain nombre d'EIAH utilisent des algorithmes d'intelligence artificielle (IA) pour détecter les faiblesses et aussi pour s'adapter à chaque apprenant. Les algorithmes et modèles de certains de ces systèmes seront analysés dans les chapitres 2 et 3. Ces chapitres présenteront leurs propriétés, leurs avantages et leurs limites.

Le système AI-VT initial était capable d'ajuster les paramètres de personnalisation d'une séance à l'autre, mais il ne pouvait pas modifier une séance en cours même si certains exercices de celle-ci était trop simples ou trop complexes. Chaque séance était figée et devait être déroulée jusqu'à son terme avant de pouvoir identifier des acquis et des lacunes. Les travaux de cette thèse ont eu pour objectif de pallier ce manque.

La problématique principale de cette thèse est la personnalisation en temps réel du parcours d'apprentissage des apprenants dans le système AI-VT (*Artificial Intelligence - Virtual Trainer*)

Ici *le temps réel* est considéré comme étant le moment où se déroule la séance que l'apprenant est en train de suivre. Par conséquent, l'objectif est de rendre AI-VT plus dynamique pour l'identification des difficultés et l'adaptation du contenu personnalisé en fonction des connaissances démontrées.

La partie suivante présente une liste des principales contributions apportées par cette thèse à la problématique générale énoncée plus haut.

Ce travail de thèse a été éffectué au sein l'Université de Franche-Comté (UFC) devenue depuis le 1er janvier 2025 l'Université Marie et Louis Pasteur (UMLP). Ces recherches ont été menées au sein de l'équipe DEODIS du département d'informatique des systèmes complexes de l'institut de recherche FEMTO-ST, unité mixte de recherche (UMR) 6174 du centre national de la recherche scientifique (CNRS).

1.1/ CONTRIBUTIONS PRINCIPALES

La problématique principale de ces travaux de recherche a été déclinée en plusieurs sous-parties. Ces dernières sont présentées ci-dessous sous la forme de questions. Pour chacune d'elles, une ou plusieurs propositions ont été faites. Voici les questions de recherche abordées et les contributions apportées :

1. **Comment permettre au système AI-VT d'évoluer et d'intégrer de multiples outils ?** Pour répondre à cette question, une architecture modulaire autour du moteur initial d'AI-VT a été conçue et implémentée (ce moteur initial étant le système de raisonnement à partir de cas (RàPC) développé avant le démarrage de ces travaux de thèse).
2. **Comment déceler qu'un exercice est plus ou moins adapté aux besoins de l'apprenant ?** Choisir les exercices les plus adaptés nécessite d'associer une valeur liée à son utilité pour cet apprenant. Les algorithmes de régression permettent d'interpoler une telle valeur. Pour répondre à cette question, nous avons proposé de nouveaux algorithmes de régression fondés sur le paradigme du raisonnement à partir de cas et celui des systèmes multi-agents.
3. **Quel modèle et quel type d'algorithmes peuvent être utilisés pour recommander un parcours personnalisé aux apprenants ?** Pour apporter une réponse à cette question, un système de recommandation fondé sur l'apprentissage par renforcement a été conçu. L'objectif de ces travaux est de proposer un module permettant de recommander des exercices aux apprenants en fonction des connaissances démontrées et en se fondant sur les réponses apportées aux exercices précédents de la séance en cours. Ce module de révision de la séance en cours du RàPC est fondé sur l'inférence Bayésienne.
4. **Comment consolider les acquis de manière automatique ?** Une séance doit non seulement intégrer des exercices de niveaux variés mais également permettre à l'apprenant de renforcer ses connaissances. Dans cette optique, notre algorithme Bayésien a été enrichi d'un processus de Hawkes incluant une fonction d'oubli.

1.2/ PLAN DE LA THÈSE

Ce manuscrit est scindé en deux grandes parties. La première partie contient trois chapitres et la seconde en contient quatre. Le premier chapitre de la première partie (chapitre 2) vise à introduire le sujet, les concepts, les algorithmes associés, présenter le contexte général et l'application AI-VT initiale. Le deuxième chapitre de cette partie présente différents travaux emblématiques menés dans le domaine des environnements informatiques pour l'apprentissage humain. Le chapitre suivant conclut cette première partie en présentant le raisonnement à partir de cas.

Dans la seconde partie du manuscrit, le chapitre 5 explicite l'architecture proposée pour intégrer les modules développés autour du système AI-VT initial et élargir ses fonctionnalités. Le chapitre 6 propose deux outils originaux fondés sur le raisonnement à partir de cas et les systèmes multi-agents pour résoudre des problèmes de régression de façon générique. Le chapitre 7 présente l'application de ces nouveaux outils de régression dans un système de recommandation intégré à AI-VT utilisant le raisonnement Bayésien. Ce chapitre montre de quelle manière il permet de réviser une séance d'entraînement en cours. Le chapitre 8 montre l'utilisation du processus de Hawkes pour renforcer l'apprentissage.

Enfin, une conclusion générale incluant des perspectives de recherche termine ce manuscrit.

2

CONTEXTE

Dans ce chapitre sont décrits plus en détails le contexte applicatif et le contexte technique de ces travaux de recherche. Il présente les concepts et algorithmes utilisés dans le développement des modules. Ces modules font partie des contributions de cette thèse à l'environnement informatique pour l'apprentissage humain (EIAH) appelé AI-VT (*Artificial Intelligence - Artificial Trainer*).

Ce chapitre commence par une présentation des EIAH suivie d'une présentation sommaire du fonctionnement d'AI-VT. AI-VT étant un système de raisonnement à partir de cas (RàPC) modélisé sous la forme d'un système multi-agents (SMA), cette présentation des EIAH est suivie d'une présentation du RàPC, puis des SMA. Ce chapitre se termine par une présentation de différents algorithmes et notions implémentés dans AI-VT pour la personnalisation et l'adaptation des séances d'entraînement proposées par l'EIAH.

2.1/ LES STRATÉGIES D'APPRENTISSAGE HUMAIN ET LES ENVIRONNEMENTS INFORMATIQUES POUR L'APPRENTISSAGE HUMAIN (EIAH)

2.1.1/ LES STRATÉGIES D'APPRENTISSAGE

Dans [Lalitha and Sreeja, 2020], l'apprentissage est défini comme une fonction du cerveau humain acquise grâce au changement permanent dans l'obtention de connaissances et de compétences par le biais d'un processus de transformation du comportement lié à l'expérience ou à la pratique. L'apprentissage implique réflexion, compréhension, raisonnement et mise en œuvre. Un individu peut utiliser différentes stratégies pour acquérir la connaissance. Comme le montre la figure 2.1, les stratégies peuvent être classées entre mode traditionnel et mode en-ligne.

L'apprentissage traditionnel se réfère à l'apprentissage-type tel qu'il est fait dans une classe. Il est centré sur l'enseignant où la présence physique est l'élément fondamental dans la mesure où elle implique une unité de temps et de lieu. Ici, l'enseignant interagit directement avec l'élève, et les ressources sont généralement des documents imprimés. L'apprentissage et la participation doivent y être actifs, la rétroaction y est instantanée et il existe des interactions sociales. En revanche, les créneaux horaires, les lieux et les contenus sont rigides (décidés par l'enseignant et la structure dans laquelle les enseignements sont dispensés).

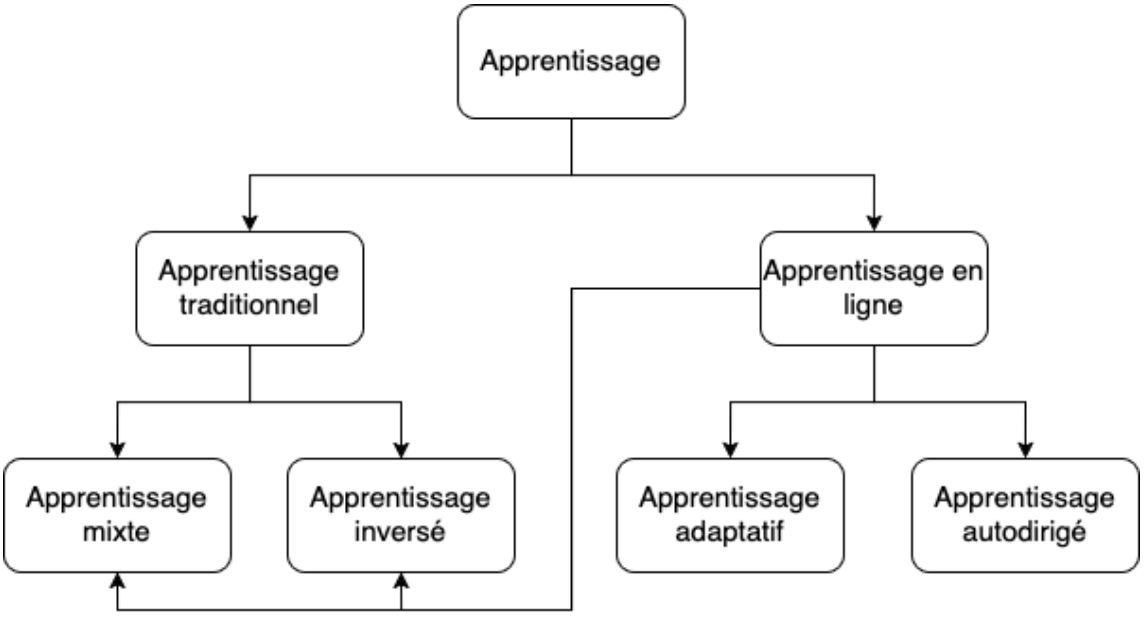

FIGURE 2.1 – Stratégies d'apprentissage (Traduit de [Lalitha and Sreeja, 2020])

L'autre stratégie globale est l'apprentissage en-ligne, qui s'appuie sur les ressources et l'information disponible sur le web. Cette stratégie incite les apprenants à être actifs pour acquérir de nouvelles connaissances de manière autonome grâce aux nombreuses ressources disponibles. Les points positifs de cette stratégie sont par exemple la rentabilité, la mise à niveau continue des compétences et des connaissances ou une plus grande opportunité d'accéder au contenu du monde entier. Pour l'apprenant, l'auto-motivation et l'interaction peuvent être des défis à surmonter. Pour l'enseignant, il est parfois difficile d'évaluer l'évolution du processus d'apprentissage de l'individu. L'apprentissage en-ligne fait aussi référence à l'utilisation de dispositifs électroniques pour apprendre (*e-learning*) où les apprenants peuvent être guidés par l'enseignant à travers des liens, du matériel spécifique d'apprentissage, des activités ou exercices.

Il existe également des stratégies hybrides combinant des caractéristiques des deux stratégies fondamentales, comme un cours en-ligne mais avec quelques séances en présentiel pour des activités spécifiques ou des cours traditionnels avec du matériel d'apprentissage complémentaire en-ligne.

2.1.2/ LES EIAH

Les EIAH sont des outils pour aider les apprenants dans le processus d'apprentissage et l'acquisition de la connaissance. Ces outils sont conçus pour fonctionner avec des stratégies d'apprentissage en-ligne et mixtes. Généralement, l'architecture de ces systèmes est divisée en quatre composants comme le montre la figure 2.2. Ces composants sont :

— Le *domaine* (*Domain Model* sur la figure 2.2), qui contient les concepts, les règles et les stratégies pour résoudre des problèmes dans un domaine spécifique. Ce composant peut détecter et corriger les erreurs des apprenants. En général, l'information est divisée en séquences pédagogiques.
— Le composant *apprenant* (*Student Model* sur la figure 2.2), qui est le noyau du

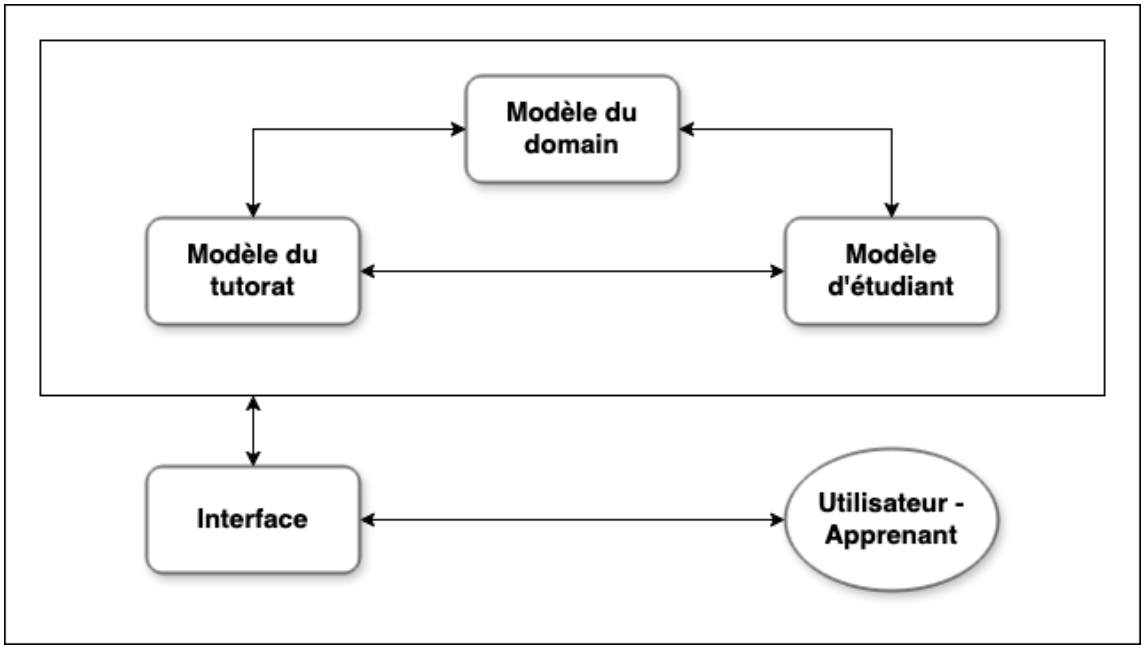

FIGURE 2.2 – L'architecture générale des EIAH, les composantes et leurs interactions (Traduit de [Nkambou et al., 2010])

système car il contient toute l'information utile à l'EIAH concernant l'apprenant (*User-Learner* sur la figure 2.2). Ce composant contient également les informations sur l'évolution de l'acquisition des compétences de l'apprenant. Ce module doit avoir les informations implicites et explicites pour pouvoir créer une représentation des connaissances acquises, non et partiellement acquises et l'évolution de l'apprentissage de l'apprenant. Ces informations doivent permettre de générer un diagnostic de l'état de la connaissance de l'apprenant, à partir duquel le système peut prédire les résultats dans certains domaines et choisir la stratégie optimale pour présenter les nouveaux contenus.

— L'*enseignant* (*Tutoring Model* sur la figure 2.2) est également modélisé sous la forme d'un composant. Il reçoit l'information des modules précédents, information grâce à laquelle il peut prendre des décisions sur le changement de parcours ou sur la stratégie d'apprentissage. Il peut également interagir avec l'apprenant.

— L'*interface* (*Interface* sur la figure 2.2) est le module chargé de la gestion des configurations de l'EIAH et des interactions entre ses composants.

Le développement et la configuration de ce type de systèmes relèvent de multiples disciplines et impliquent plusieurs domaines de recherche parmi lesquels figurent la pédagogie, l'éducation, la psychologie, les sciences cognitives et l'intelligence artificielle.

2.1.3/ L'EXERCISEUR INITIAL AI-VT

Le DISC travaille depuis plusieurs années sur l'utilisation potentielle de l'intelligence artificielle dans le cadre d'un exerciseur couvrant plusieurs domaines d'apprentissage. Cet exerciseur, appelé AI-VT(Artificial Intelligence Virtual Trainer) est fondé sur différents concepts d'intelligence artificielle et ses domaines d'application sont l'apprentissage de l'aïkido, des bases de l'algorithmique et de l'anglais. Un premier réseau de convolution

détermine les lacunes de l'apprenant en analysant les résultats de quelques exercices préliminaires. Puis un système de raisonnement à partir de cas distribué prend le relais et détermine une liste d'exercices à proposer en regard de ses lacunes et en tenant compte des séances qui ont déjà été proposées par le système à cet apprenant.

Dans le système AI-VT, une séance d'entraînement est proposée dans le cadre d'un cycle d'entraînement. Ainsi, un exercice proposé en première séance du cycle peut être reproposé ensuite dans une autre séance, afin de consolider les acquis et de remettre en mémoire certaines connaissances à l'apprenant. Une séance est entièrement consacrée à une seule et même compétence, déclinée en plusieurs sous-compétences. Des exercices dans différentes sous-compétences sont proposées à chaque séance. Les exercices d'une même sous-compétence sont regroupés. Un même exercice pouvant permettre de travailler deux sous-compétences différentes, un mécanisme de règlement des conflits a été implémenté dans AI-VT afin qu'un même exercice ne puisse être proposé plus d'une seule fois dans une même séance d'entraînement. Dans AI-VT, l'énoncé d'un exercice est constitué de deux parties : le contexte et la question. Cette structure modulaire permet d'associer plusieurs questions à un même contexte et inversement. Les réponses apportées par les apprenants à chaque exercice sont notées sur 10 points et le temps mis pour répondre à chaque question est comptabilisée. L'étudiant dispose d'une interface présentant un tableau de bord des exercices, sous-compétences et compétences qu'il ou elle a travaillé.

Pour l'apprentissage de l'Anglais, un robot est chargé d'énoncer les exercices de la séance. Au démarrage de cette thèse, les apprenants ont une liste personnalisée d'exercices proposée par le système AI-VT, mais c'est à l'enseignant de vérifier la validité des algorithmes/solutions proposés par les étudiants et d'identifier leurs difficultés. L'amélioration du système et de la personnalisation du parcours de l'apprenant implique une correction automatique des exercices et cela sans utiliser un entrainement spécifique à chaque exercice, mais cet objectif ne fait pas partie de cette thèse. La définition de profils d'apprenants par le recueil de traces reste également à travailler pour optimiser les aides et séances d'exercices.

Le système AI-VT a été implémenté en se fondant sur deux paradigmes de l'IA : la séance d'entraînement est construite par différents modules suivant la philosophie du raisonnement à partir de cas, et l'architecture logicielle a été modélisée selon un système multi-agents. Une présentation sommaire de ces deux paradigmes de l'IA sont présentés dans cette section, et celles-ci sont suivies de la présentation de différents algorithmes et fonctions implémentés dans l'EIAH AI-VT. Des états de l'art plus complets sur les EIAH et le RàPC sont présentés dans le chapitre suivant. Le système AI-VT, son architecture et les évolutions qui ont été réalisées sur celle-ci sont détaillés dans le chapitre 4.

2.2/ LE CONTEXTE TECHNIQUE

2.2.1/ LE RAISONNEMENT À PARTIR DE CAS (RÀPC)

Le raisonnement à partir de cas est un modèle de raisonnement fondé sur l'hypothèse que les problèmes similaires ont des solutions similaires. Ainsi, les systèmes de RàPC infèrent une solution à un problème posé à partir des solutions mises en œuvre auparavant pour résoudre d'autres problèmes similaires [Roldan Reyes et al., 2015].

Un cas est défini comme étant la représentation d'un problème et la description de sa solution. Les cas résolus sont stockés et permettent au système de RàPC de construire de nouveaux cas à partir de ceux-ci. Formellement, si P est l'espace des problèmes et S l'espace des solutions, alors un problème x et sa solution y appartiennent à ces espaces : $x \in P$ et $y \in S$. Si y est solution de x alors, un cas est représenté par le couple $(x, y) \in P \times S$. Le RàPC a besoin d'une base de N problèmes et de leurs solutions associées. Cette base est appelée base de cas. Ainsi tout cas d'indice $n \in [1, N]$ de cette base de cas est formalisé de la manière suivante : (x^n, y^n). L'objectif du RàPC est, étant donné un nouveau problème x^z de trouver sa solution y^z en utilisant les cas stockés dans la base de cas [Lepage et al., 2020].

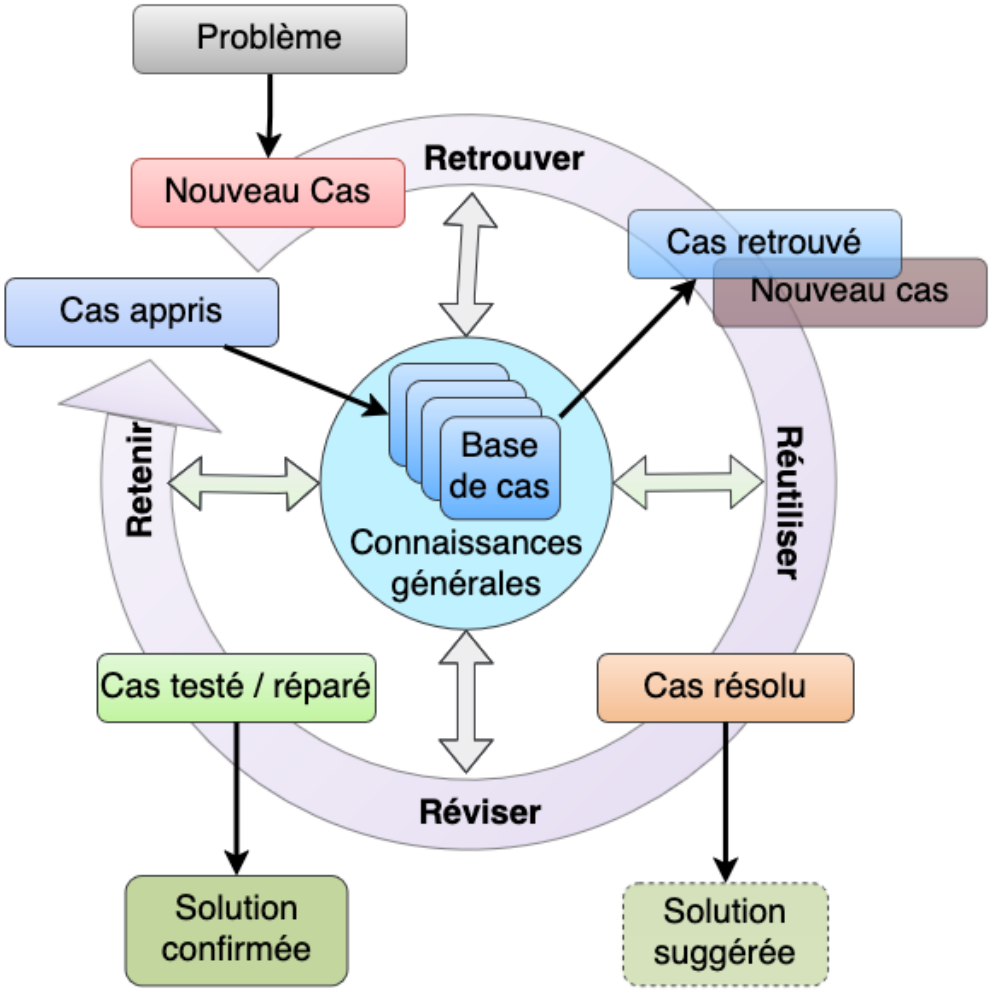

FIGURE 2.3 – Cycle fondamental du raisonnement à partir de cas (Adapté et traduit de [Leikola et al., 2018])

Le processus du RàPC est divisé en quatre étapes formant un cycle comme l'ont proposé Aamont et Plaza [Aamodt and Plaza, 1994]. La figure 2.3 montre le cycle, le flux d'informations ainsi que les relations entre chacune des étapes [Leikola et al., 2018] : re-

trouver (rechercher), réutiliser (adapter), réviser et retenir (stocker). Chacune des étapes est décrite dans [Richter and Weber, 2013] de la manière suivante.

2.2.1.1/ RETROUVER (RECHERCHER)

L'objectif de cette étape est de rechercher dans la base de cas, les cas similaires à un nouveau cas donné. Cette similarité n'est pas un concept général, mais dépend du contexte, de l'objectif et du type de données. La question fondamentale qui se pose dans cette étape est : quel est le cas le plus approprié dans la base de cas qui permet de réutiliser sa solution pour résoudre le nouveau problème donné ?.

Le nouveau cas est comparé à chaque cas de la base afin d'en évaluer la similitude. La comparaison entre les cas est différente selon la structure de la base et la manière dont les problèmes sont décrits dans celle-ci. Généralement, un problème est représenté par un ensemble d'attributs aux valeurs spécifiques. Cette représentation est connue comme la représentation attribut-valeur. La similitude entre deux cas de ce type est calculée suivant l'équation 2.1. La similitude entre deux cas attribut-valeur $c_1 = a_{1,1}, a_{1,1}, ..., a_{1,n}$ et $c_2 = a_{2,1}, a_{2,1}, ..., a_{2,n}$ est la somme pondérée des similitudes entre les attributs considérés individuellement. La pondération détermine l'importance de chaque attribut.

$$\sum_{i=1}^{n} \omega_i \; sim(a_{1,i}, a_{2,i}) \tag{2.1}$$

Dans cette étape, il est possible de sélectionner k différents cas similaires au nouveau cas donné.

2.2.1.2/ RÉUTILISER (ADAPTER)

L'idée du RàPC est d'utiliser l'expérience pour essayer de résoudre de nouveaux cas (problèmes). La figure 2.4 montre le principe de réutilisation d'un cas (problème) résolu (appelé *cas source*) auparavant pour le proposer en solution d'un nouveau cas (dénommé *cas cible*) après adaptation.

Si certains cas cibles sont très similaires au cas source le plus proche, il est cependant souvent nécessaire d'adapter la solution du cas source le plus similaire afin d'arriver à une solution satisfaisante pour le cas cible. De la même manière que pour la phase de recherche du cas le plus similaire, il existe de nombreuses stratégies d'adaptation. Celles-ci dépendent de la représentation des cas, du contexte dans lequel le RàPC est mis en œuvre et des algorithmes utilisés pour l'adaptation. L'une des stratégies les plus utilisées est la définition d'un ensemble de règles transformant la ou les solutions des cas sources les plus similaires au cas cible. L'objectif ici est d'inférer une nouvelle solution y^z du cas x^z à partir des k cas sources retrouvés dans la première étape.

2.2.1.3/ RÉVISER ET RÉPARER

La solution adaptée proposée doit ensuite être évaluée et révisée afin de satisfaire certains critères de validation. L'étape de révision est chargée d'évaluer l'applicabilité de la

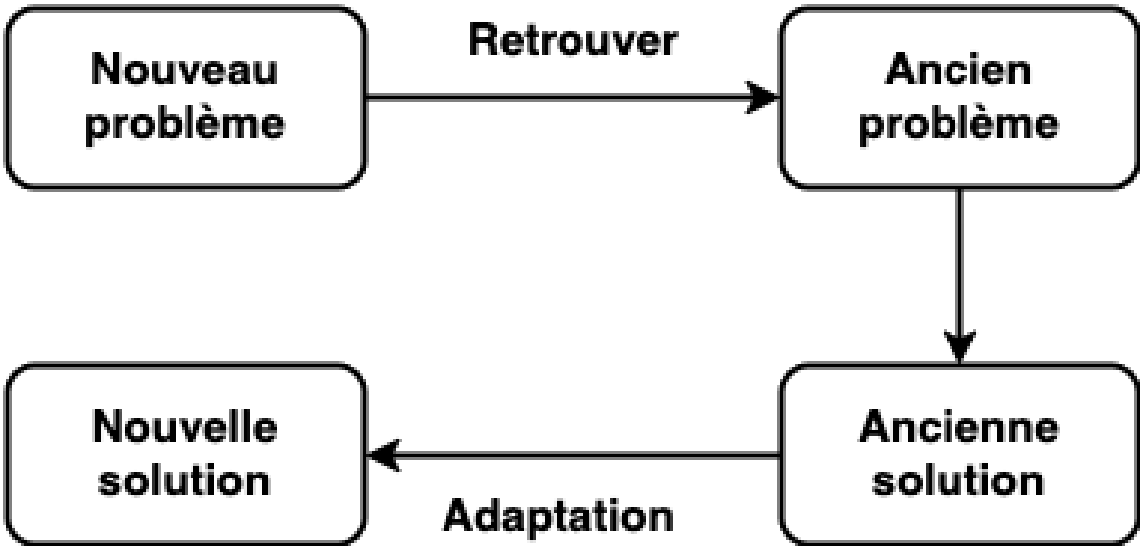

FIGURE 2.4 – Principe de réutilisation dans le RàPC (Traduit de [Richter and Weber, 2013])

solution cible obtenue suite à l'étape 'adaptation'. Cette évaluation peut être réalisée dans le monde réel ou dans une simulation.

L'objectif de cette phase de révision est la validation du couple (x^z, y^z). Autrement dit, le but est ici de vérifier si la solution trouvée y^z résout le problème x^z et satisfait les règles de validité et d'applicabilité. Si la solution n'est pas correcte, l'expert mettant en œuvre la solution doit ajuster ou modifier la solution cible proposée.

Le processus d'évaluation, de révision et de réparation peut être mis en œuvre via un processus d'apprentissage permettant d'améliorer la détection et la correction des failles des futures solutions générées. Dans les travaux de cette thèse, nous avons envisagé la possibilité d'utiliser différents outils d'apprentissage pour évaluer et réviser les solutions cibles dans certains cas particuliers.

L'évaluation peut être faite :

— par un expert humain capable d'évaluer la solution et sa pertinence dans le contexte applicatif,
— par un système fonctionnant en production et renvoyant le résultat de l'application de la solution,
— par un modèle statistique ou un système de test.

La correction des solutions peut être automatique, mais elle dépend du domaine et de l'information disponible. Un ensemble de règles peut aider à identifier les solutions non valides.

2.2.1.4/ RETENIR (STOCKER)

Si le cas cible résolu est jugé pertinent, alors celui-ci peut être stocké dans la base de cas pour aider ensuite à la résolution de futurs cas cible. La décision de stocker ou non un nouveau cas dans la base de cas doit tenir compte de la capacité de cette nouvelle base de cas à proposer de futures solutions pertinentes, évaluer et corriger les solutions

cibles générées d'une part, et maintenir une base de cas d'une taille ne gaspillant pas inutilement des ressources de stockage et de calculs du système de RàPC d'autre part.

2.2.1.5/ CONTENEURS DE CONNAISSANCE

Pour pouvoir exécuter le cycle complet, le RàPC dépend de quatre sources différentes d'information. Ces sources sont appelées les conteneurs de connaissances par [Richter, 2009]. Cet article définit les conteneurs suivants dans les systèmes de RàPC : le conteneur de cas, le conteneur d'adaptation, le conteneur du vocabulaire et le conteneur de similarité :

- Le conteneur de cas contient les expériences passées que le système peut utiliser pour résoudre les nouveaux problèmes. L'information est structurée sous la forme d'un couple (p, s), où p est la description d'un problème et s est la description de sa solution.
- Le conteneur d'adaptation stocke la ou les stratégies d'adaptation ainsi que les règles et paramètres nécessaires pour les exécuter.
- Le conteneur du vocabulaire contient l'information, sa signification et sa terminologie. Les éléments comme les entités, les attributs, les fonctions ou les relations entre les entités peuvent y être décrits.
- Le conteneur de similarité contient l'ensemble des connaissances liées et nécessaires au calcul de la similarité entre deux cas : les fonctions de calcul de similarité et les paramètres de mesure de similarité $sim(p_1, p_2)$ entre les cas p_1 et p_2.

La figure 2.5 montre les flux d'informations entre les étapes du RàPC et les conteneurs. Les flèches continues de la figure 2.5 décrivent l'ordre dans lequel les phases du cycle du RàPC sont exécutées. Les flèches avec des lignes discontinues matérialisent les flux d'information, c'est-à-dire les liens entre les étapes du cycle et les conteneurs de connaissance.

2.2.2/ LES SYSTÈMES MULTI-AGENTS

Les systèmes multi-agents sont des systèmes conçus pour résoudre des problèmes en combinant l'intelligence artificielle et le calcul distribué. Ces systèmes sont composés de multiples entités autonomes appelées agents, qui ont la capacité de communiquer entre elles et également de coordonner leurs comportements [Hajduk et al., 2019].

Les agents ont les propriétés suivantes :

- Autonomie : les agents fonctionnent sans intervention externe des êtres humains ou d'autres entités, ils ont un mécanisme interne qui leur permet de contrôler leurs états.
- Réactivité : les agents ont la capacité de percevoir l'environnement dans lequel ils sont et peuvent réagir aux changements qui se produisent.
- Pro-activité : les agents peuvent effectuer des changements, réagir à différents stimuli provenant de leur environnement et s'engager dans un processus cognitif interne.
- Coopération : les agents peuvent communiquer les uns avec les autres, échanger des informations afin de se coordonner et résoudre un même problème.
- Apprentissage : il est nécessaire qu'un agent soit capable de réagir dans un en-

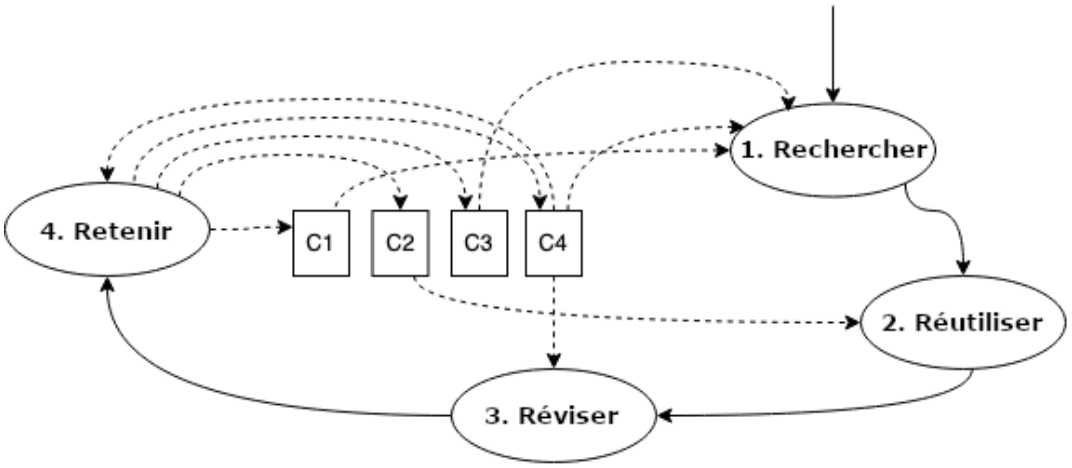

FIGURE 2.5 – Cycle du RàPC, les étapes, les conteneurs et leurs flux de données

vironnement dynamique et inconnu, il doit donc avoir la capacité d'apprendre de ses interactions et ainsi améliorer la qualité de ses réactions et comportements.

Il existe quatre types d'agent en fonction des capacités et des approches :

- Réactif : c'est l'agent qui perçoit constamment l'environnement et agit en fonction de ses objectifs.
- Fondé sur les réflexes : c'est l'agent qui considère les options pour atteindre ses objectifs et développe un plan à suivre.
- Hybride : il combine les deux modèles antérieurs en utilisant chacun d'eux en fonction de la situation et de l'objectif.
- Fondé sur le comportement : l'agent a à sa disposition un ensemble de modèles de comportement pour réaliser certaines tâches spécifiques. Chaque comportement se déclenche selon des règles prédéfinies ou des conditions d'activation. Le comportement de l'agent peut être modélisé avec différentes stratégies cognitives de pensée ou de raisonnement.

2.2.3/ DIFFÉRENTS ALGORITHMES ET FONCTIONS IMPLÉMENTÉS DANS AI-VT POUR LA PERSONNALISATION ET L'ADAPTATION DES SÉANCES D'ENTRAINEMENT PROPOSÉES

2.2.3.1/ PENSÉE BAYESIENNE

D'après les travaux de certains mathématiciens et neuroscientifiques, toute forme de cognition peut être modélisée dans le cadre de la formule de Bayes (équation 2.2), car elle permet de tester différentes hypothèses et donner plus de crédibilité à celle qui est confirmée par les observations. Étant donné que ce type de modèle cognitif permet l'apprentissage optimal, la prédiction sur les événements passés, l'échantillonnage représentatif

et l'inférence d'information manquante ; il est appliqué dans quelques algorithmes de machine learning et d'intelligence artificielle [Hoang, 2018].

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2.2}$$

La formule de Bayes calcule la probabilité a posteriori $P(A|B)$ de la plausibilité de la théorie A compte tenu des données B, et requiert trois termes : (1) une probabilité a priori $P(A)$ de la plausibilité de l'hypothèse A, (2) le terme $P(B|A)$ qui mesure la capacité de l'hypothèse A à expliquer les données observées B et (3) la fonction de partition $P(B)$ qui met en compétition toutes les hypothèses qui ont pu générer les données observées B. À chaque nouvelle évaluation de la formule, la valeur du terme a priori $P(A)$ est actualisée par la valeur du terme a posteriori $P(A|B)$. Ainsi, à chaque évaluation, le degré de plausibilité de chaque hypothèse est ajusté [Hoang, 2018].

Par la suite, nous explicitons quelques algorithmes, intègrés généralement dans l'étape "Rechercher" du RàPC, pour la recherche des cas les plus proches d'un nouveau cas.

2.2.3.2/ MÉTHODE DES K PLUS PROCHES VOISINS (K-NEAREST NEIGHBORHOOD - KNN)

Comme est défini dans [Cunningham and Delany, 2021], la méthode des 'k' plus proches voisins est une méthode pour classifier des données dans des classes spécifiques. Pour faire cela, il est nécessaire de disposer d'une base de données avec des exemples déjà identifiés. Pour classer de nouveaux exemples, la méthode attribue la même classe que celle de leurs voisins les plus proches. Généralement, l'algorithme compare les caractéristiques d'une entité avec plusieurs possibles voisins pour essayer d'obtenir des résultats plus précis. 'k' représente le nombre de voisins, sachant que l'algorithme peut être exécuté à chaque fois avec un nombre différent de voisins.

Étant donné un jeu de données D constitué de $(x_i)_{i\in[1,n]}$ données (où $n = |D|$). Chacune des données est décrite par F caractéristiques qui sont des valeurs numériques normalisées $[0, 1]$, et par une classe de labellisation $y_j \in Y$. Le but est de classifier une donnée inconnue q. Pour chaque $x_i \in D$, il est possible de calculer la distance entre q et x_i selon l'équation 2.3.

$$d(q, x_i) = \sum_{f\in F} w_f \delta(q_f, x_{if}) \tag{2.3}$$

La distance entre q et x_i est la somme pondérée de toutes les distances élémentaires $delta$ calculées pour chaque caractéristique. La fonction de distance δ peut être une métrique générique. Pour des valeurs numériques discrètes il est possible d'utiliser la définition 2.4,

$$\delta(q_f, x_{if}) = \begin{cases} 0 & q_f = x_{if} \\ 1 & q_f \neq x_{if} \end{cases} \tag{2.4}$$

et si les valeurs sont continues l'équation 2.5.

$$\delta(q_f, x_{if}) = |q_f - x_{if}| \tag{2.5}$$

Un poids plus important est généralement attribué aux voisins les plus proches. Le vote pondéré en fonction de la distance est une technique couramment utilisée (équation 2.6).

$$N(a,b) = \begin{cases} 1 & a = b \\ 0 & a \neq b \end{cases} \tag{2.6}$$

Le vote est couramment calculé selon l'équation 2.7.

$$vote(y_i) = \sum_{c=1}^{k} \frac{1}{d(q, x_c)^p} N(y_j, y_c) \tag{2.7}$$

une autre alternative est fondée sur le travail de Shepard, équation 2.8.

$$vote(y_i) = \sum_{c=1}^{k} e^{d(q,x_c)} N(y_j, y_c) \tag{2.8}$$

La fonction de distance peut être n'importe quelle mesure d'affinité entre deux objets, mais cette fonction doit répondre à quatre critères (équations 2.9).

$$\begin{aligned} &d(x,y) \geq 0 \\ &d(x,y) = 0, \ \textit{seulement si } x = y \\ &d(x,y) = d(y,x) \\ &d(x,z) \geq d(x,y) + d(y,z) \end{aligned} \tag{2.9}$$

2.2.3.3/ K-MOYENNES

Selon [Sinaga and Yang, 2020], K-Moyennes est un algorithme d'apprentissage utilisé pour partitionner et grouper des données. Considérons $X = \{x_1, ..., x_n\}$ un ensemble de vecteurs dans un espace Euclidien $\mathbb{R}^d$, et $A = \{a_1, ..., a_c\}$ où c est le nombre de groupes. Considérons également $z = [z_{ik}]_{n \times c}$, où z_{ik} est une variable binaire (i.e. $z_{ik} \in \{0, 1\}$) qui indique si une donnée x_i appartient au k-ème groupe, $k = 1, , c$. La fonction bijective k-moyenne est définie selon l'équation 2.10.

$$J(z, A) = \sum_{i=1}^{n} \sum_{k=1}^{c} z_{ik} \parallel x_i - a_k \parallel^2 \tag{2.10}$$

Cet algorithme minimise la fonction objectif des k moyennes $J(z, A)$: à chaque itération, les termes a_k et z_{ik} sont calculés selon les équations 2.11 et 2.12.

$$a_k = \frac{\sum_{i=1}^{n} z_{ik} x_{ij}}{\sum_{i=1}^{n} z_{ik}} \tag{2.11}$$

$$z_{ik} = \begin{cases} 1 & si \ \parallel x_i - a_k \parallel^2 = min_{1 \leq k \leq c} \parallel x_i - a_k \parallel^2 \\ 0, & sinon \end{cases} \tag{2.12}$$

2.2.3.4/ MODÈLE DE MÉLANGE GAUSSIEN GMM (*Gaussian Mixture Model*)

Le modèle de mélange gaussien (GMM) est un modèle probabiliste composé de plusieurs modèles gaussiens simples. Ce modèle est décrit dans [Wang et al., 2021]. En considérant une variable d'entrée multidimensionnelle $x = \{x_1, x_2, ..., x_d\}$, GMM multivarié est défini selon dans l'équation 2.13.

$$p(x|\theta) = \sum_{k=1}^{K} \alpha_k g(x|\theta_k) \tag{2.13}$$

Dans cette équation, K est le nombre de modèles gaussiens uniques dans GMM, également appelés composants ; α_k est la probabilité de mélange de la k-ème composante respectant la condition $\sum_{k=1}^{K} \alpha_k = 1$.

θ_k représente la valeur moyenne et la matrice de covariance de chaque modèle gaussien unique : $\theta_k = \{\mu_k, \Sigma_k\}$. Pour un seul modèle gaussien multivarié, nous avons la fonction de densité de probabilité $g(x)$ (équation 2.14).

$$g(x; \mu, \Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right) \tag{2.14}$$

La tâche principale est d'obtenir les paramètres α_k, θ_k pour tout k défini dans GMM en utilisant un ensemble de données avec N échantillons d'entrainement. Une solution classique pour l'estimation des paramètres requis utilise l'algorithme de maximisation des attentes (Expectation-Maximization EM), qui vise à maximiser la vraisemblance de l'ensemble de données. Il s'agit d'un algorithme itératif durant lequel les paramètres sont continuellement mis à jour jusqu'à ce que la valeur delta log-vraisemblance entre deux itérations soit inférieure à un seuil donné.

