Recommanding sequences in a multidimensional space

Pierre-Edouard Osche

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Recommandations en séquences dans un espace multidimensions

THÈSE

présentée et soutenue publiquement le 26 février 2021

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

Pierre-Edouard Osché

Composition du jury

Président :	Laurent Vigneron	Professeur, Université de Lorraine
Rapporteurs :	Sylvie Calabretto	Professeur, INSA de Lyon
	Laurent Vercouter	Professeur, INSA de Rouen
Directrice :	Anne Boyer	Professeur, Université de Lorraine
Co-Directeur :	Sylvain Castagnos	Maître de Conférences, Université de Lorraine

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thesul.

Remerciements

Ce manuscrit de thèse est le fruit d'un long travail de recherche qui a été riche en enseignements. Durant ces années, j'ai beaucoup appris tant du point de vue professionnel que personnel, plus que je n'aurais imaginé avant de commencer cette thèse. Pour tout cela, je souhaite remercier les personnes qui m'ont entouré durant cette période.

Je tiens tout d'abord à particulièrement remercier ma directrice Anne Boyer et mon co-directeur Sylvain Castagnos avec qui j'ai eu la chance de pouvoir travailler durant toute cette thèse. Anne m'a apporté son expérience et son recul dans le domaine de la recherche et Sylvain m'a guidé et soutenu dans chacune des étapes de ce travail. J'ai beaucoup apprécié travailler avec vous et je vous remercie encore une fois ici pour vos conseils avisés, votre patience, votre bonne humeur et pour tout ce que j'ai pu apprendre à vos côtés.

Je remercie Sylvie Calabretto et Laurent Vercouter qui ont rapporté ma thèse et qui ont accepté de faire partie du jury de thèse. Leurs remarques pertinentes m'ont donné un recul et une confiance nouvelle en mon travail. Je remercie Laurent Vigneron qui a apporté un regard extérieur et bienveillant sur mon travail pendant ces années de thèse et qui a également accepté de faire partie du jury comme examinateur.

Je remercie aussi l'équipe KIWI dans son ensemble. De nombreuses idées de ce travail sont le fruit de discussions avec les membres de l'équipe, dont certains sont devenus pour moi des amis proches.

Je remercie enfin ma famille et mes amis, leur soutien m'a donné la motivation de terminer ce travail dans cette période difficile de pandémie et d'isolation sociale. Tout particulièrement, je remercie Rose et Ragnar pour leur douce présence quotidienne.

Sommaire

Table des figures

Chapitre 1

Introduction

1.1 Introduction

Les systèmes de recommandation existent depuis maintenant plus de deux décennies [Resnick et al., 1994] et ont pris leur essor avec la démocratisation d'Internet et de l'accès à l'information. Le but d'un système de recommandation est, comme son nom l'indique, de recommander des ressources à l'utilisateur pour l'aiguiller parmi la masse d'informations disponibles. Le terme de ressource peut faire référence à un produit, une information, un film, une musique, etc. Ces ressources doivent correspondre à ce qu'attend l'utilisateur pour répondre à ses attentes. Là où d'autres méthodes d'aide à l'utilisateur, comme les moteurs de recherche, proposeront une liste de sites Internet relatifs à une question générale (ex. : la requête "films super-héros" résultera en une liste des films de super-héros les plus connus et/ou récents) ; un système de recommandation proposera quant à lui une réponse plus spécifique à un besoin plus précis (ex. : un système recommandera à un utilisateur de voir le dernier film de super-héros du studio Marvel car ce dernier a apprécié tous les films réalisés par ce studio spécifique ces dernières années). Une autre spécificité majeure des systèmes de recommandation réside dans le fait que ces derniers répondent à un besoin de l'utilisateur qui n'aura pas forcément été exprimé en amont par ce dernier. Contrairement aux moteurs de recherche qui se contentent de répondre à une requête de l'utilisateur, les systèmes de recommandation proposent quant à eux des informations de manière pro-active au moment le plus opportun. Un des défis majeurs des systèmes de recommandation, en plus de proposer des recommandations au bon moment, est de réussir à modéliser ce que l'utilisateur souhaite réellement alors que les seules données disponibles proviennent des traces laissées par les utilisateurs lors des précédentes interactions avec le système (consultation de pages, achat de produits, visionnage de films, etc.). Les systèmes de recommandation apportent une réelle plus-value par rapport à d'autres méthodes (comme les moteurs de recherche) car l'aide proposée se veut très personnalisée selon les préférences de chaque utilisateur. Cet objectif de personnalisation entraîne cependant une complexification du problème, car il est nécessaire de définir et de prendre en compte les facteurs humains sous-jacents à la satisfaction de l'utilisateur lors de la recherche d'information. Nous allons maintenant nous intéresser plus particulièrement aux différentes techniques employées par les systèmes de recommandation.

Les systèmes de recommandation existent depuis plus de deux décennies et, même si les techniques ont évolué, le principe fondamental reste le même : identifier les besoins et les préférences d'un utilisateur pour lui proposer des ressources adaptées. Pour ce faire, la première étape est de récolter des informations à propos de l'utilisateur dans le but d'identifier ses préférences.

Les informations récoltées sont dépendantes du domaine dans lequel est déployé le système de recommandation. Sur un site d'e-commerce, le système pourra par exemple utiliser les produits consultés par l'utilisateur ainsi que les éventuelles notes sur les produits qu'il a acheté. Un site d'écoute de musique exploitera quant à lui les titres écoutés par l'utilisateur ainsi que son comportement durant l'écoute (est-ce qu'il a écouté le titre jusqu'au bout ? etc.).

Durant ces deux décennies d'existence, les systèmes de recommandation ont changé pour accompagner l'évolution des usages d'Internet. Les premières générations de systèmes étaient naturellement adaptées aux ressources, peu nombreuses et complexes, typiques du début de l'ère d'Internet. C'est à cette époque que les deux grandes catégories de systèmes de recommandation se sont différenciées, à savoir le filtrage collaboratif [Resnick et al., 1994] et le filtrage par contenu [Belkin and Croft, 1992]. Dans le filtrage collaboratif, les préférences des utilisateurs du système sont utilisées (e.g. pour un site de recommandation de films : les notes données par les utilisateurs du site sur les films qu'ils ont vu). Ainsi, le système proposera à l'utilisateur souhaitant des recommandations les ressources appréciées des autres utilisateurs qui lui sont similaires, suivant l'hypothèse selon laquelle les préférences d'un ensemble d'utilisateurs similaires sont cohérentes entre elles. Dans le filtrage par contenu, les caractéristiques des ressources sont utilisées (e.g. pour un site de recommandation de films : genre, acteurs, année de sortie,...). Ainsi, le système proposera à l'utilisateur les ressources similaires à celles qu'il a appréciées précédemment. Pendant longtemps, ces deux techniques ont été évaluées sur la mesure de précision. Pour calculer cette mesure, les prédictions du système de recommandation sont directement confrontées aux données réelles recueillies avec de vrais utilisateurs. Par exemple, un système calcule qu'un utilisateur pourrait donner à un film une note de 4 sur 5, indiquant qu'il pourrait effectivement apprécier ce film et qu'il serait intéressant de lui recommander. Cette note calculée de 4/5 est ensuite comparée avec la note réelle que cet utilisateur a donné à ce film spécifique, note qui avait auparavant été cachée au système de recommandation pendant les calculs. Plus la note prédite et la note réelle sont proches et plus le système va être jugé "précis" [Konstan et al., 1998]. Cette précision peut se mesurer via différentes métriques, comme par exemple l'erreur moyenne absolue, qui calcule la déviation entre ce qu'a prédit le système et les préférences réelles de l'utilisateur [Sammut and Webb, 2010]. Cette méthode, bien qu'intuitive, a tendance à enfermer les utilisateurs dans leurs préférences déjà établies puisque c'est précisément de cette manière que les performances du système sont jugées. Ainsi, dans le cas du filtrage par contenu, si l'utilisateur avait apprécié par le passé un type de ressources particulier, le système tendait à lui recommander plus de ressources de ce type.

Un autre problème provenant de l'évolution des usages peut être soulevé ici. À l'origine, les systèmes de recommandation se concentraient dans quelques domaines particuliers, comme par exemple les films ou les produits de sites commerciaux. Les recommandations étaient alors présentées à l'utilisateur sous forme de ressources uniques ou de liste de ressources similaires. Cette manière de présenter les recommandations est encore très courante aujourd'hui alors que les usages ont évolué et que les systèmes de recommandation sont présents dans un nombre de plus en plus grand de domaines où recevoir une liste de ressources n'est pas forcément adapté. Pour conclure sur cette partie, il est clair que les systèmes de recommandation ont évolué depuis leur création. Cependant, des problèmes demeurent quant à l'évaluation de leurs performances et la satisfaction de l'utilisateur final, ainsi que pour la présentation des résultats à l'utilisateur final.

Prenons maintenant deux exemples qui vont servir à illustrer ce propos dans le chapitre. Le premier exemple se concentre sur le problème de l'évaluation par la précision dans le domaine de

l'e-commerce. Considérons un utilisateur qui consulte depuis quelques temps des appareils photographiques sur un site de vente en ligne. Le système de recommandation détecte ce comportement et calcule les produits les plus similaires disponibles sur le site, à savoir d'autres appareils photographiques. Après s'être décidé, l'utilisateur achète son appareil et, satisfait, le reçoit chez lui quelques jours plus tard. Lors de ses prochaines connexions au site de vente en ligne, le système de recommandation proposera encore à l'utilisateur des appareils photographiques car ce sont effectivement toujours ces produits qui correspondent le plus à ses préférences, même si l'utilisateur n'en a maintenant plus besoin. Ce premier exemple montre le problème relatif à la sur-utilisation de la précision comme mesure de la performance. Si les premières recommandations basées sur la précision pouvaient être utiles pour l'utilisateur, on voit bien qu'une fois l'achat effectué cette mesure n'est plus adaptée. Des recommandations sur des nouveaux produits pourraient par exemple ici se révéler intéressantes pour l'utilisateur. Une autre amélioration possible pourrait être la prise en compte de la temporalité. Des recommandations d'appareils photographiques pourraient être utiles à l'utilisateur, mais peut-être dans plusieurs mois ou années et non pas directement après son achat.On voit bien ici que le système de recommandation ne peut pas se contenter de la seule mesure de précision et pourrait bénéficier de l'utilisation d'autres mesures.

Prenons maintenant un deuxième exemple dans le domaine de l'e-education pour illustrer les problèmes que peut apporter la présentation des recommandations sous forme de liste à l'utilisateur final. Un utilisateur s'est inscrit récemment sur un site proposant des cours en ligne car il souhaite apprendre la programmation en Java. Il consulte ainsi un cours sur les bases de la programmation en Java et le termine avec succès. Le système de recommandation du site calcule un certain nombre de ressources éducatives similaires au cours qu'il vient de suivre et les propose sous forme de liste. L'utilisateur se voit donc proposer en tête de liste d'autres cours pour débutants en Java ainsi que quelques cours pour débutants dans d'autres langages de programmation dans le reste de la liste. L'utilisateur aurait davantage profité d'une recommandation de plusieurs ressources éducatives s'adaptant à son niveau et proposant une augmentation de la difficulté pour qu'il poursuive son apprentissage, illustrant l'intérêt de ne pas uniquement proposer des recommandations sous forme de liste. On remarque une nouvelle fois que la prise en compte d'autres mesures auraient amélioré la qualité des recommandations : temporalité, prise en compte du niveau de l'utilisateur, diversité des cours proposés, etc.