2.2.3.5/ FUZZY-C

Fuzzy C-Means Clustering (FCM) est un algorithme de clustering flou non supervisé largement utilisé [Xu et al., 2021]. Le FCM utilise comme mesure de distance la mesure euclidienne. Supposons d'abord que l'ensemble d'échantillons à regrouper est $X = \{x_1, x_2, ..., x_n\}$, où $x_j \in R^d (1 \leq j \leq n)$ dans un espace Euclidien à d dimensions, et c le nombre de clusters. L'équation 2.15 montre la fonction objectif de FCM.

$$J(U, V) = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^m (x_j - v_i)^T A (x_j - v_i) \tag{2.15}$$

où A est la matrice métrique, m est un nombre quelconque $(m > 1)$ qui dénote le degré de flou, u_{ij} est le degré d'appartenance du j-ème échantillon x_j qui appartient au i-ème cluster, dont le centre est v_i. $U = (u_{ij})$, $V = [v_1, v_2, ..., v_c], 1 \leq i \leq c, 1 \leq j \leq n, 2 \leq c < n$ satisfaisant les conditions de l'équation 2.16.

$$\sum_{i=1}^{c} u_{ij} = 1, u_{ij} \geq 0 \tag{2.16}$$

Pour compléter le modèle, les équations de mise à jour pour le centre du cluster v_i (équation 2.17) et des degrés d'appartenance u_{ij} (équation 2.18) sont définies.

$$v_i = \frac{\sum_{j=1}^{n} u_{ij}^m x_j}{\sum_{j=1}^{n} u_{ij}^m} \tag{2.17}$$

$$u_{ij} = \frac{((x_j - v_i)^T A (x_j - v_i))^{\frac{2}{m-1}}}{\left(\sum_{h=1}^{c} (x_j - v_h)^T A (x_j - v_h)\right)^{\frac{2}{m-1}}} \tag{2.18}$$

Les algorithmes qui se trouvent dans la suite de cette section sont des algorithmes qui font partie de l'intelligence artificielle et sont utilisés dans certains EIAH pour améliorer les recommandations, essayer de corriger et de détecter les faiblesses des apprenants de façon automatique.

2.2.3.6/ BANDIT MANCHOT MAB (*Multi-Armed Bandits*)

Dans [Gupta et al., 2021] MAB est décrit comme un problème qui appartient au domaine de l'apprentissage par renforcement. Celui-ci représente un processus de prise de décision séquentielle dans des instants t de temps, où un utilisateur doit sélectionner une action $k_t \in K$ à réaliser dans un ensemble K fini d'actions possibles et donnant une récompense inconnue de l'utilisateur. L'objectif est de maximiser la récompense cumulée globale dans le temps. La clé pour trouver une solution au problème est de trouver l'équilibre optimal entre l'exploration (exécuter des actions inconnues pour collecter de l'information complémentaire) et l'exploitation (continuer à exécuter les actions déjà connues pour cumuler plus de gains). L'un des algorithmes utilisés pour résoudre ce problème c'est l'échantillonnage de Thompson.

2.2.3.7/ ÉCHANTILLONNAGE DE THOMPSON TS (*Thompson Sampling*)

Comme indiqué dans [Lin, 2022], l'algorithme d'échantillonnage de Thompson est un algorithme de type probabiliste utilisé généralement pour résoudre le problème MAB. Il s'appuie sur un modèle Bayésien dans lequel une distribution de probabilités *Beta* est initialisée. Cette distribution de probabilités est ensuite affinée de manière à optimiser la valeur résultat à estimer. Les valeurs initiales des probabilités de Beta sont définies sur l'intervalle $[0, 1]$. Elle est définie à l'aide de deux paramètres α et β.

L'échantillonnage de Thompson peut être appliqué à la résolution du problème du Bandit Manchot(MAB). Dans ce cas, les actions définies dans le MAB ont chacune une incidence sur la distribution de probabilités Beta de l'échantillonnage de Thompson. Pour chaque action de MAB, les paramètres de Beta sont initialisés à 1. Ces valeurs changent et sont calculées à partir de récompenses obtenues : si au moment d'exécuter une action spécifique le résultat est un succès, alors la valeur du paramètre α de sa distribution Beta augmente mais si le résultat est un échec alors c'est la valeur du paramètre β de sa distribution Beta qui augmente. De cette façon, la distribution pour chacune des actions possibles est ajustée en privilégiant les actions qui génèrent le plus de récompenses.

L'échantillonnage de Thompson est un cas particulier de la loi de Dirichlet comme le montre la figure 2.6. L'équation 7.1 décrit formellement la famille des courbes générées

par la distribution Beta.

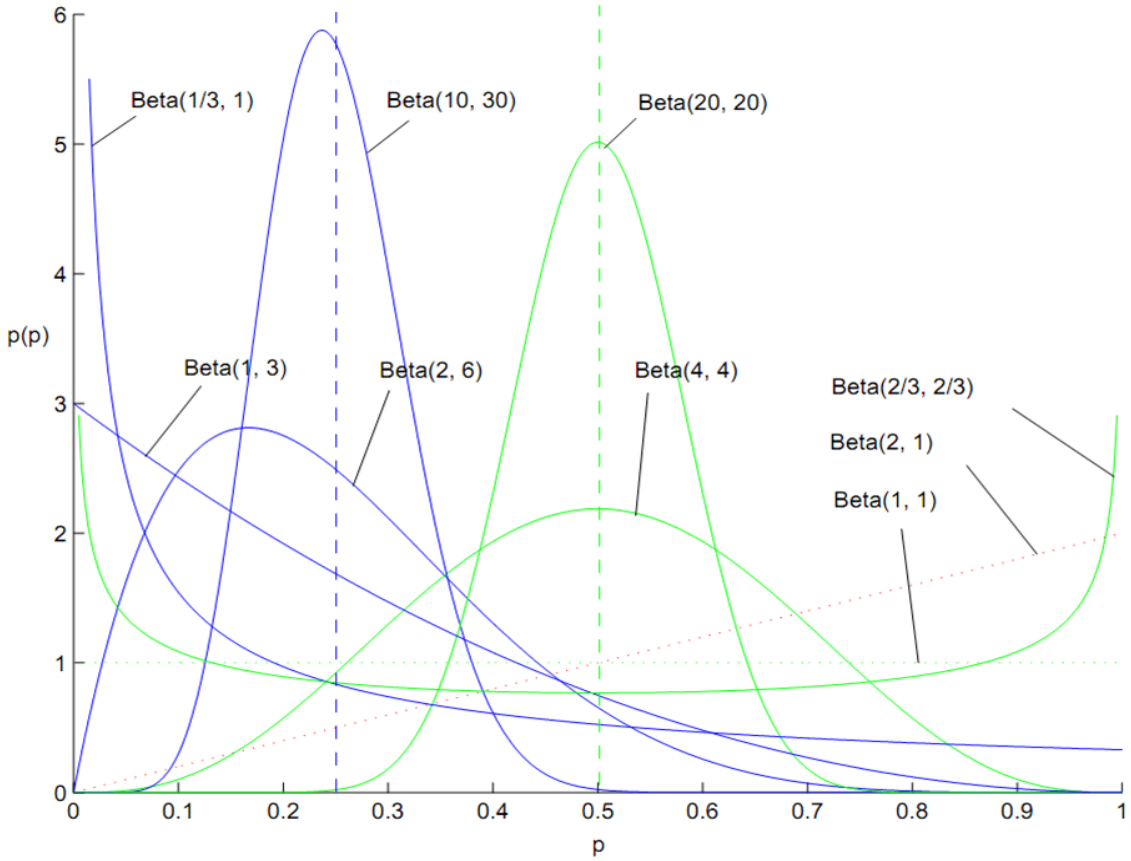

FIGURE 2.6 – Comportement de la distribution Beta avec différentes valeurs de paramètres $alpha$ et $beta$

$$B(x,\alpha,\beta) = \begin{cases} \frac{x^{\alpha-1}(1-x)^{\beta-1}}{\int_0^1 u^{\alpha-1}(1-u)^{\beta-1}du} & si\ x \in [0,1] \\ 0 & sinon \end{cases} \tag{2.19}$$

La section suivante présente un état de l'art plus détaillé des EIAH et des systèmes de RàPC dédiés aux EIAH.

II

ÉTAT DE L'ART

3

Environnements Informatiques d'Apprentissage Humain

Ce chapitre présente les travaux sur les EIAH en lien avec le travail de cette thèse. Les EIAH référencés dans cet état de l'art sont classés selon le thème principal abordé.

3.1/ L'Intelligence Artificielle

L'intelligence artificielle a été appliquée avec succès au domaine de l'éducation dans plusieurs écoles de différents pays et pour l'apprentissage de l'informatique, des langues étrangères et des sciences. [Zhang. and Aslan, 2021] référence un grand nombre de travaux sur la période de 1993-2020. Cet article montre que les besoins et contraintes des EIAH ne sont pas différentes selon l'age, le niveau culturel, ou le niveau éducatif des apprenants. Cet article montre également qu'il est possible d'adapter une stratégie d'apprentissage pour quelque soit l'apprenant et de maximiser le rendement et acquisition des connaissances grâce à différentes techniques d'intelligence artificielle.

[Chiu et al., 2023] présente une révision des travaux d'intelligence artificielle appliquée à l'éducation extraits des bases de données bibliographiques ERIC, WOS et Scopus sur la période 2012-2021. Ce travail est focalisé sur les tendances et les outils employées pour aider le processus d'apprentissage. Une classification des contributions de l'IA est faite, fondée sur 92 travaux scientifiques. Les contributions et l'utilité des outils d'IA implémentés dans les EIAH évalués dans cet article sont évaluées sur les quatre domaines suivants : l'apprentissage, l'enseignement, l'évaluation et l'administration. Pour l'apprentissage, l'IA est généralement utilisée pour attribuer des tâches en fonction des compétences individuelles, fournir des conversations homme-machine, analyser le travail des étudiants pour obtenir des commentaires et accroître l'adaptabilité et l'interactivité dans les environnements numériques. Pour l'enseignement, des stratégies d'enseignement adaptatives peuvent être appliquées pour améliorer la capacité des enseignants à enseigner et soutenir le développement professionnel des enseignants. Dans le domaine de l'évaluation, l'IA peut fournir une notation automatique et prédire les performances des élèves. Enfin, dans le domaine de l'administration, l'IA contribue à améliorer la performance des plateformes de gestion, à soutenir la prise de décision pédagogique et à fournir des services personnalisés. Cet article met également en lumière les lacunes suivantes de l'IA :

— Les ressources recommandées par les plateformes d'apprentissage personnali-

sées sont trop homogènes,
— les données nécessaires pour les modèles d'IA sont très spécifiques,
— le lien entre les technologies et l'enseignement n'est pas bien clair. En effet, la plupart des travaux sont conçus pour un domaine ou un objectif très spécifique difficilement généralisable,
— l'inégalité éducative, car les technologies ne suscitent pas la motivation chez tous les élèves. De plus, l'intelligence artificielle peut parfois engendrer des attitudes négatives et s'avérer difficile à maîtriser et à intégrer efficacement dans les cours.

Les techniques d'IA peuvent aussi aider à prendre des décisions stratégiques visant des objectifs à longue échéance comme le montre le travail de [Robertson and Watson, 2014]. Les jeux de stratégie présentent en effet plusieurs particularités singulières. Ils continent plusieurs règles et des environnements contraints. Ils nécessitent de mettre en œuvre des actions et réactions en temps réel. Ils intègrent des aléas et des informations cachées. Les techniques principales identifiées et implémentées dans ces jeux de stratégie sont l'apprentissage par renforcement, les modèles bayésiens, la recherche dans une arborescence??, le raisonnement à partir de cas, les réseaux de neurones et les algorithmes évolutifs. Il est également important de noter que la combinaison de ces techniques permet dans certains cas d'améliorer le comportement global des algorithmes et d'obtenir de meilleures réponses.

3.2/ SYSTÈMES DE RECOMMANDATION DANS LES EIAH

Des systèmes de recommandation sont régulièrement intégrés aux EIAH. Ils permettent en effet de tenir compte des exigences, des besoins, du profil, des talents, des intérêts et de l'évolution de l'apprenant pour s'adapter et recommander des ressources ou des exercices dans le but d'améliorer l'acquisition et la maîtrise de concepts et des connaissances en général. L'adaptation de ces systèmes peut être de deux types [Muangprathub et al., 2020] :

— l'adaptation de la présentation qui montre aux apprenants les ressources en concordance avec leurs faiblesses et
— l'adaptation de la navigation qui change la structure du cours en fonction du niveau et style d'apprentissage de chaque apprenant.

Ces systèmes montrent des effets positifs pour les apprenants comme le révèle le travail de [Huang et al., 2023] qui utilise l'IA pour personnaliser des recommandations de ressources en vidéo et ainsi aider et motiver les apprenants. L'évolution des résultats entre les tests préliminaires et ceux réalisés après la finalisation du cours des groupes ayant suivi les cours avec et sans système de recommandation démontre cet effet positif. La figure 3.1 montre l'architecture du système qui utilise l'intelligence artificielle dans différentes étapes du processus avec les données de l'apprenant sous la supervision et contrôle du professeur.

Le travail de [Seznec et al., 2020] propose un modèle générique de recommandation qui peut être utilisé dans les systèmes de recommandation de produits ou les systèmes d'apprentissage. Ce modèle est fondé sur l'algorithme par renforcement UCB (*Upper Confidence Bound*) qui permet de trouver une solution approchée à une variante du problème *Bandit manchot* non stationnaire (MAB, présenté dans la section **??** de ce manuscrit), où les récompenses de chaque action diminuent progressivement après chaque utilisation. Pour valider le modèle, une comparaison avec trois autres algorithmes a été menée en

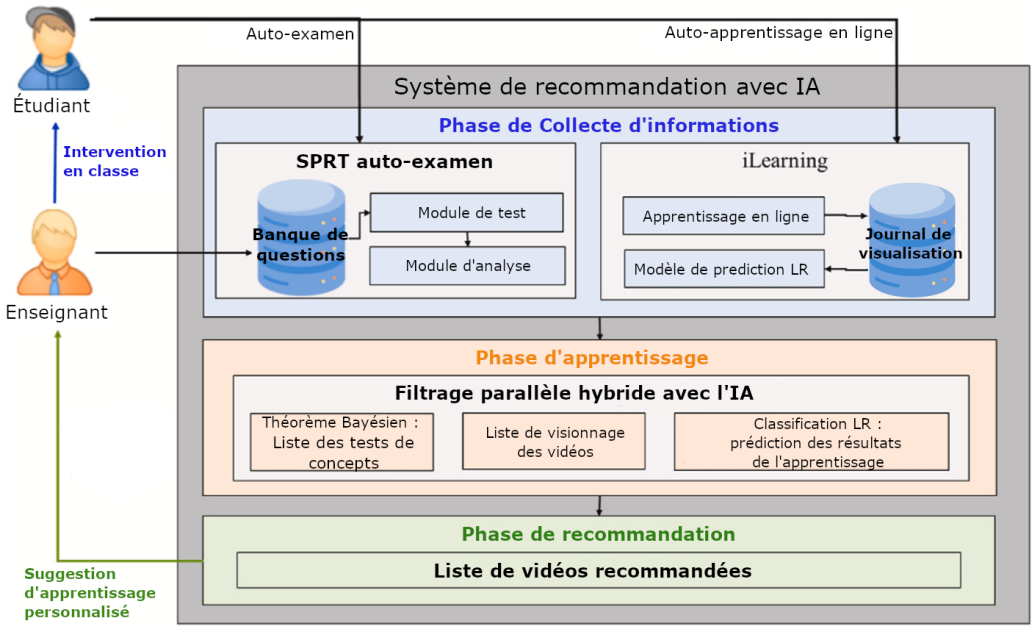

FIGURE 3.1 – Traduction de l'architecture du système de recommandation proposé dans [Huang et al., 2023]

considérant une base de données réelle. La performance du système a été démontrée en se fondant sur la moyenne accumulée du regret et la moyenne de la récompense accumulée.

Aussi, [Ingkavara et al., 2022] met en évidence que les technologies et les systèmes de recommandation peuvent s'adapter à différents besoins et aspirations et qu'ils peuvent également favoriser l'apprentissage auto-régulé. Ce type d'apprentissage aide les apprenants à acquérir les compétences pour diminuer les délais de réponse et devenir plus performants. Ce type d'apprentissage permet de définir des objectifs variables dans un environnement structuré, également d'adapter le temps nécessaire à l'acquisition et la maîtrise de connaissances et de compétences. De plus, avec les systèmes de recommandation est possible d'avoir accès à des ressources et ainsi constituer un excellent outil pour le renforcement des connaissances acquises.

L'IA est utilisée pour l'apprentissage adaptatif afin de suggérer des ressources d'étude dans le travail de [Lalitha and Sreeja, 2020]. Le système proposé par ces auteurs intègre un module de personnalisation qui collecte l'information de l'apprenant et ses exigences, un module de classification qui compare l'information avec d'autres profils en utilisant l'algorithme KNN et un module de recommandation chargé d'identifier les ressources communes entre les profiles afin d'en extraire des informations complémentaires provenant d'Internet. Ce système inclut également un algorithme *Random Forest* pour améliorer le processus d'apprentissage du nouvel apprenant. Les techniques et les stratégies sont très variées mais l'objectif est de s'adapter aux apprenants et à leurs besoins. Dans [Zhao et al., 2023] les données des apprenants sont collectées et classées dans des groupes présentant des caractéristiques similaires. Ensuite, la méthode d'analyse par enveloppement des données (DEA) permet d'identifier les besoins spécifiques de chaque groupe et ainsi proposer un parcours d'apprentissage personnalisé.

La représentation des données des enseignants, des apprenants et l'environnement sont des aspects particulièrement importants à considérer dans l'implémentation de ces systèmes de recommandation. Ces aspects influencent en effet les performances globales des EIAH dans lesquels ils sont intégrés. La proposition de [Su et al., 2022] consiste à stocker les données des apprenants dans deux graphes évolutifs qui contiennent les relations entre les questions et les réponses correctes ainsi que les réponses données par les apprenants. Ces informations et relations permettent de construire un graphe global propre à chaque apprenant en donnant une information sur son état cognitif et ainsi de personnaliser son parcours. Dans [Muangprathub et al., 2020] définit dans son EIAH trois ensembles de concepts :

— les objets (G),
— les attributs (M) et
— les relations entre G et M.

L'originalité de ce travail consiste à analyser ces concepts en via l'approche FCA (*Formal Context Analysis*). Ce type de représentation permet de mettre en relation les ressources d'étude, les questions et les sujets. En conséquence, si un apprenant étudie un sujet S_1 et si une règle relie S_1 au sujet S_4 ou la question $Q_{3,4}$, alors le système peut suggérer l'étude des ressources d'étude associés au sujet S_4 ou aux questions relies par les règles. Par ailleurs, cet algorithme se fonde sur une structure du cours prédéfinie pour recommander un parcours exhaustif d'apprentissage.

[Zhou and Wang, 2021] propose un système de recommandation pour personnaliser des exercices pour l'apprentissage de l'anglais. Le système décrit contient un module principal qui représente les apprenants comme des vecteurs selon le modèle DINA où sont stockes les points des connaissances acquises. Le vecteur de connaissances K est de dimension n ($K = \{k_1, k_2, ..., k_n\}$) où chaque dimension correspond à un point de connaissance. Ainsi, $k_i = 1$ signifie que l'apprenant maîtrise le point de connaissance k_i. A contrario, $k_i = 0$ signifie que l'apprenant doit étudier le point de connaissance k_i car car non acquise. La recommandation des questions proposées par le système se sert par ailleurs d'une librairie de ressources associes à chacune des questions possibles et permettant à l'apprenant de renforcer son apprentissage et d'avancer dans le programme du cours.

Certains travaux de recommandation et de personnalisation considèrent des variables complémentaires aux notes. C'est le cas de l'EIAH proposé par [Ezaldeen et al., 2022], qui lie l'analyse du comportement de l'apprenant à une analyse sémantique de son propre profil. La première étape consiste à collecter les données nécessaires pour créer un profil de l'apprenant. Fort de ce profil complet, des associations sont faites entre l'apprenant et un groupe de catégories d'apprentissages prédéfinies. Les apprentissages sont regroupés selon ses préférences et les données historiques. Puis les concepts associés pour les catégories sont recherchés et permettent ainsi de générer un guide pour obtenir des ressources internet qu'il est possible de recommander. Le système est divisé en quatre niveaux représentés sur la figure 3.2 où chacun des niveaux est chargé d'une tache spécifique. Les résultats d'un niveau sont envoyés au niveau suivant jusqu'à proposer des recommandations personnalisées pour l'apprenant.

Le tableau 3.1 présente un récapitulatif des articles analysés dans l'état de l'art des EIAH. Les articles apparaissent dans le même ordre qui ont été cités dans le chapitre. Les limites décris correspondent au temps de calcul, quantité de données d'entrainement, complexité de définitions ou des règles ainsi comme la quantité de variables subjectives.

Une limitation générale qui présentent les algorithmes d'IA dans les EIAH est que ce type

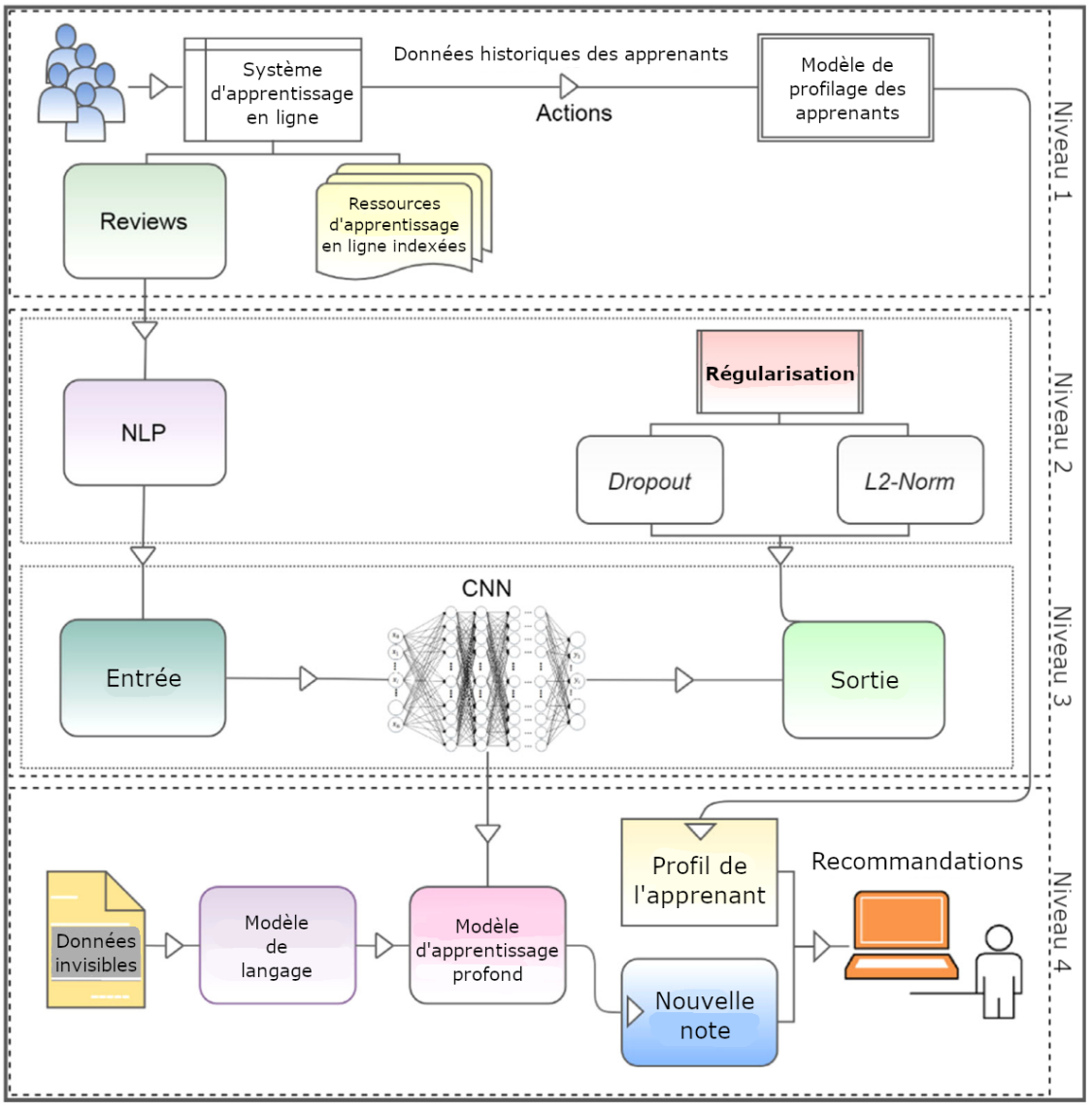

FIGURE 3.2 – Traduction des niveaux du système de recommandation dans [Ezaldeen et al., 2022]

d'algorithmes ne permettent pas d'oublier l'information avec laquelle ils ont été entrainés, une propriété nécessaire pour les EIAH dans certaines situations où peut se produire un dysfonctionnement du système, la réévaluation d'un apprenant ou la restructuration d'un cours. Par ailleurs, les représentations et algorithmes peuvent changer et évoluer. Aujourd'hui il y a encore beaucoup de potentiel pour développer encore les capacités de calcul et d'adaptation.

Référence	Limites
[Zhang. and Aslan, 2021]	Niveau global d'application de EIAH. Analyse des effets de l'IA
[Chiu et al., 2023]	Aspects non cognitifs en IA. Motivation des apprenants
[Robertson and Watson, 2014]	Peu de test. Pas de standard d'évaluation
[Huang et al., 2023]	Recommandation en fonction de la motivation seulement. N'est pas général pour tous les apprenants
[Seznec et al., 2020]	Le modèle n'a pas été testé avec données d'étudiants. UCB prends beaucoup de temps pour explorer les alternatives
[Ingkavara et al., 2022]	Modèle très complexe. Définition de constantes subjectif et beaucoup de variables.
[Lalitha and Sreeja, 2020]	A besoin de beaucoup de données pour entrainement. Ne marche pas dans le cas 'cold-start', n'est pas totalement automatisé
[Su et al., 2022]	Estimation de la connaissance de façon subjective. Il est nécessaire une étape d'entrainement
[Muangprathub et al., 2020]	Structure des règles complexe. Définition de la connaissance de base complexe
[Zhou and Wang, 2021]	Utilisation d'un filtre collaboratif sans stratification. Utilisation d'une seule métrique de distance
[Ezaldeen et al., 2022]	Il n y a pas de comparaison avec d'autres modèles différents de CNN. Beaucoup de variables à valeurs subjectives

TABLE 3.1 – Tableau de synthèse des articles analysés dans l'état de l'art des EIAH

4

ÉTAT DE L'ART (RAISONNEMENT À PARTIR DE CAS)

Le raisonnement à partir de cas est une approche fondée sur la connaissance. Il s'agit d'une technique d'intelligence artificielle dont l'idée est de résoudre un nouveau problème grâce aux connaissances déjà acquises par le système et en tenant un raisonnement fondé sur l'analogie. Le RàPC est apparu comme une alternative pour améliorer les systèmes experts. Shank et Abelson [Schank and Abelson, 1977] ont initialement mené des travaux sur l'organisation hiérarchique de la mémoire pour imiter le raisonnement humain. Ceux-ci ont servi de fondement aux travaux de Janet Kolodner [Kolodner, 1983] et ont abouti à l'implémentation un système fondé sur ces principes en 1983. Le terme *raisonnement à partir de cas* est utilisé pour la première fois en 1989 par Riesbeck et Shank [C.K. and R.C., 1989].

Comme vu dans le chapitre du contexte, le raisonnement à partir de cas utilise quatre conteneurs de connaissance pour représenter la connaissance complète du système. Chaque conteneur stocke l'information associée à une fonction spécifique et tout est fondé sur le carré d'analogie : des solutions ayant permis de résoudre un problème sont réutilisées afin de résoudre un nouveau problème similaire. La plupart des systèmes de RàPC fonctionnent suivant un cycle composé de quatre étapes : retrouver, réutiliser, réviser et retenir.

Les travaux présentés dans ce chapitre ont nouri notre réflexion et ont ouvert des perspectives d'amélioration de performance du système AI-VT. Ce chapitre commence par la présentation de travaux emblématiques intégrant un réseau de neurones au cycle de RàPC. Cette première section est suivie d'une présentation de différents travaux liés à l'explicabilité, toujours en utilisant le RàPC. Des travaux récents sur la représentation des cas et le cycle du RàPC, puis sur l'intégration d'autres algorithmes au cycle du RàPC sont ensuite présentés. Pour finir, la prédiction et la recommandation via RàPC terminent ce chapitre.

4.1/ INTÉGRATION D'UN RÉSEAU DE NEURONES DANS LE CYCLE DU RÀPC

Plusieurs travaux combinent le RàPC avec les réseaux de neurones avec l'objectif d'améliorer les réponses générées et optimiser les performances de chaque algorithme. C'est une idée explorée dès 2009 par Jung *et al.* [Jung et al., 2009]. Les auteurs de cet article

ont en effet développé un système fondé sur l'hybridation du RàPC avec des réseaux de neurones pour concevoir des produits. Cette hybridation est iméplementée dans les phases "rechercher" et "réutiliser" : le système détermine de façon automatique les valeurs pour les paramètres nécessaires à la conception d'un produit particulier en suivant le cycle traditionnel du RàPC. Avec l'algorithme de k-moyennes, est extrait un cas représentatif de la base de cas et l'adaptation des solutions des voisins trouvées est faite avec le réseau de neurones RBFN (*Radial Basis Function Network*). L'évaluation du système est faite en utilisant une base de données contenant 830 tests, les résultats démontrent que les produits conçus s'ajustent avec un grand pourcentage aux normes définies.

Dans [Butdee and Tichkiewitch, 2011] un réseau de neurones classique est implémenté pour définir la géométrie d'une matrice pour l'extrusion de l'aluminium. En effet, actuellement c'est un processus qui se fait manuellement et par essai et erreur. Le RàPC est alors utilisé pour aider à déterminer les valeurs optimales des paramètres du réseau, en utilisant l'information des matrices d'extrusion déjà testées.

Petrovic *et al.* [Petrovic et al., 2016] proposent un système de raisonnement à partir de cas pour calculer la dose optimale de radiation pour le traitement du cancer. Dans ce domaine particulier de la radiothérapie, administrer une dose nécessite de connaître avec précision le nombre de faisceaux et l'angle de chacun d'eux. L'algorithme proposé tente de trouver la combinaison de valeurs optimales pour ces deux paramètres en utilisant les réseaux de neurones. L'utilisation des réseaux de neurones intervient lors de l'adaptation des cas connus : ils modifient le nombre et les angles des faisceaux. La validation de l'algorithme est évaluée avec une base de 80 cas réels de cancer du cerveau extraits de l'hôpital de Nottingham City. Le nombre de neurones et de couches ont été définis de façon empirique. Les résultats montrent que l'utilisation des cas historiques et la construction des solutions à partir des solutions déjà connues permet une amélioration de 12% concernant la décision du nombre de faisceaux et de 29% concernant la décision liée à leur angle.

4.2/ LE RÀPC POUR CONSTRUIRE OU CORRIGER UN TEXTE, POUR CONSOLIDER OU EXPLIQUER LES RÉSULTATS OBTENUS

La génération, analyse et correction de texte constitue également un domaine d'application intéressant du RàPC. Pour la réalisation de ces tâches il est parfois nécessaire de transformer le texte en représentation numérique ou de mesurer la proximité sémantique de mots. Le travail de [Ontañón et al., 2015] utilise le RàPC pour générer des histoires en utilisant le texte d'autres histoires. Le système décrit explore les transformations possibles afin que la nouvelle histoire ne soit pas très similaire aux histoires déjà connues mais que elle soit cohérente. Le travail est plus focalisé sur la phase de révision, car les résultats de celle-ci déterminent si l'histoire générée correspond aux critères spécifiés ou si le cycle d'adaptation doit recommencer ; l'adaptation se fait en recherchant les personnages, les contextes, les objets dans la base de cas, et en les unifiant avec des actions ; la plupart des histoires générées sont cohérentes mais elles sont constituées de paragraphes très courts.

Dans [Lepage et al., 2020] les phrases écrites en langue française sont corrigées. Ce travail n'utilise ni la transformation numérique des phrases, ni de connaissances linguistiques, mais retrouve les phrases similaires en utilisant l'algorithme LCS (*Longest Com-*

mon Subsequence) et en calculant une mesure de distance avec les phrases correctes et incorrectes de la base de connaissances. Si les phrases similaires sont incorrectes, le système peut proposer une correction en changeant certains mots selon le contexte et recalculer les distances afin de mesurer la cohérence et pertinence de la phrase proposée.

Plus récemment Wolf *et al.* [Wolf et al., 2024] ont présenté un système dans lequel le RàPC est employé pour expliquer les résultats générés par un réseau qui classifie les images en fonction de certains attributs et zones dans les images. Les résultats obtenus permettent de conclure que le RàPC permet d'expliquer fidèlement le contenu des images testées.

[Parejas-Llanovarced et al., 2024] utilisent également le RàPC pour sélectionner la meilleure explication au résultat donné par le classificateur d'images fondé sur l'apprentissage profond (*Deep Learning*). Dans ce système, la base de connaissances est constituée d'images avec des étiquettes décrivant l'information associée. Les résultats obtenus par ce système sont très prometteurs.

4.3/ TRAVAUX RÉCENTS SUR LA REPRÉSENTATION DES CAS ET LE CYCLE DU RÀPC

Certains travaux ont appliqué le raisonnement à partir de cas à un problème spécifique en modifiant les représentations des cas et des solutions. D'autres ont modifié le cycle conceptuel comme le montre la figure 4.1 avec l'objectif d'obtenir des réponses dynamiques et plus aptes à proposer des solutions à des problèmes complexes.

En effet, la représentation des cas peut permettre d'améliorer les résultats d'un système de RàPC. La performance d'un système de RàPC dépend de la quantité d'informations stockées, mais également des algorithmes implémentés. C'est le cas dans [Müller and Bergmann, 2015] où les recettes de cuisine sont codées comme un processus de transformation et mélangent des ingrédients en suivant une suite d'étapes ordonnés. Pour créer des recettes innovantes, une mesure de distance est utilisée entre les ingrédients. Cette mesure permet de trouver une recette en substituant certains ingrédients par d'autres, similaires ou aux qualités et/ou caractéristiques similaires. De la même manière, il est possible de créer des recettes plus proches des exigences des utilisateurs. Les étapes de transformation appelées aussi opérateurs sont stockées et catégorisées grâce à une métrique permettant de les échanger afin d'obtenir une nouvelle recette.

Les auteurs de [Grace et al., 2016] introduisent un cycle complémentaire incluant un outil d'apprentissage profond pour améliorer le résultat du système fondé sur le RàPC.

Dans [Butdee and Tichkiewitch, 2011], la phase de stockage est modifiée en retenant les cas dont les résultats n'ont pas eu de succès, pour guider le processus dans la fabrications de nouvelles pièces.

Enfin, dans [Robertson and Watson, 2014], les auteurs proposent d'ajouter à chaque cas une valeur d'utilité espérée en fonction de chaque action possible. Cet ajout s'accompagne d'une prédiction probabiliste des actions que l'application engendrera en réponse. Cette prédiction probabiliste dépend bien entendu de l'état initial du système avant mise en oeuvre de l'action.

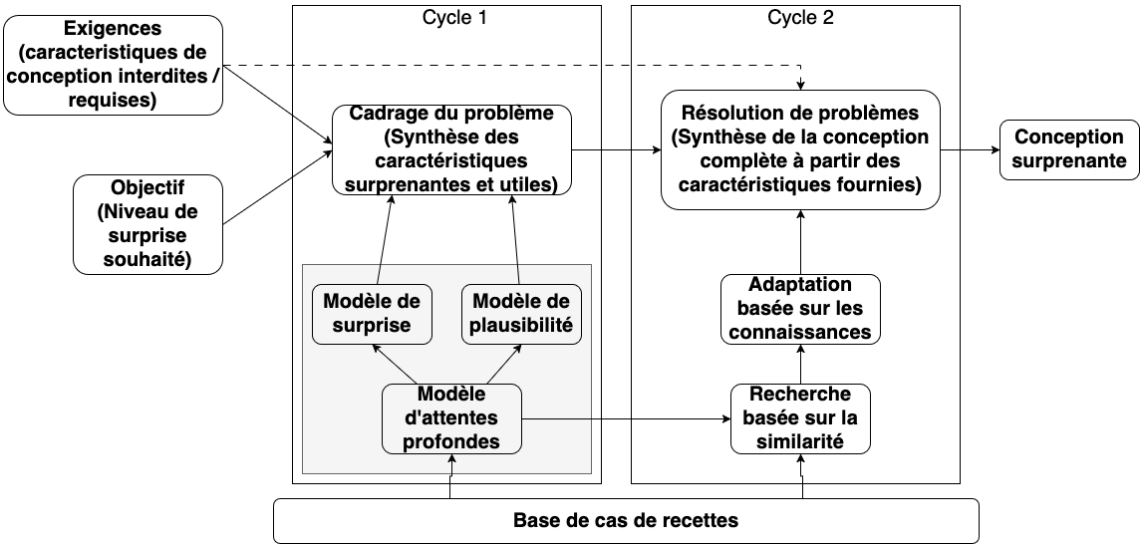

FIGURE 4.1 – Ajout d'un cycle complémentaire avec *Deep Learning* au RàPC (Traduit de [Grace et al., 2016])

Plusieurs travaux appliquent le RàPC avec succès en proposant des modifications dans chacune des phases ou en combinant différents algorithmes. Certains systèmes de RàPC appliqués au domaine de la conception de produits sont remarquables à ce titre.

Dans [Roldan Reyes et al., 2015], les auteurs proposent un algorithme pour produire le propylène glycol dans un réacteur chimique comme le montre la figure 4.2. Dans ce cas, la phase de réutilisation du RàPC est couplée à la recherche des états satisfaisant le nouveau problème (*Constraint satisfaction problems CSP*) en utilisant l'information des cas de base déjà résolus. Les solutions trouvées sont évaluées selon le nombre de changements réalisés sur les solutions déjà connues (parcimonie), le nombre de solutions possibles trouvées (précision), l'évaluation de commentaires faits par des experts et la complexité des transformations réalisées.

Dans [Grace et al., 2016], les auteurs ajoutent un nouveau cycle au cycle traditionnel du RàPC. Le premier cycle génère des descriptions abstraites du problème avec un réseau de neurones et des algorithmes génétiques ; le second cycle prend les descriptions abstraites comme des nouveaux cas, cherche les cas similaires et adapte les solutions rencontrées. L'exécution des deux cycles prend en compte certains critères prédéfinis par l'utilisateur. En comparant le même problème avec le cycle traditionnel du RàPC, les auteurs mesurent une amélioration de la qualité des recettes proposées et montrent que celles-ci sont plus en accord avec les critères définis.

[Maher and Grace, 2017] s'intéressent quant à eux à la génération de recettes de cuisine originales. Les auteurs modifient le cycle traditionnel du RàPC et créent deux cycles, combinant l'apprentissage profond et la récupération de cas fondée sur la similarité, le premier cycle génère des descriptions abstraites de la conception avec des algorithmes génétiques et un réseau neuronal profond, le second cycle utilise les descriptions abstraites pour récupérer et adapter des objets, cette structure donne lieu à un prototype appelé Q-chef qui génère une recette fondée sur d'autres ingrédients connus du système et les demandes de l'utilisateur. Ce travail ne montre pas de métriques standard génériques mais utilise deux nombres indicatifs (plausibilité et surprise) pour démontrer la

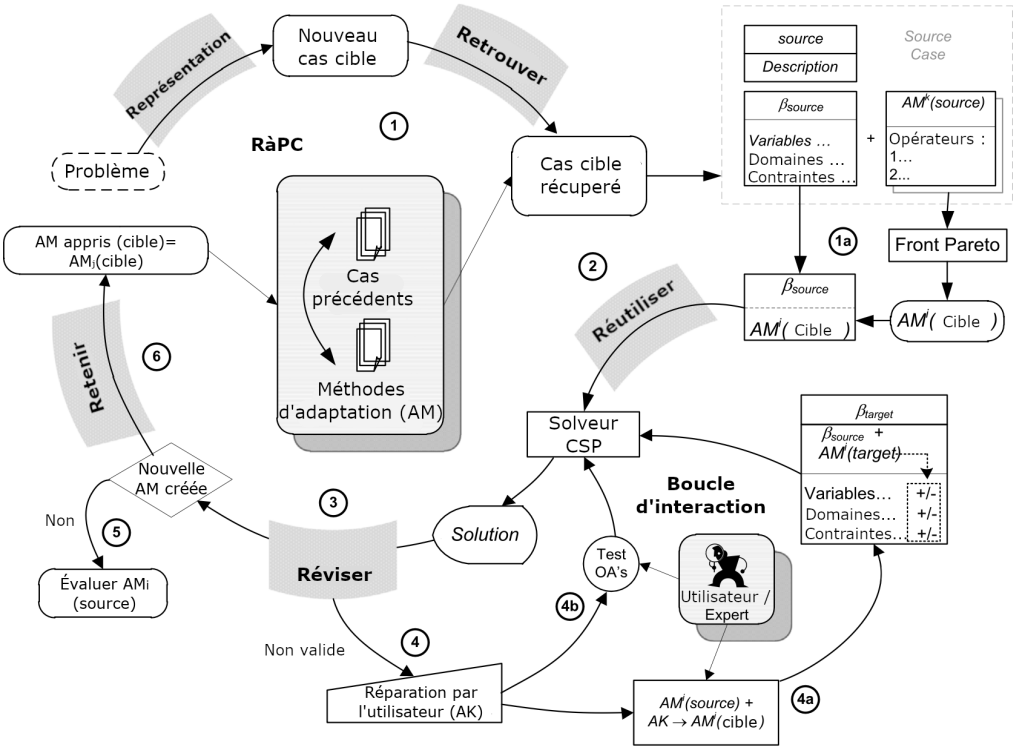

FIGURE 4.2 – Cycle du RàPC modifié. (Traduit de [Roldan Reyes et al., 2015])

génération efficace de nouvelles recettes selon les critères de l'utilisateur en comparant le RàPC à deux cycles avec le RàPC à un cycle, démontrant plus de plausibilité et de surprise dans les recettes générées.

4.4/ INTÉGRATION D'AUTRES ALGORITHMES DANS LE CYCLE DU RÀPC

Par rapport à l'intégration du RàPC avec d'autres algorithmes nous pouvons citer [Uysal and Sonmez, 2023] qui mettent en œuvre un RáPC avec une méthode d'agrégation *bootstrap* (*bagging*) pour améliorer la précision du RàPC lorsqu'il n'y a pas suffisamment de cas et réduire la variance. L'une des problématiques auxquelles les auteurs tentent de répondre dans cet article est l'estimation des coûts conceptuels dans les décisions de faisabilité d'un projet lorsque l'on ne dispose pas de suffisamment d'informations sur la conception détaillée et les exigences du projet. Les résultats ont révélé que le RàPC associé à *bootstrap* est plus performant que le RàPC non associé à *bootstrap*.

Un modèle d'ensemble fondé sur le raisonnement à partir de cas est proposé par [Yu and Li, 2023]. Celui-ci est appliqué à la prédiction financière et au remplissage des données manquantes. Dans ce système, pour retrouver les plus proches voisins, le modèle utilise trois mesures de distance différentes et une étape de vote pour l'intégration. Le modèle a été testé avec une base de données comportant onze caractéristiques (informations) financières provenant de 249 entreprises. La comparaison est faite avec deux objectifs. Premièrement, le remplissage des données manquantes avec d'autres

algorithmes tels que KNN ou *Random Forest*, et deuxièmement, la comparaison de la prédiction avec des algorithmes uniques utilisant une métrique de distance spécifique. Les résultats obtenus montrent que le système est plus performant aussi bien pour le remplissage des données manquantes que pour la prédiction des données financières.

Une étape très importante dans le RàPC est l'adaptation. Dans [Malburg et al., 2024] l'objectif est changer la représentation de la connaissance pour évaluer s'il y a un impact positif en les solutions générées, ce changement permet aussi d'établir règles de adaptation, les résultats permettent de dire que choisir une représentation adéquate et des règles correctes en fonction du problème étudié peut améliorer la qualité des solutions pour résoudre des problèmes complexes.

4.5/ PRÉDICTION ET INTERPOLATION EN UTILISANT LE RÀPC

L'objectif du travail de [Smyth and Cunningham, 2018] est la prédiction du temps de course pour un athlète. L'algorithme KNN est utilisé pour rechercher les cas similaires. Le système interpole un temps final grâce au calcul d'une moyenne pondérée des meilleurs temps des cas similaires retrouvés dans la première étape du RàPC.

Dans [Smyth and Willemsen, 2020], les auteurs essaient de prédire le meilleur temps d'un patineur par analogie avec ceux des patineurs ayant des caractéristiques et une histoire de course similaires. Cependant parfois, calculer une moyenne des temps similaires trouvés ne suffit pas. Certaines caractéristiques liées au contexte, à l'environnement et à la nature de la course (le type de course, le type de piste, la distance à parcourir, etc.), peuvent en effet influencer de manière importante la performance du patineur. L'algorithme a été testé avec une base de données contenant les informations de 21 courses de 500m, 700m, 1000m, 1500m, 3km, 5km and 10km réalisées entre Septembre 2015 et Janvier 2020.

Un système multi-fonctionnel est décrit dans [Feely et al., 2020]. Celui-ci permet d'obtenir une prédiction du temps de course, de suggérer un plan du rythme de la course et il recommande également un plan d'entraînement pour une course donnée. Les trois fonctionnalités sont implémentées en utilisant le RàPC. Les calculs de la similarité sont fondés sur un historique et des caractéristiques physiques des coureurs. Les plans d'entraînement sont génériques et sont proposés sur les 16 semaines précédant le début du marathon ciblé (selon les auteurs, c'est en effet le temps usuel pour une préparation à ce type d'épreuve). Le système a été évalué avec une base de données constituée de caractéristiques de 21000 coureurs des marathons de Dublin, Londres ou New-York pour la période de 2014 à 2017.

Les auteurs de [Yu et al., 2024] proposent un algorithme en deux étapes pour prédire avec précision et robustesse la détresse financière. Les deux étapes utilisent le RàPC. La première étape consiste à compléter les valeurs manquantes de la base de données considérée. Un cycle de RàPC est ensuite combiné avec un LVQ (*Learning Vector Quantization*) pour interpoler une classification. Les résultats obtenus sont prometteurs.

Les auteurs de [Sadeghi Moghadam et al., 2024] présentent un nouvel algorithme pour la prévision de la demande de matériel de secours. Le système décrit dans cet article combine le RàPC avec la méthode du meilleur-pire (*Best-Worst Method*, BWM) et les modèles de Markov cachés (*Hidden Markov Model*, HMM). Le HMM est entrainé avec des

données historiques de demande de matériel. Lorsque l'algorithme détecte un accident, le RàPC recherche les cas similaires pour proposer le matériel nécessaire. D'après les résultats, l'indice d'erreur de prévision est inférieur à 10%, démontrant ainsi la robustesse du système CBR-BWM-HMM proposé.

4.6/ RECOMMANDATION, EIAH ET RÀPC

Les systèmes de recommandation et le RàPC peuvent aussi être combinés comme dans le système proposé par [Supic, 2018]. Cet article montre les bénéfices de l'utilisation du RàPC dans les environnements informatiques pour l'apprentissage humain (EIAH). Le modèle proposé suit le cycle traditionnel du RàPC en combinant les modèles d'apprentissages traditionnels et numériques. Les principales contributions sont la représentation des cas et la recommandation des parcours d'apprentissage personnalisés selon les informations issues des autres apprenants. Une base de cas initiaux a été créée pour mesurer l'efficacité du modèle. Celle-ci stocke la recommandation du parcours de 120 apprenants. Des examens sont réalisés avant et après avoir suivi le parcours recommandé par le système permettent de mesurer l'efficacité de la recommandation proposée.

Le système décrit dans [Obeid et al., 2022] présente la particularité d'être capable d'analyser des données hétérogènes et multidimensionnelles. Dans ce travail, un parcours de carrière et les universités/collèges est recommandé aux élèves du secondaire en fonction de leurs intérêts. Ce travail montre également une taxonomie des techniques algorithmiques généralement utilisées dans les systèmes de recommandation pour les EIAH (figure 4.3).

En fonction de certains paramètres comme le type d'école, le nombre d'étudiants admis et non admis, le nombre de classes, les auteurs de [Skittou et al., 2024] développent un modèle hybride combinant RàPC et raisonnement à partir de règles pour recommander des actions de planification en réponse à des cas éducatifs des écoles primaires. Le modèle a été testé sur des données réelles et a donné de bons résultats avec une grande précision.

4.7/ RÉCAPITULATIF DES LIMITES DES TRAVAUX PRÉSENTÉS DANS CE CHAPITRE

Le tableau 4.1 récapitule les articles analysés dans cet état de l'art du RàPC et rappelle les limitations que nous avons identifiées. Ces limitations sont liées aux données, à la flexibilité, à la généralisation de l'algorithme proposé, à la validité des solutions proposées, l'automatisation du processus et complexité du modèle proposée.

En complément, deux problèmes très communs des systèmes de recommandation peuvent être ajoutées à ces limitations :

- le *cold-start* qui se produit au début d'une recommandation lorsqu'il n'y a pas suffisamment de données pour calculer ou inférer une recommandation appropriée [Hu et al., 2025], et
- le *gray-sheep* qui se produit lorsqu'un utilisateur présente un comportement très différent de ceux qui sont stockés dans la base de données. Le système ne peut

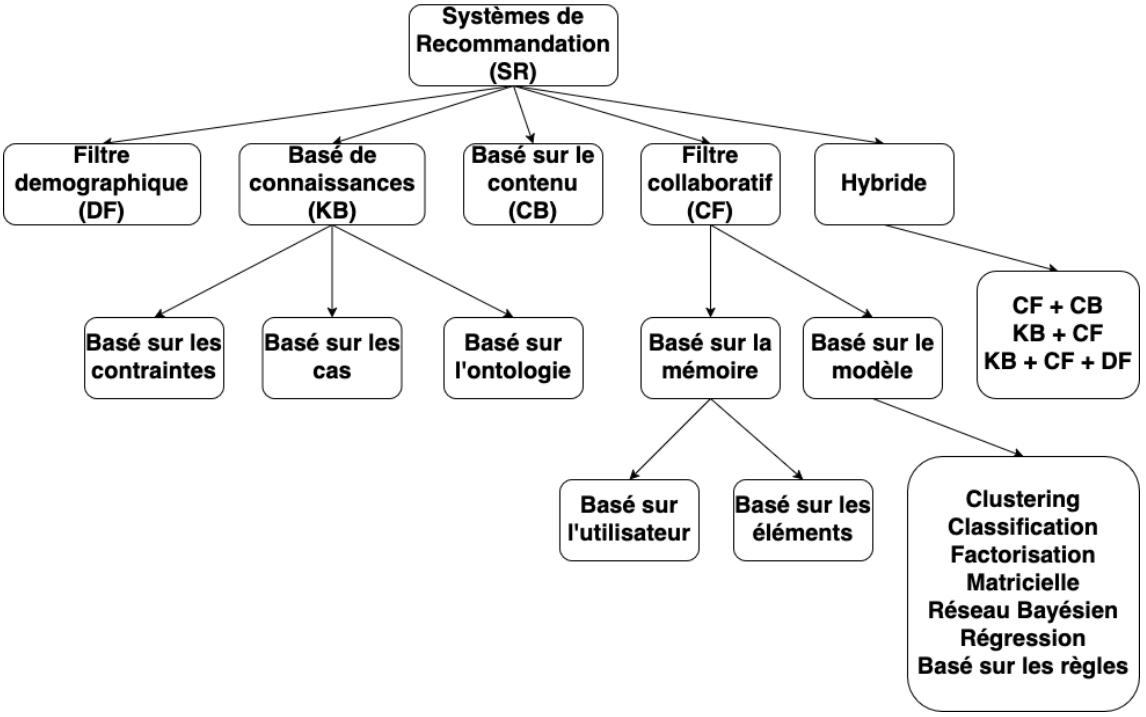

FIGURE 4.3 – Taxonomie des techniques algorithmiques employées pour des modules de recommandation dans les EIAH (Traduit de [Obeid et al., 2022])

donc pas générer des recommandations en se basant sur l'information disponible [Alabdulrahman and Viktor, 2021].

Ref	Limites
[Jung et al., 2009]	Le modèle d'adaptation proposé fonctionne seulement avec des cas très proches. L'apprentissage dans le RàCP se limite à stocker les nouveaux cas.
[Butdee and Tichkiewitch, 2011]	Le RàPC n'est pas modifié ou amélioré. La révision dans le RàPC n'est pas automatique
[Petrovic et al., 2016]	Grande quantité de données pour que l'algorithme fonctionne correctement. Recommandation limitée à un problème très spécifique.
[Wolf et al., 2024]	Le modèle n'est actuellement appliqué qu'à la classification d'images à un seul label
[Parejas-Llanovarced et al., 2024]	La base de données de test a été construite avec des évaluations subjectives. Le modèle ne considère pas les incertitudes.
[Roldan Reyes et al., 2015]	Une seule méthode d'adaptation est utilisée. Le modèle n'exploite pas les cas qui présentent une erreur.
[Grace et al., 2016]	Beaucoup de données sont nécessaires pour entrainer le modèle de 'Deep Learning'. Les solutions générées n'ont pas été validées.
[Maher and Grace, 2017]	L'évaluation des solutions générées n'est pas automatique. Les solutions sont produites avec un seul point de vue.
[Uysal and Sonmez, 2023]	La fonction d'unification des solutions est une moyenne des solutions générées avec la division des données et un seul approche
[Yu and Li, 2023]	Une base de données déséquilibrée peut affecter négativement la performance du modèle proposé
[Müller and Bergmann, 2015]	Une seule approche pour adapter les cas dans le RàPC. La révision dans le RàPC n'est pas automatique.
[Ontañón et al., 2015]	Les solutions générées peuvent présenter des incohérences. Il n'y a pas une métrique objective pour mesurer la qualité des réponses.
[Lepage et al., 2020]	Les résultats ne sont pas très bons. Le modèle n'a pas de connaissances linguistiques.
[Malburg et al., 2024]	Les règles d'adaptation définies sont fixes et peuvent être limitées pour certaines situations
[Smyth and Cunningham, 2018]	Les prédictions n'ont pas été testées dans le monde réel. Les données sont très spécifiques et peu variées.
[Smyth and Willemsen, 2020]	Les prédictions sont pour des cas limités. Le modèle est entrainé pour un type d'utilisateur spécifique.
[Feely et al., 2020]	Seulement une technique de sélection de solutions. Les données nécessaires pour le modèle sont complexes.
[Yu et al., 2024]	Il n'y a pas de révision des cas dans la méthode d'imputation ni dans le modèle de classification fondé sur RàPC. Cela peut accroître le biais d'imputation et introduire des échantillons avec bruit
[Sadeghi Moghadam et al., 2024]	Le temps de calcul peut être très grand en fonction de la quantité de données associées, du a l'étape d'entrainement du modèle HMM
[Supic, 2018]	Le modèle a été validé avec peu de données. Les recommandations se basent seulement sur l'information des autres apprenants.
[Obeid et al., 2022]	L'ontologie peut être limitée pour évaluer plusieurs cas inconnus ou imprévus. Il est nécessaire d'avoir une forte connaissance du domaine.
[Skittou et al., 2024]	Les règles de décision établies sont fixes et avec l'hybridation peut avoir de l'incertitude et l'imprécision

TABLE 4.1 – Tableau de synthèse des articles analysés dans l'état de l'art du RàPC

III

CONTRIBUTIONS

5

ARCHITECTURE GLOBALE DU SYSTÈME AI-VT

5.1/ INTRODUCTION

Il est important de définir et décrire l'architecture logicielle d'un système complexe afin de mieux comprendre son fonctionnement. L'architecture montre les composants, les couches, les fonctionnalités et les interactions. Ce chapitre présente l'architecture modulaire proposée pour le système AI-VT. Les modules intègrent des outils issus de différents paradigmes, des modèles et des algorithmes d'intelligence artificielle permettant au système d'être plus performant. Les objectifs de l'architecture proposée sont de rendre le logiciel AI-VT (Artificial Intelligence Virtual Trainer) stable, évolutif, performant et facile à entretenir. L'architecture proposée se compose de quatre couches indépendantes, chacune dédiée à une fonctionnalité spécifique : correction automatique, identification, révision et test. Le contenu de ce chapitre a été publié dans *et al.* [Soto-Forero et al., 2024b].

Il est possible de classer les architectures selon deux catégories : les architectures monolithiques et les architectures modulaires. Dans une architecture monolithique, le système logiciel est considéré comme une entité unique et unitaire avec une seule source de code, une seule base de données et un seul déploiement pour l'ensemble du système ; ce type de système est simple à développer et à tester mais il est peu adapté à la mise à jour et à l'évolution en raison de sa rigidité. Une architecture modulaire divise le système en modules indépendants qui peuvent communiquer entre eux, chaque module contenant alors tout ce qui est nécessaire pour fonctionner. De nombreux systèmes logiciels ont été conçus avec une architecture modulaire en raison des multiples avantages qu'elle présente [Auer et al., 2021] [Zuluaga et al., 2022].

L'EIAH AI-VT a ainsi évolué vers une architecture modulaire que nous présentons dans ce chapitre. Après avoir rappelé le fonctionnement et les différents composants initiaux du système AI-VT, ce chapitre présente l'architecture modulaire implémentée lors de cette thèse et permettant à AI-VT d'intégrer de nouvelles fonctionnalités. La section 5.2 décrit le système AI-VT qui a été développé et qui existait avant le début de cette thèse.

5.2/ DESCRIPTION DU SYSTÈME AI-VT

Pour rappel, le système AI-VT est un outil pédagogique générique qui vise à accompagner les apprenants dans leur apprentissage en leur proposant des fiches d'exercices appelées sessions. Durant chaque session, les capacités attendues sont divisées en compétences, elles-mêmes divisées en sous-compétences. L'apprenant choisit une compétence à travailler et le système génère une session composée d'exercices associés à plusieurs sous-compétences de la compétence choisie. Le système propose une liste d'exercices au début d'une session en utilisant le paradigme du raisonnement à partir de cas avec une base de données de questions.

Le système AI-VT est un EIAH générique dont la structure globale est présentée sur la figure 5.1. Il intègre une base de données de questions. Chacune des questions est associée à un contexte, au texte de la question considérée et à un niveau de complexité. Les questions sont liées à des sous-compétences, elles-mêmes liées à des compétences. Les principaux acteurs du système sont l'enseignant et l'apprenant. L'enseignant a la capacité de configurer l'ensemble du système, le nombre de compétences, les sous-compétences d'une compétence, le nombre de questions, la complexité de chacune d'entre elles, le niveau de complexité et le temps par session. L'apprenant quant à lui, peut commencer l'entraînement d'une compétence spécifique, accéder à des ressources de soutien complémentaires et répondre aux questions de test dans les sessions proposées par le système. Le tableau 5.1 montre les caractéristiques du système AI-VT selon les 20 descripteurs permettant de caractériser le profil de l'apprenant décrits dans [Jean-Daubias, 2011].

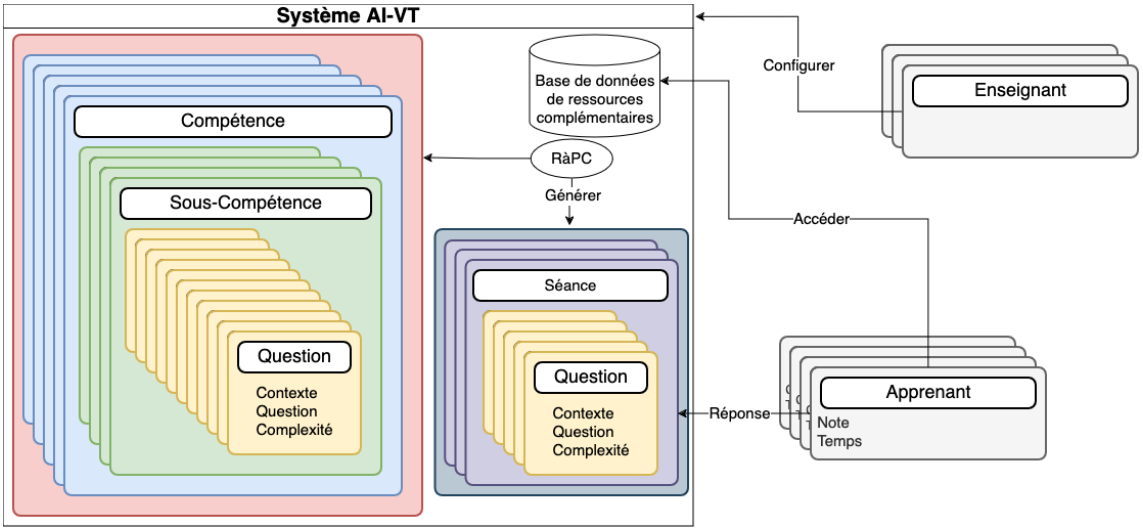

FIGURE 5.1 – Structure du système AI-VT

En s'appuyant sur la philosophie du RàPC, le système global AI-TV part du principe que la création d'une session pour un apprenant peut être réalisé par analogie avec celles proposées à des étudiants ayant des acquis, besoins et capacités similaires. AI-VT propose une liste d'exercices en fonction de la compétence ou de la sous-compétence sélectionnée par similitude avec celles proposées auparavant à d'autres apprenants.

L'algorithme d'AI-VT tente de tenir compte de l'équilibre entre répétitivité et variété. De

plus, le nombre d'exercices par session change en fonction du domaine, de la compétence choisie et du niveau de l'apprenant. La liste d'exercices est générée au début de chaque séance et elle n'est pas modifiée au cours de la séance en fonction des réponses fournies par l'apprenant : les listes d'exercices sont statiques pendant la séance [Henriet and Greffier, 2018].

Type	Definition	Système AI-VT
Subject	Acteur humain concerné par le profil	Apprentissage seul et en groupe
Collaboration	Le rôle de la collaboration dans les activités du profil	Individuel, Collaboratif
Distance	Le rôle de la distance dans les activités du profil	Presentiel, Distanciel
Discipline	Discipline des informations contenues dans le profil	Generic
Niveau	Le niveau scolaire de la matière concernée par le profil	Generic
Initiateur	L'acteur humain à l'origine de la décision de créer le profil de création	Professeur, Administrateur
Créateur	L'acteur humain ou logiciel qui compose le profil	Professeur, Administrateur
Destinataire	Acteur humain ou logiciel exploitant le profil	Apprenant
Temps	Période du profil	Asynchrone
Évolution	L'évolutivité du profil	Profil évolutif
Type	Le type d'informations contenues dans le profil	Profil de l'apprenant
Nature	La nature des informations contenues dans le profil	Connaissances et compétences
Évaluation	La forme sous laquelle l'information est évaluée	Rating, Taux de maîtrise
Représentation interne	Représentation interne utilisée par le système informatique pour manipuler les profils	Tables
Représentation externe	Représentation utilisée pour stocker le profil	Liste de valeurs
Visualisation	Représentation utilisée pour présenter le profil à ses destinataires	Représentation textuel et graphique standard
Norme	Norme ou standard éducatif	-
Format	Format de stockage du profil	Base de données relationnelle
Plate-forme	plate-forme informatique compatible	Web
Dispositifs	le type de dispositif d'affichage du profil	Ordinateur ou appareils connectés

TABLE 5.1 – Un tableau décrivant les caractéristiques du système AI-VT

5.3/ MODÈLE D'ARCHITECTURE PROPOSÉ

Cette section explicite l'architecture proposée à partir du système AI-VT déjà existante.

La nouvelle architecture proposée a pour objectif d'intégrer de nouvelles fonctionnalités complémentaires au système initial sans en modifier les fonctionnalités. L'idée consiste donc à pouvoir activer simplement le module correspondant en envoyant et en recevant les informations nécessaires à son fonctionnement. Une évolution vers une architecture plus modulaire permettra ainsi d'ajouter de nouvelles fonctionnalités au système d'origine en intégrant de nouveaux modules ou en faisant évoluer les modules existants. La conception modulaire facilite également la maintenance du code, le développement et l'intégration de nouvelles extensions. De plus, le système pourra être configuré et adapté module par module, réduisant ainsi les risques de pannes et les coûts de maintenance et d'évolution. La modularité permet également d'exécuter les algorithmes de chaque module de manière asynchrone, en parallèle ou en mode distribué si nécessaire.

L'architecture se compose de deux éléments principaux représentés sur la figure 5.2 : le système central AI-VT, tel qui décrit dans la section 5.2 et les modules fonctionnels proposées pour compléter et améliorer certaines fonctionnalités. Le système central gère l'ensemble du processus d'apprentissage ; il génère les séances ; il stocke les données relatives aux compétences, aux questions, aux ressources, aux apprenants, aux enseignants et aux réponses ; il contient les commandes et l'interface générale ; il gère le flux d'informations et active les modules nécessaires. Les modules sont un ensemble de fonctionnalités indépendantes mises en œuvre avec des algorithmes d'intelligence artificielle qui reçoivent et envoient des données depuis le composant central. Chaque module fonctionne selon des critères spécifiques liés à son propre objectif. Les modules sont regroupés en couches selon leur fonctionnalité : correction automatique, identification, adaptation, révision et test. L'enseignant et l'apprenant n'utilisent pas les modules directement ; ceux-ci sont appelés par le système pour compléter certaines fonctionnalités.

La couche de correction automatique (LC) contient les modules chargés de recevoir les réponses des apprenants et, conformément aux algorithmes et critères définis, d'établir une note cohérente avec une réponse de référence à une question spécifique. Dans cette couche, le module routeur (LC0) est chargé d'identifier le type de correction nécessaire et d'instancier le module approprié pour l'exécution de la tâche spécifique.

La couche d'identification (LD) contient les modules qui identifient les faiblesses ou les variables externes des apprenants lors de l'exécution des exercices proposés par le système ou après l'analyse des résultats. Ces modules aident à personnaliser le processus d'apprentissage en fonction de l'analyse des résultats obtenus par les apprenants.

La couche de révision (LR) comprend les modules qui prennent les données des résultats obtenus dans la couche LC et les résultats de l'analyse de la couche LD pour modifier le parcours de l'apprenant en essayant de renforcer l'apprentissage dans les faiblesses détectées. Elle comprend également les modules qui obtiennent des informations de la part des apprenants et tentent de prédire leurs résultats en fonction des différentes compétences et des différents niveaux de complexité.

Pour évaluer les modules dans différents scénarios, il est nécessaire de produire des données selon différents critères et complexités. C'est la raison pour laquelle la couche de test (LT) a été définie. Les modules qui permettent de générer des données synthétiques selon des critères variables font partie de cette couche. Les modules de cette couche

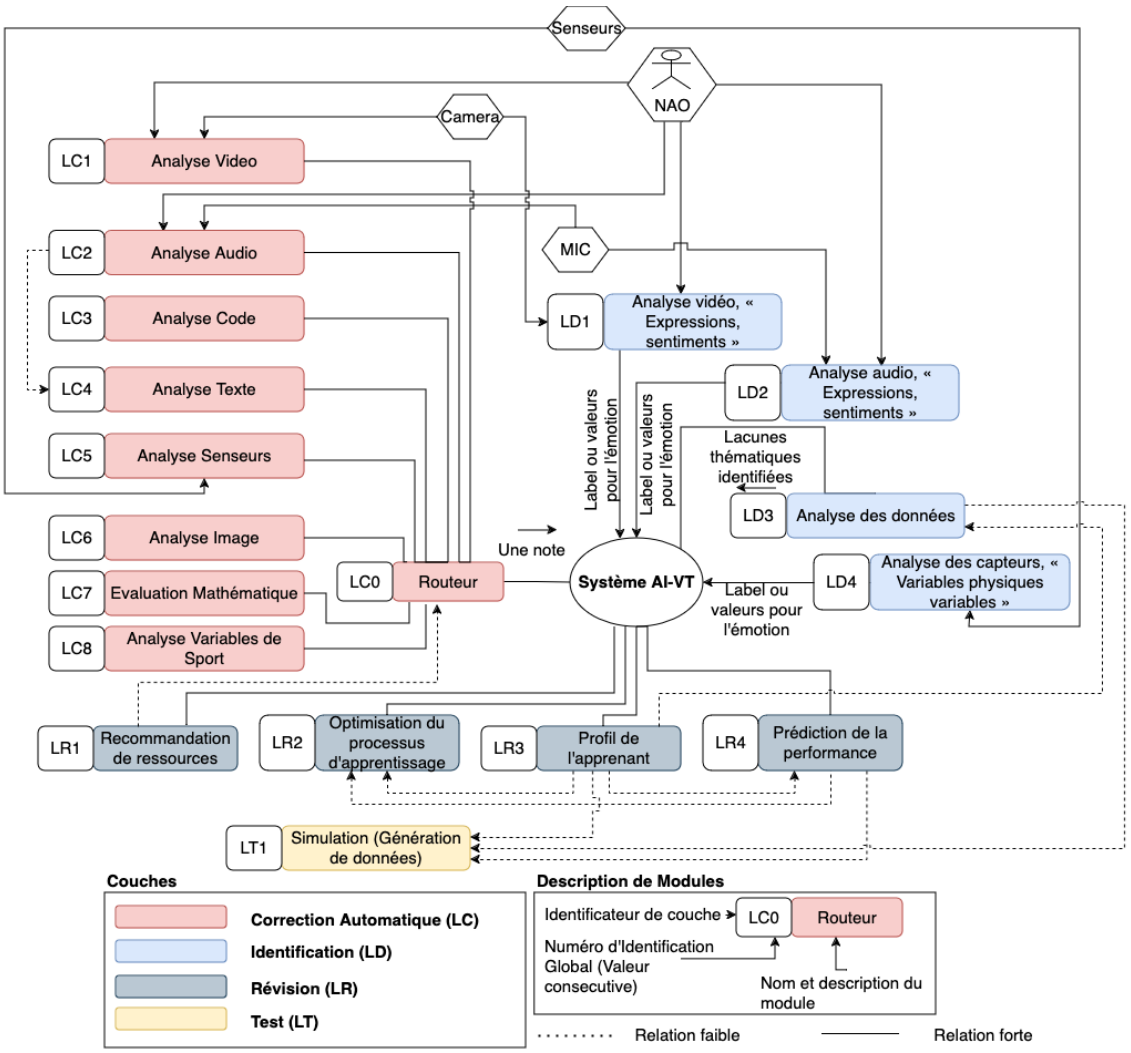

FIGURE 5.2 – Schème de l'architecture proposée

permettent d'obtenir des résultats numériques des modules, d'appliquer des métriques, et ainsi d'évaluer les modules selon différents critères et complexités.

Le schéma complet de l'architecture est représenté sur la figure 5.2. Sur cette figure, les lignes pleines représentent un flux d'informations bidirectionnel, les lignes pleines terminées par une flèche représentent le flux unidirectionnel et les lignes en pointillé représentent la dépendance de l'information entre les modules. Les dispositifs externes qui peuvent être utilisés par les modules pour exécuter leurs fonctionnalités et les étiquettes indiquant le type d'information transmis au système central par le module sont également représentés. Certains des algorithmes d'intelligence artificielle mis en œuvre dans chaque module et le stade de développement dans lequel chacun d'eux se trouve y figurent également. Comme le montre cette figure, certains dispositifs ont besoin de données provenant de sources externes telles que le robot NAO, des capteurs, caméra vidéo ou microphone.

5.3.1/ CORRECTION AUTOMATIQUE

L'une des couches importantes et la couche de correction automatique. Dans cette couche, les modules ont la capacité de recevoir et d'évaluer différents types de réponses données par les apprenants en fonction du contexte et de la question que le système a proposée. Les modules représentés ont la capacité d'évaluer une réponse donnée par l'apprenant. Des études et certains programmes ont déjà été réalisés dans l'analyse de vidéo, de texte audio (langage naturel) et de code source (Java, Python). D'autres modules permettent également d'analyser des images, des expressions mathématiques, des valeurs générées par des capteurs physiques et des variables définies pour des activités sportives spécifiques.

Le module routeur (LC0) a pour fonction d'identifier le type de réponse donnée par l'apprenant, c'est à dire reconnaître si la réponse donnée par l'apprenant est vidéo, audio, texte, image, etc et de la rediriger vers le module d'analyse correspondant ; une fois le module spécifique obtient les résultats de l'analyse de la réponse donnée, le router les redirige vers le système AI-VT principal.

Le module vidéo (LC1) permet de capturer un flux d'images à partir d'un dispositif externe (caméra vidéo ou robot NAO) et de les analyser pour déterminer si une réponse donnée est correcte, actuellement le module est utilisé pour évaluer la réponse à la question : montrer *n* nombre de doigts. L'algorithme implémenté détecte les doigts qui apparaissent sur la caméra, les compte et détermine s'il s'agit de la bonne réponse à la question donnée.

Le module audio (LC2) analyse une piste audio et convertit son contenu en texte. Une fois le texte généré, il peut être comparé à une réponse attendue à la question posée à l'aide de techniques NLP. Un score d'approximation estimé peut alors être calculé. Pour la comparaison est utilisé le module de texte (LC4) qui permet d'établir le score de similarité entre le texte envoyé comme réponse de l'apprenant ou converti depuis une réponse audio et un texte de référence.

Le module d'analyse du code (LC3) génère un score après l'exécution de plusieurs étapes. En premier lieu, il détermine l'existence ou non de faiblesses dans certaines compétences prédéfinies. Il transforme ensuite la représentation du code en un vecteur numérique à comparer avec une réponse de référence : le résultat de la pondération entre les faiblesses détectées et le pourcentage de comparaison donne un score.

Le module d'analyse de texte (LC5) transforme le texte envoyé par l'apprenant en un vecteur numérique qui peut être comparé au vecteur numérique d'une réponse attendue. Dans ce cas, la réponse donnée ne doit pas nécessairement être exactement la même que la réponse attendue. La représentation vectorielle permet d'établir des similitudes dans l'espace, même si les termes utilisés et la longueur du texte diffèrent.

Les modules LC6, LC7, LC8 et LC9 sont capables d'analyser différents types de réponses potentielles qui peuvent éventuellement gérées par le système AI-VT.

Les modules LC1, LC2, LC3 ont été partiellement développés et n'ont pas été suffisamment testés pour les intégrer dans le système de façon fonctionnelle, les autres modules pour le moment sont théoriques.

5.3.2/ IDENTIFICATION

Les modules de la couche d'identification visent à extraire des informations complémentaires à la note de l'étudiant pour chacune des réponses envoyées au système et ainsi affiner encore plus les recommandations générées. Ces informations complémentaires permettent d'obtenir principalement des informations sur les émotions (bonheur, dégoût, surprise, colère, peur, neutre et triste) grâce à la détection des expressions faciales et aussi les lacunes de connaissance de l'apprenant qui peuvent se manifester dans chaque sous-compétence et niveau de complexité. Ces modules d'AI-VT sont capables d'obtenir une meilleure estimation de l'état de l'apprentissage afin de mieux adapter le parcours de l'apprenant.

Les modules d'identification LD1, LD2 et LD4 tentent de détecter les comportements, les émotions et les sentiments à l'aide de dispositifs externes tels que la caméra vidéo et le microphone. Dans le cas de l'analyse vidéo, des réseaux neuronaux d'apprentissage profond sont utilisés pour capturer et analyser les images statiques obtenues à partir du flux de la caméra vidéo. Le modèle d'IA a été entraîné à détecter des émotions prédéfinies. Le module audio vise à détecter le même type d'émotions, mais à partir de l'analyse des signaux obtenus à partir d'un microphone. Il utilise également l'apprentissage profond entrainé avec des signaux qui présentent différentes émotions prédéfinies. Le module capteur est plus générique, mais il peut être subdivisé en modules spécifiques en fonction du type de capteur et du signal à analyser. Toutefois, l'idée de la détection est la même : détecter des émotions prédéfinies.

Le module Analyse des données (LD3) est différent car, en plus d'être générique, il tente d'identifier les faiblesses dans des compétences spécifiques en fonction du type d'évaluation. Ce module peut contenir différents modèles entraînés pour chaque type de cas. Pour les exercices linguistiques, les faiblesses identifiables peuvent être la conjugaison des verbes, l'utilisation des temps, le vocabulaire, la correspondance des genres de mots, etc. Si l'exercice est de type programmation le module est capable d'identifier des faiblesses liées à la syntaxe, la déclaration de variables, l'appel de fonctions, la construction de structures, etc. Ce module fonctionne sur la base d'implémentation de modèles d'apprentissage profond et de modèles collaboratifs tels que le raisonnement à partir de cas

Les modules LD1 et LD3 ont été partiellement développés et n'ont pas été suffisamment testés pour les intégrer dans le système de façon fonctionnelle, les autres modules pour le moment sont théoriques.

5.3.3/ RÉVISION

Les modules qui se trouvent dans cette section sont les principales de ce travail. Dans la couche de révision, les modules utilisent les informations générées par les modules des couches de correction automatique et d'identification ainsi que les informations complémentaires de l'apprenant stockées dans la base de données du système. Ces modules valident le fait que la recommandation générée est optimale pour l'apprenant. Toutes les informations récoltées permettent d'établir la meilleure façon de guider l'apprenant vers une meilleure compréhension en surmontant les faiblesses et les lacunes qui ont été identifiées.

Les ressources recommandées par le module LR1 proviennent d'une base de données

déjà établie que le module consulte et suggère à l'étudiant en fonction du résultat de la comparaison de l'état d'apprentissage, du niveau et des caractéristiques et spécifications de chacune des ressources.

Le module LR2 a deux variantes fondées sur les valeurs générées par l'apprenant lorsque la séance d'entrainement est en cours : l'une déterministe et l'autre stochastique. Le modèle déterministe utilise un tableau prédéterminé de rangs sur lequel l'apprenant est positionné pour générer la suggestion d'adaptation. Le modèle stochastique utilise des distributions de probabilités dynamiques qui changent en fonction des résultats de l'apprenant à chaque niveau de complexité de la même sous-compétence.

D'autres modules proposant une prédiction avec les mêmes données que celles qui ont été utilisées pour la génération de l'adaptation sont aujourd'hui implémentés dans AI-VT. Ils permettent d'estimer plus précisément la performance de l'apprenant et si effectivement l'adaptation proposée lui permet d'acquérir les compétences nécessaires et d'améliorer les notes obtenues.

Le module LR3 obtient dynamiquement des structures variables à partir des informations extraites du système AI-VT central. Ce module peut également effectuer des transformations dans la structure, le contenu et la représentation des données des apprenants.

Le module LR4 utilise les informations produites par l'apprenant et les informations collaboratives pour tenter de prédire les performances futures de l'apprenant. En particulier avec la recommandation générée par les modules de la couche adaptative, la prédiction est utilisée pour valider l'adaptation recommandée et l'ajuster si nécessaire.

Le module LR1 est fonctionnelle dans le système AI-VT, mais encore il est nécessaire ajouter plus de ressources en fonction des questions configurées. L'algorithme proposé pour le module LR2 est décrit dans le chapitre 7, l'algorithme proposé pour les modules LR3, et LR4 apparait dans le chapitre 6

5.3.4/ TEST

La couche de test fonctionne en amont et en aval de la génération des séances d'entrainement. Elle permet d'évaluer chacun des modules indépendamment et de générer des données spécifiques pour divers scénarios de test qui peuvent être pris en compte lors de la validation d'un module.

Les outils des différentes couches de l'architecture peuvent communiquer entre eux, car, pour que certains de leurs modules internes fonctionnent, ils ont besoin des informations générées par les modules des autres couches. La figure 5.3 montre les interactions qui peuvent se produire lorsque le système génère une séance ou l'adapte ou encore lorsqu'un module expérimental doit être évalué. La couche de test (LT) a besoin des informations générées par les modules de toutes les autres couches pour évaluer leurs performances individuelles. La couche d'identification (LD) doit obtenir les données relatives au profil des apprenants. Ces informations sont générées par le module de profil qui se trouve dans la couche de révision (LR). Pour certains modules qui ont la capacité de modifier une solution, cette couche doit connaître les résultats obtenus par l'apprenant dans chacun des tests ou exercices proposés par le système. Ces résultats sont attribués par la couche de correction automatique (LC). De plus, il est possible d'obtenir une estimation du résultat de la révision proposée avant qu'elle ne soit envoyée à l'apprenant, pour cela il faut invoquer les modules spécifiques de prédiction qui appartiennent à la

couche de révision (LR).

L'algorithme spécifique pour le module LT1 et les caractéristiques de la base de données générée sont détaillées dans le chapitre 7.

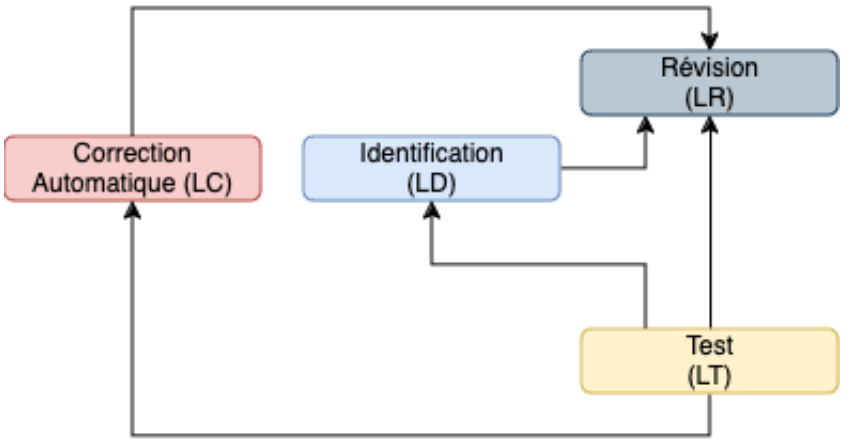

FIGURE 5.3 – Relation entre les couches définies de l'architecture.

Le flux complet d'informations est représenté sur la figure 5.4. La première étape consiste à envoyer le test généré par le système AI-VT à l'apprenant. Lors de cette étape, le module d'identification correspondant à l'analyse à effectuer est également lancé. Lorsque l'étudiant envoie la réponse à une question, le module d'identification envoie à l'apprenant l'analyse effectuée sur cette même réponse. Grâce à ces informations, le système active les modules de correction automatique pour attribuer une note à la réponse envoyée, en tenant compte également des résultats du module d'identification. À l'étape 6, la note générée est envoyée au système, puis pour effectuer l'adaptation, le système obtient les informations spécifiques de l'apprenant et lance les modules d'adaptation, en envoyant les informations obtenues dans les étapes précédentes. Les algorithmes d'adaptation évaluent les variables et déterminent le parcours optimal pour l'apprenant, mais avant de le renvoyer au système principal, l'étape 9 évalue sa pertinence à l'aide du module de prédiction. Si le résultat de la prédiction détermine que le chemin suggéré est acceptable, il est envoyé au système principal qui décide, à l'étape 11, de le présenter à l'apprenant comme une alternative au chemin sélectionné à l'origine.