Ces deux exemples mettent en exergue la complexité du domaine de la recommandation, tant les besoins des utilisateurs peuvent être variés et complexes à satisfaire. La satisfaction de l'utilisateur final est l'objectif central de tout système de recommandation, tout en étant un problème ouvert et complexe. En effet, la satisfaction d'un utilisateur n'est pas binaire mais continue (un utilisateur n'est pas soit satisfait soit non satisfait, mais sera plutôt satisfait de la recommandation à un certain degré). Nous proposons donc ici d'aborder ce problème sous un autre angle : comment améliorer la qualité des recommandations fournies à l'utilisateur dans le but d'obtenir de l'utilisateur final une plus haute satisfaction ? Les deux exemples ci-dessus illustrent l'effet potentiellement néfaste de proposer des recommandations innapropriées, c'est-à-dire qui ne satisfont pas les besoins de l'utilisateur au moment où celui-ci les reçoit . En effet, un utilisateur à qui l'on propose de manière récurrente des recommandations innapropriées sera de plus en plus méfiant quant à la capacité du système de produire des recommandations adaptées à ses besoins au moment t. De la même manière, un mauvais système de recommandation ne sera pas utile à l'utilisateur pour satisfaire ses besoins. Il a été prouvé que ces deux notions de confiance et d'utilité diminuent chez l'utilisateur face à un système de recommandation ne les satisfaisant pas [Benbasat and Wang, 2005]. À terme, ce genre de recommandations peut amener l'utilisateur à

ne plus utiliser le système s'il ne perçoit pas la valeur ajoutée de celui-ci. Les exemples précédents donnent aussi quelques pistes sur la manière d'améliorer la qualité des recommandations. En effet, on peut remarquer que, même si les propositions étaient satisfaisantes du point de vue interne du système, elles ne l'étaient pas du point de vue externe de l'utilisateur. Cette contradiction peut s'expliquer par le fait que le système ne prend simplement pas assez en compte les facteurs déterminant les besoins de l'utilisateur. Un système réussissant à mieux modéliser un utilisateur et prenant en compte les facteurs les plus importants pour ce dernier dans le processus de recommandation permettrait de mieux satisfaire ses besoins. La problématique scientifique principale de cette thèse peut donc se formuler ainsi :

SP0 : Comment améliorer la qualité des recommandations faites aux utilisateurs ?

S'intéresser à la qualité des recommandations implique de s'intéresser à l'évaluation de ces dernières. Dans la prochaine section, nous détaillerons les défis et enjeux de cette problématique et son lien avec l'évaluation, puis nous définirons plusieurs sous-problématiques permettant d'y apporter des réponses.

1.2 Préambule

L'amélioration de la qualité des recommandations des utilisateurs est un vaste sujet auquel il est possible de répondre de multiples façons. Il est d'abord nécessaire de se demander ce que signifie "améliorer la qualité" dans cette question. Nous avons vu jusqu'à maintenant qu'un système de recommandation mesure classiquement la qualité de ses résultats selon une métrique, la précision. Cependant, la satisfaction de l'utilisateur n'est pas forcément en accord avec cette mesure comme cela a été illustré dans l'exemple des appareils photographiques où, malgré un score de précision potentiellement très haut, l'utilisateur peut ne pas être satisfait des propositions du système. La satisfaction de l'utilisateur final peut être évaluée de deux manières différentes. Premièrement, elle peut être mesurée directement via un système d'évaluation des recommandations, où l'on demande par exemple à l'utilisateur de noter la recommandation qui vient de lui être proposée. Deuxièmement, elle peut être mesurée indirectement en observant le comportement de l'utilisateur après lui avoir fourni la recommandation [Shani and Gunawardana, 2011]. Si ce dernier consulte ou consomme la ressource recommandée plutôt qu'une autre, il est raisonnable de faire l'hypothèse qu'il est satisfait par cette recommandation : c'est ce qu'on appelle le taux d'acception des recommandations d'un système par rapport à un utilisateur. Ces deux méthodes d'évaluation permettent d'avoir une idée de la corrélation entre les métriques internes du système et la satisfaction réelle de l'utilisateur. Il est plus simple de se baser sur des mesures internes pour évaluer un système, comme la précision, plutôt que sur des mesures externes, comme par exemple la satisfaction réelle de l'utilisateur, étant donné que celle-ci demande de mettre le système en ligne et de l'évaluer après un certain temps d'utilisation. L'évaluation sur les seules performances du système est appelée "évaluation hors ligne" tandis que l'évaluation en conditions réelles est appelée "évaluation en ligne" [Shani and Gunawardana, 2011]. L'évaluation hors ligne, même si potentiellement moins apte à juger de la satisfaction de l'utilisateur, offre tout de même de nombreux avantages. Elle permet premièrement de réaliser des expériences rapidement et à bas coûts étant donné qu'il n'y a pas besoin d'attendre que les utilisateurs se servent du système. Elle permet ensuite d'utiliser des bases de données déjà créées, permettant aux chercheurs de se concentrer uniquement sur la conception du système de recommandation (comme par exemple la très connue base de données MovieLens qui rassemble un grand nombre

d'évaluations de films par des utilisateurs sous la forme de notes [Resnick et al., 1994]). Enfin, elle permet de comparer les performances de différentes techniques de manière objective en les appliquant à un même jeu de données. Autrement dit, l'évaluation en ligne permet d'obtenir des résultats provenant directement d'utilisateurs dans un contexte réel, mais est très couteuse en temps et en ressource. L'évaluation hors ligne est quant à elle peu couteuse en temps et en ressources et permet de réaliser autant d'expérimentations que nécessaire. Dans le contexte de cette thèse où la question de la qualité des recommandations est centrale, ces deux méthodes d'évaluations possèdent chacune des avantages et des inconvénients et seront discutées dans le chapitre dédiée aux expérimentations du manuscrit.