5.4/ CONCLUSION

L'architecture modulaire proposée est fondée sur des concepts et des modèles couramment utilisés pour concevoir des systèmes complexes. Ils utilisent des algorithmes et des outils d'intelligence artificielle. Ce type de conception permet la mise en œuvre d'un système fonctionnel doté d'une capacité d'adaptation, nécessaire à l'exécution de l'une des principales exigences des systèmes d'apprentissage intelligents. En outre, comme l'indiquent les travaux cités en référence, l'architecture modulaire permet une mise en œuvre plus souple et donne au système la possibilité d'évoluer rapidement et même d'ajouter des fonctionnalités complémentaires sans affecter le système et les données qui y sont stockées. Les modules principales qui vont être développés dans cette thèse

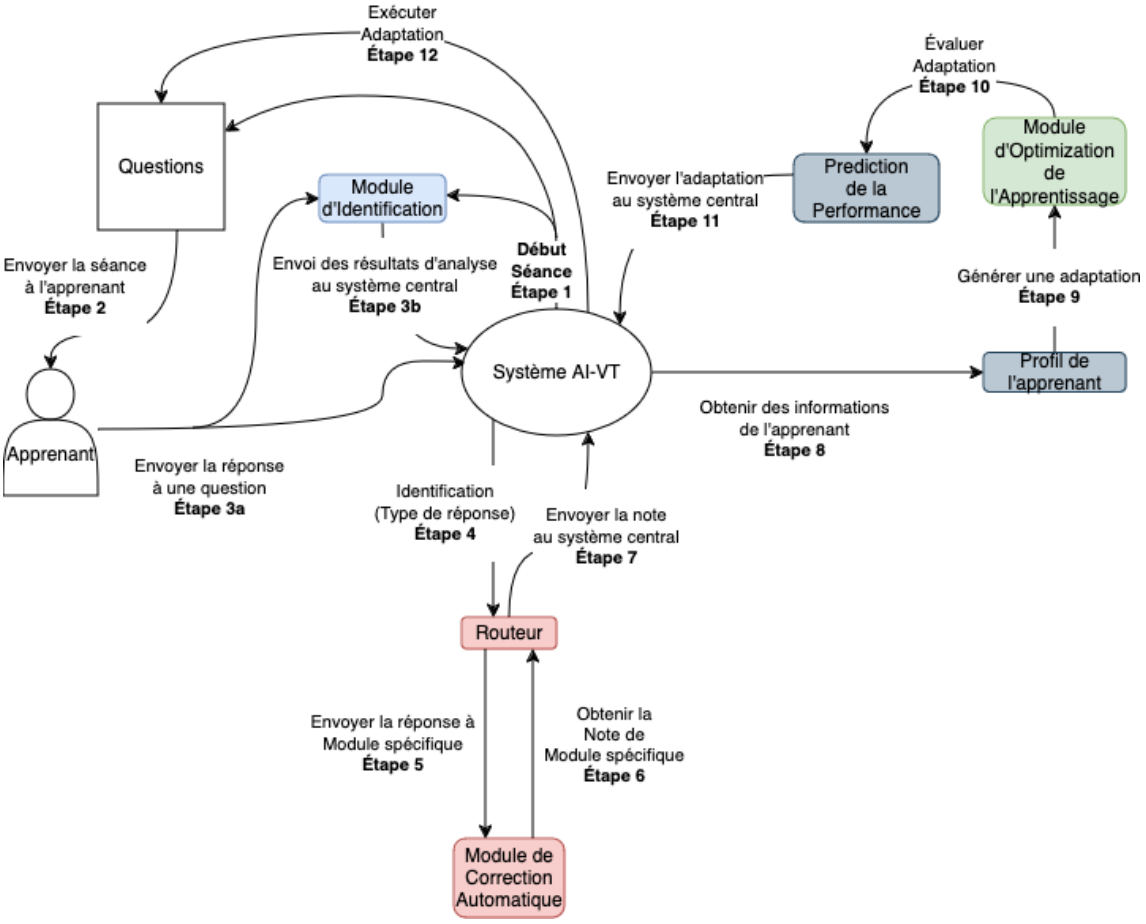

FIGURE 5.4 – Étapes du flux de l'information pour la fonctionnalité de recommandation globale.

et explicités dans les chapitres suivantes sont : Optimisation du processus d'apprentissage (LR2), Profil de l'apprenant (LR3), Prédiction de la performance (LR4), Analyse de données (LD3) et Simulation (LT1).

6

LE RAISONNEMENT À PARTIR DE CAS (RÀPC) POUR LA RÉGRESSION

6.1/ INTRODUCTION

Ce chapitre est divisé en deux parties. La première partie présente un algorithme fondé sur le raisonnement à partir de cas et les méthodes d'ensemble avec un double empilement itératif. Nous l'avons baptisé *ESCBR* (*Ensemble Stacking Case Based Reasoning*). Ce processus se fait en deux étapes pour trouver des solutions approximatives à des problèmes de régression unidimensionnels et multidimensionnels. Une partie de cette proposition est publiée dans [Soto-Forero et al., 2024c]. La seconde partie montre la conception et l'implémentation d'un SMA intégré à l'ESCBR présenté dans la première partie. Nous considérons dans ce chapitre que le choix des exercices les plus adaptés revient à résoudre un problème de régression. C'est la raison pour laquelle nous testons notre approche sur des jeux de données classiques de régression. L'algorithme présenté dans cette seconde partie associe un raisonnement à partir de cas à des systèmes multi-agents cognitifs implémentant un raisonnement Bayésien. Cette association, combinant échange d'informations entre agents et apprentissage par renforcement, permet l'émergence de prédictions plus précises et améliore ainsi les performances d'ESCBR [Soto-Forero et al., 2024c].

Avant de présenter nos propositions, rappelons quelques caractéristiques importantes des outils combinés dans notre approche.

Le système de raisonnement à partir de cas ne nécessite pas d'entrainement, et peut fonctionner avec des données dynamiques au moment de l'exécution. Les solutions proposées par notre système de RàPC sont générées à l'aide d'algorithmes stochastiques guidant l'exploration de l'espace des solutions. L'évaluation des solutions possibles est effectuée en transformant le problème de régression en un problème d'optimisation avec une fonction objectif associée. La fonction objectif calcule un rapport de la distance entre la solution générée et les solutions connues avec les problèmes à résoudre et les problèmes connus. La définition formelle se trouve dans la section 6.2.1.3. Les prédictions de ce nouvel algorithme ont été comparées à celles de neuf algorithmes de régression classiques sur dix jeux de données pour la régression extraits du site de l'UCI [UCI, 2024]. En évaluant les résultats obtenus selon la métrique RMSE (Erreur quadratique moyenne - *Root Mean Squared Error*), ce nouvel algorithme se classe parmi les six meilleurs. Selon la métrique MAE (Erreur moyenne absolue - *Mean Absolute Error*) il est le troisième

meilleur algorithme des dix évalués, ce qui suggère que les résultats produits par ESCBR sont raisonnablement satisfaisants.

La technique d'ensemble permet de résoudre des problèmes de classification et de régression en combinant les résultats de plusieurs algorithmes exécutés indépendamment. Certaines de ces méthodes utilisent des algorithmes différents et des ensembles de données différents tandis que d'autres utilisent les mêmes algorithmes avec des paramètres différents. La combinaison des résultats provenant de multiples algorithmes peut être réalisée selon différentes stratégies comme l'application de règles simples ou des approches plus complexes [Bakurov et al., 2021]. Plusieurs travaux de recherche explorent la possibilité d'utiliser cette technique d'ensemble en la combinant à des outils d'apprentissage automatique.

Les méthodes d'apprentissage automatique appliquées à la régression permettent quant à elles de prédire des valeurs pour différents types de problèmes en construisant, évaluant et formant des algorithmes linéaires et non linéaires complexes. Mais il est possible d'en améliorer la précision en les associant. Les stratégies d'intégration les plus courantes utilisées pour l'apprentissage d'ensemble sont le *Stacking* (empilement), le *Boosting* (stimulation) et le *Bagging* (ensachage) [Mang et al., 2021]. Le *Stacking* est une technique d'apprentissage profond d'ensemble dont l'objectif est d'utiliser diverses algorithmes d'apprentissage automatique pour surmonter les limites des algorithmes individuels. Plusieurs études démontrent que l'association de ces algorithmes permet d'améliorer la précision des résultats [Choi et al., 2023]. Dans ces méthodes d'empilement, les algorithmes de base sont appelés *niveau-0*. Il s'agit généralement d'algorithmes d'apprentissage automatique hétérogènes qui travaillent tous avec les mêmes données. Le méta-algorithme (appelé *niveau-1*) qui unifie les résultats peut être une autre technique d'apprentissage automatique ou un ensemble de règles qui reçoit en entrée les résultats des algorithmes de *niveau-0* [Liang et al., 2021].

La modélisation de systèmes implémentant différentes techniques d'apprentissage nous amène tout naturellement à considérer les avantages et inconvénients proposés par un système multi-agents. Un SMA est un système décentralisé composé de plusieurs entités appelées agents qui ont la capacité d'effectuer des actions spécifiques et de réagir à l'environnement en fonction des informations partielles à leur disposition. Ils peuvent également collaborer les uns avec les autres et coopérer pour atteindre un objectif spécifique. À partir des informations qu'ils perçoivent et échangent, les agents peuvent apprendre de manière autonome et atteindre leurs objectifs efficacement [Kamali et al., 2023]. Les actions d'un système multi-agents peuvent être exécutées en parallèle grâce à l'indépendance des agents. Ils sont également robustes aux problèmes présentant une incertitude [Didden et al., 2023].

Le raisonnement Bayésien clos ce premier tour d'horizon des techniques implémentées dans l'approche présentée dans ce chapitre. Celui-ci est fondé sur l'observation du raisonnement humain et la relation de l'humain avec l'environnement. Son principe repose sur le postulat que les expériences passées permettent de déduire les états futurs, guidant ainsi ses actions et ses décisions [Hipólito and Kirchhoff, 2023]. Avec le raisonnement Bayésien certaines informations peuvent également être déduites à partir d'informations incomplètes [Zhang et al., 2023].

6.2/ APPRENTISSAGE PAR EMPILEMENT ET RAISONNEMENT À PARTIR DE CAS

Cette section présente la première version de l'algorithme que nous avons conçu pour résoudre des problèmes de régression. Celui-ci est fondé sur le raisonnement à partir de cas et l'empilement.

6.2.1/ ALGORITHME PROPOSÉ

L'algorithme proposé, ESCBR (*Ensemble Stacking Case-Based Reasoning*), est fondé sur le paradigme générique du RàPC combiné à plusieurs algorithmes de recherche de voisins et de génération de solutions. Ces algorithmes ont été intégrés selon une variation de l'empilement en deux étapes itératives. Cette intégration donne à l'algorithme la capacité de s'adapter à différents types de problèmes, d'éviter les biais et le surentraînement. Les résultats de l'exécution des niveaux d'empilement stockent dans la mémoire des conteneurs de connaissances du système de RàPC des informations qui aident non seulement à l'apprentissage de l'algorithme au fil des itérations, mais facilitent également la génération de solutions à divers problèmes sur différents ensembles de données sans nécessité d'entraînement préalable. La conception itérative en deux cycles améliore la capacité du système de RàPC à travailler et à s'adapter à des problèmes dynamiques en cours d'exécution, comme le montre la figure 6.1.

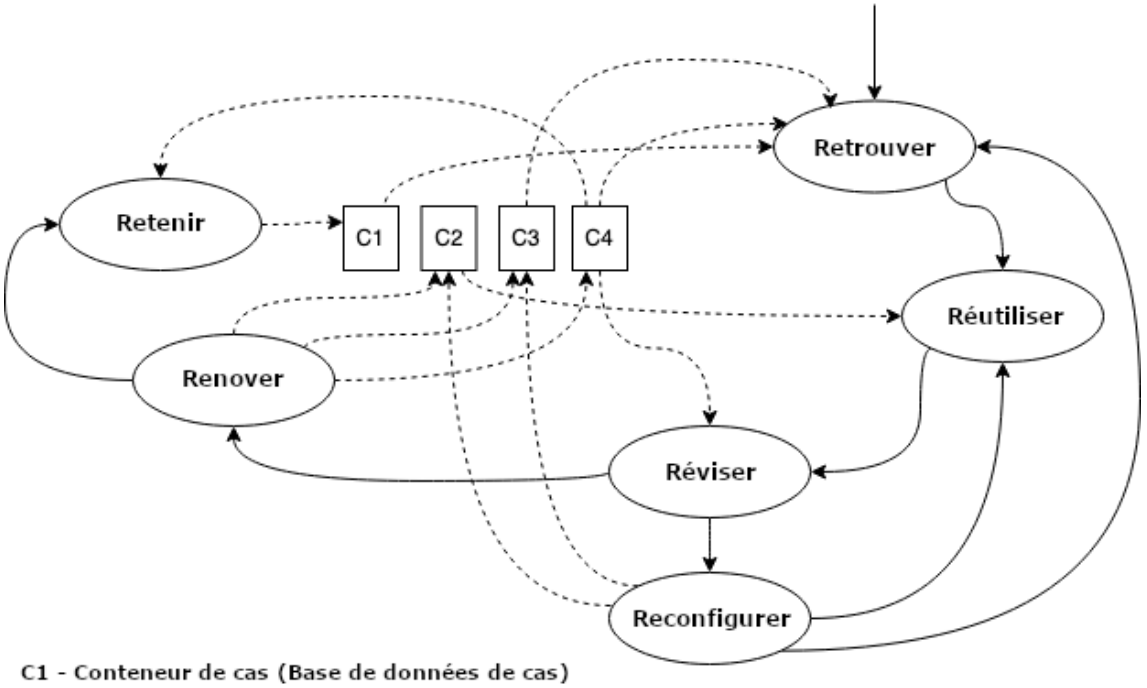

FIGURE 6.1 – Les deux cycles proposés pour le RàPC

L'étape de récupération utilise les algorithmes de recherche et le jeu de données de cas (conteneurs $C1$ et $C3$ de la figure 6.1) pour trouver les voisins les plus proches d'un problème cible. Puis l'étape de réutilisation utilise les algorithmes de génération de solutions (conteneur $C2$). L'étape de révision évalue ensuite les solutions générées et permet d'en générer de nouvelles itérativement en fonction des paramètres stockés dans le conteneur $C4$. Vient ensuite l'étape de révision prenant en considération la solution sélectionnée. Faisant suite à cette révision, l'étape de renouvellement met à jour les paramètres et les données du conteneur. Enfin, dans l'étape de capitalisation, la base de cas est mise à jour avec l'intégration du nouveau cas. Les flux d'information de l'algorithme proposé sont présentés sur la figure 6.2, tandis que le tableau 6.1 présente l'ensemble des variables et paramètres de l'algorithme proposé.

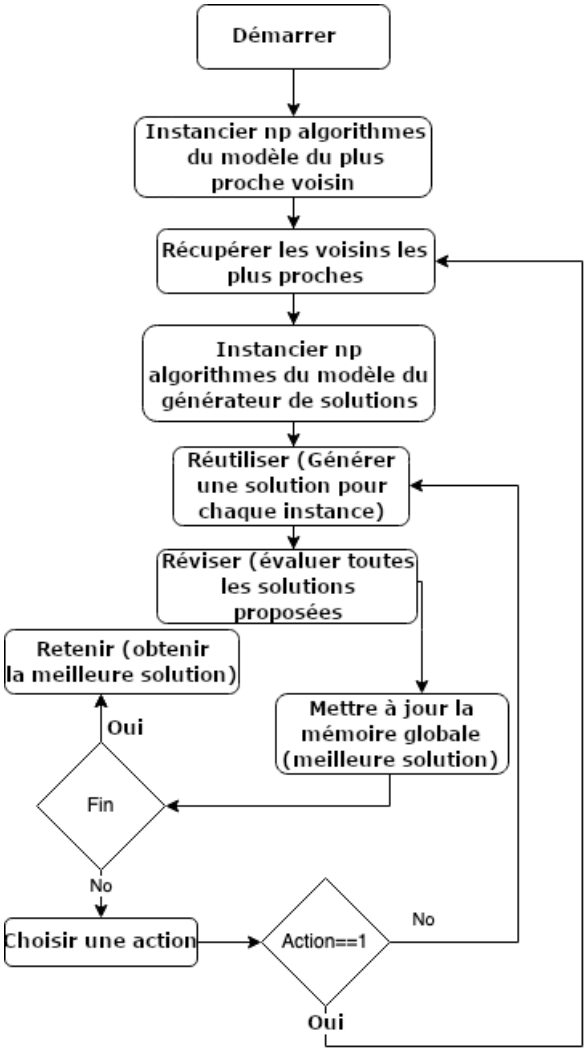

FIGURE 6.2 – Flux du *Stacking* RàPC

6.2.1.1/ RECHERCHER

La première étape de l'algorithme consiste à trouver les cas les plus similaires à un nouveau cas (appelé *cas cible*). Pour ce faire, l'empilement de différents processus présenté

ID	Type	Description	Domain
it	p	Nombre d'itérations	$\mathbb{N}, it > 0$
np	p	Nombre de processus	$\mathbb{N}, np > 2$
nl	p	Nombre maximal de voisins locaux	$\mathbb{N}, nl > 0$
ng	p	Nombre de voisins globaux	$\mathbb{N}, ng > 2$
n	v	Dimension de l'espace du problème	$\mathbb{N}, n > 0$
m	v	Dimension de l'espace de solution	$\mathbb{N}, m > 0$
z	v	Taille du jeu de données	$\mathbb{N}, z > 0$
p	v	Description du problème	$\mathbb{R}^n$
s	v	Description de la solution	$\mathbb{R}^m$
r_a	v	Nombre d'algorithmes pour l'étape retrouver	$\mathbb{N}, r_a > 2$
r_b	v	Nombre d'algorithmes pour l'étape de réutilisation	$\mathbb{N}, r_b > 2$
at	v	Identificateur des actions	$[0, 2] \in \mathbb{N}$
nl_i	v	Nombre de voisins locaux pour l'algorithme i	$\mathbb{N}, nl_i \leq nl$
g	v	Description de la meilleure solution globale	$\mathbb{R}^m$
v	v	Évaluation de la meilleure solution globale	$\mathbb{R}$
$d(x_1, x_2)$	f	Fonction de distance entre x_1 et x_2	$\mathbb{R}$
$MP(x_1^z, x_2, a)$	f	Fonction pour retrouver entre x_1 et x_2	$\mathbb{R}^{a \times z}$
$MS(x_1^m)$	f	Fonction pour réutiliser avec x_1	$\mathbb{R}^m$
$f_s(p^n, s^m)$	f	Évaluation des solutions	$\mathbb{R}$

TABLE 6.1 – Variables et paramètres de l'algorithme proposé (Type : p - paramètre, v - variable, f - fonction)

sur la figure 6.3 est utilisé. Au niveau-0, chaque processus sélectionne et exécute un algorithme de recherche de voisins différents choisi parmi r_a algorithmes dans le conteneur $C3$, avec un nombre de voisins nl_i choisi aléatoirement dans l'intervalle $[0, nl]$. Puis au niveau-1, les résultats sont unifiés en construisant un ensemble global de cas similaires. Cinq algorithmes pour le niveau-0 ont été mis en œuvre pour l'étape de récupération : KNN (*K Nearest Neighbors* - K plus proches voisins), KMeans, GMM (*Gaussian Mixture Model*), FuzzyC et KNN Ponderation.

Formellement, le premier empilement proposé fonctionne avec deux paramètres : un jeu de données de z cas où un cas est composé de la description du problème et de la description de la solution $(p^n, s^m)^z$ et un nouveau cas sans solution p_w^n. Le but de tous les algorithmes de niveau-0 est de générer une liste locale de cas similaires au cas cible. Ainsi, pour chaque algorithme j exécuté, l'ensemble $X_j = \{x_1, x_2, ..., x_z \mid x_i = MP_i((p^n)^z, p_w^n, nl_i)\}$ est généré. Au niveau-1, un ensemble global est créé à l'aide de tous les ensembles locaux j $X_g = \cup_{n=1}^{ng} min\,; ((\cup_{j=1}^{np} X_j) - X_g)$. Le résultat du premier empilement est l'ensemble X_g avec les ng voisins les plus proches.

6.2.1.2/ RÉUTILISER

Une fois la liste globale des cas similaires établie, les informations correspondant aux solutions de chacun de ces cas sont extraites et utilisées pour générer une nouvelle solution qui s'adapte au cas cible, en suivant le processus et les flux d'informations représentés sur la figure 6.5. La génération est effectuée avec un deuxième empilement de différents processus. Cet empilement est représenté sur la figure 6.6. Au niveau-0, chaque processus sélectionne et exécute un algorithme de génération différent à partir des algorithmes

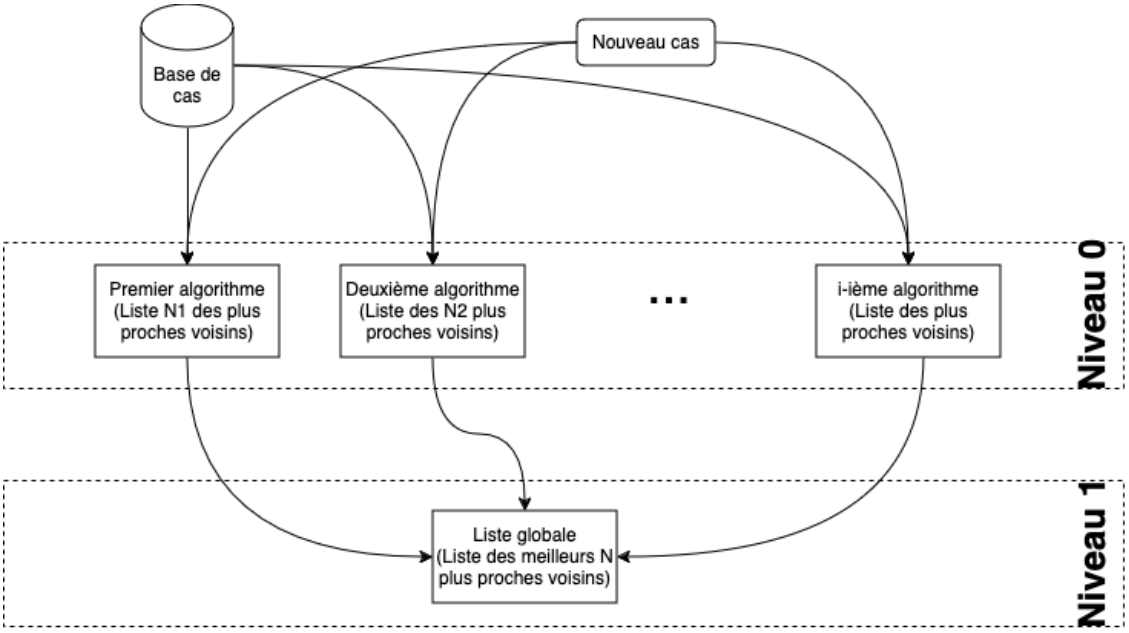

FIGURE 6.3 – *Stacking* pour chercher les plus proches voisins

r_b dans le conteneur $C2$. Au niveau-1, toutes les solutions générées sont stockées dans une mémoire globale. Toutes les solutions connues et générées sont représentées avec la même structure comme le montre la figure 6.4.

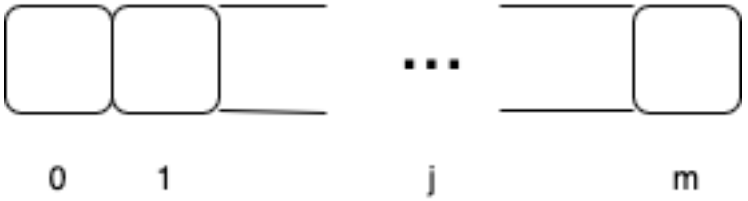

FIGURE 6.4 – Représentation des solutions connues et générées

Neuf algorithmes ont été mis en œuvre pour l'étape de réutilisation au niveau-0 : moyenne avec probabilité, moyenne sans probabilité, valeurs médianes, sélection aléatoire avec probabilité, copie et changement, vote, interpolation, PCA (analyse en composantes principales) et marche aléatoire. Tous les algorithmes proposés pour générer une nouvelle solution combinent et modifient l'ensemble de solutions construit dans l'étape de récupération.

La moyenne pondérée avec probabilité est construite en considérant les nl cas les plus proches du cas cible. Ces nl cas sont les cas sélectionnés par le premier empilement. Pour chacun de ces nl cas, un ratio α_j est calculé en considérant la distance entre celui-ci et le cas cible, selon l'équation 6.1. Nous considérons alors l'ensemble de ces ratios comme une distribution de probabilité discrète. Chacune des composantes du vecteur solution est ensuite calculée selon l'équation 6.2. Cet algorithme permet de donner plus de poids aux solutions des cas les plus proches du cas cible.

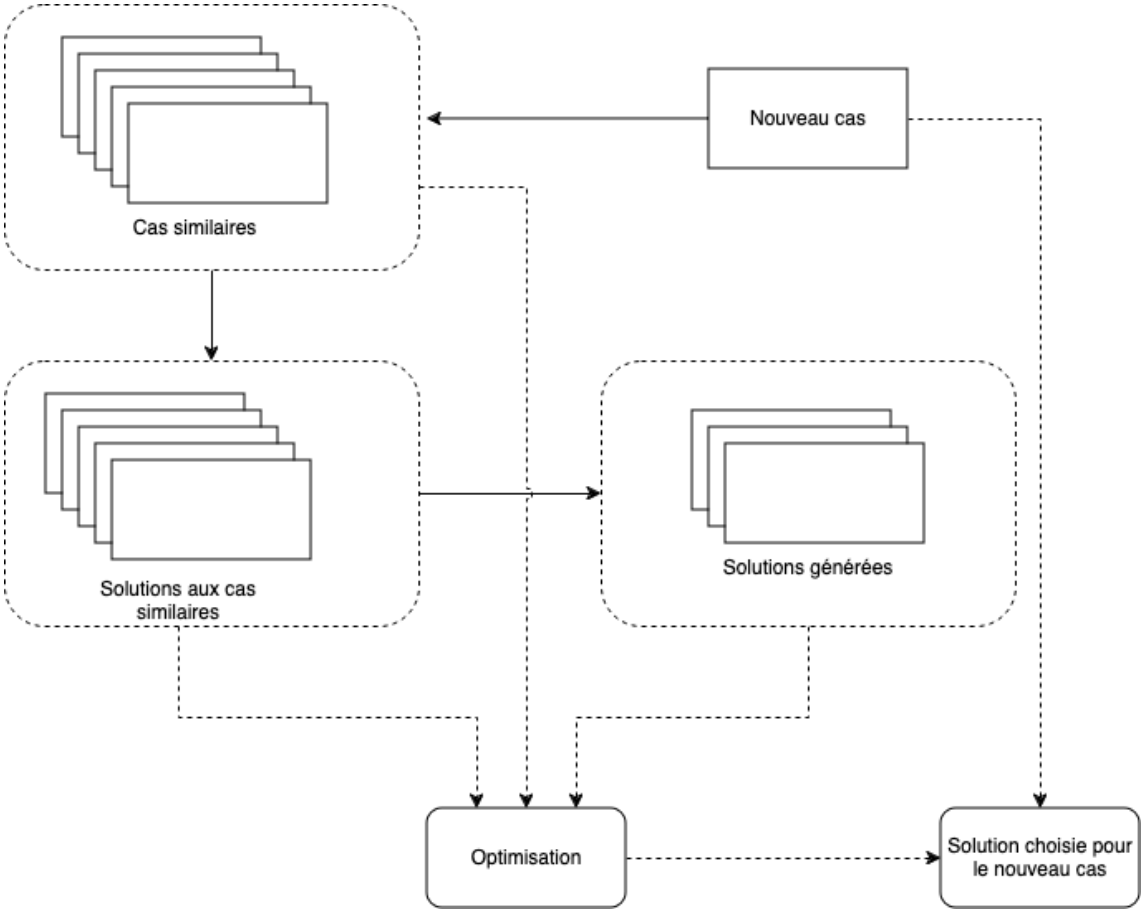

FIGURE 6.5 – Génération et vérification automatique des solutions

$$\alpha_j = 1 - \left(\frac{d(p_j^n, p_w^n)}{\sum_{i=0}^{nl} d(p_i^n, p_w^n)} \right) \tag{6.1}$$

$$s_{j,w}^m = \sum_{i=0}^{nl-1} \alpha_j s_{i,j} \tag{6.2}$$

La moyenne sans probabilité génère la nouvelle solution où la valeur de chaque composante est la moyenne calculée avec une distribution de probabilité uniforme (attribuant la même probabilité à toutes les composantes de toutes les solutions associées aux cas les plus proches du cas cible). Formellement la génération de la solution est calculée selon l'équation 6.3.

$$s_{w,j}^m = \frac{1}{nl} \sum_{i=0}^{nl-1} s_{i,j} \tag{6.3}$$

La génération par valeurs médianes (équation 6.4) construit la solution en utilisant la valeur médiane de toutes les solutions pour chaque composante j. X_j représente l'ensemble des valeurs ordonnées des j-ièmes composantes de toutes solutions S.

$$s^m_{j,w} = \begin{cases} x_{j,\frac{m+1}{2}}, \ si\ m \in \{2k+1 : k \in \mathbb{Z}\} \\ \frac{x_{j,\frac{m}{2}} + x_{j,\frac{m}{2}+1}}{2}, \ sinon \end{cases} \tag{6.4}$$

La sélection aléatoire avec probabilité (équation 6.5) génère une solution en copiant la j-ième composante de l'une des solutions tirée au sort. Le tirage au sort suit une loi donnée par une distribution de probabilité discrète. Cette distribution de probabilité est identique à celle de la moyenne pondérée avec probabilité.

$$s^m_{j,w} = s^m_{j,k}; k \sim \left(1 - \left(\frac{d(p_k, p_w)}{\sum_{i=1}^{n} l(p_k, p_i)}\right)\right) \tag{6.5}$$

"Copie/changement" copie les informations d'une solution aléatoire et en remplace une partie par celles d'une autre solution sélectionnée aléatoirement selon l'équation 6.6.

$$s^m_{j,w} = s^m_{j,k}; k \sim \left(\frac{1}{nl}\right) \tag{6.6}$$

Le vote permet de copier l'information la plus fréquente de toutes 6.7.

$$s^m_{j,w} = s^m_{j,k}; k = argmax_i\ \mathbb{P}(X = s^m_{j,i}), 1 \leq i \leq nl \tag{6.7}$$

L'interpolation construit une distribution de probabilité continue pour chaque composante j. Chaque probabilité de ces distributions est le résultat d'une fonction d'interpolation linéaire à une composante. La valeur de la j-ième composante de la solution est calculée selon l'équation 6.8.

$$s^m_{w,j} \sim \left(\left(\frac{y_i - y_{i+1}}{x_i - x_{i+1}}\right)(x - x_i) + y_i\right), \forall i\ 0 \leq i < nl \tag{6.8}$$

L'analyse en composantes principales (PCA) consiste à établir une transformation de la description du problème en description de la solution associée. La moyenne de la distance entre toutes les paires problème-solution est calculée et permet de générer une solution par transposition.

La PCA permet donc de générer une matrice de transformation $\mathcal{M}$ d'un vecteur de l'espace des problèmes en vecteur de l'espace des solutions. Pour chaque cas source, la distance entre le vecteur solution obtenu et le vecteur solution stocké est calculée. Une moyenne de ces distances md est ensuite considérée.

L'application de $\mathcal{M}$ au vecteur problème du cas cible donne donc dans un premier temps un vecteur solution. La solution cible est alors le vecteur $md \times \mathcal{M}$.

La marche aléatoire consiste à choisir une solution et à changer la valeur de l'une de ses composantes. Cette composante est tirée au sort et un pas Δ est ajouté à sa valeur. La valeur de Δ est générée en suivant une distribution de probabilité normale de moyenne 0 et de variance 1.

$$k \sim \left(\frac{1}{nl}\right) \tag{6.9}$$

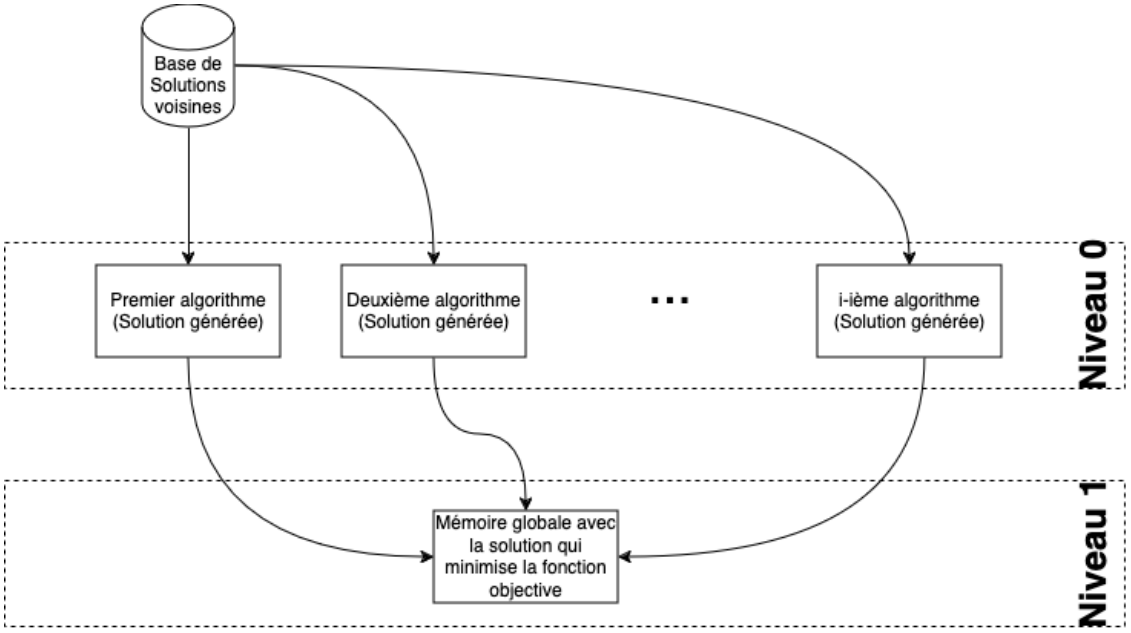

FIGURE 6.6 – *Stacking* pour la génération de solutions

$$s^m_{w,j} = \begin{cases} s^m_{j,k} + \mathcal{N}(0,1) & si\ (\mathcal{U}_{int}(0,10))\%2 = 0 \\ s^m_{j,k} - \mathcal{N}(0,1) & sinon \end{cases} \tag{6.10}$$

La description des solutions s et l'ensemble $(s^m)^{ng}$ sont les paramètres du deuxième empilement. Chaque algorithme décrit ci-dessus génère une solution candidate $s_{i,c} = MS_i((s^m)^{ng})$. Vient ensuite la construction de l'ensemble d'unification de toutes les solutions candidates $Y_g = \cup_{i=1}^{np} s_{i,c}$ (niveau-1). Cet ensemble est évalué à l'aide d'une fonction permettant de déterminer la qualité de la solution.

6.2.1.3/ RÉVISION

Dans cette phase, le problème de l'évaluation automatique d'une solution candidate est transformé en un problème d'optimisation, où la fonction objectif est 6.12. Ce problème revient à calculer la moyenne géométrique ou encore à résoudre le problème de "Fermat-Weber". Dans le cas d'un espace multidimensionnel, résoudre ce problème revient à calculer la moyenne spatiale [Minsker and Strawn, 2024]. La figure 6.7 montre un exemple de la formulation du problème en deux dimensions. Sur cette figure, le point rouge représente une solution possible minimisant la somme des distances entre ce point et tous les autres points définis dans l'espace.

La fonction objectif 6.12 établit un rapport entre la distance de la solution générée s^m_w et les x solutions connues s^m_x avec un facteur aléatoire de *drift* d'une part, et la distance entre le problème cible p^n_w et les x problèmes connus p^n_x d'autre part. Ici, la difficulté à trouver le point optimal réside dans le fait que les points de l'espace ne peuvent pas tous convenir en tant que solution cible. La solution cible finale dépend donc des informations des solutions connues. L'objectif est d'utiliser les informations disponibles de chaque cas (problème et solution) pour valider et générer l'ensemble de solutions proposées.

$$\lambda_x(p_w, s_w) = \left(\frac{d(s_w^m, (s_x^m + rn(0, d(p_w^n, p_i^n))))}{d(p_w^n, p_x^n)^2} \right) \tag{6.11}$$

$$min \ \ (f_s(p_w^n, s_w^m)) = min \left(\sum_{i=1}^{ng} \lambda_i(p_w, s_w) \right) \tag{6.12}$$

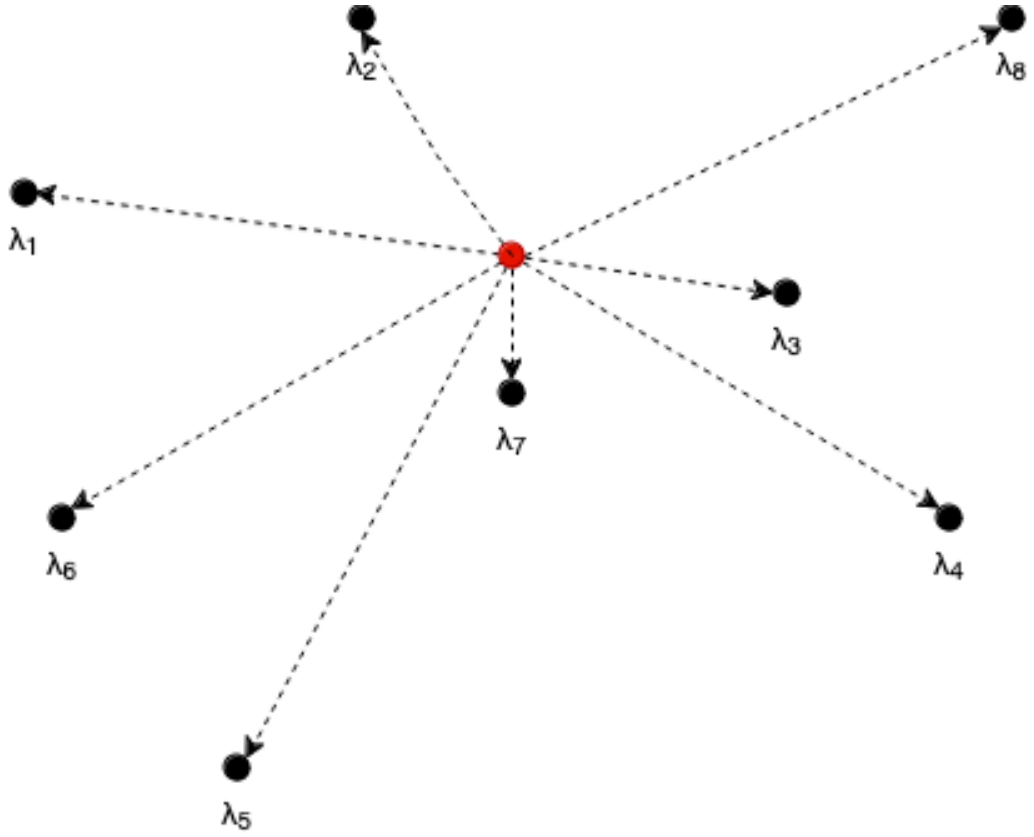

FIGURE 6.7 – Représentation graphique en deux dimensions du problème de moyenne géométrique. (Points associés au problème $(\lambda_1, .., \lambda_7)$ et point rouge solution au problème)

Le cycle d'optimisation permet d'exécuter les phases de récupération et de réutilisation en fonction de l'action sélectionnée selon une distribution de probabilité parmi $[0, at]$ à chaque itération it. À chaque itération, la valeur minimale évaluée par la fonction objectif est sauvegardée.

6.2.1.4/ MÉMORISATION

L'étape de mémorisation consiste simplement à prendre la meilleure solution proposée et à déterminer s'il s'agit d'une solution nouvelle ou existante. S'il s'agit d'une nouvelle solution, elle est enregistrée dans la base de connaissances.

6.2.2/ RÉSULTATS

Les performances de prédiction de l'algorithme proposé ont été comparées à celles de neuf autres algorithmes sur dix jeux de données classiquement utilisés pour évaluer des méthodes de régression. Ces jeux de données présentent des caractéristiques différentes. Les jeux de données et leurs caractéristiques sont consignées dans le tableau 6.2. Les valeurs des paramètres de l'algorithme sont les suivantes : $it = 100$, $np = 50$, $nl = 10$ et $ng = 10$.

ID	DataSet	Features	Instances	Output Dimension	Input Domain	Output Domain
DS1	Yatch Hydrodynamics	6	308	1	$\mathbb{R}$	$\mathbb{R}$
DS2	Electrical Grid Stability	12	10000	1	$\mathbb{R}$	$\mathbb{R}$
DS3	Real State Valuation	6	414	1	$\mathbb{R}_+$	$\mathbb{R}_+$
DS4	Wine Quality (Red)	11	1598	1	$\mathbb{R}_+$	$\mathbb{N}$
DS5	Wine Quality (White)	11	4897	1	$\mathbb{R}_+$	$\mathbb{N}$
DS6	Concrete Compressive Strength	8	1030	1	$\mathbb{R}_+$	$\mathbb{R}_+$
DS7	Energy Efficiency	8	768	2	$\mathbb{R}_+$	$\mathbb{R}_+$
DS8	Gas Turbine CO, NOx Emission (2015)	9	7384	2	$\mathbb{R}_+$	$\mathbb{R}_+$
DS9	Student Performace Portuguese	30	649	3	$\mathbb{N}$	$\mathbb{N}$
DS10	Student Performance Math	30	395	3	$\mathbb{N}$	$\mathbb{N}$

TABLE 6.2 – Description des jeux de données évaluées.

L'algorithme proposé est comparé à neuf algorithmes de régression largement utilisés dans divers travaux de recherche et problèmes appliqués. La liste des algorithmes est présentée dans le tableau 6.3. Tous les algorithmes ont été exécutés cent fois, et les algorithmes qui nécessitent un entraînement et des validations croisées sont exécutés avec $k = 10$ blocs.

ID	Algorithme	ID	Algorithme
A1	Linear Regression	A6	Polinomial Regression
A2	K-Nearest Neighbor	A7	Ridge Regression
A3	Decision Tree	A8	Lasso Regression
A4	Random Forest (Ensemble)	A9	Gradient Boosting (Ensemble)
A5	Multi Layer Perceptron	A10	Proposed Case Based Reasoning

TABLE 6.3 – Liste des algorithmes évalués

Le tableau 6.4 présente l'erreur quadratique moyenne (RMSE) obtenue par chaque algorithme pour chaque jeu de données. Le tableau 6.5 présente l'erreur absolue médiane (MAE) obtenue par chaque algorithme pour chaque jeu de données.

Dataset	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
DS1	9.010	10.780	1.224	0.982	3.369	9.009	8.985	9.629	**0.668**	5.871
DS2	0.022	0.025	0.020	0.012	0.017	0.022	0.022	0.037	**0.011**	0.015
DS3	8.633	8.033	9.334	**7.203**	8.470	8.705	8.842	9.009	7.324	8.491
DS4	0.651	0.746	0.782	**0.571**	0.694	0.651	0.651	0.792	0.617	0.762
DS5	0.753	0.806	0.820	**0.599**	0.853	0.754	0.757	0.863	0.688	0.748
DS6	10.439	8.871	6.144	**4.738**	6.553	10.423	10.422	10.428	5.053	8.766
DS7	2.948	2.116	0.541	**0.465**	3.726	2.949	2.979	4.094	0.467	1.973
DS8	1.315	1.161	1.513	**1.109**	1.566	1.303	1.308	1.318	1.125	2.157
DS9	**2.304**	2.624	3.217	2.315	2.898	**2.304**	**2.304**	2.551	2.342	2.802
DS10	3.052	3.404	4.158	**3.014**	3.607	3.061	3.061	3.150	3.020	3.874
Avg. Rank	5.7	6.3	7.2	2.1	6.6	5.6	5.5	8.6	1.8	5.6

TABLE 6.4 – RMSE calculée sur les dix jeux de données sélectionnés obtenue après exécution des dix algorithmes de régression considérés

La dispersion globale, la médiane et les valeurs aberrantes pour quatre jeux de données représentatifs sont présentées sur la figure 6.8. Nous pouvons voir que l'algorithme

Dataset	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
DS1	6.776	2.385	0.231	0.207	3.632	6.778	6.307	5.186	**0.162**	1.193
DS2	0.015	0.017	0.012	0.008	0.012	0.015	0.015	0.030	**0.007**	0.011
DS3	5.092	4.320	4.1	3.632	4.435	5.092	5.20	5.132	**3.504**	3.90
DS4	0.413	0.495	0.18	0.325	0.451	0.413	0.412	0.544	0.387	**0.154**
DS5	0.509	0.548	0.285	0.374	0.550	0.509	0.509	0.633	0.456	**0.113**
DS6	6.989	5.709	3.134	**2.839**	4.306	6.989	6.989	6.986	3.084	5.439
DS7	1.393	1.372	**0.217**	0.218	2.523	1.393	1.529	2.346	0.243	1.008
DS8	0.549	0.297	0.365	**0.289**	0.742	0.549	0.549	0.540	0.309	0.861
DS9	**1.496**	1.788	2.080	1.612	2.005	**1.496**	**1.496**	1.714	1.538	1.721
DS10	2.344	2.534	2.910	2.331	2.543	2.344	2.344	2.481	**2.258**	2.602
Avg. Rank	6.45	6.4	4.35	2.3	7.35	6.55	6.6	7.9	2.4	4.7

TABLE 6.5 – MAE calculée sur les dix jeux de données sélectionnés obtenue après exécution des dix algorithmes de régression considérés

ESCBR proposé génère plus de valeurs aberrantes que les autres algorithmes. Ceci peut être expliqué par le fait que les algorithmes de niveau-0 de ESCBR tendent à privilégier l'exploration de l'espace des solutions. Toutefois, nous pouvons observer que ESCBR est plus stable (la variance est plus faible) et converge mieux que la plupart des autres algorithmes testés.

6.2.3/ DISCUSSION

De par ses performances, ESCBR se révèle compétitif par rapport à certains des neuf autres algorithmes testés pour la prédiction dans les problèmes de régression. En particulier, dans ce travail, nous avons effectué les tests sur dix jeux de données aux caractéristiques variées, telles que le nombre d'instances, le nombre de caractéristiques, le domaine des variables d'entrée, les dimensions de la variable de sortie et le domaine d'application. Cela démontre la polyvalence de l'algorithme proposé et son applicabilité à différentes configurations. Étant donné la nature exploratoire et stochastique d'ESCBR, il présente une grande diversité de solutions générant plusieurs valeurs aberrantes. Malgré cela, dans la plupart des cas, il est possible d'atteindre une solution approximative convergeant vers la solution optimale. Pour cette raison, les valeurs de la moyenne sont parfois élevées, mais celles de la médiane restent faibles, la médiane atténuant les disparités.

On constate également que l'intégration des algorithmes de recherche produit de meilleurs résultats que les algorithmes simples (tels que le KNN ou la régression linéaire).

Globalement, si l'on observe les RMSE obtenues, les algorithmes d'ensemble (*Random forest* et *Gradient Boosting*) sont globalement plus performants que les algorithmes classiques, même si les performances sont variables. Les performances calculées selon la RMSE classent notre algorithme ESCBR à la sixième place. En revanche, en considérant les performances mesurées selon la MAE, ESCBR est classé en première place pour trois des dix jeux de données et il se classe globalement à la troisième place.

Un aspect important de l'algorithme proposé est la fonction objectif, qui peut être modifiée dynamiquement en fonction des caractéristiques du problème évalué. L'une des perspectives envisagées serait de modifier le fonctionnement du paramètre de *drift* dans la fonction objectif en rendant sa valeur dynamique.

De plus, ESCBR peut intégrer des algorithmes différents et des règles spécifiques à

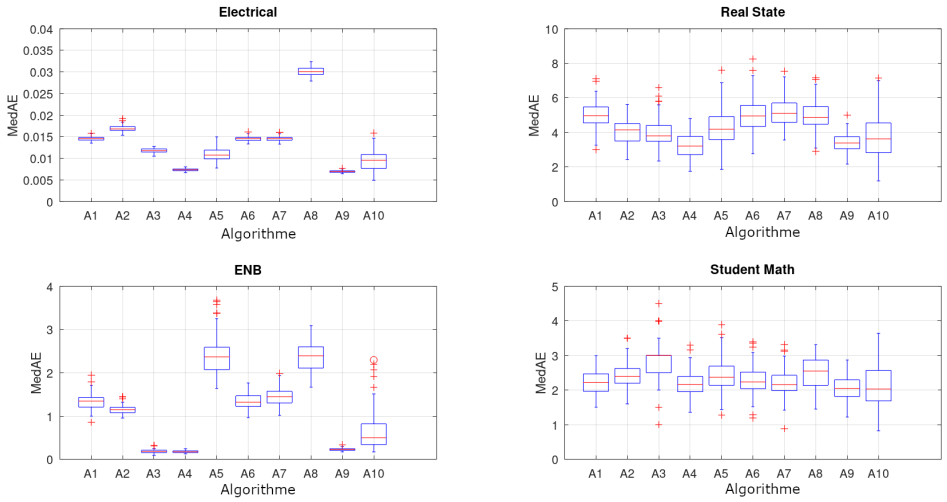

FIGURE 6.8 – Résultats de la métrique MAE (*Median Absolute Error*) pour les dix algorithmes et quatre bases de données représentatives

certaines problèmes dans chaque empilement et, grâce à la conception en deux cycles, il peut travailler avec des problèmes dynamiques en cours d'exécution. Par ailleurs, la variance faible obtenue lors des tests sur les dix jeux de données montrent qu'ESCBR fournit des résultats stables.

6.2.4/ CONCLUSION

Ce chapitre propose une technique de régression générique utilisant le raisonnement à partir de cas et une technique d'empilement que nous avons baptisé ESCBR. Cet algorithme ne nécessite pas d'entrainement préalable et grâce au cycle itératif interne, il peut s'adapter à des problèmes dynamiques en temps réel. Les résultats numériques obtenus lors des tests effectués montrent le potentiel de l'algorithme avec des données variées et des jeux de données de différentes tailles. Les tests effectués dans cette première partie de chapitre montrent ainsi la compétitivité d'ESCBR par rapport à d'autres algorithmes standards et robustes couramment utilisés pour résoudre des problèmes de régression.

6.3/ ESCBR-SMA : INTRODUCTION DES SYSTÈMES MULTI-AGENTS DANS ESCBR

La méthode ESCBR que nous avons proposée dans la section précédente est fondée sur une technique d'apprentissage d'ensemble par empilement. Cette technique mettant en oeuvre plusieurs solutions proposées par des processus intelligents et concurrents pour en générer une, nous nous sommes tout naturellement tournés vers la possibilité d'inclure des agents dans le processus de décision. L'une des ambitions des systèmes

multi-agents consiste en effet à faire émerger une intelligence collective capable de fournir une solution. Cette section présente donc l'évolution d'ESCBR en ESCBR-SMA.

6.3.1/ ALGORITHME PROPOSÉ

L'algorithme ESCBR-SMA est fondé sur le paradigme du RàPC et incorpore divers algorithmes de recherche de voisins et de génération de solutions. Ceux-ci sont intégrés à l'aide d'une variante de la technique d'empilement en deux étapes itératives, exécutées par des agents qui exploitent indépendamment les algorithmes définis dans les conteneurs de connaissances de la révision et de la réutilisation du cycle classique du RàPC. Les agents possèdent une mémoire locale qui leur permet d'ajuster et de développer leur recherche de voisins, en générant des solutions à chaque itération. Chaque agent a un comportement individuel autorisant l'échange d'informations avec les autres agents. La figure 6.9 décrit le processus de décision et les comportements des agents au sein du système élargi.

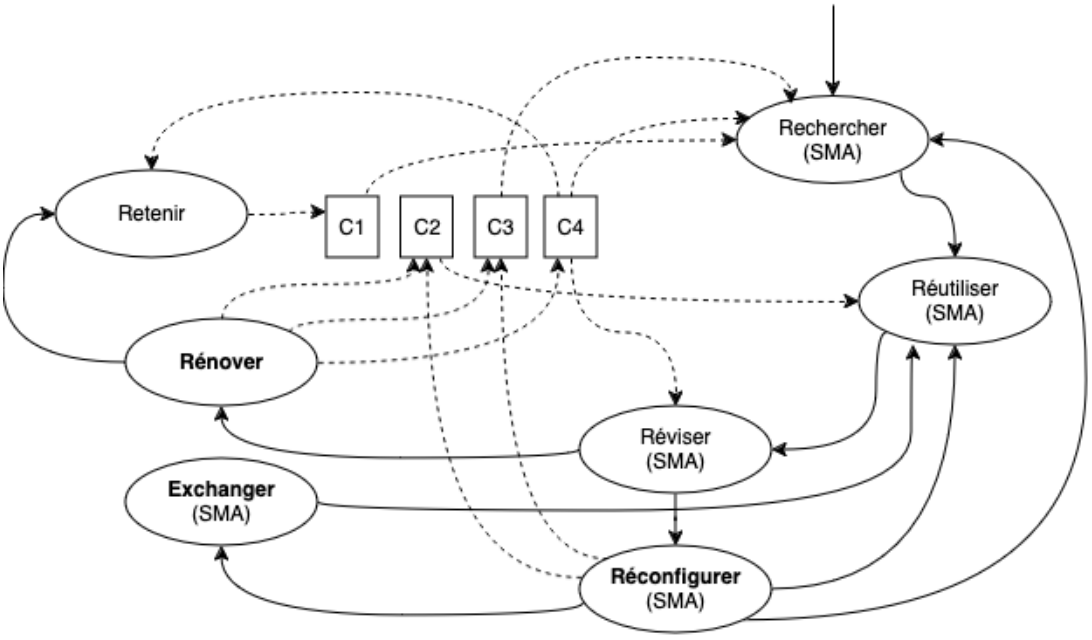

FIGURE 6.9 – Deux cycles du système ESCBR-SMA proposé

Dans ESCBR-SMA, chaque agent effectue un algorithme de recherche des voisins du problème au cours de la première étape puis, au cours de la seconde, génère une solution en se référant à la liste des solutions obtenues au cours de la première étape. À chaque itération, les agents peuvent choisir parmi trois comportements programmés : la recherche de problèmes voisins, la génération de solutions ou l'échange d'informations avec un autre agent. L'exécution de ces trois comportements programmés est asynchrone. En revanche, la création et la sélection de la liste des voisins se font de manière synchronisée.

Les étapes d'extraction, de réutilisation, de révision et de conservation restent conformes au RàPC conventionnel. En revanche, ESCBR-SMA comprend trois nouvelles étapes :

— durant la phase de reconfiguration, les agents mettent à jour les valeurs de leurs paramètres locaux afin d'améliorer la qualité de la solution proposée lors de l'itération suivante,
— durant la phase d'échange, les agents échangent des informations pour améliorer leurs processus internes de recherche et de génération,
— enfin, l'étape de révision met à jour les valeurs des paramètres globaux de l'algorithme.

Le flux complet de l'algorithme proposé est décrit sur la figure 6.10.

En premier lieu, ESCBR-SMA crée n agents. Lors de l'initialisation, les agents sélectionnent au hasard un algorithme de récupération et un algorithme de réutilisation, et initialisent également les vecteurs bayésiens correspondants. Tous les agents travaillent avec le même ensemble de données.

Les agents exécutent en parallèle l'algorithme de récupération sélectionné, à partir des problèmes similaires trouvés ; chaque agent extrait les solutions et exécute l'algorithme spécifique de génération d'une nouvelle solution. Toutes les solutions proposées par les agents sont évaluées à l'aide d'une fonction objectif : la solution qui minimise la fonction objectif est la meilleure solution trouvée à ce stade.

Au cours de l'étape suivante, les agents sélectionnent au hasard un algorithme de récupération et un algorithme de réutilisation, puis ils mettent à jour les vecteurs bayésiens en fonction des résultats obtenus avec la fonction objectif.

Lors de l'itération suivante, chaque agent peut choisir l'une des trois actions possibles : changer les algorithmes de récupération et de réutilisation, ne changer que l'algorithme de réutilisation ou échanger des informations avec un autre agent choisi au hasard. De plus, l'agent choisit aléatoirement une action pour tenter d'améliorer la solution candidate.

ESCBR-SMA utilise l'ensemble des variables et paramètres du tableau 6.1 ainsi que ceux consignés dans le tableau 6.6. Le système multi-agents est composé d'un nombre variable d'agents, tous homogènes mais dotés de processus cognitifs internes indépendants et stochastiques.

ID	Type	Description	Domain
p_w	v	Description du nouveau problème	$\mathbb{R}^n$
s_w	v	Description de la nouvelle solution	$\mathbb{R}^m$
n_{rt}	v	Nombre d'algorithmes pour l'étape de rétrouver	$\mathbb{Z}$
n_{rs}	v	Nombre d'algorithmes de réutilisation	$\mathbb{Z}$
$rn(x,y)$	f	Valeur aléatoire avec distribution normale x moyenne, y écart-type	$\mathbb{R}_+$
$rnp(x,y)$	f	Valeur aléatoire discrète, x nombre d'options y vecteur discret de probabilités	$\mathbb{Z}$

TABLE 6.6 – Variables et paramètres supplémentaires du ESCBR-SMA (Type : p - paramètre, v - variable, f - fonction)

6.3.1.1/ ALGORITHMES

Cette section présente de manière plus détaillée les comportements des agents d'ESCBR-SMA.

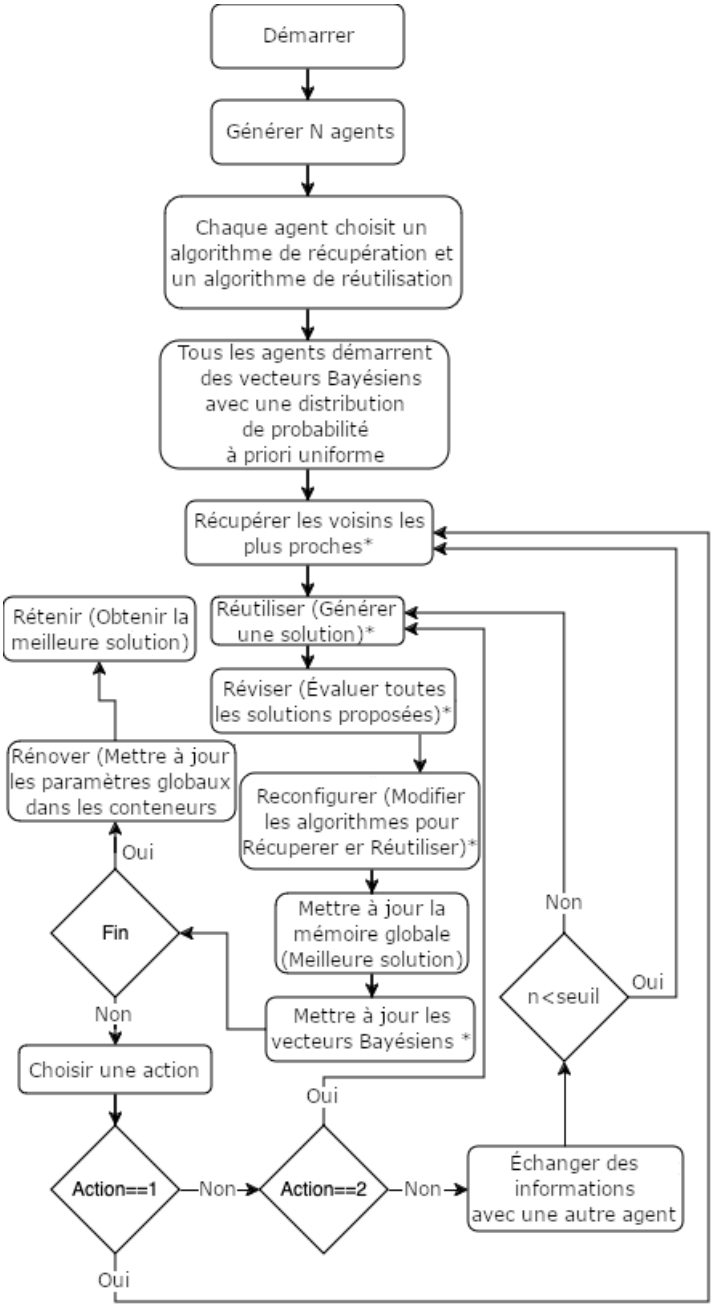

FIGURE 6.10 – Flux du *Stacking* du RàPC (* : tâche effectuée par chaque agent)

Lors de la première étape de l'empilement, chaque agent peut sélectionner l'un des algorithmes suivants pour rechercher des problèmes similaires au problème cible : KNN (*K-Nearest Neighbor*), *KMeans*, GMM (*Gaussian Mixture Model*), *FuzzyC* et KNN pondéré.

Lors de la deuxième étape de l'empilement, les agents peuvent choisir un algorithme parmi les algorithmes explicités dans la section 6.2.1.2.

6.3.1.2/ STRUCTURE DES AGENTS

Tous les agents ont une structure similaire, mais chaque agent suit un processus cognitif individuel qui lui permet d'adopter un comportement indépendant et différent de tous les autres. La figure 6.11 montre les actions et les variables nécessaires à l'exécution de l'ensemble du processus.

Chaque agent peut exécuter trois actions différentes :
- « Récupérer et réutiliser »,
- « Réutiliser » et
- « Échanger ».

Chaque agent accède aux valeurs de huit variables qui lui sont propres :
- le nombre de voisins définissant le nombre de problèmes similaires au nouveau problème posé que l'agent doit rechercher dans la base de connaissances,
- la liste des voisins les plus proches consigne les agents avec lesquels des informations peuvent être échangées,
- l'identifiant de l'algorithme de récupération que l'agent exécutera pour trouver les problèmes similaires au problème cible,
- l'identifiant de l'algorithme de réutilisation que l'agent exécutera pour générer une solution candidate au cas cible,
- la description de la solution générée,
- l'évaluation de la solution renseignant sur la qualité de la solution générée et calculée selon une fonction d'optimisation (équations 6.13 et 6.14),
- le vecteur bayésien nécessaire aux algorithmes de récupération contenant les valeurs de probabilité, et
- le vecteur bayésien nécessaire aux algorithmes de réutilisation contenant les valeurs de probabilité.

$$min\ (f_s(p_w^n, s_w^m)) = min\left(\sum_{i=1}^{ng} \frac{d(s_w^m, s_i^t)}{d(p_w^n, p_i^n)^2}\right) \tag{6.13}$$

$$s_i^t = s_i^m + rn(0, d(p_w^n, p_i^n)) \tag{6.14}$$

6.3.1.3/ APPRENTISSAGE DES AGENTS

Tous les agents considèrent un vecteur bayésien de probabilité pour la phase de récupération (conteneur $C3$ de la figure 6.9) et un vecteur bayésien de probabilité pour la phase de réutilisation (conteneur $C2$). Initialement, ces vecteurs sont configurés avec une probabilité uniforme pour tous les algorithmes de récupération et de réutilisation (probabilité *a priori*). À chaque itération, les vecteurs sont mis à jour selon l'équation bayésienne 6.15 en utilisant les meilleurs résultats du même agent comme paramètre de vraisemblance. L'agent apprend ainsi de son expérience et sélectionne les algorithmes les plus aptes à fournir les meilleures réponses au regard de l'objectif défini.

Par conséquent, d'un point de vue plus global au niveau du SMA, l'apprentissage est fondé sur un raisonnement bayésien où les vecteurs de récupération et de réutilisation évoluent. Un exemple d'évolution est présenté sur la figure 6.12. Au cours d'une itération, si un algorithme a contribué à la construction de la meilleure solution, il reçoit un point pour chaque agent qui l'a utilisé. Ces informations sont stockées dans un vecteur

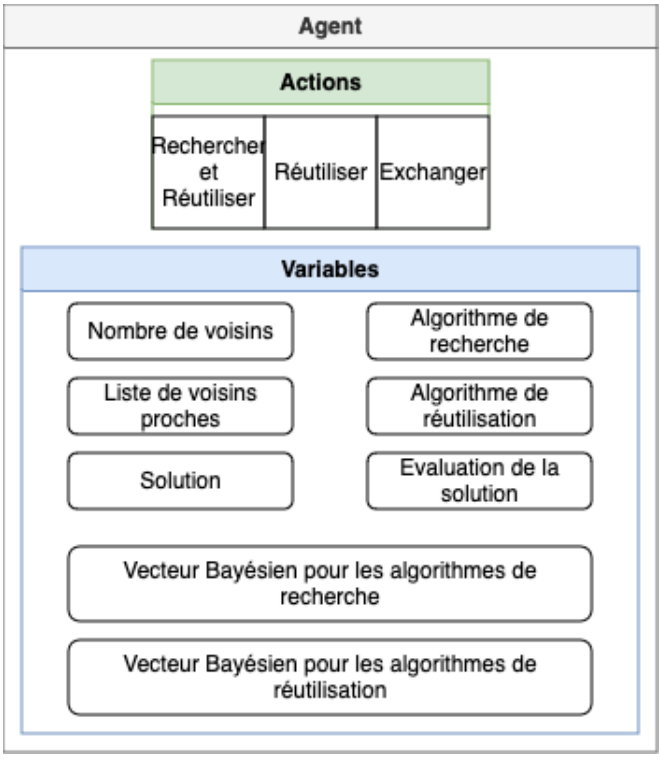

FIGURE 6.11 – Structure interne des agents

normalisé qui est ensuite utilisé comme vecteur de vraisemblance $P(A)$ pour calculer les nouvelles probabilités dans l'équation bayésienne 6.15 (dans cette équation, $P(B)$ est le terme de normalisation global).

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{6.15}$$

La fonction rnp de l'équation 6.16 choisit l'algorithme de recherche a_{rt} de l'étape rt. Cette fonction prend en paramètres l'ensemble des algorithmes possibles n_{rt} et une probabilité associée $P(A|B)_{rt}$. $P(A|B)_{rt}$ représente la probabilité que l'algorithme de recherche associé fournisse la solution optimale à l'étape rt.

$$a_{rt} = rnp(n_{rt}, P(A|B)_{rt}) \tag{6.16}$$

La fonction rnp est de nouveau utilisée pour choisir l'algorithme de réutilisation a_{rs} (équation 6.17).

$$a_{rs} = rnp(n_{rs}, P(A|B)_{rs}) \tag{6.17}$$

Le processus d'apprentissage est réalisé individuellement par chaque agent, mais comporte un aspect collaboratif puisque les agents échangent les informations qu'ils ont calculées, les optimisations locales qu'ils ont réalisées, ainsi que les algorithmes et les paramètres qu'ils ont utilisés.

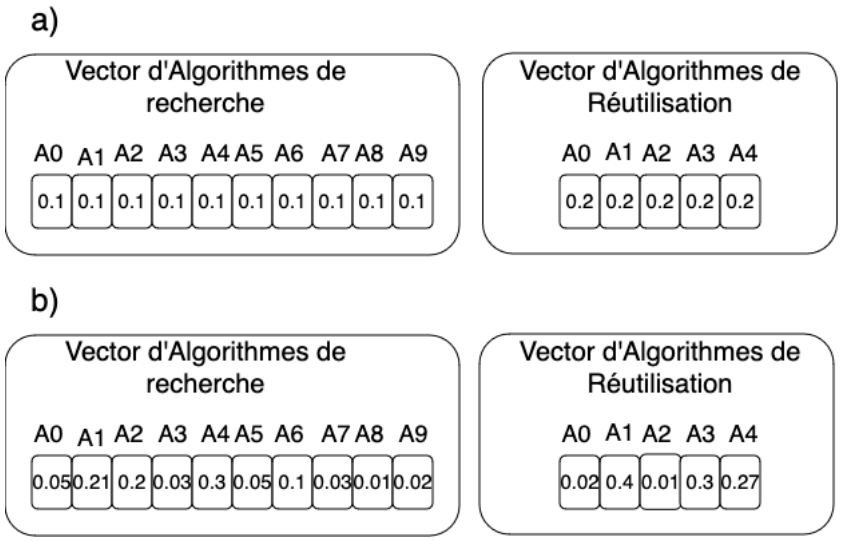

FIGURE 6.12 – Exemple d'évolution Bayésienne des vecteurs pour un agent. a) Initialisation des probabilités $P(B|A)$ vecteurs pour Retrieve et Reuse, b) Probabilités après quelques itérations $P(A|B)$ vecteurs pour Retrieve et Reuse

6.3.1.4/ ÉCHANGES ENTRE LES AGENTS

Un agent peut modifier ses informations et leur affecter les valeurs de celles d'un voisin au cours de chaque itération en choisissant au hasard un voisin dans sa liste de voisins les plus proches. L'ensemble de ces changements permet de propager les paramètres menant aux meilleurs résultats et d'effectuer des actions rétrospectivement. Les informations modifables sont choisies aléatoirement : il peut s'agir du nombre de voisins, de la liste des voisins les plus proches, de l'algorithme de récupération, de l'algorithme de réutilisation ou même des vecteurs bayésiens.

6.3.2/ RÉSULTATS

Des expérimentations sur onze jeux de données relatifs à des problèmes de régression et présentant des caractéristiques différentes ont été réalisées. Les jeux de données et leurs caractéristiques sont consignés dans le tableau 6.7. Les valeurs des paramètres d'ESCBR-SMA sont les suivantes : $it = 100$ itérations, $np = 50$ agents, $nl = 10$ voisins par agent au maximum lors de l'étape de *niveau-0* et $ng = 10$ voisins par agent au maximum lors de l'étape de *niveau-1*.

Le tableau 6.8 présente les valeurs des paramètres de tous les autres algorithmes.

Les tableaux 6.9 et 6.10 présentent respectivement les RMSE et les MAE obtenues pour chaque jeu de données et chaque algorithme testé.

La figure 6.13 représente graphiquement la dispersion globale, la médiane et les valeurs aberrantes obtenues durant ces tests. Cette figure montre qu'ESCBR-SMA génère parfois plus de valeurs aberrantes que d'autres algorithmes. Toutefois, la variance est très faible. De plus, la convergence est proche de la valeur réelle et même meilleure que celle de la plupart des autres algorithmes testés. Il est également possible de remarquer que les valeurs aberrantes sont plus nombreuses que les valeurs réelles.

ID	DataSet	Features	Instances	Output. Dimension	Input. Domain	Output Domain
DS1	Yatch Hydrodynamics	6	308	1	$\mathbb{R}$	$\mathbb{R}$
DS2	Electrical Grid Stability	12	10000	1	$\mathbb{R}$	$\mathbb{R}$
DS3	Real State Valuation	6	414	1	$\mathbb{R}_+$	$\mathbb{R}_+$
DS4	Wine Quality (Red)	11	1598	1	$\mathbb{R}_+$	$\mathbb{N}$
DS5	Wine Quality (White)	11	4897	1	$\mathbb{R}_+$	$\mathbb{N}$
DS6	Concrete Compressive Strength	8	1030	1	$\mathbb{R}_+$	$\mathbb{R}_+$
DS7	Energy Efficiency	8	768	2	$\mathbb{R}_+$	$\mathbb{R}_+$
DS8	Gas Turbine CO, NOx Emission (2015)	9	7384	2	$\mathbb{R}_+$	$\mathbb{R}_+$
DS9	Student Performace Portuguese	30	649	3	$\mathbb{N}*$	$\mathbb{N}$
DS10	Student Performance Math	30	395	3	$\mathbb{N}*$	$\mathbb{N}$
DS11	Generated Student Performance	5	1000	1	$\mathbb{R}_+$	$\mathbb{R}_+$

TABLE 6.7 – Description des jeux de données évalués.

ID	Parameter	Value	ID	Parameter	Value
A1	Intercept	True	A6	Degree	4
	Positive	True		Bias	True
A2	Neighbors	5	A7	Fit Intercept	True
	Weights	Uniform		alpha	0.2
	Metric	Minkowsky		tol	1e-4
	Power Minkowsky	2			
A3	Error	Squared Error	A8	Fit Intercept	True
	Min samples split	2		alpha	[0.00001, 0.4]
				Max iter	1000
				tol	1e-4
A4	Estimators	10	A9	Error	Squarred Error
	Error	Squared Error		Learning Rate	0.1
	Min samples split	2		Estimators	100
	Bootstrap	True		Min Split	2
A5	Hidden Layers	100			
	Activation	Relu			
	Solver	Adam			
	alpha	0.0001			
	Learning Rate	0.001			
	Max Iter	200			
	beta1	0.9			
	beta2	0.999			
	epsilon	1e-8			

TABLE 6.8 – Paramètres de tous les algorithmes comparés

Les résultats montrent que l'algorithme ESCBR-SMA est compétitif pour la plupart des jeux de données considérés dans cette étude. Il obtient en effet les meilleurs résultats sur $DS2$, $DS4$, $DS5$ et $DS9$. Globalement, ESCBR-SMA est le troisième meilleur algorithme. Ses performances sont donc comparables à celles des autres algorithmes d'ensemble testés. Par rapport à ESCBR, une amélioration des résultats et une réduction de la variance ont été obtenues, démontrant ainsi que les systèmes multi-agents et le raisonnement stochastique bayésien contribuent à l'apprentissage et à la convergence vers des solutions plus proches de la solution réelle.

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
DS1	9.081	12.301	1.228	1.066	7.763	9.081	9.078	9.764	0.750	8.225
DS2	0.022	0.025	0.019	0.013	0.017	0.022	0.022	0.037	0.011	0.016
DS3	8.756	8.465	9.656	7.665	8.716	8.756	9.005	9.177	7.369	7.991
DS4	0.647	0.752	0.794	0.602	0.688	0.647	0.646	0.798	0.616	0.607
DS5	0.767	0.824	0.877	0.66	0.826	0.767	0.775	0.87	0.703	0.662
DS6	10.525	9.174	6.93	5.372	6.662	10.525	10.525	10.527	5.131	9.070
DS7	2.961	2.451	0.589	0.528	3.955	2.961	3.009	4.083	0.490	2.941
DS8	1.298	1.125	1.360	1.197	1.486	1.298	1.298	1.306	1.128	2.553
DS9	2.256	2.565	3.174	2.377	2.817	2.256	2.255	2.468	2.293	2.468
DS10	3.136	3.415	4.173	3.165	3.710	3.136	3.135	3.161	3.108	3.621
DS11	0.625	0.565	0.741	0.56	0.606	0.626	0.626	0.681	0.541	0.54
Avg. Rank	6.4	6.9	8.2	2.6	7.2	6.45	6.35	9.55	2.1	4.75

TABLE 6.9 – Résultat selon la métrique RMSE (Root Mean Squared Error) pour les jeux de données évalués avec les différents algorithmes de régression considérés.

Dataset	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
DS1	6.776	2.385	0.231	0.207	3.632	6.778	6.307	5.186	0.162	1.218
DS2	0.015	0.017	0.012	0.008	0.012	0.015	0.015	0.030	0.007	0.010
DS3	5.092	4.320	4.1	3.632	4.435	5.092	5.20	5.132	3.504	3.771
DS4	0.413	0.495	0.18	0.325	0.451	0.413	0.412	0.544	0.387	0.135
DS5	0.509	0.548	0.285	0.374	0.550	0.509	0.509	0.633	0.456	0.085
DS6	6.989	5.709	3.134	2.839	4.306	6.989	6.989	6.986	3.084	5.072
DS7	1.393	1.372	0.217	0.218	2.523	1.393	1.529	2.346	0.243	1.006
DS8	0.549	0.297	0.365	0.289	0.742	0.549	0.549	0.540	0.309	0.794
DS9	1.496	1.788	2.080	1.612	2.005	1.496	1.496	1.714	1.538	1.556
DS10	2.344	2.534	2.910	2.331	2.543	2.344	2.344	2.481	2.258	2.371
DS11	0.387	0.35	0.46	0.338	0.384	0.387	0.387	0.453	0.327	0.347
Avg. Rank	7.15	6.9	5.35	2.6	7.95	7.25	7.3	9.0	2.5	4.5

TABLE 6.10 – Comparaison des résultats MAE (Median Absolute Error) pour les bases de données évaluées avec des algorithmes de régression

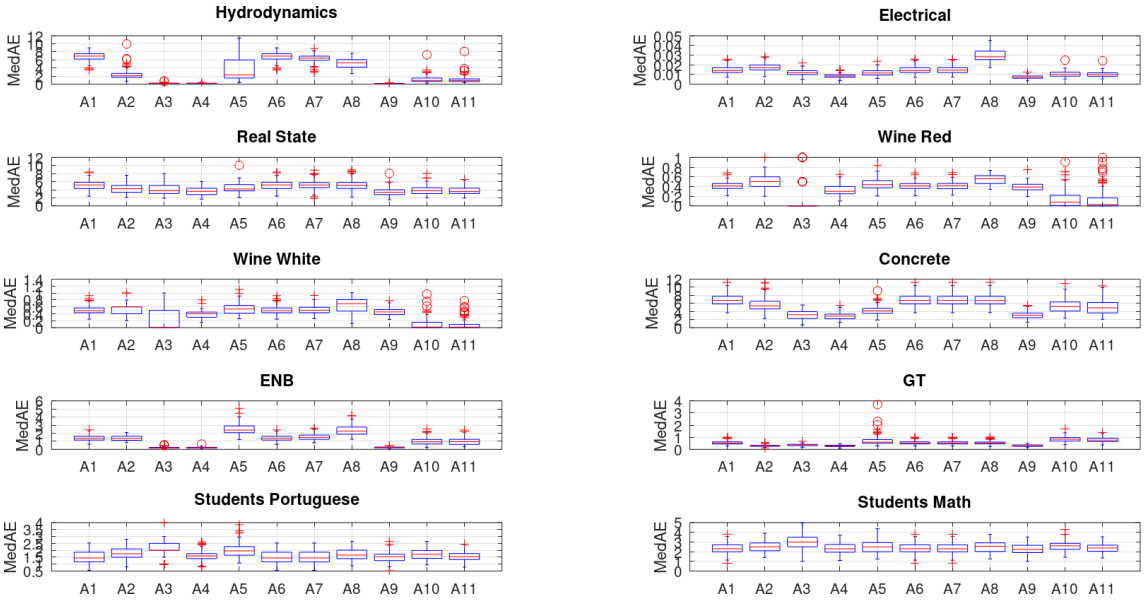

FIGURE 6.13 – Résultats selon la métrique MAE (Median Absolute Error) pour les dix algorithmes testés

6.3.3/ CONCLUSION

Cette version d'ECBR intégrant un SMA propose d'utiliser les avantages des systèmes multi-agents pour améliorer la qualité des solutions proposées et également le processus d'apprentissage global en couvrant un plus large spectre de possibilités et en le couvrant de manière intelligente à l'aide d'un raisonnement bayésien. ESCBR-SMA permet ainsi d'obtenir de bonnes approximations avec peu de données.

Ce travail a démontré la capacité d'ESCBR-SMA à trouver des solutions proches de l'optimum global pour la majorité des ensembles de données analysés. Ces jeux de données présentent une certaine diversité, ils peuvent être déséquilibrés et ils sont de tailles différentes. Grâce aux caractéristiques inhérentes aux systèmes multi-agents (possibilités de rétroactions, d'échanges d'informations, l'émergence d'une intelligence collective, l'utilisation de raisonnements cognitifs), les performances d'ESCBR-SMA sont meilleures que celles d'ESCBR.

Dans le chapitre suivant, nous explorons la possibilité d'intégrer de nouveaux outils à ESCBR-SMA pour la recommandation d'exercices en cours de séance d'entraînement dans l'EIAH AI-VT.

7

SYSTÈME DE RECOMMANDATION DANS AI-VT

7.1/ INTRODUCTION

L'un des principaux modules d'un EIAH est le système de recommandation visant à trouver les faiblesses et à réviser la séance d'entraînement proposée initialement par celui-ci. Ce type de module permet donc au système de personnaliser les contenus et les exercices en fonction des besoins et des résultats de chacun des apprenants. Certains auteurs n'hésitent pas à considérer que l'efficacité d'un EIAH dans l'acquisition des connaissances et l'adaptation aux différents types d'apprentissage dépend de ce type de module fondé sur la recommandation [Liu and Yu, 2023].

Les systèmes de recommandation dans les environnements d'apprentissage prennent en compte les exigences, les besoins, le profil, les acquis, compétences, les intérêts et l'évolution de l'apprenant pour adapter et recommander des ressources ou des exercices. Dans ces systèmes, l'adaptation peut être de deux types : l'adaptation de la présentation qui montre aux apprenants des ressources d'étude en fonction de leurs faiblesses et/ou l'adaptation du parcours qui change la structure du cours en fonction du niveau et du style d'apprentissage de chaque apprenant [Muangprathub et al., 2020].

Parmi les algorithmes les plus prometteurs pouvant aider à proposer des recommandations, nous avons identifié l'algorithme d'échantillonnage de Thompson (TS). Il s'agit d'un algorithme probabiliste appartenant à la catégorie des algorithmes d'apprentissage par renforcement. À l'instant t, TS choisit l'action a_t d'un ensemble A d'actions possibles, et obtient une récompense pour celle-ci. À $t+1$, une action a_{t+1} est sélectionnée en tenant compte de la récompense précédente. L'objectif consiste à maximiser la récompense. Selon le principe bayésien, cette maximisation itérative est opérée en suivant une distribution de probabilité évoluant à chaque itération. Cette évolution peut être calculée selon la variante de Bernoulli où la récompense n'a que deux valeurs possibles 0 ou 1 (échec ou succès), ou selon une distribution *Beta* définie sur l'intervalle $[0, 1]$ et calculée en fonction de deux valeurs α et β [Lin, 2022].

Ce chapitre est divisé en trois parties, la première partie présente un algorithme délivrant des recommandations en fonction des résultats produits par l'apprenant en temps réel. Une partie de cette proposition est publiée dans [Soto-Forero et al., 2024a]. Cet algorithme permet l'adaptation automatique en temps réel d'une séance prédéterminée dans l'EIAH AI-VT. Nous considérons qu'elle intervient durant la phase de révision du cycle

classique du raisonnement à partir de cas (RàPC). L'algorithme proposé est stochastique et il a été testé selon trois scénarios différents. Les résultats montrent de quelle manière AI-VT peut proposer des recommandations pertinentes selon les faiblesses identifiées de l'apprenant.

La deuxième partie de ce chapitre montre l'intégration de tous les algorithmes présentés dans les chapitres précédents à AI-VT. L'algorithme intégré est appliqué au système AI-VT sur des données générées et des données réelles. Plusieurs types de test sont exécutés pour montrer que l'algorithme final permet en effet d'améliorer les capacités d'identification et d'adaptation. Les performances de ce nouveau algorithme sont comparées à celles d'autres algorithmes. Enfin, l'évolution de l'acquisition des connaissances induites par ces nouveaux algorithmes de recommandation stochastiques est analysée dans cette deuxième partie du présent chapitre.

Pour terminer, dans la troisième partie de ce chapitre, nous présentons une évolution de ce système de recommandation intégrant le processus de Hawkes. L'intérêt de ce dernier réside dans le fait qu'il utilise une courbe d'oubli, nous permettant ainsi de tenir compte du fait que certaines connaissances et certains mécanismes doivent être rappelés aux apprenants. Cette troisième partie intègre une étude des performances du système de recommandation incluant ce processus stochastique de Hawkes.

7.2/ SYSTÈME DE RECOMMANDATION STOCHASTIQUE FONDÉ SUR L'ÉCHANTILLONNAGE DE THOMPSON

7.2.1/ ALGORITHME PROPOSÉ

L'algorithme proposé, en tant que système de recommandation, prend en compte les notes antérieures des apprenants pour estimer leurs connaissances et leur maîtrise des différentes compétences, sous-compétences et niveaux de complexité au sein du système AI-VT. Puis il adapte les séances pour maximiser l'acquisition des connaissances et la maîtrise des différents domaines contenus dans la même compétence définie.

La famille de distributions de probabilité Beta est utilisée pour définir dynamiquement le niveau de complexité (équation 7.1) à proposer à l'apprenant. Cet algorithme permet de recommander des niveaux de complexité non contigus et dans lesquels des lacunes ont été détectées. Les paramètres initiaux des distributions de probabilité peuvent forcer le système à recommander des niveaux de complexité contigus (juste inférieur ou supérieur).

$$B(x,\alpha,\beta) = \begin{cases} \frac{x^{\alpha-1}(1-x)^{\beta-1}}{\int_0^1 u^{\alpha-1}(1-u)^{\beta-1}du} & si\ x \in [0,1] \\ 0 & sinon \end{cases} \tag{7.1}$$

Le tableau 7.1 présente les variables de l'algorithme de recommandation. Nous avons considéré que les notes g_c obtenues au niveau de complexité c tiennent compte du temps de réponse. C'est la raison pour laquelle, nous avons défini le grade de l'apprenant ng_c au niveau de complexité c. Ce grade calculé selon l'équation 7.2, tient compte d'un poids de pénalisation temporelle λ.