L'amélioration de la qualité des recommandations est un point crucial de nos jours, tant les systèmes de recommandation sont omniprésents dans notre monde. Presque tous les grands sites commerciaux sont désormais dotés d'un système de ce genre dans le but de garder l'utilisateur plus longtemps sur leur site (Youtube [1], Deezer [2], Netflix [3]), de le satisfaire pour qu'il revienne à son prochain achat, voire même de susciter chez lui de nouvelles envies grâce à la caractéristique de ces systèmes d'être force de proposition (Amazon [4]). Les entreprises commerciales ne sont pas les seules à exploiter le potentiel des systèmes de recommandations. Les écoles et universités commencent elles aussi à se doter de systèmes de recommandation permettant aux étudiants de s'auto-former en utilisant les vastes ressources éducatives mises à disposition par celles-ci [Drachsler et al., 2015] [Boyer et al., 2015]. De la même manière, le domaine de l'héritage culturel et du tourisme s'équipe petit à petit d'applications smartphone permettant aux touristes, perdus devant les possibilités offertes par certains lieux de culture, une aide automatisée et personnalisée [Borràs et al., 2014].

Devant l'explosion de la popularité des systèmes de recommandation, devant la complexité grandissante des ressources à recommander et devant l'augmentation du nombre d'utilisateurs de ces systèmes, un constat a été fait : proposer constamment des ressources similaires par rapport aux préférences de l'utilisateur n'est plus suffisant [McNee et al., 2006]. De nouvelles études et expérimentations ont montré que proposer des recommandations selon d'autres modalités que la précision pouvait être bénéfique. Citons ici la mesure de diversité qui a été définie par [Smyth and McClave, 2001] comme étant l'inverse de la similarité entre deux ressources. Ainsi, plus deux ressources sont similaires entre elles et moins elles seront diverses, et vice versa. Proposer des recommandations plus diversifiées permet d'éviter aux utilisateurs de recevoir toujours les mêmes recommandations similaires à leurs préférences, ou encore d'augmenter le nombre des ressources pouvant être recommandées à un utilisateur (cette proportion entre ressources existantes et ressources pouvant être recommandées est appelée la "couverture" d'un système). Citons aussi la mesure de nouveauté qui fait quant à elle référence à une ressource qui n'a jamais été vue par l'utilisateur [Castells et al., 2015]. La nouveauté permet de remplir un des buts principaux d'un système de recommandation, à savoir faire découvrir à l'utilisateur des ressources qu'il n'aurait peut-être pas trouvées sans le système. Il est cependant important de préciser que l'apport de ces mesures dans les systèmes de recommandation est aussi discuté. Par exemple, [Ekstrand et al., 2014] ont montré que, si la diversité était corrélée positivement avec la satisfaction des utilisateurs, ce n'était pas le cas de la nouveauté. Les auteurs expliquent que cette corrélation négative pourrait être due au fait qu'un système proposant beaucoup de ressources nouvelles sans

1. https ://www.youtube.com/
2. https ://www.deezer.com/
3. https ://www.netflix.com/
4. https ://www.amazon.com/

lien avec les préférences de l'utilisateur serait mal vu par ce dernier. Les auteurs suggèrent ainsi de ne pas intégrer de ressources issues de la métrique de nouveauté aux nouveaux utilisateurs pour leur laisser le temps d'avoir confiance dans le système, avant éventuellement d'intégrer de la nouveauté dans les recommandations à mesure que l'utilisateur s'habitue au système.

Il apparaît donc important d'inclure plusieurs critères dans le calcul les recommandations tout en équilibrant ces différentes recommandations selon l'utilisateur. Cependant, cet équilibre est loin d'être évident à trouver alors que les informations disponibles sur les ressources et les utilisateurs sont souvent limitées et que certaines de ces métriques semblent opposées les unes aux autres, comme par exemple la similarité et la diversité. De plus, cet équilibre est non seulement amené à changer selon chaque utilisateur, mais il peut aussi varier selon les attentes d'un même utilisateur. Ce dernier souhaitera par exemple découvrir les ressources les plus populaires dans un premier temps après s'être inscrit sur un site. Après avoir consulté ces ressources, ce même utilisateur souhaitera peut-être découvrir des ressources moins populaires ou plus récentes. Il est donc important de pouvoir modifier l'importance de chacun de ces facteurs à tout moment lors de la recommandation.