ID	Description	Domaine
c_n	Niveaux de complexité	$\mathbb{N} \mid c_n > 0$
g_m	Valeur maximale dans l'échelle des notes	$\mathbb{N} \mid g_m > 0$
g_t	Seuil de notation	$(0, g_m) \in \mathbb{R}$
s	Nombre de parcours définis	$\mathbb{N} \mid s > 0$
s_c	Parcours courant fixe défini	$[1, s] \in \mathbb{N}$
Δs	Pas pour les paramètres de la distribution bêta dans le parcours s	$(0, 1) \in \mathbb{R}$
t_m	Valeur maximale du temps de réponse	$\mathbb{R} \mid t_m > 0$
g_c	Note de l'apprenant à une question de complexité c	$[0, g_m] \in \mathbb{R}$
ng_c	Grade de l'apprenant avec pénalisation du temps	$[0, g_m] \in \mathbb{R}$
t_c	Le temps de réponse à une question de complexité c	$[0, t_m] \in \mathbb{R}$
ncl	Nouveau niveau de complexité calculé	$\mathbb{N}$
α_c	Valeur de α dans la complexité c	$\mathbb{R} \mid \alpha_c > 0$
β_c	Valeur de β dans la complexité c	$\mathbb{R} \mid \beta_c > 0$
$\Delta\beta$	Pas initial du paramètre bêta	$\mathbb{N} \mid \Delta\beta > 0$
λ	Poids de la pénalisation temporelle	$(0, 1) \in \mathbb{R}$
G_c	Ensemble de d notes dans le niveau de complexité c	$\mathbb{R}^d, d \in \mathbb{N} \mid d > 0$
x_c	Notes moyennes normalisées	$[0, 1] \in \mathbb{R}$
n_c	Nombre total de questions dans une séance	$\mathbb{N} \mid n_c > 0$
ny_c	Nombre de questions dans le niveau de complexité c	$\mathbb{N} \mid 0 < ny_c \leq n_c$
y_c	Proportion de questions dans le niveau de complexité c	$[0, 1] \in \mathbb{R}$
r	Valeur totale de la métrique définie pour l'adaptabilité	$[0, c_n] \in \mathbb{R}$
sc	Valeur totale de la métrique de similarité cosinus	$[-1, 1] \in \mathbb{R}$

TABLE 7.1 – Variables et paramètres du système de recommandation proposé

$$ng_c = g_c - \left(g_c * \lambda * \frac{t_c}{t_m} \right) \tag{7.2}$$

Dans cet algorithme, la variable de seuil de grade g_t détermine la variabilité de la distribution de probabilité pour chaque niveau de complexité. Les niveaux de complexité des exercices proposés à l'apprenant sont calculés par récompense inverse selon les équations 7.3 et 7.4. Chaque niveau de complexité est associé à une distribution de probabilité Beta avec des valeurs initiales α et β prédéfinies.

$$ng_c \geq g_t \rightarrow \begin{cases} \beta_c = \beta_c + \Delta_s \\ \beta_{c-1} = \beta_{c-1} + \frac{\Delta_s}{2} \\ \alpha_{c+1} = \alpha_{c+1} + \frac{\Delta_s}{2} \end{cases} \tag{7.3}$$

$$ng_c < g_t \rightarrow \begin{cases} \alpha_c = \alpha_c + \Delta_s \\ \alpha_{c-1} = \alpha_{c-1} + \frac{\Delta_s}{2} \\ \beta_{c+1} = \beta_{c+1} + \frac{\Delta_s}{2} \end{cases} \tag{7.4}$$

Pour chaque niveau de complexité c, $Beta(\alpha_c, \beta_c)$ fournit une distribution de probabilité θ_c dont nous calculons l'espérance $\mathbb{E}[\theta_c]$. Le nouveau niveau de complexité ncl correspond à l'espérance maximale obtenue.

Le détail des pas d'exécution de l'algorithme proposé sont dans l'algorithme 1.

Algorithm 1 Algorithme de recommandation stochastique

Initialisation de la distribution de probabilité
for each question q **do**
  Soit le niveau de complexité i
  $ng_i = g_i - \left(g_i * \lambda * \frac{t_i}{t_m}\right)$ ▷ eq 7.2
  Calculs des paramètres α_i et β_i ▷ eq 7.3 et eq 7.4
  Choisir θ_c selon la distribution de probabilité Beta ▷ $\forall c, \theta_c = Beta(\alpha_c, \beta_c)$
  $ncl = max(\mathbb{E}[\theta_c]), \forall c$
end for

7.2.2/ RÉSULTATS

Le comportement du module de recommandation a été testé avec des données générées contenant les notes et les temps de réponse de mille apprenants pour cinq niveaux de complexité différents. Ces données sont décrites dans le tableau 7.2. Les notes des apprenants sont générées selon la loi de probabilité logit-normale considérée comme la plus fidèle dans ce contexte par [Arthurs et al., 2019].

L'ensemble de données générées résulte d'une simulation des notes obtenues par des apprenants virtuels ayant répondu à quinze questions réparties sur cinq niveaux de complexité. L'ensemble de données simule, via la distribution de probabilité logit-normale, une faiblesse dans chaque niveau de complexité pour 70% des apprenants sur les dix premières questions. La difficulté de la complexité est quant à elle simulée en réduisant le score moyen et en augmentant la variance. La figure 7.1 montre la manière dont sont réparties les notes selon le niveau de complexité.

ID	Description	Domaine
q_c	Niveau de complexité de une question q	$[0, c_n] \in \mathbb{N}$
$q_{g,c}$	Note obtenue g pour la question q avec complexité c	$[0, g_m] \in \mathbb{R}$
$q_{t,c}$	Temps employé t pour une question q avec complexité c	$[0, t_m] \in \mathbb{R}$

TABLE 7.2 – Description des données utilisées pour l'évaluation.

Toutes les valeurs des paramètres pour tester l'algorithme sont dans le tableau 7.3.

ID	c_n	g_m	t_m	s	s_c	λ	g_t	$\alpha_{x,1}$	$\alpha_{x,y}$	$\beta_{x,1}$	$\Delta\beta_{x,y}$	Δ_1	Δ_2	Δ_3
Valeur	5	10	120	3	2	0.25	6	2	1	1	1	0.3	0.5	0.7

TABLE 7.3 – Valeurs des paramètres pour les scénarios évalués

La figure 7.2 permet de comparer les résultats obtenus par le module proposé, un système de recommandation déterministe et le système AI-VT initial lors d'un *démarrage à froid* (c'est-à-dire sans données historiques ni informations préalables sur le profil de l'apprenant). Sur les graphiques de cette figure, les numéros des questions posées sont reportées en abscisse selon l'ordre chronologique d'apparition durant la séance d'entraînement, le niveau de complexité de chaque question posée est représenté par une couleur différente, et le nombre d'apprenants ayant eu des questions de ce niveau de complexité sont reportés en ordonnées. Ainsi, le système AI-VT initial (premier graphique de la figure) et le système de recommandation déterministe (deuxième graphique) ont tous deux proposé trois questions de niveau de complexité 0 (le plus faible) à tous les

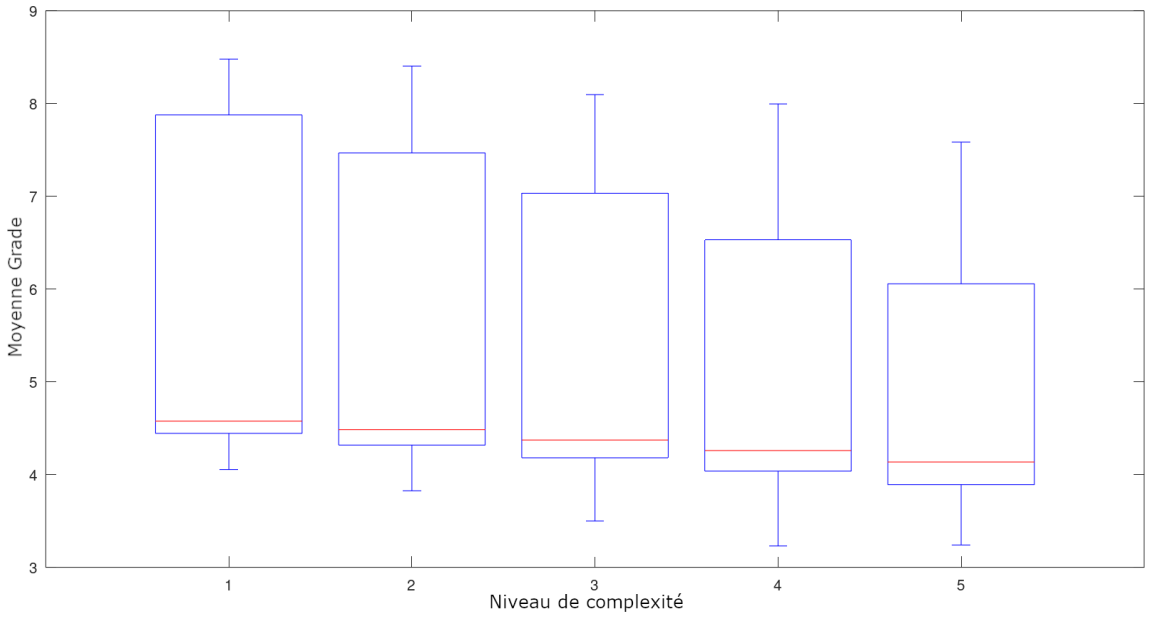

FIGURE 7.1 – Répartition des notes générées selon le niveau de complexité.

apprenants au démarrage de la séance d'entrainement. Nous pouvons remarquer que le système initial est resté sur ce niveau de complexité durant toute la séance (pour les 15 questions du test), tandis que le système de recommandation déterministe a progressivement mixé les complexités des questions posées. Le système de recommandation stochastique décrit dans ce chapitre a quant à lui mixé ces niveaux de complexité dès la première question.

Ainsi, les systèmes de recommandation permettent de proposer une adaptation progressive du niveau de complexité en fonction des notes obtenues. L'algorithme déterministe génère quatre grandes transitions avec un grand nombre d'apprenants dans les questions 5, 6, 8 et 12, toutes entre des niveaux de complexité contigus. La tendance est à la baisse pour les niveaux 0, 1 et 2 après la huitième question et à la hausse pour les niveaux 1 et 3. L'algorithme stochastique commence par proposer tous les niveaux de complexité possibles tout en privilégiant le niveau 0. Avec ce système, les transitions sont constantes mais pour un petit nombre d'apprenants. La tendance après la dixième question est à la baisse pour les niveaux 0 et 4 et à la hausse pour les niveaux 1, 2 et 3.

Après la génération de la première séance, le système peut continuer avec une deuxième liste d'exercices. Pour cette partie des tests, les trois algorithmes ont été initialisés avec les mêmes données, et des valeurs égales pour tous les apprenants. La figure 7.3 permet de voir que la première transition du système initial n'intervient qu'entre deux séances. Les transitions sont très lentes, et tous les apprenants doivent suivre un chemin identique même s'ils obtiennent des notes différentes au cours de celle-ci.

Pour leur part, les deux autres systèmes de recommandation testés proposent un fonctionnement différent. L'algorithme déterministe présente trois transitions aux questions 3, 5 et 12. Les tendances sont y relativement homogènes et progressives pour le niveau 3, très variables pour le niveau 2 et fortement décroissantes pour le niveau 0. L'algorithme stochastique quant à lui, propose des transitions douces mais il a tendance à toujours privilégier le niveau le plus faible. Nous pouvons observer une prépondérance du niveau 1 avec ce système. Ici, les niveaux 0 et 1 sont décroissants, le niveau 2 est statique et

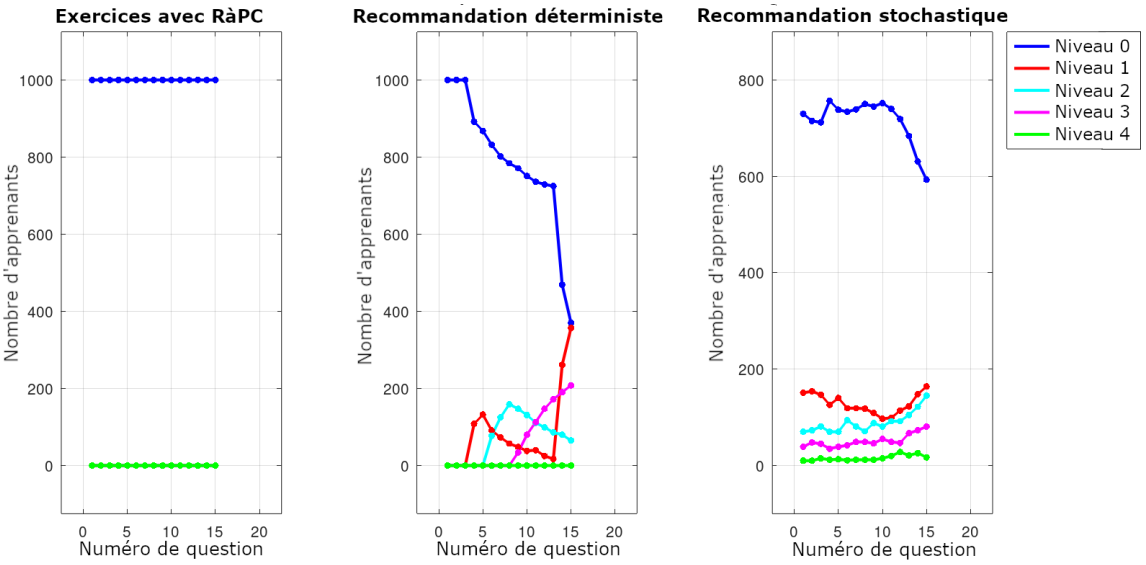

FIGURE 7.2 – Niveaux de complexité des questions posées aux apprenants par les trois systèmes testés lors de la première séance avec un démarrage à froid (sans données initiales sur les apprentants).

les niveaux 3 et 4 sont ascendants.

Les questions de la première et la deuxième séance étant de niveaux 0 et 1, le système a proposé des niveaux de complexité 1 ou 2 pour la troisième séance. La figure 7.4 montre que le système initial est très lent à passer d'un niveau à l'autre. La figure 7.3 permet de voir la première transition du système initial ne réagissant qu'aux notes obtenues dans les séances précédentes et non à celles de la séance en cours. Dans ce cas, l'algorithme de recommandation déterministe adopte la même stratégie et propose un changement brutal à tous les apprenants autour de la cinquième question. L'algorithme stochastique continue avec des changements progressifs tout en privilégiant le niveau 2.

Pour comparer numériquement le système initial, l'algorithme déterministe et l'algorithme de recommandation proposé, un ensemble d'équations a été défini (équation 7.5 et équation 7.6). Celles-ci permettent de décrire le système de recommandation idéal si l'objectif de l'apprenant est de suivre un apprentissage standard. Une valeur est calculée pour chaque niveau de complexité en fonction de la moyenne des notes et du nombre de questions recommandées dans ce niveau de complexité. L'objectif de cette mesure est d'attribuer un score élevé aux systèmes de recommandation qui proposent plus d'exercices au niveau de complexité où l'apprenant a obtenu une note moyenne plus basse, lui permettant ainsi de renforcer ses connaissances pour ce niveau de complexité. De la même manière, il est attendu que le système de recommandation propose moins d'exercices aux niveaux de complexité pour lesquels les notes moyennes sont élevées, l'étudiant ayant acquis des connaissances suffisantes à ces niveaux de complexité. Les scores faibles sont attribués aux systèmes qui recommandent peu d'exercices à des niveaux de complexité dont les notes moyennes sont faibles et, inversement, s'ils proposent beaucoup d'exercices à des niveaux de complexité dont les notes moyennes sont élevées.

$$rp_c(x) = e^{-2(x_{0,c}+x_{1,c}-1)^2}; \{x \in \mathbb{R}^2 | 0 <= x <= 1\} \tag{7.5}$$

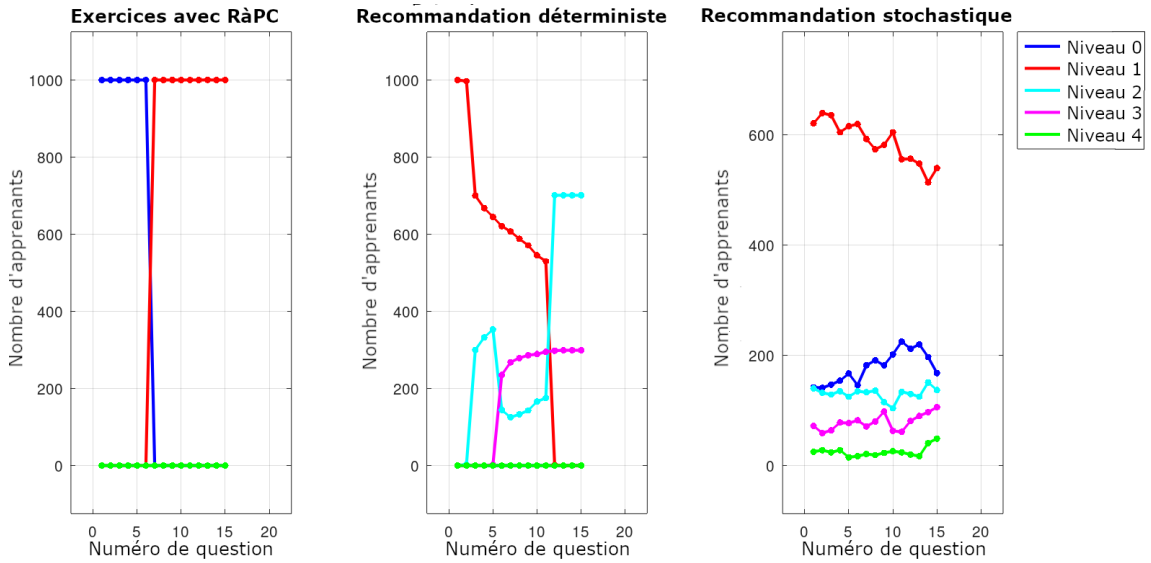

FIGURE 7.3 – Niveaux de complexité des questions posées aux apprenants par les trois systèmes testés lors de la deuxième séance.

$$r = \sum_{c=0}^{c_n-1} rp_c \tag{7.6}$$

Les propriétés de la métrique sont :

— $\{\forall x \in \mathbb{R}^2 | 0 <= x <= 1\}, rp_c(x) > 0$
— $max(rp_c(x)) = 1;\ if\ x_{0,c} + x_{1,c} = 1$
— $min(rp_c(x)) = 0.1353;\ if\ \left(\sum_{i=1}^{2} x_{i,c} = 0\ \vee\ \sum_{i=1}^{2} x_{i,c} = 2\right)$

Dans l'équation 7.5, $x_{0,c}$ est la moyenne normalisée des notes dans le niveau de complexité c (équation 7.7), et $x_{1,c}$ est le nombre normalisé de questions auxquelles des réponses ont été données dans le niveau de complexité c (équation 7.8). Ainsi, plus la valeur de r est élevée, meilleure est la recommandation.

$$x_{0,c} = \frac{< g_c >_{G_c}}{g_m} \tag{7.7}$$

$$x_{1,c} = \frac{ny_c}{n_c} \tag{7.8}$$

La figure 7.5 représente la fonction $rp_c(x)$. La valeur maximale de r dans un niveau de complexité spécifique étant égale à 1, la valeur maximale globale pour les scénarios testés est égale à 5.

Les résultats des calculs de la métrique $rp_c(x)$ établie pour le système initial et les deux algorithmes dans les trois scénarios testés sont présentés dans le tableau 7.4.

Les équations 7.9 et 7.10 permettent de caractériser un apprentissage progressif. Dans ce cas, un score élevé est attribué aux systèmes proposant plus d'exercices dans un niveau de complexité où les notes moyennes sont légèrement insuffisantes (4/10), plus flexibles avec des notes moyennes plus basses, et un petit nombre d'exercices pour

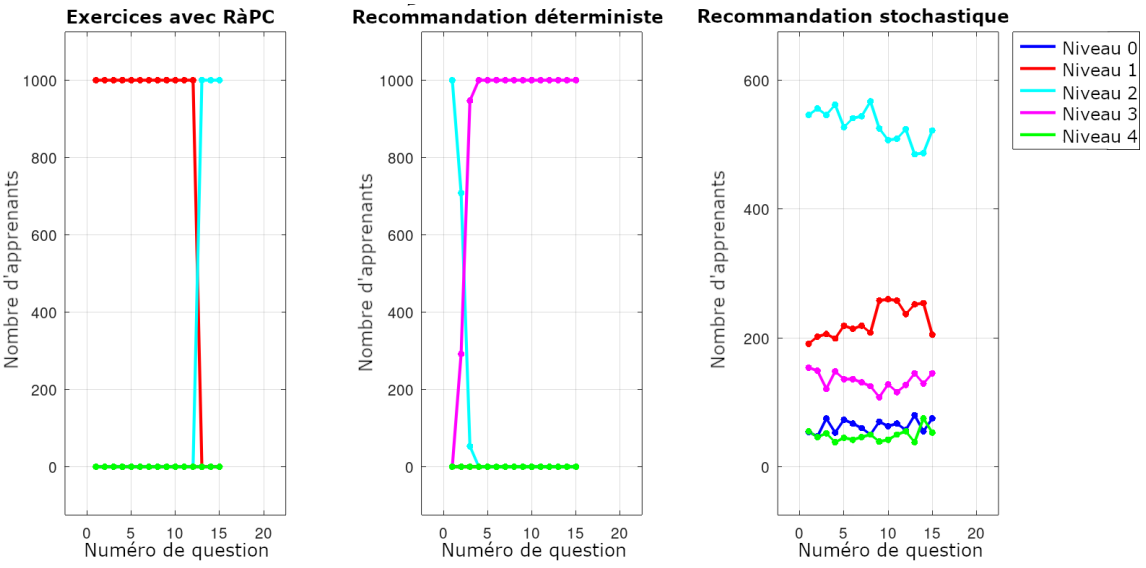

FIGURE 7.4 – Niveaux de complexité des questions posées aux apprenants par les trois systèmes testés lors de la troisième séance.

	c_0	c_1	c_2	c_3	c_4	Total (r)	Total (%)
Test 1							
RàPC	0.5388	-	-	-	-	0.5388	10.776
DM	0.8821	0.7282	**0.9072**	**0.8759**	-	3.3934	67.868
SM	**0.9463**	**0.8790**	0.7782	0.7108	0.6482	**3.9625**	**79.25**
Test 2							
RàPC	0.9445	**0.9991**	-	-	-	1.9436	38.872
DM	-	0.9443	**0.8208**	**0.9623**	-	2.7274	54.548
SM	**0.9688**	0.9861	0.8067	0.7161	0.6214	**4.0991**	**81.982**
Test3							
RàPC	-	0.8559	0.7377	-	-	1.5936	31.872
DM	-	-	0.5538	**0.7980**	-	1.3518	27.036
SM	0.9089	**0.9072**	**0.9339**	0.7382	0.6544	**4.1426**	**82.852**

TABLE 7.4 – Résultats de la métrique $rp_c(x)$ (RàPC - Système sans module de recommandation, DM - Module de recommandation déterministe, SM - Module de recommandation stochastique)

des notes moyennes élevées. Les scores faibles sont attribués aux systèmes qui recommandent de nombreuses questions dans un niveau de complexité avec des notes moyennes élevées ou faibles.

$$rs_c(x) = e^{-\frac{2}{100}(32x_{0,c}^2 - 28x_{0,c} + 10x_{1,c} - 4)^2}; \{x \in \mathbb{R}^2 | 0 <= x <= 1\} \tag{7.9}$$

$$r = \sum_{c=0}^{c_n - 1} rs_c \tag{7.10}$$

Les propriétés de la métrique sont :

— $\{\forall x \in \mathbb{R}^2 | 0 <= x <= 1\}, rs_c(x) > 0$

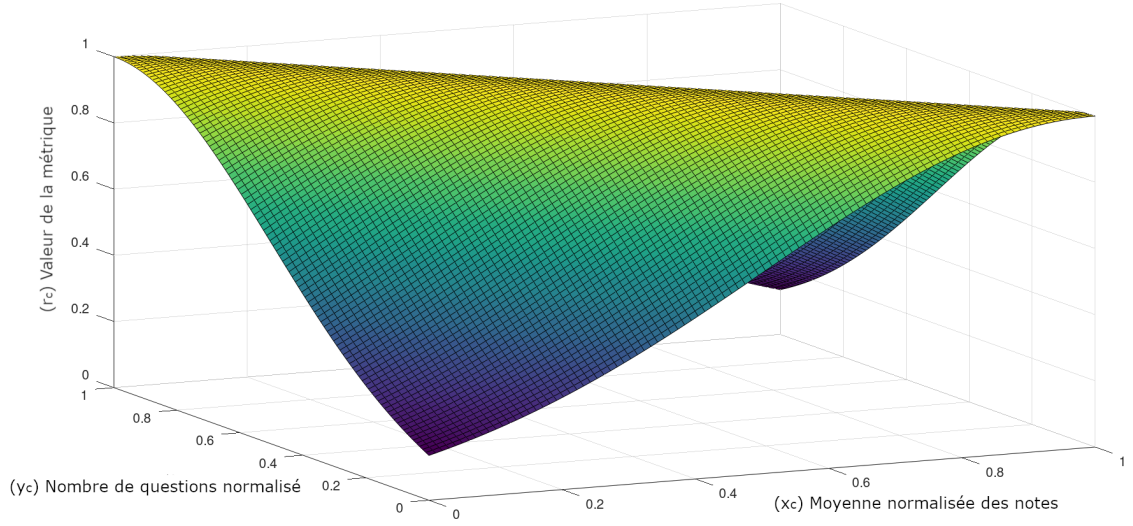

FIGURE 7.5 – Fonction d'évaluation de la qualité de la recommandation pour un parcours standard

— $max(rs_c(x)) = 1;\ if\ 16x_{0,c}^2 - 14x_{0,c} + 5x_{1,c} - 2 = 0$

La figure 7.6 représente la fonction $rs_c(x)$. Comme pour rp_c, la valeur maximale de r dans un niveau de complexité spécifique étant égale à 1, la valeur maximale globale pour les scénarios testés est égale à 5.

Les résultats du calcul des métriques pour le système initial et les deux algorithmes dans les trois scénarios définis sont présentés dans le tableau 7.5.

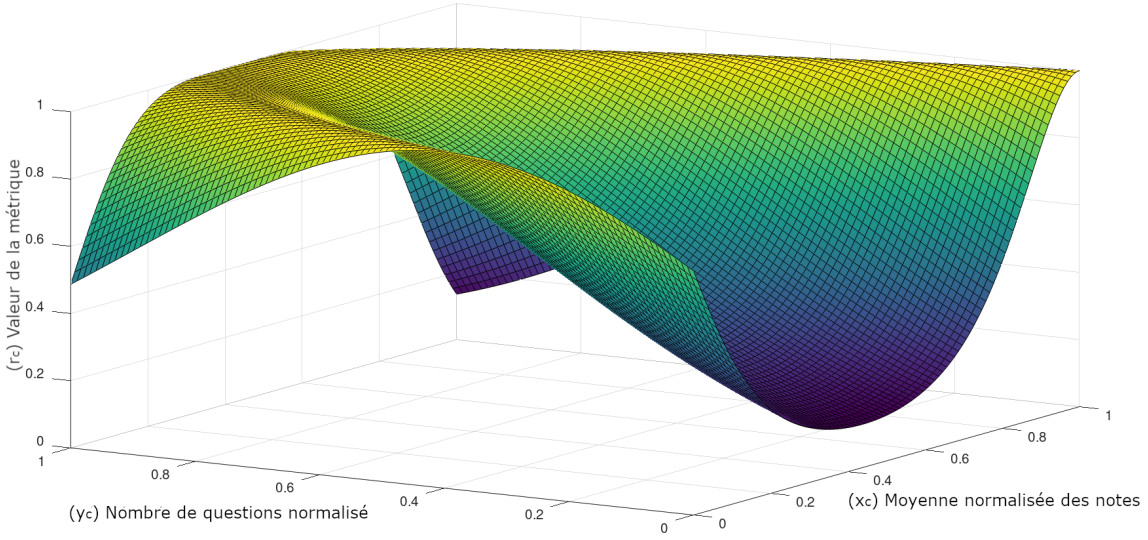

FIGURE 7.6 – Fonction d'évaluation de la qualité de la recommandation pour un apprentissage progressif.

En complément, le tableau 7.6 présente les similarités entre toutes les recommandations faites aux apprenants par les trois systèmes et les trois séances d'entrainement. Pour ce faire, nous avons choisi d'appliquer l'équation 7.11 permettant de calculer une similarité cosinus entre deux vecteurs A et B.

	c_0	c_1	c_2	c_3	c_4	Total (r)	Total (%)
Séance 1							
RàPC	**0.9979**	-	-	-	-	0.9979	19.96
DM	0.8994	0.1908	**0.3773**	**0.2990**	-	1.7665	35.33
SM	0.8447	**0.3012**	0.2536	0.2030	**0.1709**	**1.7734**	**35.47**
Séance 2							
RàPC	**0.4724**	**0.7125**	-	-	-	1.1849	23.70
DM	-	0.6310	**0.3901**	**0.4253**	-	1.4464	28.93
SM	0.2697	0.7089	0.2634	0.2026	**0.1683**	**1.6129**	**32.26**
Séance 3							
RàPC	-	**0.9179**	0.2692	-	-	1.1871	23.74
DM	-	-	0.2236	**0.9674**	-	1.191	23.82
SM	0.1873	0.3038	**0.6345**	0.2394	**0.1726**	**1.5376**	**30.75**

TABLE 7.5 – Évaluation des recommandations proposées selon $rs_c(x)$ par les différents systèmes de recommandation testés : RàPC - Système sans module de recommandation, DM - Algorithme deterministique, SM - Algorithme stochastique

$$sc = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{7.11}$$

Système de recommandation	Séance 1	Séance 2	Séance 3
RàPC	1	1	1
DM	0.9540	0.9887	0.9989
SM	**0.8124**	**0.8856**	**0.9244**

TABLE 7.6 – Moyenne de la diversité des propositions pour tous les apprenants. Une valeur plus faible représente une plus grande diversité. (RàPC - Système sans module de recommandation, DM - Module deterministe, SM - Module stochastique)

7.2.3/ DISCUSSION ET CONCLUSION

Avec la génération d'exercices par le système de RàPC initial, AI-VT propose les mêmes exercices à tous les apprenants, et l'évolution des niveaux de complexité est très lente, un changement toutes les trois ou quatre séances environ. En effet, le système ne prend pas en compte les notes obtenues pendant la séance. Les systèmes intégrant l'un des modules de recommandation testés sont plus dynamiques et les évolutions sont plus rapides. En considérant les notes des apprenants, l'algorithme déterministe suggère des changements de niveaux à un grand nombre d'apprenants de manière soudaine, tandis que l'algorithme stochastique est plus axé sur la personnalisation individuelle et les changements de niveau de complexité sont produits pour un petit nombre d'apprenants. Les deux modules de recommandation proposés ont la capacité de détecter les faiblesses des apprenants et d'adapter la séance à leurs besoins particuliers.

Les données générées ont permis de simuler diverses situations avec les notes de mille apprenants, permettant ainsi d'évaluer le comportement des systèmes de recommandation avec différentes configurations.

Les résultats numériques montrent que les distributions des questions dans une séance par les deux modules de recommandation sont différentes bien que la tendance générale soit similaire. Les modules de recommandation proposés tentent de répartir les questions dans tous les niveaux de complexité définis. Globalement, le module de recommandation stochastique a obtenu un meilleur score. En comparaison du système initial, les modules de recommandation (déterministe et stochastique) proposent 15% à 68% d'adaptations de la complexité pour tous les niveaux. Pour cette raison, l'approche stochastique sera préférée à l'approche déterministe dans la suite des travaux de recherche.

Selon la métrique de la similarité cosinus, le module de recommandation stochastique augmente la diversité des propositions par rapport au système initial dans les trois séances d'entrainement testées, ce qui indique qu'en plus d'atteindre l'adaptabilité, des propositions personnalisées sont générées tout en maintenant l'objectif de progression des niveaux de compétence des apprenants. La diversité des propositions est une caractéristique essentielle de l'algorithme de recommandation dans ses deux versions.

Les modules de recommandation sont un élément essentiel pour certains EIAH car ils aident à guider le processus d'apprentissage individuel. Ils permettent également d'identifier les faiblesses et de réorienter le processus complet afin d'améliorer les connaissances et les compétences. Les deux modules de recommandation proposés peuvent détecter en temps réel les faiblesses de l'apprenant et tentent de réorienter la séance vers le niveau de complexité le plus adapté. Même si l'ensemble des données générées est une simulation de temps de réponse et de notes fictives d'apprenants fictifs, les tests démontrent la flexibilité et la robustesse des modules de recommandation proposés : les données relatives aux apprenants présentent en effet une grande diversité et obligent le système à s'adapter à différents types de configuration. Par conséquent, il est possible de conclure que les modules de recommandation proposés ont la capacité de fonctionner dans différentes situations et de proposer des chemins alternatifs et personnalisés pour améliorer le processus d'apprentissage global.

7.3/ ESCBR-SMA ET ÉCHANTILLONNAGE DE THOMPSON

La section précédente a démontré l'intérêt de l'intégration d'un module de recommandation afin de proposer des exercices d'un niveau de difficulté adapté aux besoins de l'apprenant en fonction des difficultés décelées au cours de la séance d'entraînement. Le système AI-VT initial fondé sur le cycle du raisonnement à partir de cas et ne proposant que des adaptations entre deux séances d'entraînement consécutives, a été supplanté par l'intégration de modules de recommandation utilisés durant la phase de révision du cycle classique du RàPC. Les deux modules de recommandation testés dans la section précédente étaient l'un déterministe, l'autre stochastique.

La section précédente a également démontré qu'il était possible et intéressant que les niveaux de complexité des exercices proposés puissent suivre des fonctions permettant de les faire fluctuer de manière progressive au cours de la séance, et ce afin que les apprenants ne soient pas confrontés à des difficultés changeant de manière trop abrupte durant l'entraînement. Cette étude nous amène donc à considérer la résolution de la génération d'une séance d'exercices sous l'angle de la régression. Nous proposons donc dans cette partie de montrer de quelle manière nous avons intégré et vérifié l'intérêt des outils définis dans le chapitre précédent dans l'EIAH AI-VT.

7.3.1/ Concepts Associés

Cette section présente les concepts, les définitions et les algorithmes nécessaires à la compréhension du module proposé. Le paradigme fondamental utilisé dans ce travail est le raisonnement à partir de cas (RàPC), qui permet d'exploiter les connaissances acquises et l'expérience accumulée pour résoudre un problème spécifique. L'idée principale est de rechercher des situations antérieures similaires et d'utiliser l'expérience acquise pour résoudre de nouveaux problèmes. Le RàPC suit classiquement un cycle de quatre étapes pour améliorer la solution d'inférence [Louvros et al., 2023].

L'un des algorithmes les plus couramment utilisés dans les EIAH pour adapter le contenu et estimer la progression du niveau de connaissance des apprenants est le BKT (*Bayesian Knowledge Tracing*) [Zhang and Yao, 2018]. Cet algorithme utilise quatre paramètres pour estimer la progression des connaissances. $P(k)$ estime la probabilité de connaissance dans une compétence spécifique. $P(w)$, est la probabilité que l'apprenant démontre ses connaissances. $P(s)$, est la probabilité que l'apprenant fasse une erreur. $P(g)$, est la probabilité que l'apprenant ait deviné une réponse. La valeur estimée de la connaissance est mise à jour selon les équations 7.12, 7.13 et 7.14. Si la réponse de l'apprenant est correcte, l'équation 7.12 est utilisée, mais si la réponse est incorrecte, l'équation 7.13 est utilisée.

$$P(k_{t-1}|Correct_t) = \frac{P(k_{t-1})(1-P(s))}{P(k_{t-1})(1-P(s)) + (1-P(k_{t-1}))P(g)} \tag{7.12}$$

$$P(k_{t-1}|Incorrect_t) = \frac{P(k_{t-1})P(s)}{P(k_{t-1})(P(s)) + (1-P(k_{t-1}))(1-P(g))} \tag{7.13}$$

$$P(k_t) = P(k_{t-1}|evidence_t) + (1-P(k_{t-1}|evidence_t))P(w) \tag{7.14}$$

Le module de recommandation proposé, associé à AI-VT, est fondé sur le paradigme de l'apprentissage par renforcement. L'apprentissage par renforcement est une technique d'apprentissage automatique qui permet, par le biais d'actions et de récompenses, d'améliorer les connaissances du système sur une tâche spécifique [Abel et al., 2023]. Nous nous intéressons ici plus particulièrement à l'échantillonnage de Thompson, qui, par le biais d'une distribution de probabilité initiale (distribution a priori) et d'un ensemble de règles de mise à jour prédéfinies, peut adapter et améliorer les estimations initiales d'un processus [Ou et al., 2024]. La distribution de probabilité initiale est généralement définie comme une distribution spécifique de la famille des distributions Beta (équation 7.15) avec des valeurs initiales prédéterminées pour α et β [Uguina et al., 2024], [Nguyen, 2024].

$$Beta(\theta|\alpha,\beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1} \tag{7.15}$$

En utilisant la définition formelle de la fonction Γ (équation 7.16) et en remplaçant certaines variables, une nouvelle expression de la fonction Beta est obtenue (équation 7.17).

$$\Gamma(z) = \int_0^\infty e^{-x}x^{z-1}dx \tag{7.16}$$

$$Beta(\theta|\alpha,\beta) = \frac{\int_0^\infty e^{-s}s^{\alpha+\beta-1}ds}{\int_0^\infty e^{-u}u^{\alpha-1}du\int_0^\infty e^{-v}v^{\beta-1}dv}\theta^{\alpha-1}(1-\theta)^{\beta-1} \tag{7.17}$$

En exprimant les deux intégrales du dénominateur comme une seule intégrale, l'équation 7.18 est obtenue.

$$\int_{u=0}^\infty\int_{v=0}^\infty e^{-u-v}u^{\alpha-1}v^{\beta-1}dudv \tag{7.18}$$

$u = st$, $v = s(1-t)$, $s = u+v$ et $t = u/(u+v)$ sont ensuite remplacées par le résultat du Jacobien 7.19, menant ainsi à l'expression finale définie par l'équation 7.20.

$$\begin{pmatrix}\frac{\partial u}{\partial t} & \frac{\partial u}{\partial s}\\ \frac{\partial v}{\partial t} & \frac{\partial v}{\partial s}\end{pmatrix} = \begin{pmatrix} sdt & tds\\ -sdt & (1-t)ds\end{pmatrix} = s\ dtds \tag{7.19}$$

$$\int_{s=0}^\infty\int_{t=0}^1 e^{-s}(st)^{\alpha-1}(s(1-t))^{\beta-1}s\ dsdt \tag{7.20}$$

Viennent ensuite les équations 7.21 et 7.22 en exprimant les intégrales en fonction des variables de substitution indépendantes s et t.

$$\int_{s=0}^\infty e^{-s}s^{\alpha+\beta-1}ds\int_{t=0}^1 t^{\alpha-1}(1-t)^{\beta-1}dt \tag{7.21}$$

$$Beta(\theta|\alpha,\beta) = \frac{\int_0^\infty e^{-s}s^{\alpha+\beta-1}ds}{\int_{s=0}^\infty e^{-s}s^{\alpha+\beta-1}ds\int_{t=0}^1 t^{\alpha-1}(1-t)^{\beta-1}dt}\theta^{\alpha-1}(1-\theta)^{\beta-1} \tag{7.22}$$

Finalement, la famille de fonctions de distribution Beta peut être calculée selon l'équation 7.23.

$$Beta(\theta|\alpha,\beta) = \frac{\theta^{\alpha-1}(1-\theta)^{\beta-1}}{\int_0^1 t^{\alpha-1}(1-t)^{\beta-1}dt} \tag{7.23}$$

L'évolution de l'algorithme de recommandation TS résulte du changement des distributions de probabilité. Il est à noter qu'au moment de quantifier l'évolution, le changement et la variabilité doivent être calculés en fonction du temps. Les distributions de probabilités peuvent être comparées pour déterminer leur degré de similitude.

Par ailleurs, l'apprentissage automatique utilise la divergence de Kullback-Liebler, qui décrit l'entropie relative de deux distributions de probabilités. Cette fonction est fondée sur le concept d'entropie et le résultat peut être interprété comme la quantité d'informations nécessaires pour obtenir la distribution de probabilité q à partir de la distribution de probabilité p. Bien que largement utilisée, la divergence de Kullback-Liebler (équation 7.24) présente toutefois l'inconvénient de ne pas être une mesure symétrique car elle ne satisfait pas à l'inégalité triangulaire et n'est pas bornée [Li et al., 2024]. Pour remédier à cette difficulté, il est possible d'utiliser la divergence de Jensen-Shannon.

$$D_{KL}(p(x), q(x)) = \int_{-\infty}^{\infty} p(x) log\left(\frac{p(x)}{q(x)}\right) dx \tag{7.24}$$

La divergence de Jenser-Shannon est fondée sur la divergence de Kullback-Liebler. Une distribution de probabilité auxiliaire m est créée dont la définition est fondée sur les distributions initiales p et q [Kim, 2024]. L'équation 7.25 montre la définition formelle de la divergence de Jensen-Shannon, où $m(x)$ est une distribution de mélange de probabilités fondée sur $p(x)$ et $q(x)$. Celle-ci est calculée selon l'équation 7.26. Les distributions de probabilité à comparer doivent être continues et définies dans le même domaine.

$$D_{JS}(p(x), q(x)) = \frac{1}{2} D_{KL}(p(x), m(x)) + \frac{1}{2} D_{KL}(q(x), m(x)) \tag{7.25}$$

$$m(x) = \frac{1}{2} p(x) + \frac{1}{2} q(x) \tag{7.26}$$

La prédiction utilisée dans le module proposé ici a été présentée dans le chapitre 6. Il s'agit d'un algorithme d'empilement de raisonnement à partir de cas mettant en œuvre deux niveaux d'intégration. Le module utilise globalement la stratégie d'empilement pour exécuter plusieurs algorithmes afin de rechercher des informations dans un ensemble de données et générer des solutions à différents problèmes génériques. En outre une étape d'évaluation permet de sélectionner la solution la plus optimale pour un problème donné en fonction d'une métrique adaptative définie pour les problèmes de régression.

7.3.2/ ALGORITHME PROPOSÉ

Nous proposons ici une intégration de l'algorithme d'adaptation stochastique (fondé sur l'échantillonnage de Thompson) avec ESCBR-SMA. Ainsi, le module de recommandation révise la séance en fonction des notes de l'apprenant et ESCBR-SMA effectue une prédiction pour valider l'adaptation générée.

L'idée est d'unifier les deux modules en se fondant à la fois sur des informations locales (recommandation fondée sur l'échantillonnage de Thompson (TS) et les informations propres à l'apprenant), et sur des informations globales (cas similaires de la base de connaissances du système de RàPC).

L'architecture de l'algorithme est présentée sur la figure 7.14, où l'on peut voir que les deux algorithmes TS et RàPC sont exécutés en parallèle et indépendamment. Des synchronisations sont faites après obtention des résultats de chaque module. Ces résultats sont unifiés via d'une fonction de pondération. La recommandation finale est calculée selon l'équation 7.29. Le *paradoxe de Simpson* est un paradoxe dans lequel un phénomène observé dans plusieurs groupes s'inverse lorsque les groupes sont combinés[Xu et al., 2023]. L'unification d'ensembles de données différents peut atténuer ce paradoxe [Lei, 2024].

La première étape est l'adaptation avec l'échantillonnage de Thompson. Vient ensuite la prédiction via ECBR-SMA. Enfin, le processus se termine par la prise de décision concernant la suite de la séance à délivrer à l'apprenant. Le système de recommandation obtient une valeur de probabilité pour tous les niveaux de complexité et l'ECBR-SMA évalue la proposition avec une prédiction pour chaque niveau de complexité. Le tableau

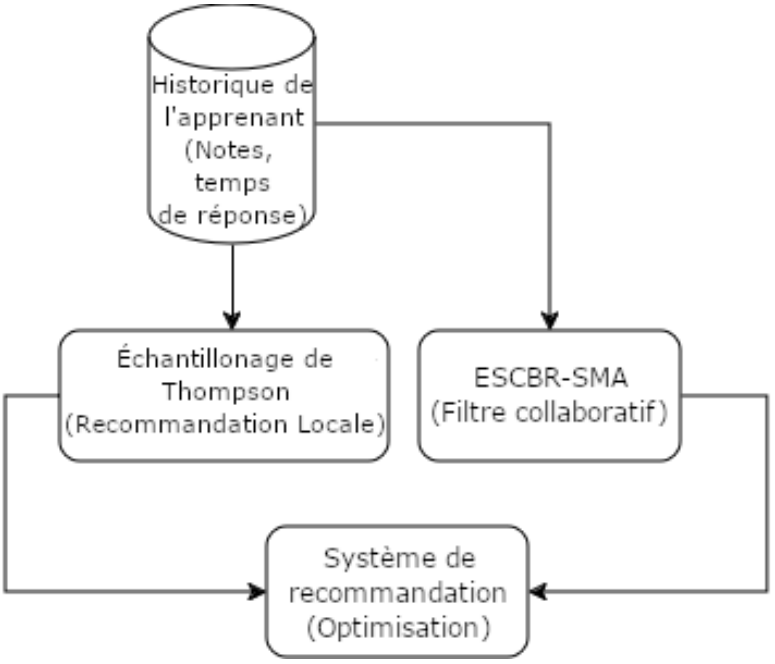

FIGURE 7.7 – Schéma de l'architecture de l'algorithme proposé

7.7 présente les variables et les paramètres du module proposé ainsi que les mesures employées.

ID	Type	Description	Domain
α	p	Paramètre de la distribution beta	$[1,\infty] \in \mathbb{R}$
β	p	Paramètre de la distribution beta	$[1,\infty] \in \mathbb{R}$
t	p	Numéro de l'itération	$\mathbb{N}$
c	p	Niveau de complexité	$\mathbb{N}$
x_c	p	Notes moyennes par niveau de complexité c	$\mathbb{R}$
y_c	p	Nombre de questions par niveau de complexité c	$\mathbb{N}$
r	f	Fonction suivie pour la recommandation	$[0,1] \in \mathbb{R}$
$k_{t,c}$	v	Évolution de la connaissance dans le temps t pour le niveau de complexité c	$[0,1] \in \mathbb{R}$
$vk_{t,c}$	v	Évolution de la connaissance pour chaque niveau de complexité c	$\mathbb{R}$
TS_c	v	Récompense d'échantillonnage de Thompson pour un niveau de complexité c	$[0,1] \in \mathbb{R}$
TSN_c	v	Normalisation de TS_c avec d'autres niveaux de complexité	$[0,1] \in \mathbb{R}$
$ESCBR_c$	v	Prédiction de la note pour un niveau de complexité c	$\mathbb{R}_+$
p_c	f	Fonction de densité de probabilité pour le niveau de complexité c	$\mathbb{R}_+$
D_{JS}	f	Divergence de Jensen-Shannon	$[0,1] \in \mathbb{R}$

TABLE 7.7 – Paramètres (p), variables (v) et fonctions (f) de l'algorithme proposé et des métriques utilisées

Pour rappel, le processus de recommandation se fait en trois étapes. Tout d'abord, il est nécessaire d'avoir des valeurs aléatoires pour chaque niveau de complexité c en utilisant les distributions de probabilité générées avec le algorithme TS (équation 7.27). Une fois que toutes les valeurs de probabilité correspondant à tous les niveaux de complexité ont été obtenues, la normalisation de toutes ces valeurs est calculée selon l'équation

7.28. Les valeurs de normalisation servent de paramètres de priorité pour les prédictions effectuées par ESCBR-SMA (équation 7.29). La recommandation finalement proposée est celle dont la valeur est la plus élevée.

$$TS_c = rand(Beta(\alpha_c, \beta_c)) \tag{7.27}$$

$$TSN_c = \frac{TS_c}{\sum_{i=0}^{4} TS_i} \tag{7.28}$$

$$n_c = argmax_c(TSN_c * ESCBR_c) \tag{7.29}$$

7.3.3/ Résultats et Discussion

Le principal inconvénient posé par la validation d'un tel système « en situation réelle » est la difficulté à collecter des données et à évaluer des systèmes différents dans des conditions strictement similaires. Cette difficulté est accentuée dans les contextes d'apprentissage autorégulés, puisque les apprenants peuvent quitter la plateforme d'apprentissage à tout moment rendant ainsi les données incomplètes [Badier et al., 2023].

Pour cette raison, les différentes approches proposées ont été testées sur des données générées : les notes et les temps de réponse de 1000 apprenants fictifs et cinq questions par niveau de complexité. Les notes des apprenants ont été créées en suivant la loi de distribution *logit-normale* que nous avons jugée proche de la réalité de la progression d'un apprentissage. lien vers la base générée ? ==> Ref de l'url du git

Quatre séries de tests ont été effectuées. La première série a été menée sur le système AI-VT intégrant le système de RàPC pour la régression afin de démontrer la capacité de l'algorithme à prédire les notes à différents niveaux de complexité. La deuxième série de tests a évalué la progression des connaissances avec TS afin d'analyser la capacité du module à proposer des recommandations personnalisées. Lors de la troisième série de tests, nous avons comparé les algorithmes de recommandation BKT et TS. Enfin, lors de la quatrième série de tests, nous avons comparé TS seul et TS avec ESCBR-SMA.

7.3.3.1/ Régression avec ESCBR-SMA pour l'aide à l'apprentissage humain

Le SMA que nous avons implémenté utilise un raisonnement bayésien, ce qui permet aux agents d'apprendre des données et d'intéragir au cours de l'exécution et de l'exploration.

ESCBR-SMA utilise une fonction noyau pour obtenir la meilleure approximation de la solution du problème cible. Dans notre cas, l'obtention de la meilleure solution est un problème NP-Difficile car la formulation est similaire au problème de Fermat-Weber à N dimensions [Minsker and Strawn, 2024].

Les différents scénarios du tableau 7.8 ont été considérés dans un premier temps. Dans le scénario E_1, il s'agit de prédire la note d'un apprenant au premier niveau de complexité, après 3 questions. Le scénario E_2 considère les notes de 8 questions et l'objectif est de prédire la note de la neuvième question dans le même niveau de complexité. Le scénario E_3 interpole la neuvième note que l'apprenant obtiendrait si la neuvième question était de niveau de complexité supérieur à celui de la huitième question. Cette interpolation est

faite sur la base des notes obtenues aux quatre questions précédentes. Le scénario E_4 considère 4 questions et le système doit interpoler 2 notes dans un niveau de complexité supérieur.

Scenario	Caractéristiques du problème	Dimension de la solution
E_1	5	1
E_2	15	1
E_3	9	1
E_4	9	2

TABLE 7.8 – Description des scénarios

ESCBR-SMA a été comparé aux neuf outils classiquement utilisés pour résoudre la régression consignés dans le tableau 7.9 et selon l'erreur quadratique moyenne (RMSE - *Root Mean Squared Error*), l'erreur médiane absolue (MedAE - *Median Absolute Error*) et l'erreur moyenne absolue (MAE - *Mean Absolute Error*).

ID	Algorithm	ID	Algorithm
A1	Linear Regression	A6	Polinomial Regression
A2	K-Nearest Neighbor	A7	Ridge Regression
A3	Decision Tree	A8	Lasso Regression
A4	Random Forest (Ensemble)	A9	Gradient Boosting (Ensemble)
A5	Multi Layer Perceptron	A10	Proposed Ensemble Stacking RàPC

TABLE 7.9 – Liste des algorithmes évalués

Le tableau 7.10 présente les résultats obtenus par les 10 algorithmes sur les quatre scénarios. Ces résultats montrent qu'ESCBR-SMA (A10) et le *Gradient Boosting* (A9) obtiennent toujours les deux meilleurs résultats. Si l'on considère uniquement la RMSE, ESCBR-SMA occupe toujours la première place sauf pour E_3 où il est deuxième. Inversement, en considérant l'erreur médiane absolue ou l'erreur moyenne absolue, A10 se classe juste après A9. ESCBR-SMA et le *Gradient Boosting* sont donc efficaces pour interpoler les notes des apprenants.

	Algorithme									
Scenario (Metrique)	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
E_1 (RMSE)	0.625	0.565	0.741	0.56	0.606	0.626	0.626	0.681	0.541	**0.54**
E_1 (MedAE)	0.387	0.35	0.46	0.338	0.384	0.387	0.387	0.453	**0.327**	0.347
E_1 (MAE)	0.485	0.436	0.572	0.429	0.47	0.485	0.485	0.544	**0.414**	0.417
E_2 (RMSE)	0.562	0.588	0.78	0.571	0.61	0.562	0.562	0.622	0.557	**0.556**
E_2 (MedAE)	0.351	0.357	0.464	0.344	0.398	0.351	0.351	0.415	**0.334**	0.346
E_2 (MAE)	0.433	0.448	0.591	0.437	0.478	0.433	0.433	0.495	**0.422**	0.429
E_3 (RMSE)	0.591	0.59	0.79	0.57	0.632	0.591	0.591	0.644	**0.555**	0.558
E_3 (MedAE)	0.367	0.362	0.474	0.358	0.404	0.367	0.367	0.433	**0.336**	0.349
E_3 (MAE)	0.453	0.45	0.598	0.441	0.49	0.453	0.453	0.512	**0.427**	0.43
E_4 (RMSE)	0.591	0.589	0.785	0.568	0.613	0.591	0.591	0.644	0.554	**0.549**
E_4 (MedAE)	0.367	0.362	0.465	0.57	0.375	0.367	0.367	0.433	**0.336**	0.343
E_4 (MAE)	0.453	0.45	0.598	0.438	0.466	0.453	0.453	0.512	0.426	**0.417**

TABLE 7.10 – Erreurs moyennes et médianes des interpolations des 10 algorithmes sélectionnés sur les 4 scénarios considérés et obtenues après 100 exécutions.

7.3.3.2/ PROGRESSION DES CONNAISSANCES

L'algorithme de recommandation TS est fondé sur le paradigme bayésien le rendant ainsi particulièrement adapté au problèmes liés à la limitation de la quantité de données et à une incertitude forte. Afin de quantifier la connaissance et de voir sa progression dans le temps avec TS, la divergence de Jensen-Shannon avec la famille de distribution Beta en t et $t-1$ a été mesurée. L'équation 7.30 décrit formellement le calcul à effectuer avec les distributions de probabilité en un temps t pour un niveau de complexité c, en utilisant la définition m (équation 7.31).

$$k_{t,c} = \frac{1}{2}\int_0^1 p_c(\alpha_t,\beta_t,x) log\left(\frac{p_c(\alpha_t,\beta_t,x)}{m(p_c(\alpha_{t-1},\beta_{t-1},x),p_c(\alpha_t,\beta_t,x))}\right)dx + \frac{1}{2}\int_0^1 p_c(\alpha_{t-1},\beta_{t-1},x) log\left(\frac{p_c(\alpha_{t-1},\beta_{t-1},x)}{m(p_c(\alpha_{t-1},\beta_{t-1},x),p_c(\alpha_t,\beta_t,x))}\right)dx \quad (7.30)$$

$$m(p(\alpha_{(t-1)},\beta_{(t-1)},x),p(\alpha_t,\beta_t,x)) = \frac{1}{2}\left(\frac{x^{\alpha_{(t-1)}-1}(1-x)^{\beta_{(t-1)}-1}}{\int_0^1 u^{\alpha_{(t-1)}-1}(1-u^{\beta_{(t-1)}-1})du}\right) + \frac{1}{2}\left(\frac{x^{\alpha_t-1}(1-x)^{\beta_t-1}}{\int_0^1 u^{\alpha_t-1}(1-u^{\beta_t-1})du}\right) \quad (7.31)$$

La progression du nombre total de connaissances en t est la somme des différences entre t et $t-1$ pour tous les c niveaux de complexité calculés avec la divergence de Jensen-Shannon (équation 7.33). Pour ce faire, nous évaluons la progression de la variabilité données par équation 7.32.

$$vk_{t,c} = \begin{cases} D_{JS}(Beta(\alpha_{t,c},\beta_{t,c}),Beta(\alpha_{t+1,c},\beta_{t+1,c})), & \frac{\alpha_{t,c}}{\alpha_{t,c}+\beta_{t,c}} < \frac{\alpha_{t+1,c}}{\alpha_{t+1,c}+\beta_{t+1,c}} \\ -D_{JS}(Beta(\alpha_{t,c},\beta_{t,c}),Beta(\alpha_{t+1,c},\beta_{t+1,c})), & Otherwise \end{cases} \quad (7.32)$$

$$k_t = \sum_{c=4}^{c=0\vee k_t\neq 0} \begin{cases} \alpha_{c-1} vk_{t,c-1}; & vk_{t,c} > 0 \\ 0; & Otherwise \end{cases} \quad (7.33)$$

La figure 7.8 montre la progression cumulée des connaissances sur les quinze questions d'une même séance d'entrainement. L'augmentation de la moyenne du niveau de connaissance entre la première et la dernière question de la même séance montre que tous les apprenants ont statistiquement augmenté leur niveau de connaissance. La variabilité augmente à partir de la première question jusqu'à la question neuf, où le système a acquis plus d'informations sur les apprenants. À ce stade, la variabilité diminue et la moyenne augmente.

7.3.3.3/ SYSTÈME DE RECOMMANDATION AVEC UN JEU DE DONNÉES D'ÉTUDIANTS RÉELS

Le système de recommandation TS a été testé avec un ensemble de données adaptées extraites de données réelles d'interactions d'étudiants avec un environnement d'ap-

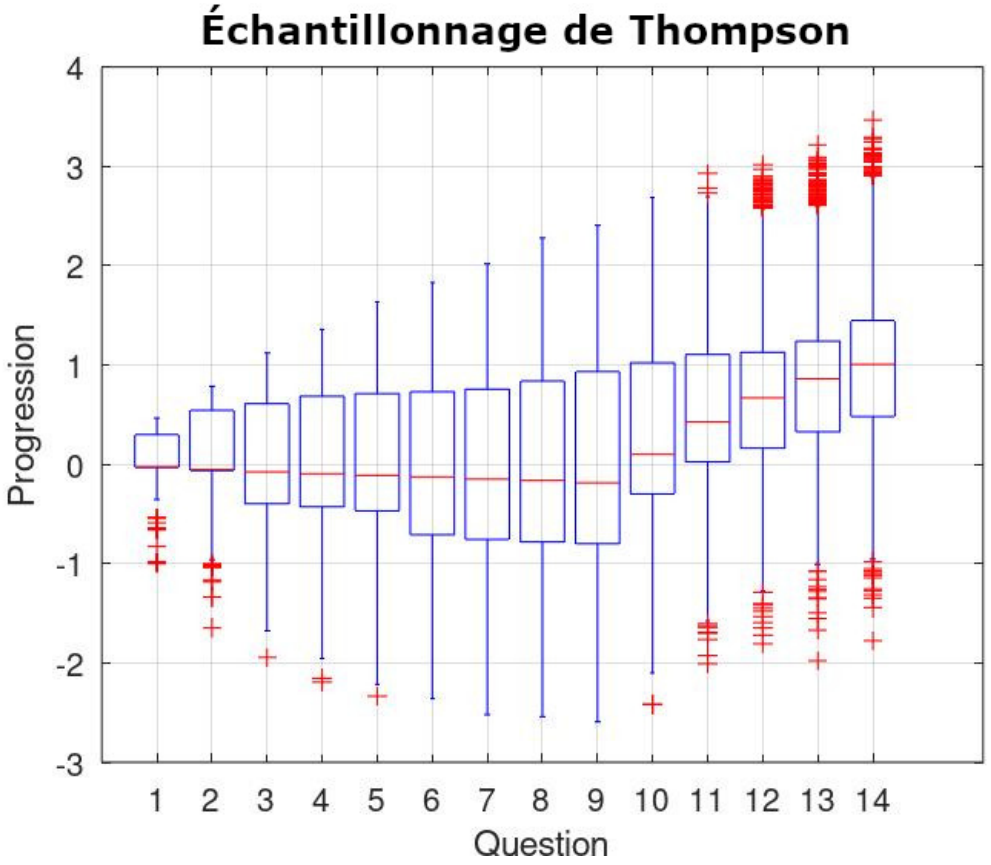

FIGURE 7.8 – Progression des connaissances avec l'échantillonnage de Thompson selon la divergence de Jensen-Shannon

prentissage virtuel pour différents cours [Kuzilek et al., 2017]. Cet ensemble contient les notes de 23366 apprenants dans différents cours. Les apprenants ont été évalués selon différentes modalités (partiels, projets, QCM). Cet ensemble de données a pu être intégré au jeu de données d'AI-VT (notes, temps de réponse et 5 niveaux de complexité). Le test a consisté à générer une recommandation pour l'avant dernière question en fonction des notes précédentes. Ce test a été exécuté 100 fois pour chaque apprenant. Les nombres de questions recommandées sont reportés sur la figure 7.15 pour chaque niveau de complexité. Celle-ci montre que malgré la stochasticité, la variance globale dans tous les niveaux de complexité est faible en fonction du nombre total d'apprenants et du nombre total de recommandations, et démontre ainsi la stabilité de l'algorithme.

La précision de la recommandation pour tous les apprenants est évaluée en considérant comme comportement correct deux états : i) l'algorithme recommande un niveau où l'apprenant a une note supérieure ou égal à 6 et ii) l'algorithme recommande un niveau inférieur au niveau réel évalué par l'apprenant. Le premier cas montre que l'algorithme a identifié le moment précis où l'apprenant doit augmenter le niveau de complexité, le second cas permet d'établir que l'algorithme propose de renforcer un niveau de complexité plus faible. Puis la précision est calculée comme le rapport entre le nombre d'états correspondant aux comportements corrects définis et le nombre total de recommandations. La figure 7.10 montre les résultats de cette métrique après 100 exécutions.

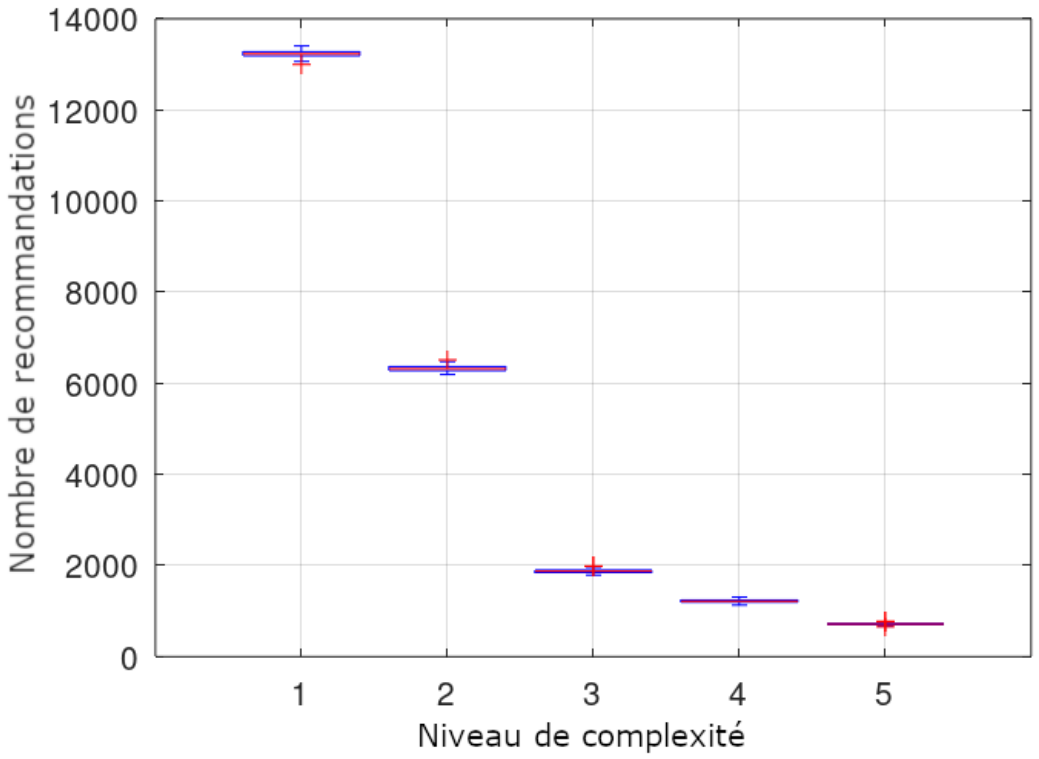

FIGURE 7.9 – Nombre de recommandations par niveau de complexité

7.3.3.4/ COMPARAISON ENTRE TS ET BKT

La figure 7.11 permet de comparer la recommandation fondée sur l'échantillonnage de Thompson et celle fondée sur BKT. Cette figure montre l'évolution des notes des apprenants en fonction du nombre de questions auxquelles ils répondent dans la même séance. Dans ce cas, le TS génère moins de variabilité que BKT, mais les évolutions induites par les deux systèmes restent globalement très similaires.

Toutefois, si l'on considère l'évolution du niveau de complexité recommandé (figure 7.12), TS fait évoluer le niveau de complexité des apprenants, alors que BKT a tendance à laisser les apprenants au même niveau de complexité. Autrement dit, avec BKT, il est difficile d'apprendre de nouveaux sujets ou des concepts plus complexes au sein du même domaine. En comparant les résultats des deux figures (figures 7.11 et 7.12), TS permet de faire progresser la moyenne des notes et facilite l'évolution des niveaux de complexité.

7.3.3.5/ SYSTÈME DE RECOMMANDATION AVEC ESCBR-SMA

Le troisième étape est l'association des deux algorithmes afin de combiner des observations qui ne sont pas directement liées l'une à l'autre, c'est-à-dire en utilisant l'information individuelle (recommandation avec Thomson) et le filtre collaboratif (interpolation avec ESCBR-SMA). Cette partie présente une comparaison entre le système de recommandation TS et le système de recommandation TS intégrée à la prédiction ESCBR-SMA.

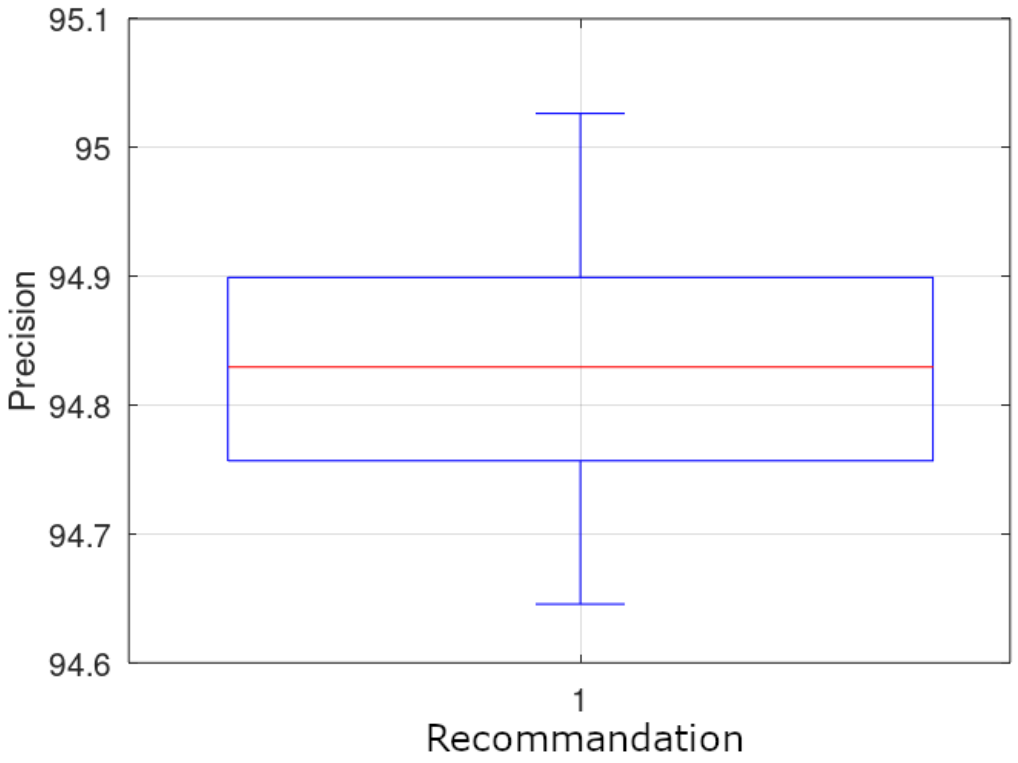

FIGURE 7.10 – Précision de la recommandation

Les résultats proposés sont comparés après réponse des apprenants à six questions car il est nécessaire de disposer d'informations préalables pour utiliser l'algorithme ESCBR-SMA et prédire les notes dans tous les niveaux de complexité à la question suivante.

7.3.3.6/ PROGRESSION DES CONNAISSANCES TS VS TS ET ESCBR-SMA

Pour établir la différence entre le système de recommandation TS et ce même système associé à la prédiction fondée sur le raisonnement à partir de cas ESCBR-SMA, nous utiliserons la métrique de Jensen-Shannon (définie suivant les équations 7.34 et 7.35). Dans ce cas la comparaison est faite sur le même niveau de complexité et dans le même temps t.

$$k_{t,c} = \frac{1}{2}\int_0^1 p_c(\alpha_{p1,t},\beta_{p1,t},x)log\left(\frac{p_c(\alpha_{p1,t},\beta_{p1,t},x)}{m(p_c(\alpha_{p1,t},\beta_{p1,t},x),p_c(\alpha_{p2,t},\beta_{p2,t},x))}\right)dx + \frac{1}{2}\int_0^1 p_c(\alpha_{p2,t},\beta_{p2,t},x)log\left(\frac{p_c(\alpha_{p2,t},\beta_{p2,t},x)}{m(p_c(\alpha_{p1,t},\beta_{p1,t},x),p_c(\alpha_{p2,t},\beta_{p2,t},x))}\right)dx \quad (7.34)$$

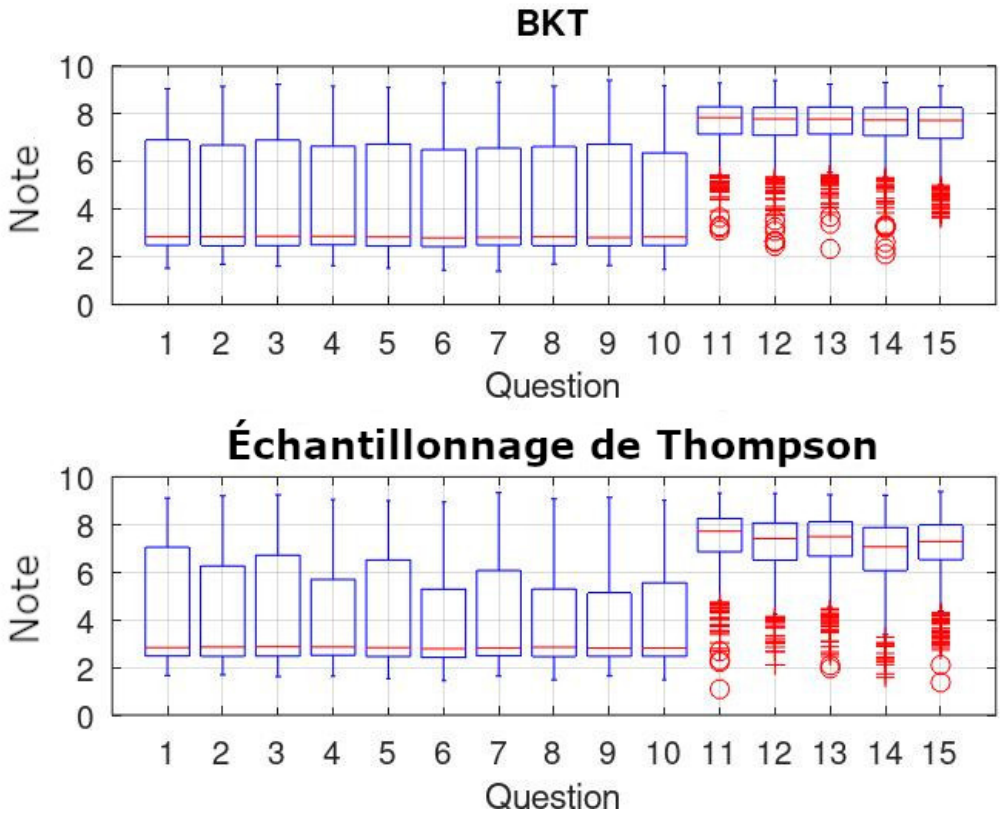

FIGURE 7.11 – Comparaison de l'évolution des notes entre les systèmes fondés sur TS et BKT.

$$m(p(\alpha_{p1,t},\beta_{p1,t},x),p(\alpha_{p2,t},\beta_{p2,t},x)) = \frac{1}{2}\left(\frac{x^{\alpha_{p1,t}-1}(1-x)^{\beta_{p1,t}-1}}{\int_0^1 u^{\alpha_{p1,t}-1}(1-u^{\beta_{p1,t}-1})du}\right) + \frac{1}{2}\left(\frac{x^{\alpha_{p2,t}-1}(1-x)^{\beta_{p2,t}-1}}{\int_0^1 u^{\alpha_{p2,t}-1}(1-u^{\beta_{p2,t}-1})du}\right) \quad (7.35)$$

La figure 7.13 présente une visualisation normalisée des différences de progression entre TS et TS intégré à ESCBR-SMA. Cette figure montre une rupture après la septième question. Pour toutes les questions de la même séance, en moyenne, la progression avec TS associé à ESCBR-SMA est meilleure que celle avec TS seul.

7.3.4/ CONCLUSION

Cette partie montre que l'intégration du module d'échantillonnage de Thompson au module de régression ESCBR-SMA est profitable à l'EIAH AI-VT dans la mesure où leur utilisation conjointe permet de réviser une séance d'entrainement en proposant des exercices de difficulté adaptée à l'apprenant en fonction des notes obtenues aux réponses précédentes.

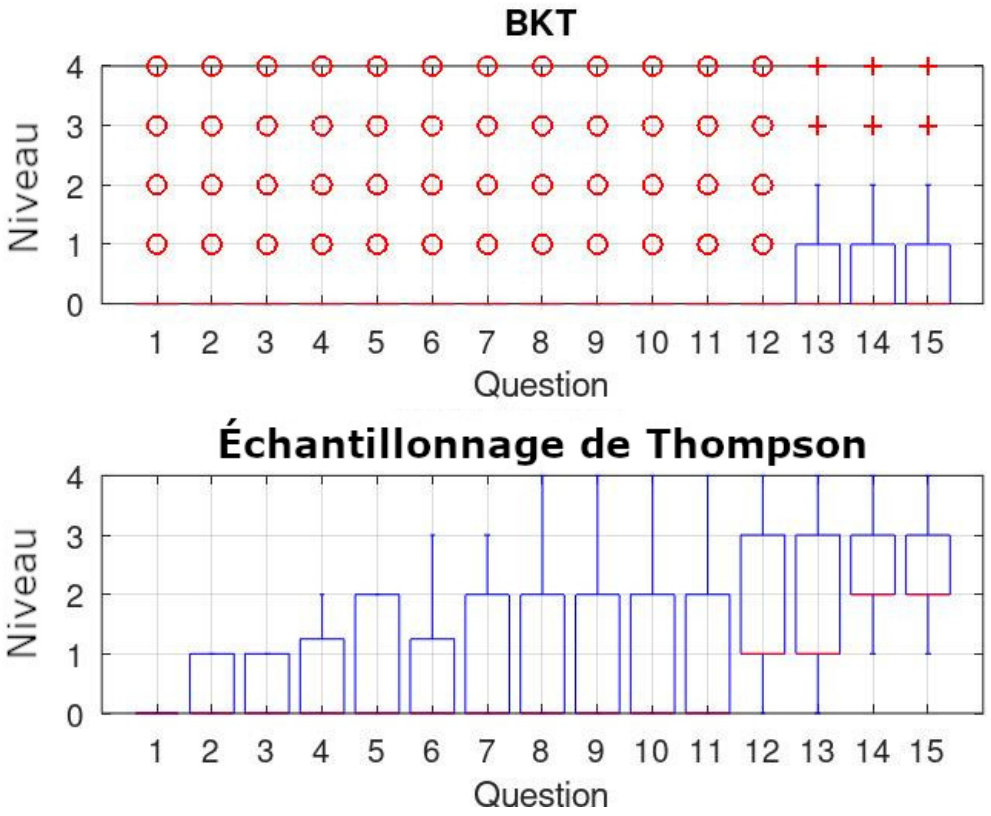

FIGURE 7.12 – Comparaison de l'évolution des niveaux entre les systèmes de recommandation fondés sur BKT et TS

Le système proposé permet de générer des recommandations personnalisées pour chaque apprenant avec relativement peu de données historiques. Il permet de réviser une séance en se fondant sur des informations générales (progression standardisée) et locales (niveau acquis par l'apprenant).

Toutefois, un apprentissage n'est pas linéaire et il est nécessaire de proposer des rappels de cours ou de notions à l'apprenant. La partie suivante de ce chapitre tente d'apporter une solution à cette observation pouvant limiter les performances d'AI-VT en introduisant le processus de Hawkes.

7.4/ ESCBR-SMA, ÉCHANTILLONNAGE DE THOMPSON ET PROCESSUS DE HAWKES

Dans la partie précédente, nous avons proposé un système capable de permettre à l'apprenant de suivre une progression relativement linéaire. Toutefois, certaines notions sont parfois oubliées par les apprenants et des rappels doivent être proposés afin de mieux consolider les connaissances. C'est la raison pour laquelle nous nous intéressons dans cette partie à l'intégration d'un processus de Hawkes qui permet de simuler une courbe d'oubli.

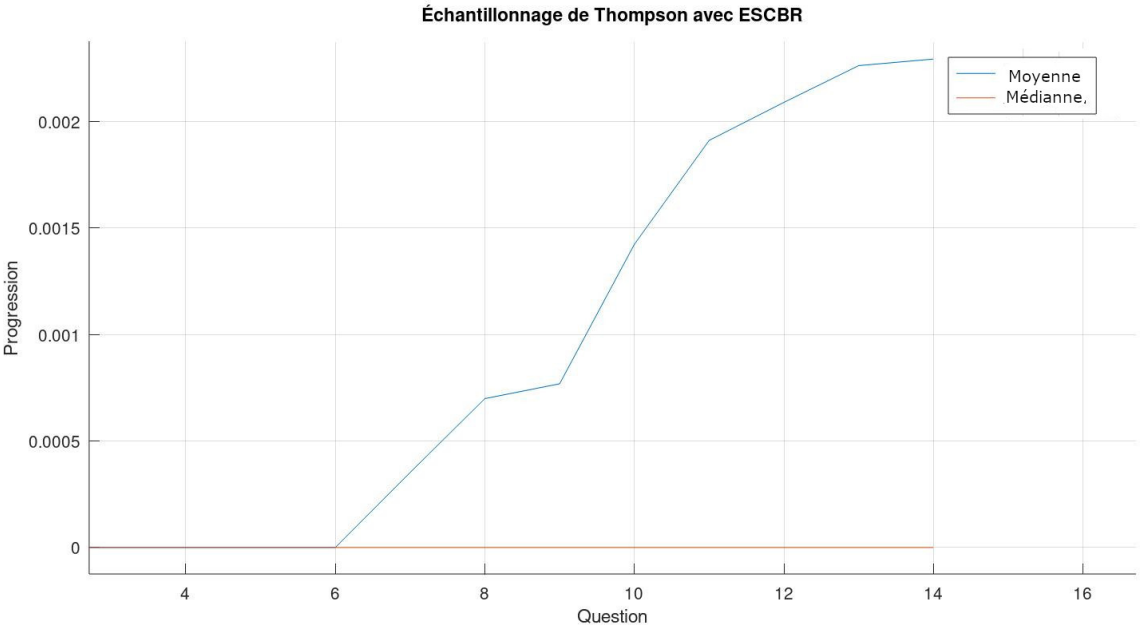

FIGURE 7.13 – Différence normalisée entre la progression avec échantillonnage de Thompson seul et échantillonnage de Thompson aassocié à ESCBR-SMA pour 1000 apprenants

7.4.1/ ALGORITHME PROPOSÉ

L'algorithme proposé est une intégration de la recommandation stochastique (fondée sur l'échantillonnage de Thompson), du raisonnement à partir de cas (ESCBR-SMA) et du processus de Hawkes. Dans ce cas, l'algorithme de recommandation produit une adaptation en fonction des notes de l'apprenant et l'ESCBR-SMA effectue une prédiction pour valider l'adaptation générée, le processus de Hawkes simule la courbe d'oubli dans le processus d'apprentissage.

L'idée de cette unification est d'obtenir des informations d'un point de vue local où une recommandation est obtenue en se fondant sur les informations des apprenants individuels (rôle de l'échantillonnage de Thompson), la prédiction globale fondée sur les expériences acquises précédemment par le système (le système de RàPC fournit des exercices ayant permis par le passé à d'autres apprenants aux profils similaires d'acquérir les connaissances visées), et le processus d'apprentissage dynamique avec le processus de Hawkes.

La figure 7.14 montre de quelle manière sont organisés ces différents modules les uns par rapport aux autres. TS et ESCBR-SMA sont exécutés en parallèle et indépendamment avec les informations extraites de la même base de connaissances. Une fois que les résultats de chaque module sont obtenus, ils sont unifiés par une fonction de pondération et une distribution de probabilité mise à jour dynamiquement selon les événements passés et le niveau de complexité sélectionné. Une dernière étape permet de consolider et d'atténuer l'effet du paradoxe de Simpson [Xu et al., 2023].

La première étape est l'adaptation avec l'échantillonnage de Thompson et la prédiction de l'ECBR-SMA. Celle-ci est suivie par la prise de décision pour l'envoi à l'apprenant. Le système de recommandation obtient une valeur de probabilité pour tous les niveaux

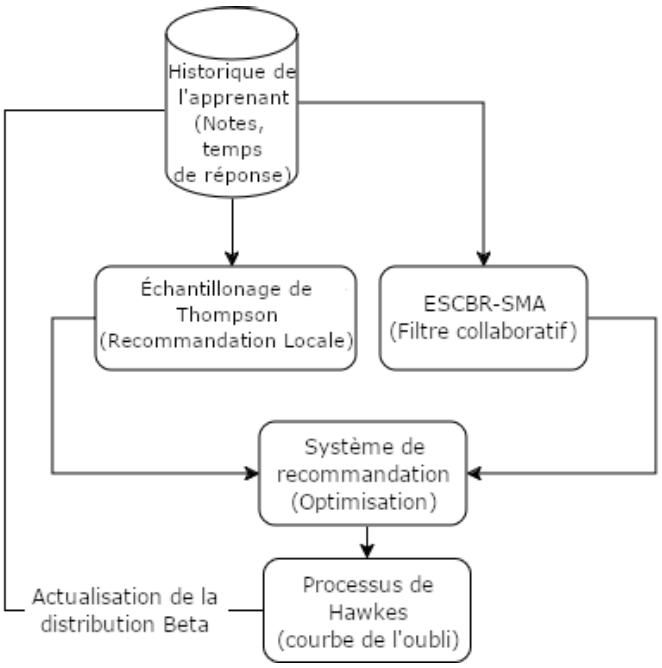

FIGURE 7.14 – Organisation des modules TS, ESCBR-SMA et processus de Hawkes.

de complexité pour l'apprenant et l'ECBR-SMA évalue la proposition avec une prédiction pour chaque niveau de complexité. Le tableau 7.7 présente les variables et les paramètres du module proposé et les mesures employées.

Après la sélection du niveau de complexité, toutes les distributions de probabilité sont mises à jour selon le processus de Hawkes (équation 7.36) pour chaque paramètre α et β en utilisant la fonction d'intensité définie constante (équations 7.37 et 7.38) et la fonction d'excitation (équation 7.39 avec des valeurs $\alpha = 10$ et $\beta = 0.02$ équation 7.40), afin de simuler la courbe d'oubli dans l'évolution de la distribution de probabilité Beta.

$$\lambda(t) = \mu(t) + \sum_{t_i<t} \phi(t - t_i) \tag{7.36}$$

$$\mu_{\alpha,c}(t) = \begin{cases} 2, & c = 0 \\ 1, & 1 \leq c \leq 4 \end{cases} \tag{7.37}$$

$$\mu_{\beta,c}(t) = \begin{cases} 1, & c = 0 \\ 3, & c = 1 \\ 5, & c = 2 \\ 7, & c = 3 \\ 9, & c = 4 \end{cases} \tag{7.38}$$

$$\phi_h(t) = \alpha\beta e^{-\beta t} \tag{7.39}$$

$$\phi_h(t) = (10)(0.02)e^{-0.02t} \tag{7.40}$$

L'équation 7.41 montre la définition complète pour tous les α et l'équation 7.42 la définition pour les paramètres β.

$$\lambda_\alpha(t) = \mu_{\alpha,c}(t) + \sum_{t_i<t} \phi_h(t - t_i) \tag{7.41}$$

$$\lambda_\beta(t) = \mu_{\beta,c}(t) + \sum_{t_i<t} \phi_h(t - t_i) \tag{7.42}$$

Et pour chaque niveau de complexité c, l'équation 7.43 décrit la distribution des probabilités.

$$P_c(x, \lambda_\alpha(t), \lambda_\beta(t)) = \frac{x^{\lambda_\alpha(t)}(1-x)^{\lambda_\beta(t)}}{\int_0^1 u^{\lambda_\alpha(t)}(1-u)^{\lambda_\beta(t)}du} \tag{7.43}$$

7.4.2/ Résultats et Discussion

7.4.2.1/ Système de recommandation avec un jeu de données d'étudiants réels (TS avec Hawkes)

Le système de recommandation TS a été testé avec un ensemble de données adaptées extraites des données réelles des interactions des étudiants avec un environnement d'apprentissage virtuel pour différents cours [Kuzilek et al., 2017]. Cet ensemble de données compte 23366 apprenants répartis dans différents cours et évalués de différentes manières.

Le format de cet ensemble de données est adapté à la structure de la connaissance dans AI-VT (notes, temps de réponse et niveau de complexité), et les exercices y sont répartis en cinq niveaux de complexité. La figure 7.15 montre le nombre de recommandations générées par un processus d'apprentissage statique et par le module dynamique associant TS au processus de Hawkes, sur les différents niveaux de complexité. Cette figure montre les moyennes obtenues avec 100 exécutions des algorithmes. La faible variance globale obtenue dans tous les niveaux de complexité montre la stabilité des recommandations.