Reprenons maintenant le premier exemple de l'introduction sur les appareils photographiques en proposant cette fois-ci une solution grâce aux différentes métriques explicitées ci-dessus. Un utilisateur arrive sur un site de vente en ligne et commence à consulter du matériel photographique. Le système n'étant pas encore certain du but de l'utilisateur, ses recommandations sont relativement diverses. Après un certain temps, l'utilisateur, qui a affiné ses recherches sur des appareils photographiques, se voit proposer d'autres types d'appareils similaires à celui qu'il est en train de consulter dans le but de lui faire découvrir des alternatives possibles. Le système détecte finalement que l'utilisateur est sur le point de se décider. Il propose alors une liste de recommandations plus diverses, permettant à l'utilisateur d'effectuer un dernier tour d'horizon général avant l'achat et de confirmer son choix [Castagnos et al., 2010]. Après s'être décidé avec l'aide du système, l'utilisateur achète son appareil. Lors de ses prochaines connexions sur le site de vente en ligne, le système de recommandation ne proposera peu ou plus d'appareils photographiques. À la place, l'utilisateur se verra recommander du matériel photographique dont il pourrait maintenant avoir l'utilité, comme un étui de protection ou encore différents objectifs. Le système lui recommandera aussi de nouveaux produits qu'il n'a encore jamais consulté dans le but de lui faire découvrir la grande variété des produits vendus par le site, comme par exemple du matériel informatique. On peut voir ici l'intérêt d'intégrer de nouveaux facteurs comme la diversité dans la recommandation. La variation du niveau de diversité au cours du temps et en fonction du contexte permet, dans cet exemple, d'améliorer la qualité de la recommandation finale en s'adaptant plus finement aux besoins de l'utilisateur et en l'aidant à des moments clés, comme juste avant ou après son achat. L'ajout d'un facteur comme la diversité permet de plus d'éviter la sclérose des recommandations passé un certain temps, c'est-à-dire la recommandation des mêmes ressources en boucle à un utilisateur. Ce phénomène se produit inévitablement dans un système basé uniquement sur la précision où les ressources recommandées finissent toutes par être déjà connues de l'utilisateur ou trop similaires à ses goûts. Ce problème peut en fait être considéré comme du sur-apprentissage, dans le sens où le système est tellement spécialisé sur les préférences de l'utilisateur que quelques ressources obtiennent un très haut score de précision et sont recommandées en permanence. L'introduction de nouveaux facteurs permettant d'autres manières de calculer des recommandations, ainsi que des variations d'importance entre ces facteurs au cours du temps, peuvent réduire ce problème.

De cet exemple, la première problématique scientifique permettant de répondre à la question générale posée à la fin de l'introduction peut être définie comme suit :

SP1 : Quels sont les facteurs à prendre en compte pour satisfaire les utilisateurs et comment adapter leur importance au cas par cas pour chaque utilisateur ?

Comme nous l'avons évoqué précédemment, les systèmes de recommandation se sont grandement démocratisés et existent désormais dans la plupart des domaines où se trouvent de nombreuses ressources mises à disposition de nombreux utilisateurs, et ce, quelque soit le type de ces ressources. Classiquement, un système de recommandation va calculer par divers moyens les ressources que l'utilisateur pourraient le plus apprécier. Le résultat de cette opération est bien souvent une valeur réelle d'utilité ou d'intérêt comprise dans l'intervalle $[0; 1]$ associée à chaque ressource, où 1 correspond à la prédiction selon laquelle l'utilisateur devrait très fortement apprécier la ressource en question, et inversement. Ainsi, les n ressources ayant les plus hautes valeurs sont recommandées à l'utilisateur. Certains domaines, comme l'e-commerce, se satisfont très bien de recommandations proposées de cette manière sous forme de listes. Cependant, d'autres domaines comme l'e-éducation ou le tourisme bénéficieraient de recommandations proposées aux utilisateurs sous la forme d'une séquence. Nous définissons ici une séquence comme une suite ordonnée de ressources possédant un début, une fin, un but et éventuellement une ou plusieurs contrainte(s) à respecter lors de sa création. Cette définition implique, entre autres, de définir ce que signifie la notion d'ordre dans la séquence. Un ordre correspond à un agencement particulier de ressources permettant d'atteindre le but recherché de manière optimisée. Reprenons maintenant le deuxième exemple de l'introduction sur le site d'e-éducation en ajoutant cette fois-ci la notion de séquence afin d'illustrer son intérêt. Un utilisateur s'est inscrit récemment sur un site proposant des cours en ligne car il souhaite apprendre la programmation en Java. Cependant et avant de commencer, le site lui propose de fournir quelques informations sur son objectif éducatif. Ainsi, l'utilisateur explicite l'objectif qu'il souhaite atteindre (apprendre les bases de Java) ainsi que le temps dont il dispose (exprimé par exemple ici sous la forme d'un nombre n de ressources éducatives voulues pour atteindre l'objectif). Avec ces informations, le système de recommandation du site va calculer, parmi toutes les ressources éducatives disponibles, lesquelles correspondent aux besoins de l'utilisateur. Le niveau de difficulté de ces ressources sélectionnées est ensuite calculé par le système grâce à leurs caractéristiques. Enfin, le système construit une séquence de taille n où la place de chaque ressource éducative dépend de son niveau de difficulté, de manière à ce que l'utilisateur débute avec la ressource la moins complexe puis progresse avec des ressources abordant des notions de plus en plus avancées. La séquence créée va donc lui permettre d'atteindre son objectif grâce à des cours à la difficulté croissante, tout en respectant la contrainte de temps imposée.

Avec cet exemple, nous pouvons maintenant expliciter les deuxième et troisième problématiques scientifiques :

SP2 : Connaissant les facteurs humains à prendre en compte (SP1), comment passer d'une recommandation unitaire à une séquence de recommandations ?

SP3 : Connaissant les facteurs humains à prendre en compte (SP1), comment les combiner dans un même modèle permettant de faire varier leur importance les uns par rapport aux autres à tout moment ?

Le domaine de la recommandation est en plein essor et, même si des problèmes subsistent encore actuellement, des solutions existent pour y répondre comme l'ont illustré les exemples ci-dessus. La compréhension des besoins des utilisateurs, la prise en considération de la dimension temporelle et d'objectifs multiples et parfois antagonistes dans le processus de recommandation sont désormais les objectifs principaux du domaine de la recommandation, et c'est au sein de ce mouvement que cette thèse s'inscrit. De plus, les nouveaux domaines applicatifs s'ouvrant à la recommandation (e.g. l'e-éducation, le tourisme,...) sont à la fois une chance et un défi ; une chance car il devient possible d'assister l'utilisateur dans un panel de plus en plus grand de situations ; un défi dans le sens où les modèles doivent être assez génériques pour s'adapter à cette grande variété de situations possibles. Une dernière problématique scientifique est toutefois nécessaire dans le cadre de cette thèse pour compléter ce tour d'horizon. Il est en effet nécessaire de pouvoir quantifier à quel point SP1 et SP2 améliorent la qualité des recommandations. Nous avons vu dans les exemples précédents que cette qualité n'était pas une notion simple à mesurer, mais il est pourtant nécessaire de s'atteler à ce problème pour quantifier les performances d'un modèle. La 3ème et dernière problématique de cette thèse peut donc être exprimée ainsi :

SP4 : Comment évaluer les séquences produites en SP2 et SP3 ?

L'évaluation des recommandations est au cœur des nouveaux défis évoqués précédemment. Le domaine de la recommandation est intimement lié à l'être humain et il est important de ne plus uniquement se concentrer sur une seule mesure comme la précision. L'être humain est complexe dans ses besoins et l'évaluation des recommandations doit refléter cette complexité.

1.3 Contributions

Dans cette thèse, une réponse à été apportée à chacune des problématiques scientifiques énoncées ci-dessus. Ces contributions scientifique ont fait l'objet de trois publications :

- Workshop international SMAP[5] : ***Walk the line : Toward an efficient user model for recommendations in museums*** [Osche et al., 2016] ;
- Conférence internationale ECIR[6] : ***AntRS : Recommending lists through a multi-objective ant colony system*** [Osche et al., 2019a] ;
- Workshop international ALA[7] organisé avec la conférence internationale AAMAS[8] : ***From Music to Museum : Applications of Multi-Objective Ant Colony Systems to Real World Problems*** [Osche et al., 2019b].

Dans la suite de cette section, un aperçu plus en détail des réponses apportées aux problématique scientifique est donné.

5. *SMAP : International Workshop on Semantic Media Adaptation and Personalization*
6. *ECIR : European Conference on Information Retrieval* : http ://ecir2019.org/
7. *ALA : Adaptive and Learning Agents*
8. *AAMAS : International Conference on Autonomous Agents and Multiagent Systems*

1.3.1 SP1 : les facteurs à prendre en compte pour satisfaire les utilisateurs

Pour répondre à la première problématique scientifique SP1, nous avons sélectionné un ensemble de quatre critères d'évaluation représentant au mieux les besoins de l'utilisateur. Ces critères ont été choisis car ils apparaissent dans la littérature scientifique comme étant liés à la satisfaction finale de l'utilisateur. Ils représentent un panel intéressant de facteurs humains importants pour les utilisateurs, mais il est important de noter que ce choix n'est pas exhaustif. Comme nous l'avons mentionné, le nombre de facteurs humains rentrant en compte dans la prise de décision est potentiellement infini et varie en fonction de l'utilisateur. Nous avons donc choisi ces quatre critères car ils sont les plus étudiés dans la littérature, mais le but de cette thèse est de pouvoir intégrer de nouveaux critères, de faire varier leur importance si nécessaire, et de les traiter comme un problème d'optimisation multi-critères. Ces quatre critères nous permettent de nous situer par rapport aux travaux existants de la littérature tout en établissant un modèle générique capable de s'adapter à de multiples besoins. Voici ces quatre critères :

1. **La similarité** : garantissant à l'utilisateur des recommandations similaires aux ressources qu'il apprécie déjà ;
2. **La diversité** : permettant d'offrir de la variété dans les recommandations ;
3. **La nouveauté** : proposant à l'utilisateur des ressources qu'il ne connaît et qu'il n'aurait peut-être jamais trouvé lui-même ;
4. **Les préférences** : offrant à l'utilisateur la possibilité de revoir les ressources qu'il a déjà aimé par le passé.

1.3.2 SP2 : la conception de séquences de recommandations en accord avec les facteurs sélectionnés

La notion de séquence est centrale pour comprendre son intérêt dans le paysage actuel des systèmes de recommandation, ainsi que pour être capable d'en générer. Dans ce travail, nous avons proposé une méthode permettant de garantir des transitions optimales entre ressources au sein d'une séquence. Nous avons ensuite mis en pratique ces contributions dans un jeu de données existant provenant de Deezer, un site d'écoute de musique en ligne. Ce jeu de données correspond à un mois d'écoutes de titres par plusieurs milliers d'utilisateurs et possède des informations sur les titres ainsi que sur les utilisateurs. Grâce à ce jeu de données, nous avons pu extraire des séquences d'écoutes et nous avons utilisé ces séquences réelles comme évaluation pour notre système. Nous avons enfin créé un modèle permettant de générer des séquences de la taille voulue, possédant un point de départ et d'arrivée définis.

1.3.3 SP3 et SP4 : la création d'un modèle générique prenant en compte plusieurs métriques et produisant des séquences

Après avoir défini les métriques à utiliser, la manière de les calculer et après avoir défini la notion de séquence et la manière de les construire, nous avons développé un modèle réunissant l'ensemble de ces caractéristiques. Pour cela, nous nous sommes inspirés de solutions développées pendant des millions d'années par la sélection naturelle pour résoudre des problèmes d'exploration et d'optimisation de chemins dans la nature. Il existe en effet tout un domaine de recherche dédié à la modélisation informatique de systèmes vivants complexes et à l'application de ces modèles à des problèmes complexes humains : **les Systèmes Multi-Agents** [Ferber and Weiss, 1999]. Nous avons utilisé l'un de ces systèmes inspiré des colonies de fourmis pour répondre à