Le module incluant le processus de Hawkes recommande plus de niveaux de faible complexité car la connaissance tend à diminuer avec le temps, avant de tendre à renforcer la connaissance dans tous les niveaux de complexité. La configuration initiale privilégiant les niveaux inférieurs (grâce à une plus grande probabilité), le module tend à répéter les niveaux plus accessibles nécessaires pour atteindre les niveaux supérieurs.

7.4.2.2/ Mesures de performances

Pour mesurer les performances des différents modules, nous avons utilisé le même jeu de données simulées que dans la section précédentes. La comparaison est faite avec la métrique décrite dans l'équation 7.44, où x_c est la moyenne des notes, y_c est le nombre de questions pour chaque niveau de complexité c.

$$r(x_c, y_c) = e^{-2(x_c+y_c-1)^2} \tag{7.44}$$

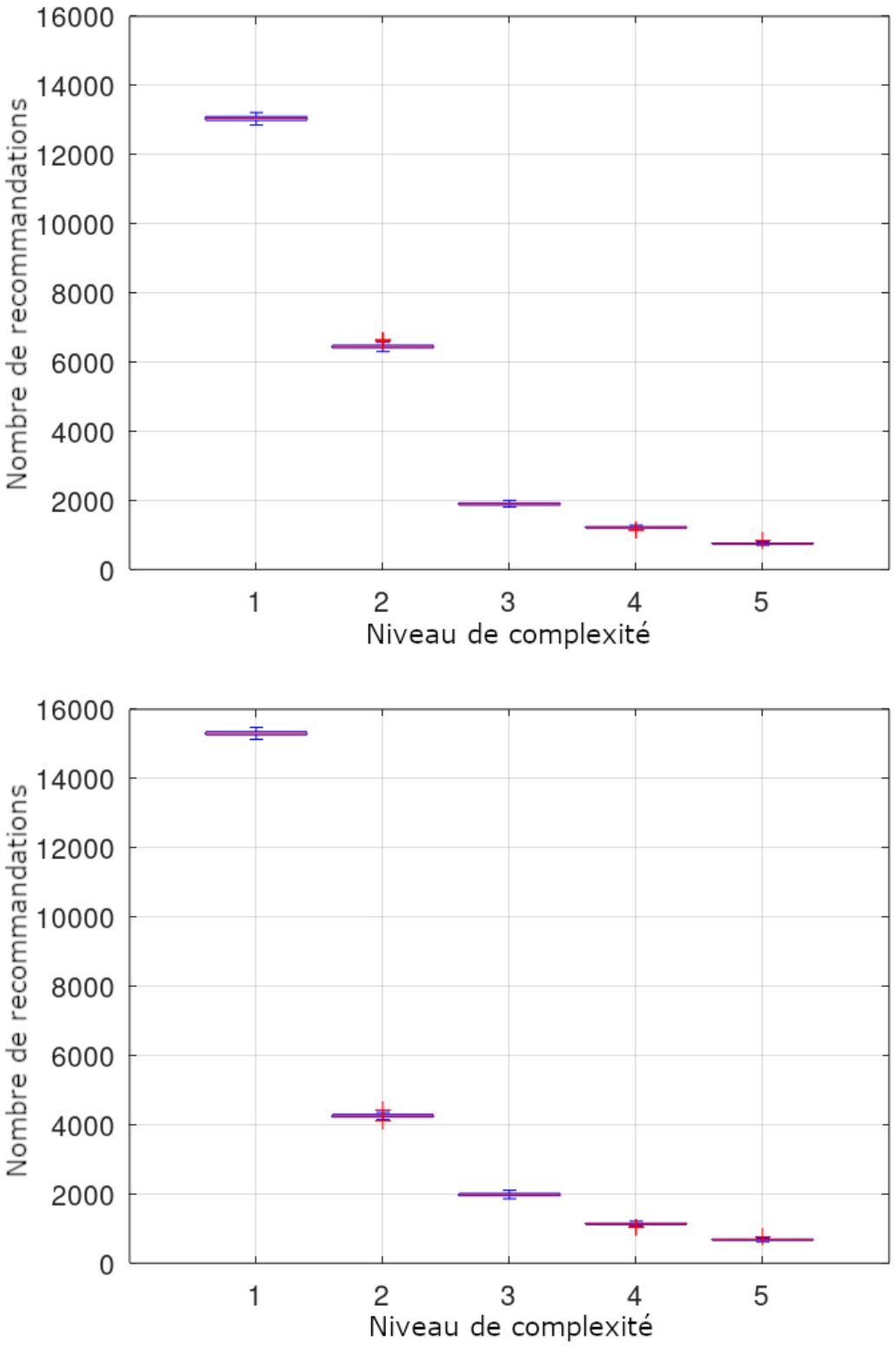

FIGURE 7.15 – Nombre de recommandations par niveau de complexité (processus d'apprentissage statique en haut, processus d'apprentissage dynamique avec processus de Hawkes en bas)

Un scénario spécifique a été défini sans données initiales (notes et temps de réponse) en simulant un démarrage à froid. Le tableau 7.11 montre les résultats obtenus après 10000 exécutions pour TS et TS associé au processus de Hawkes. Malgré les changements dans chaque niveau de complexité, les deux approches donnent des résultats similaires.

	r_{C0}	r_{C1}	r_{C2}	r_{C3}	r_{C4}	Total	Percent
TS	0.9695	0.7945	0.6377	0.5615	0.5184	3.4816	69.632
TS-Hawkes	0.9414	0.7175	0.6434	0.5761	0.5448	3.4232	68.464

TABLE 7.11 – Comparaison entre ESCBR-TS et ESCBR-TS-Hawkes lors d'un démarrage à froid.

La variance (figure 7.16) montre qu'avec le processus de Hawkes, les valeurs sont maintenues autour de la configuration initiale, ce qui permet une plus grande adaptabilité aux changements dynamiques des connaissances qui se produisent dans le processus d'apprentissage. Étant donné que la distribution de probabilité Beta converge rapidement vers une valeur unique, plus on obtient de valeurs, plus la variance est faible et s'il y a un changement dans la valeur de convergence, la distribution a besoin de plus de données pour converger vers la nouvelle valeur. Ce dernier comportement peut être expliqué par le fait que les changements dans la moyenne sont proportionnels à la valeur de la variance avec un pas de changement constant pour les paramètres. Dans le cas de la modélisation du processus d'apprentissage, il est donc préférable de maintenir une valeur de variance relativement élevée pour faciliter l'adaptation aux changements imprévus.

7.4.3/ CONCLUSION

Cette partie a présenté un algorithme intégrant un système de recommandation fondé sur l'algorithme d'échantillonnage de Thompson, un algorithme de régression d'ensemble fondé sur le raisonnement par cas et système multi-agents, et une fonction d'oubli fondée sur le processus de Hawkes.

L'algorithme de régression d'ensemble ESCBR-SMA permet d'interpoler la note qu'un apprenant aurait à la question suivante selon le niveau de complexité de celle-ci. L'échantillonnage de Thompson permet de recommander la complexité de la question suivante en fonction d'informations locales : les notes obtenues et les temps de réponses de cet apprenant aux questions précédentes. Enfin, le processus de Hawkes incite le système à proposer des rappels de compétences acquises en intégrant un modèle d'oubli.

Nous avons mesuré ses performances sur un ensemble de données générées permettant de simuler les notes et temps de réponses obtenues par 1000 apprenants sur un EIAH proposant des exercices répartis sur cinq niveaux de complexité et aussi avec une base de données réel non-symétrique de 23366 apprenants. Ces tests ont été réalisés en vue d'une intégration à l'EIAH AI-VT. Les résultats montrent que le module final permet de réviser les séances d'entrainement, de proposer des exercices progressivement plus difficiles, tout en intégrant des rappels de notions.

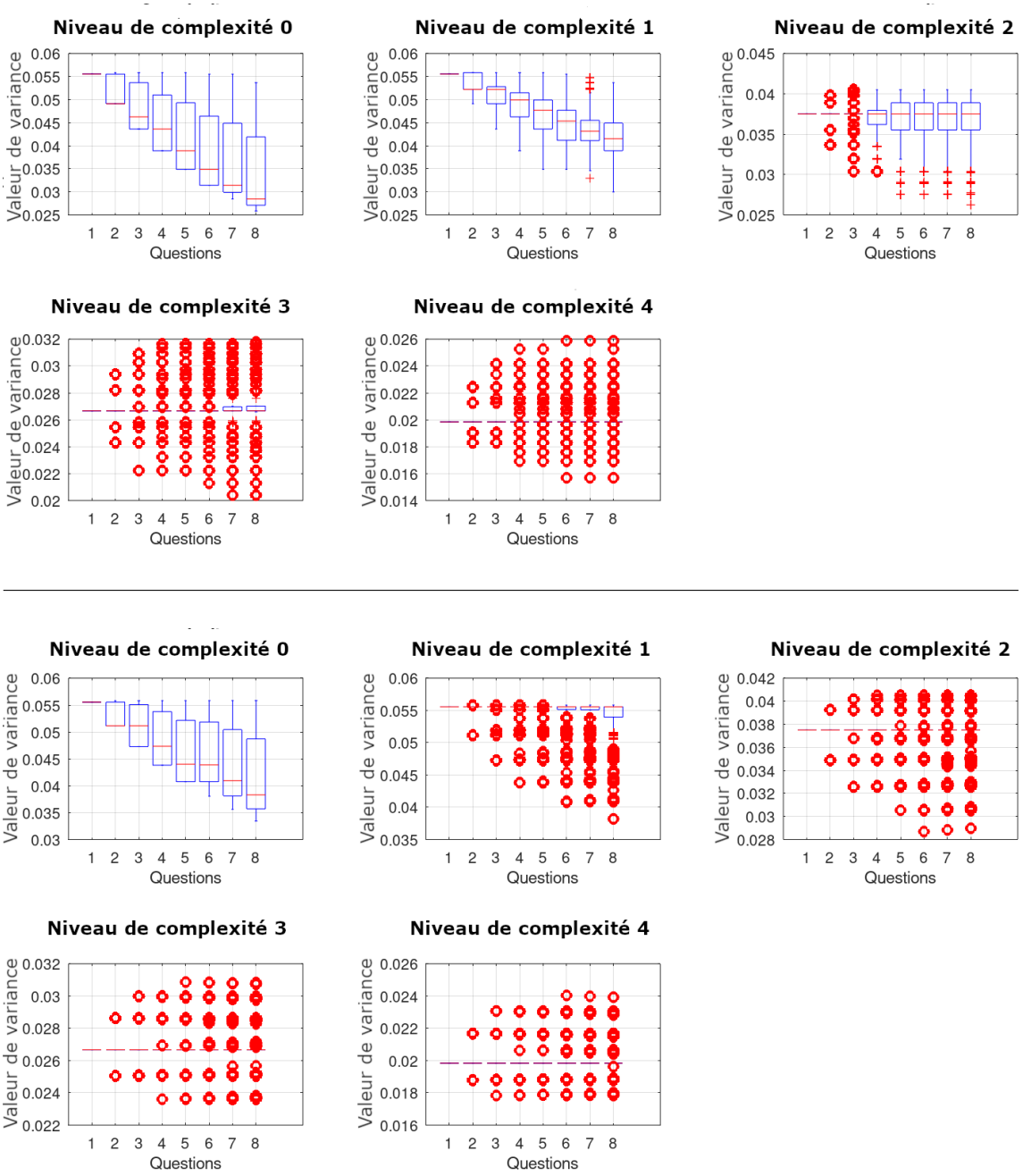

FIGURE 7.16 – Variance pour la distribution de probabilité bêta et tous les niveaux de complexité (en haut : processus d'apprentissage statique. En bas : processus d'apprentissage dynamique avec processus de Hawkes)

8

CONCLUSIONS ET PERSPECTIVES

8.1/ CONCLUSION GÉNÉRALE

Les environnements informatiques pour l'apprentissage humain (EIAH) doivent trouver des stratégies adaptées afin d'exploiter et d'analyser toute l'information récoltée sur l'évolution de l'apprenant et ses éventuelles lacunes. L'intégration d'un système de recommandation en temps réel proposant des alternatives de parcours dynamiques et variées aux apprenants présente donc de nombreux avantages.

Nous avons montré dans ces travaux de quelle manière les outils d'IA et d'apprentissage automatique peuvent répondre à cette problématique. Ils permettent en effet de travailler avec des données structurées selon un contexte déterminé, d'explorer et d'exploiter les espaces où ces données sont définies, d'analyser et d'extraire l'information utile pour connaître les comportements, les tendances et les faiblesses des apprenants. Parmi tous les outils d'IA, nous nous sommes plus particulièrement tournés vers ceux dont le fonctionnement nécessite peu de données car ils permettent à des apprenants inconnus de l'EIAH de se voir proposer des adaptations en temps réel dès les premiers exercices.

L'EIAH AI-VT qui a servi d'environnement d'application et d'expérimentation à ces travaux est donc aujourd'hui capable de proposer des recommandations dynamiques et personnalisées en utilisant l'apprentissage par renforcement, la génération de solutions stochastique, le raisonnement Bayésien, les approches d'ensemble avec plusieurs algorithmes via deux empilements, l'optimisation stochastique itérative, l'intégration d'approches locales et collaboratives, le raisonnement à partir de cas avec les systèmes multi-agents et le processus de Hawkes.

Les jeux de données testés (réelles et générées) ont permis de tester la solidité du système, son amélioration progressive et l'unification de tous les algorithmes, outils et modules proposés. Deux métriques ont également été proposés afin de mieux mesurer le degré d'adaptabilité du système au regard de chaque apprenant.

à un endroit tu parles de parcours stantard, est ce que c'est défini, paramétrable ?

La version finale du système AI-VT qui intègre tous les modules développés est capable de prendre en compte des aspects importants du processus d'apprentissage tels que le renforcement des connaissances, la dynamique de l'information, la variation progression, le temps d'apprentissage non-linéaire en étant capable de suivre un algorithme intégrant une courbe d'oubli.

Chaque module proposé s'est montré compétitif et a présenté des performances de qualités comparables à d'autres algorithmes et modules de référence dans le domaine de la régression et de la recommandation. L'architecture proposée permet de tirer partie des avantages de chacun de ces modules et ainsi d'améliorer globalement l'adaptation en cours de séance d'entraînement dans le système AI-VT en fonction des résultats partiels des apprenants.

Au regard des problématiques soulevées en introduction de ce manuscrit, voici les contributions apportées par ces travaux de recherche.

Résumé de la conclusion du chapitre 5

Résumé de la conclusion du chapitre 6

Résumé de la conclusion du chapitre 7

8.2/ PERSPECTIVES

Différentes perspectives peuvent être envisagées. En particulier, l'intégration de résultats d'analyse vidéo et audio pourrait aider à mieux interpréter les comportements des apprenants et ainsi leur proposer des recommandations encore plus pertinentes.

Il pourrait également être intéressant de proposer différentes configurations pour le système de recommandation. Le point de départ pourrait consister à étudier la possibilité d'améliorer encore la fonction déterminant quelles sont les meilleures solutions proposées au second niveau d'empilement de l'outil d'ensemble (ESCBR-SMA). Peut-être intéressant de pouvoir paramétrer cette fonction, qu'AI-VT en propose plusieurs et qu'elles soient dépendantes du profil de l'apprenant et des matières enseignées.

Les scénarios d'apprentissage testés dans ces travaux ont mis en lumière le fait qu'il existait des points de rupture. Il serait intéressant de les analyser afin de mieux les prédire. Ces travaux ont notamment proposé d'utiliser un échantillonnage de Thompson pour que les modifications des niveaux de complexité des exercices proposés par le système soient progressifs. Cet outil est fondé sur une distribution de probabilité. Il pourrait donc être envisagé de tester d'autres familles de distributions de probabilité et de calculer de manière dynamique les taux corrélés d'actualisation des distributions de probabilité pour chaque niveau de complexité.

Une autre piste de travail est la correction automatique. Cet aspect n'est pas encore proposé dans AI-VT. Il pourrait l'être en considérant des algorithmes ou modèles intégrant des gabarits ou même des cartographies d'erreurs ou de réponses incorrectes à la base de données de l'EIAH.

Pour finir, il pourrait être intéressant de définir des intervalles pour chaque paramètre et exécuter un analyse de sensibilité pour évaluer les résultats avec l'objectif d'estimer la stabilité du système et les limites numériques.

9

PUBLICATIONS

Pendant le travail de cette thèse, plusieurs articles ont été produits en explicitant les contributions réalisées. D'autres publications ont été produites en marge du sujet de cette thèse dans le cadre de la recherche en informatique.

9.1/ PUBLICATIONS LIÉES AU SUJET DE THÈSE

Daniel Soto Forero, Julien Henriet and Marie-Laure Betbeder. Stochastic Recommendation and Case-Based Reasoning Model Applied to the Intelligent Tutoring System AI-VT. (En évaluation)

Daniel Soto Forero, Julien Henriet and Marie-Laure Betbeder. Integration of Stacking Case-Based Reasoning with a Multi-Agent System Applied to Regression Problems. (En évaluation) 2025.

Daniel Soto Forero, Julien Henriet and Marie-Laure Betbeder. Ensemble Stacking Case-Based Reasoning and a Stochastic Recommender Algorithm with the Hawkes Process Applied to ITS AI-VT. (ITS - International Conference on Intelligent Tutoring Systems) 2025.

Daniel Soto Forero, Julien Henriet and Marie-Laure Betbeder. Modèle de Recommandation Stochastique et de Raisonnement à Partir de Cas Appliqué à AI-VT. (EIAH - Conférence sur les Environnements Informatiques pour l'Apprentissage Humain) 2025.

Daniel Soto Forero, Simha Ackermann, Marie-Laure Betbeder and Julien Henriet. The Intelligent Tutoring System AI-VT with Case-Based Reasoning and Real Time Recommender Models. International Conference on Case Based Reasoning (ICCBR - International Conference on Case-Based Reasoning). 2024.

Daniel Soto Forero, Marie-Laure Betbeder and Julien Henriet. Ensemble Stacking Case-Based Reasoning for Regression. International Conference on Case Based Reasoning (ICCBR - International Conference on Case-Based Reasoning). 2024.

Daniel Soto Forero, Simha Ackermann, Marie-Laure Betbeder and Julien Henriet. Automatic Real-Time Adaptation of Training Session Difficulty Using Rules and Reinforcement Learning in the AI-VT ITS. International Journal of Modern Education and Computer Science (IJMECS - International Journal of Modern Education and Computer Science). 2024.

9.2/ AUTRES PUBLICATIONS

Daniel Soto Forero and Yony Ceballos. Metaheuristics Hybridation and Parametrization using Machine Learning (En évaluation). 2025.

Daniel Soto Forero and Wilson Soto Forero. Efficient Exploration in Unknown Environments : An Adaptative Multi-Agent Approach. 19th Iberian Conference on Information Systems and Technologies. 2025.

Daniel Soto Forero and Yony Ceballos. Stochastic agent-based models optimization applied to the problem of rebalancing bike-share systems. International Journal of Electrical and Computer Engineering. 2024.

Daniel Soto Forero and Wilson Soto Forero. A Multi-Agent Based Algorithm for Quadratic Assignment Problem. IEEE Latin American Conference on Computational Intelligence. 2023.

BIBLIOGRAPHIE

[UCI, 2024] (2024). Markelle kelly, rachel longjohn, kolby nottingham, the uci machine learning repository. https://archive.ics.uci.edu. Accessed : 2024-09-30.

[Aamodt and Plaza, 1994] Aamodt, A. and Plaza, E. (1994). Case-based reasoning : Foundational issues, methodological variations, and system approaches. *AI Communications*, 7(1) :39–59.

[Abel et al., 2023] Abel, D., Barreto, A., Van Roy, B., Precup, D., van Hasselt, H. P., and Singh, S. (2023). A definition of continual reinforcement learning. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S., editors, *Advances in Neural Information Processing Systems*, volume 36, pages 50377–50407. Curran Associates, Inc.

[Alabdulrahman and Viktor, 2021] Alabdulrahman, R. and Viktor, H. (2021). Catering for unique tastes : Targeting grey-sheep users recommender systems through one-class machine learning. *Expert Systems with Applications*, 166 :114061.

[Arthurs et al., 2019] Arthurs, N., Stenhaug, B., Karayev, S., and Piech, C. (2019). Grades are not normal : Improving exam score models using the logit-normal distribution. In *International Conference on Educational Data Mining (EDM)*, page 6.

[Auer et al., 2021] Auer, F., Lenarduzzi, V., Felderer, M., and Taibi, D. (2021). From monolithic systems to microservices : An assessment framework. *Information and Software Technology*, 137 :106600.

[Badier et al., 2023] Badier, A., Lefort, M., and Lefevre, M. (2023). Comprendre les usages et effets d'un système de recommandations pédagogiques en contexte d'apprentissage non-formel. In *EIAH'23*, Brest, France.

[Bakurov et al., 2021] Bakurov, I., Castelli, M., Gau, O., Fontanella, F., and Vanneschi, L. (2021). Genetic programming for stacked generalization. *Swarm and Evolutionary Computation*, 65 :100913.

[Butdee and Tichkiewitch, 2011] Butdee, S. and Tichkiewitch, S. (2011). Case-based reasoning for adaptive aluminum extrusion die design together with parameters by neural networks. In Bernard, A., editor, *Global Product Development*, pages 491–496, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Chiu et al., 2023] Chiu, T. K., Xia, Q., Zhou, X., Chai, C. S., and Cheng, M. (2023). Systematic literature review on opportunities, challenges, and future research recommendations of artificial intelligence in education. *Computers and Education : Artificial Intelligence*, 4 :100118.

[Choi et al., 2023] Choi, J., Suh, D., and Otto, M.-O. (2023). Boosted stacking ensemble machine learning method for wafer map pattern classification. *Computers, Materials & Continua*, 74(2) :2945–2966.

[C.K. and R.C., 1989] C.K., R. and R.C., S. (1989). *Inside Case-Based Reasoning*. Psychology Press.

[Cunningham and Delany, 2021] Cunningham, P. and Delany, S. J. (2021). K-nearest neighbour classifiers - a tutorial. *ACM Comput. Surv.*, 54(6).

[Didden et al., 2023] Didden, J. B., Dang, Q.-V., and Adan, I. J. (2023). Decentralized learning multi-agent system for online machine shop scheduling problem. *Journal of Manufacturing Systems*, 67 :338–360.

[Ezaldeen et al., 2022] Ezaldeen, H., Misra, R., Bisoy, S. K., Alatrash, R., and Priyadarshini, R. (2022). A hybrid e-learning recommendation integrating adaptive profiling and sentiment analysis. *Journal of Web Semantics*, 72 :100700.

[Feely et al., 2020] Feely, C., Caulfield, B., Lawlor, A., and Smyth, B. (2020). Using case-based reasoning to predict marathon performance and recommend tailored training plans. In Watson, I. and Weber, R., editors, *Case-Based Reasoning Research and Development*, pages 67–81, Cham. Springer International Publishing.

[Grace et al., 2016] Grace, K., Maher, M. L., Wilson, D. C., and Najjar, N. A. (2016). Combining cbr and deep learning to generate surprising recipe designs. In Goel, A., Díaz-Agudo, M. B., and Roth-Berghofer, T., editors, *Case-Based Reasoning Research and Development*, pages 154–169, Cham. Springer International Publishing.

[Gupta et al., 2021] Gupta, S., Chaudhari, S., Joshi, G., and Yağan, O. (2021). Multi-armed bandits with correlated arms. *IEEE Transactions on Information Theory*, 67(10) :6711–6732.

[Hajduk et al., 2019] Hajduk, M., Sukop, M., and Haun, M. (2019). *Cognitive Multi-agent Systems : Structures, Strategies and Applications to Mobile Robotics and Robosoccer*. Studies in Systems, Decision and Control. Springer International Publishing.

[Henriet et al., 2017] Henriet, J., Christophe, L., and Laurent, P. (2017). Artificial intelligence-virtual trainer : An educative system based on artificial intelligence and designed to produce varied and consistent training lessons. *Proceedings of the Institution of Mechanical Engineers, Part P : Journal of Sports Engineering and Technology*, 231(2) :110–124.

[Henriet and Greffier, 2018] Henriet, J. and Greffier, F. (2018). Ai-vt : An example of cbr that generates a variety of solutions to the same problem. In Cox, M. T., Funk, P., and Begum, S., editors, *Case-Based Reasoning Research and Development*, pages 124–139, Cham. Springer International Publishing.

[Hipólito and Kirchhoff, 2023] Hipólito, I. and Kirchhoff, M. (2023). Breaking boundaries : The bayesian brain hypothesis for perception and prediction. *Consciousness and Cognition*, 111 :103510.

[Hoang, 2018] Hoang, L. (2018). *La formule du savoir. Une philosophie unifiée du savoir fondée sur le théorème de Bayes*. EDP Sciences.

[Hu et al., 2025] Hu, B., Ma, Y., Liu, Z., and Wang, H. (2025). A social importance and category enhanced cold-start user recommendation system. *Expert Systems with Applications*, 277 :127130.

[Huang et al., 2023] Huang, A. Y., Lu, O. H., and Yang, S. J. (2023). Effects of artificial intelligence–enabled personalized recommendations on learners' learning engagement, motivation, and outcomes in a flipped classroom. *Computers and Education*, 194 :104684.

[Ingkavara et al., 2022] Ingkavara, T., Panjaburee, P., Srisawasdi, N., and Sajjapanroj, S. (2022). The use of a personalized learning approach to implementing self-regulated online learning. *Computers and Education : Artificial Intelligence*, 3 :100086.

[Jean-Daubias, 2011] Jean-Daubias, S. (2011). Ingénierie des profils d'apprenants.

[Jung et al., 2009] Jung, S., Lim, T., and Kim, D. (2009). Integrating radial basis function networks with case-based reasoning for product design. *Expert Systems with Applications*, 36(3, Part 1) :5695–5701.

[Kamali et al., 2023] Kamali, S. R., Banirostam, T., Motameni, H., and Teshnehlab, M. (2023). An immune inspired multi-agent system for dynamic multi-objective optimization. *Knowledge-Based Systems*, 262 :110242.

[Kim, 2024] Kim, W. (2024). A random focusing method with jensen–shannon divergence for improving deep neural network performance ensuring architecture consistency. *Neural Processing Letters*, 56(4) :199.

[Kolodner, 1983] Kolodner, J. L. (1983). Reconstructive memory : A computer model. *Cognitive Science*, 7(4) :281–328.

[Kuzilek et al., 2017] Kuzilek, J., Hlosta, M., and Zdrahal, Z. (2017). Open university learning analytics dataset. *Scientific Data*, 4(1) :170171.

[Lalitha and Sreeja, 2020] Lalitha, T. B. and Sreeja, P. S. (2020). Personalised self-directed learning recommendation system. *Procedia Computer Science*, 171 :583–592. Third International Conference on Computing and Network Communications (CoCoNet'19).

[Lei, 2024] Lei, Z. (2024). Analysis of simpson's paradox and its applications. *Highlights in Science, Engineering and Technology*, 88 :357–362.

[Leikola et al., 2018] Leikola, M., Sauer, C., Rintala, L., Aromaa, J., and Lundström, M. (2018). Assessing the similarity of cyanide-free gold leaching processes : A case-based reasoning application. *Minerals*, 8(10).

[Lepage et al., 2020] Lepage, Y., Lieber, J., Mornard, I., Nauer, E., Romary, J., and Sies, R. (2020). The french correction : When retrieval is harder to specify than adaptation. In Watson, I. and Weber, R., editors, *Case-Based Reasoning Research and Development*, pages 309–324, Cham. Springer International Publishing.

[Li et al., 2024] Li, Z., Ding, Z., Yu, Y., and Zhang, P. (2024). The kullback–leibler divergence and the convergence rate of fast covariance matrix estimators in galaxy clustering analysis. *The Astrophysical Journal*, 965(2) :125.

[Liang et al., 2021] Liang, M., Chang, T., An, B., Duan, X., Du, L., Wang, X., Miao, J., Xu, L., Gao, X., Zhang, L., Li, J., and Gao, H. (2021). A stacking ensemble learning framework for genomic prediction. *Frontiers in Genetics*, 12.

[Lin, 2022] Lin, B. (2022). Evolutionary multi-armed bandits with genetic thompson sampling. In *2022 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8.

[Liu and Yu, 2023] Liu, M. and Yu, D. (2023). Towards intelligent e-learning systems. *Education and Information Technologies*, 28(7) :7845–7876.

[Louvros et al., 2023] Louvros, P., Stefanidis, F., Boulougouris, E., Komianos, A., and Vassalos, D. (2023). Machine learning and case-based reasoning for real-time onboard prediction of the survivability of ships. *Journal of Marine Science and Engineering*, 11(5).

[Maher and Grace, 2017] Maher, M. L. and Grace, K. (2017). Encouraging curiosity in case-based reasoning and recommender systems. In Aha, D. W. and Lieber, J., editors, *Case-Based Reasoning Research and Development*, pages 3–15, Cham. Springer International Publishing.

[Malburg et al., 2024] Malburg, L., Hotz, M., and Bergmann, R. (2024). Improving complex adaptations in process-oriented case-based reasoning by applying rule-based adaptation. In Recio-Garcia, J. A., Orozco-del Castillo, M. G., and Bridge, D., editors, *Case-Based Reasoning Research and Development*, pages 50–66, Cham. Springer Nature Switzerland.

[Mang et al., 2021] Mang, L., Tianpeng, C., Bingxing, A., Xinghai, D., Lili, D., Xiaoqiao, W., Jian, M., Lingyang, X., Xue, G., Lupei, Z., Junya, L., and Huijiang, G. (2021). A stacking ensemble learning framework for genomic prediction. *Frontiers in Genetics*.

[Minsker and Strawn, 2024] Minsker, S. and Strawn, N. (2024). The geometric median and applications to robust mean estimation. *SIAM Journal on Mathematics of Data Science*, 6(2) :504–533.

[Muangprathub et al., 2020] Muangprathub, J., Boonjing, V., and Chamnongthai, K. (2020). Learning recommendation with formal concept analysis for intelligent tutoring system. *Heliyon*, 6(10) :e05227.

[Müller and Bergmann, 2015] Müller, G. and Bergmann, R. (2015). Cookingcake : A framework for the adaptation of cooking recipes represented as workflows. In *International Conference on Case-Based Reasoning*.

[Nguyen, 2024] Nguyen, A. (2024). Dynamic metaheuristic selection via thompson sampling for online optimization. *Applied Soft Computing*, 158 :111566.

[Nkambou et al., 2010] Nkambou, R., Bourdeau, J., and Mizoguchi, R. (2010). *Advances in Intelligent Tutoring Systems*. Springer Berlin, Heidelberg, 1 edition.

[Obeid et al., 2022] Obeid, C., Lahoud, C., Khoury, H. E., and Champin, P. (2022). A novel hybrid recommender system approach for student academic advising named cohrs, supported by case-based reasoning and ontology. *Computer Science and Information Systems*, 19(2) :979–1005.

[Ontañón et al., 2015] Ontañón, S., Plaza, E., and Zhu, J. (2015). Argument-based case revision in cbr for story generation. In Hüllermeier, E. and Minor, M., editors, *Case-Based Reasoning Research and Development*, pages 290–305, Cham. Springer International Publishing.

[Ou et al., 2024] Ou, T., Cummings, R., and Avella Medina, M. (2024). Thompson sampling itself is differentially private. In Dasgupta, S., Mandt, S., and Li, Y., editors, *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pages 1576–1584. PMLR.

[Parejas-Llanovarced et al., 2024] Parejas-Llanovarced, H., Caro-Martínez, M., del Castillo, M. G. O., and Recio-García, J. A. (2024). Case-based selection of explanation methods for neural network image classifiers. *Knowledge-Based Systems*, 288 :111469.

[Petrovic et al., 2016] Petrovic, S., Khussainova, G., and Jagannathan, R. (2016). Knowledge-light adaptation approaches in case-based reasoning for radiotherapy treatment planning. *Artificial Intelligence in Medicine*, 68 :17–28.

[Richter and Weber, 2013] Richter, M. and Weber, R. (2013). *Case-Based Reasoning (A Textbook)*. Springer-Verlag GmbH.

[Richter, 2009] Richter, M. M. (2009). The search for knowledge, contexts, and case-based reasoning. *Engineering Applications of Artificial Intelligence*, 22(1) :3–9.

[Robertson and Watson, 2014] Robertson, G. and Watson, I. D. (2014). A review of real-time strategy game ai. *AI Mag.*, 35 :75–104.

[Roldan Reyes et al., 2015] Roldan Reyes, E., Negny, S., Cortes Robles, G., and Le Lann, J. (2015). Improvement of online adaptation knowledge acquisition and reuse in case-based reasoning : Application to process engineering design. *Engineering Applications of Artificial Intelligence*, 41 :1–16.

[Sadeghi Moghadam et al., 2024] Sadeghi Moghadam, M. R., Jafarnejad, A., Heidary Dahooie, J., and Ghasemian Sahebi, I. (2024). A hidden markov model based extended case-based reasoning algorithm for relief materials demand forecasting. *Mathematics Interdisciplinary Research*, 9(1) :89–109.

[Schank and Abelson, 1977] Schank, R. C. and Abelson, R. P. (1977). *Scripts, Plans, Goals and Understanding : an Inquiry into Human Knowledge Structures*. L. Erlbaum, Hillsdale, NJ.

[Seznec et al., 2020] Seznec, J., Menard, P., Lazaric, A., and Valko, M. (2020). A single algorithm for both restless and rested rotting bandits. In Chiappa, S. and Calandra, R., editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 3784–3794. PMLR.

[Sinaga and Yang, 2020] Sinaga, K. P. and Yang, M.-S. (2020). Unsupervised k-means clustering algorithm. *IEEE Access*, 8 :80716–80727.

[Skittou et al., 2024] Skittou, M., Merrouchi, M., and Gadi, T. (2024). A recommender system for educational planning. *Cybernetics and Information Technologies*, 24(2) :67–85.

[Smyth and Cunningham, 2018] Smyth, B. and Cunningham, P. (2018). An analysis of case representations for marathon race prediction and planning. In Cox, M. T., Funk, P., and Begum, S., editors, *Case-Based Reasoning Research and Development*, pages 369–384, Cham. Springer International Publishing.

[Smyth and Willemsen, 2020] Smyth, B. and Willemsen, M. C. (2020). Predicting the personal-best times of speed skaters using case-based reasoning. In Watson, I. and Weber, R., editors, *Case-Based Reasoning Research and Development*, pages 112–126, Cham. Springer International Publishing.

[Soto-Forero et al., 2024a] Soto-Forero, D., Ackermann, S., Betbeder, M.-L., and Henriet, J. (2024a). Automatic real-time adaptation of training session difficulty using rules and reinforcement learning in the ai-vt its. *International Journal of Modern Education and Computer Science(IJMECS)*, 16 :56–71.

[Soto-Forero et al., 2024b] Soto-Forero, D., Ackermann, S., Betbeder, M.-L., and Henriet, J. (2024b). The intelligent tutoring system ai-vt with case-based reasoning and real time recommender models. In Recio-Garcia, J. A., Orozco-del Castillo, M. G., and Bridge, D., editors, *Case-Based Reasoning Research and Development*, pages 191–205, Cham. Springer Nature Switzerland.

[Soto-Forero et al., 2024c] Soto-Forero, D., Betbeder, M.-L., and Henriet, J. (2024c). Ensemble stacking case-based reasoning for regression. In Recio-Garcia, J. A., Orozco-del Castillo, M. G., and Bridge, D., editors, *Case-Based Reasoning Research and Development*, pages 159–174, Cham. Springer Nature Switzerland.

[Su et al., 2022] Su, Y., Cheng, Z., Wu, J., Dong, Y., Huang, Z., Wu, L., Chen, E., Wang, S., and Xie, F. (2022). Graph-based cognitive diagnosis for intelligent tutoring systems. *Knowledge-Based Systems*, 253 :109547.

[Supic, 2018] Supic, H. (2018). Case-based reasoning model for personalized learning path recommendation in example-based learning activities. In *2018 IEEE 27th International Conference on Enabling Technologies : Infrastructure for Collaborative Enterprises (WETICE)*, pages 175–178.

[Uguina et al., 2024] Uguina, A. R., Gomez, J. F., Panadero, J., Martínez-Gavara, A., and Juan, A. A. (2024). A learnheuristic algorithm based on thompson sampling for the heterogeneous and dynamic team orienteering problem. *Mathematics*, 12(11).

[Uysal and Sonmez, 2023] Uysal, F. and Sonmez, R. (2023). Bootstrap aggregated case-based reasoning method for conceptual cost estimation. *Buildings*, 13(3).

[Wang et al., 2021] Wang, F., Liao, F., Li, Y., and Wang, H. (2021). A new prediction strategy for dynamic multi-objective optimization using gaussian mixture model. *Information Sciences*, 580 :331–351.

[Wolf et al., 2024] Wolf, T. N., Bongratz, F., Rickmann, A.-M., Pölsterl, S., and Wachinger, C. (2024). Keep the faith : Faithful explanations in convolutional neural networks for case-based reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 5921–5929.

[Xu et al., 2021] Xu, S., Cai, W., Xia, H., Liu, B., and Xu, J. (2021). Dynamic metric accelerated method for fuzzy clustering. *IEEE Access*, 9 :166838–166854.

[Xu et al., 2023] Xu, S., Ge, Y., Li, Y., Fu, Z., Chen, X., and Zhang, Y. (2023). Causal collaborative filtering. In *Proceedings of the 2023 ACM SIGIR International Conference on Theory of Information Retrieval*, ICTIR '23, page 235–245, New York, NY, USA. Association for Computing Machinery.

[Yu and Li, 2023] Yu, L. and Li, M. (2023). A case-based reasoning driven ensemble learning paradigm for financial distress prediction with missing data. *Applied Soft Computing*, 137 :110163.

[Yu et al., 2024] Yu, L., Li, M., and Liu, X. (2024). A two-stage case-based reasoning driven classification paradigm for financial distress prediction with missing and imbalanced data. *Expert Systems with Applications*, 249 :123745.

[Zhang. and Aslan, 2021] Zhang., K. and Aslan, A. B. (2021). Ai technologies for education : Recent research and future directions. *Computers and Education : Artificial Intelligence*, 2 :100025.

[Zhang and Yao, 2018] Zhang, K. and Yao, Y. (2018). A three learning states bayesian knowledge tracing model. *Knowledge-Based Systems*, 148 :189–201.

[Zhang et al., 2023] Zhang, P., Wang, L., Fei, Z., Wei, L., Fei, M., and Menhas, M. I. (2023). A novel human learning optimization algorithm with bayesian inference learning. *Knowledge-Based Systems*, 271 :110564.

[Zhao et al., 2023] Zhao, L.-T., Wang, D.-S., Liang, F.-Y., and Chen, J. (2023). A recommendation system for effective learning strategies : An integrated approach using context-dependent dea. *Expert Systems with Applications*, 211 :118535.

[Zhou and Wang, 2021] Zhou, L. and Wang, C. (2021). Research on recommendation of personalized exercises in english learning based on data mining. *Scientific Programming*, 2021 :5042286.

[Zuluaga et al., 2022] Zuluaga, C. A., Aristizábal, L. M., Rúa, S., Franco, D. A., Osorio, D. A., and Vásquez, R. E. (2022). Development of a modular software architecture for underwater vehicles using systems engineering. *Journal of Marine Science and Engineering*, 10(4).

Document réalisé avec LaTeX et :
le style LaTeX pour Thèse de Doctorat créé par S. Galland — http://www.multiagent.fr/ThesisStyle
la collection de paquets `tex-upmethodology` — http://www.arakhne.org/tex-upmethodology/

Résumé :

L'objectif de ce travail de thèse est la prise en compte en temps réel du travail d'un apprenant pour une meilleure personnalisation de l'environnement informatique pour l'apprentissage humain AI-VT (Artificial Intelligence Virtual Trainer).
Certains des problèmes les plus courants et les plus typiques dans le domaine des environnements informatiques d'apprentissage humain (EIAH) sont (i) l'identification correcte des difficultés des apprenants dans le processus d'apprentissage, (ii) l'adaptation du contenu ou de la présentation du système en fonction des difficultés rencontrées, et (iii) la capacité à s'adapter sans données initiales (démarrage à froid). Dans certains cas, le système tolère des modifications après la réalisation et l'évaluation des compétences. D'autres systèmes nécessitent une adaptation compliquée en temps réel car seul un nombre limité de données peut être capturé. Dans ce cas, elles doivent être analysées correctement et avec une certaine précision afin d'obtenir les adaptations appropriées. L'architecture proposée et les modules développés avec techniques d'intelligence artificielle fonctionnent de manière associée en exploitant les avantages de chacun pour obtenir des propositions d'adaptation performantes selon l'évolution des connaissances des apprenants, permettant ainsi l'acquisition des connaissances et la progression dans le processus d'apprentissage.

Mots-clés : Environnements informatiques d'apprentissage humain, Intelligence artificielle, raisonnement à partir de cas, systèmes multi-agents, raisonnement bayésien, échantillonage de Thompson, processus de Hawkes

Abstract:

The aim of this thesis work is to take into account in real time a learner's work for a better personalization of the intelligent tutoring system named AI-VT (Artificial Intelligence Virtual Trainer).
Some of the most common and typical problems in the field of intelligent tutoring systems (ITS) are (i) correctly identifying learners' difficulties in the learning process, (ii) adapting the content or presentation of the system according to the difficulties encountered, and (iii) the ability to adapt without initial data (cold start). In some cases, the system tolerates modifications after the skills have been acquired and assessed. Other systems require complicated real-time adaptation, as only a limited amount of data can be captured. In this case, they must be analyzed correctly and with a certain degree of precision in order to obtain the appropriate adaptations. The proposed architecture and the modules developed with artificial intelligence techniques work in tandem, exploiting the advantages of each to obtain effective adaptation proposals according to the evolution of learners' knowledge, thus enabling knowledge acquisition and progression in the learning process.

Keywords: Intelligent tutoring systems, Artificial intelligence, Case-based reasoning, Multi-agent systems, Bayesian reasoning, Thompson sampling, Hawkes process

SPIM

■ École doctorale SPIM 16 route de Gray F - 25030 Besançon cedex
■ tél. +33 (0)3 81 66 66 02 ■ ed-spim@univ-fcomte.fr ■ www.ed-spim.univ-fcomte.fr

UBFC