notre problématique [Dorigo et al., 1996] [Dorigo et al., 2006]. Dans la nature, il a été observé que les fourmis empruntaient la plupart du temps le plus court chemin possible entre leur fourmilière et une source de nourriture. Une fourmi prise séparément n'a pas des capacités cognitives très développées avec ses quelques centaines de milliers de neurones seulement. Cependant, l'interaction entre toutes les fourmis de la fourmilière produit ce qu'on appelle un effet d'émergence. L'interaction d'animaux simples (ou d'agents si l'on parle d'un système informatique) provoque l'émergence d'une intelligence collective capable de résoudre des problèmes dont la complexité dépasse la somme des capacités des individus [Bonabeau et al., 1999]. C'est bien ce phénomène qui est à l'œuvre lors de l'optimisation de la distance d'un chemin par toutes les fourmis d'une fourmilière. Nous reviendrons plus en détail sur l'ensemble des mécanismes permettant l'émergence de cette intelligence collective dans ce manuscrit. Afin de répondre à notre problématique, nous avons développé un modèle où une multitude d'agents très simples (correspondant aux fourmis) sont déployés dans un graphe où les sommets représentent les ressources d'un domaine applicatif et où les arêtes représentent les liens entre ces ressources (correspondant à l'environnement à parcourir pour les fourmis). Afin d'intégrer les quatre métriques discutées ci-dessus, nous avons créé différentes colonies possédant chacune leurs agents spécialisés dans l'optimisation d'une des métriques. De par le phénomène d'intelligence collective, les agents produisent des séquences optimisées dans le graphe pouvant être recommandées à l'utilisateur. Cette approche apporte ainsi des solutions aux problèmes soulevés par les problématiques scientifiques sus-mentionnés :

— Nous sommes capables de prendre en compte les différentes métriques explicitées dans un seul et même modèle. De plus, nous produisons des solutions représentant un compromis entre des objectifs parfois divergents entre eux (similarité et diversité) ;
— Dans la nature, les fourmis créent un chemin entre leur fourmilière et la source de nourriture. Nous utilisons cette aptitude inhérente afin de créer des séquences entre une ressource de départ (la fourmilière) et une ressource d'arrivée (la nourriture) ;
— L'adaptation aux changements de l'environnement est rapide (nouvel utilisateur, nouvelle ressource, suppression d'une ressource, etc.). De la même manière que dans la nature, les fourmis sont capables de retrouver un autre chemin optimisé si le premier venait à être bloqué par une branche venant de tomber par exemple ;
— Notre modèle est souple car il permet de modifier le poids de chaque métrique dans la séquence finale, permettant de s'adapter à chaque utilisateur ;
— Il est possible d'ajouter ou d'enlever des colonies à la volée dans le système, permettant de s'adapter rapidement à différents domaines applicatifs et à différents objectifs.

1.4 Plan de la thèse

La suite de ce manuscrit se présente ainsi : dans le chapitre 2, nous présenterons les travaux de la littérature portant sur les différents facteurs humains utilisés dans les systèmes de recommandation, les approches existantes de recommandations en séquences et les systèmes multi-agents. Dans le chapitre 3, nous présenterons notre modèle et son fonctionnement en détail. Ensuite, nous présenterons les résultats obtenus par ce modèle dans le domaine applicatif de l'écoute de musique en ligne dans le chapitre 4. Enfin, nous conclurons et discuterons des perspectives de ce travail dans le chapitre 5.

Chapitre 2

État de l'art

Sommaire

Dans cet état de l'art, nous allons tout d'abord nous pencher sur le fonctionnement des systèmes de recommandation classiques ainsi que sur leurs limites. Nous allons ensuite étudier quelques réponses à ces limites proposées par la littérature et discuter des améliorations possibles.

2.1 Les systèmes de recommandation

Les systèmes de recommandation ont pour but principal de proposer aux utilisateurs des ressources adaptées à leurs attentes. Une ressource peut être un document, une page web, une musique, un film, un livre, un objet, un cours, etc. Pour effectuer une recommandation, une hypothèse simple a été exploitée par les premiers systèmes : les préférences passées d'un utilisateur peuvent être utilisées pour prédire les préférences futures de ce même utilisateur. Cette hypothèse s'est déclinée dans les deux approches principales des systèmes de recommandation que sont le filtrage par contenu [Belkin and Croft, 1992, Lops et al., 2011] et le filtrage collaboratif [Resnick et al., 1994]. La principale différence entre ces deux approches repose sur l'exploitation de différents types des données : le filtrage par contenu utilise les informations descriptives caractérisant les utilisateurs et les ressources du système tandis que le filtrage collaboratif exploite uniquement les interactions entre utilisateurs et ressources, comme les consultations ou les votes [Aggarwal et al., 2016]. Ces deux familles ont par la suite donné naissance a de nombreuses techniques que nous n'aborderons pas ici. Toutes peuvent cependant être rapportées à l'une ou l'autre de ces familles ou à une combinaison des deux. Dans la suite de cette section, nous reviendrons rapidement sur ces deux familles qui ont marqué les débuts des systèmes de recommandation et qui regroupent à elles seules un grand nombre d'articles de la littérature du domaine. Cette introduction aux systèmes de recommandation servira aussi à présenter quelques concepts-clés liés à ces systèmes.

2.1.1 Le filtrage par contenu

Le filtrage par contenu est une des deux approches majeures dans les systèmes de recommandation [Belkin and Croft, 1992, Lops et al., 2011]. Le but d'un système de filtrage par contenu est de proposer des recommandations à l'utilisateur uniquement en fonction des caractéristiques des ressources appréciées. Ce genre de système va donc enregistrer les actions de l'utilisateur sur les ressources qu'il a consultées dans le but de trouver d'autres ressources qu'il pourrait apprécier. Cette méthode de fonctionnement implique deux principes :

1. Le système doit pouvoir enregistrer les actions de l'utilisateur dans le système [9]. Le système doit ensuite pouvoir interpréter ces différentes actions afin d'être capable d'évaluer à quel point un utilisateur apprécie la ressource sur laquelle il a effectué des actions. Nous discuterons de ces actions et des manières de les interpréter plus en détail dans la suite de cette section.
2. Le système doit pouvoir comparer des ressources entre elles afin de déterminer lesquelles pourraient être appréciées par l'utilisateur en fonction de son historique de consultations. Pour réaliser cela, il est nécessaire que les ressources soient décrites par ce que l'on appelle des méta-données, c'est-à-dire des données décrivant d'autres données ou, comme ici, des données décrivant des ressources. Il est d'usage de définir ou d'utiliser des standards pour

9. Des exemples d'actions possibles : consulter une ressource, donner une note à une ressource, acheter une ressource, bannir une ressource pour ne plus la voir, placer une ressource dans ses favoris, etc.

les méta-données afin de décrire toutes les ressources de la même manière et, éventuellement, de pouvoir partager ou utiliser des bases de données utilisant les mêmes standards. On peut par exemple citer le *Learning Object Metadata*, qui est un standard utilisé en e-éducation pour décrire des ressources éducatives [RISK, 2002]. Des domaines comme le web sémantique ou les systèmes à base d'ontologie ont comme but de structurer ces données descriptives lorsqu'elles ne le sont initialement pas.

Le filtrage par contenu a l'avantage de pouvoir fournir des recommandations à un utilisateur même s'il est le seul à utiliser le système, tant que ce dernier possède des informations sur les ressources consultées. Cependant, la qualité des recommandations dépendra directement de la qualité de ces informations. Si les actions de l'utilisateur ou les descriptions de ressources sont insuffisantes, un système de filtrage par contenu ne sera pas capable de proposer de recommandations de grande qualité. Une autre limite majeure de ce genre de système réside dans le manque de nouveauté. En effet, après un certain temps, les recommandations proposées par le système ont un risque de sclérose provoqué par un manque de nouvelles informations dans le système.

2.1.2 Le filtrage collaboratif

La deuxième approche majeure des systèmes de recommandation est donc le filtrage collaboratif [Resnick et al., 1994]. Cette technique utilise les préférences des autres utilisateurs du système sur les ressources pour estimer les préférences de l'utilisateur. L'hypothèse sous-jacente a cette méthode est qu'il est possible de prédire les préférences d'un utilisateur en étudiant ce que d'autres utilisateurs similaires en termes de préférences ont aimé. Si beaucoup de ces utilisateurs similaires ont apprécié une ressource, alors il est probable que l'utilisateur apprécie aussi cette ressource. Cette méthode repose ici aussi sur deux principes :

1. Comme pour le filtrage par contenu, il faut d'abord pouvoir collecter les actions de nombreux utilisateurs sur les ressources du système afin d'en déterminer leurs préférences.
2. Le système doit ensuite déterminer les utilisateurs les plus similaires à l'utilisateur en terme de préférences, avec pour but final de lui recommander des ressources qu'il n'a pas encore consulté et qui sont appréciées par ces utilisateurs similaires.

Le filtrage collaboratif permet, par rapport au filtrage par contenu, de proposer des recommandations à l'utilisateur sans forcément connaître le contenu des ressources recommandées. En effet, le système ne manipule que des préférences générales, souvent représentées numériquement par une note associée à une ressource et à un utilisateur. En d'autres termes, le filtrage collaboratif s'intéresse à l'usage qui est fait des ressources, et non aux ressources en elles-mêmes. Une fois ces préférences établies, ce genre de système peut donc s'adapter à des domaines où les ressources sont différentes. Cependant, il est aussi nécessaire d'avoir à disposition un grand nombre d'utilisateurs ainsi que leurs préférences afin de pouvoir utiliser ce genre de système correctement.

2.1.3 Les actions des utilisateurs

Comme nous l'avons décrit plus haut, peu importe le type de système de recommandation utilisé, il est nécessaire de récolter les actions des utilisateurs afin d'obtenir une évaluation des ressources qu'ils ont consultées. Cette évaluation prend la plupart du temps la forme d'une note r entière variant de 1 à 5 ou d'une valeur continue $r \in [0; 1]$ où, dans les deux cas, une préférence

marquée correspond à une valeur haute et inversement. D'une manière générale, on nomme "retours" les informations récupérées par un système sur un utilisateur permettant d'aboutir à cette note. Dans les systèmes de recommandation, on peut distinguer deux types de retours : le retour explicite et le retour implicite. Le retour explicite est le plus simple des deux puisqu'il correspond à la situation où l'utilisateur fournit naturellement de lui-même la note associée à une ressource qu'il a consultée. Cette manière de procéder est courante dans l'e-commerce ou dans le divertissement, où les utilisateurs peuvent généralement noter de 1 à 5 le produit qu'ils ont acheté ou encore le film qu'ils ont vu. Cette méthode permet d'accéder à l'évaluation réelle de l'utilisateur et à ses préférences, offrant de manière directe des données de haute qualité [Koren et al., 2009]. À l'inverse, le retour implicite représente les traces laissées par l'utilisateur lors de sa présence dans le système. Ces traces ne sont pas directement des évaluations ou des notes mais représentent bien les différentes possibilités d'actions qu'offre un système à ses utilisateurs. Ces actions diffèrent donc selon le domaine d'application du système. Pour un site web d'e-commerce, les actions pourront être la consultation d'une page, la consultation d'une fiche produit, la mise d'un produit dans le panier ou encore l'achat d'un produit. Pour un site web d'écoute de musique, les actions pourront être l'écoute d'une musique en entier, l'arrêt de l'écoute d'une musique avant la fin, la consultation de la page d'un artiste, etc. On voit bien ici que les actions possibles sont dépendantes du domaine, mais aussi que toutes les actions n'ont pas la même importance ni la même signification. On peut en effet supposer qu'un utilisateur appréciera davantage un produit s'il l'achète que s'il consulte seulement sa page. On retrouve cette distinction dans [Oard et al., 1998] où les auteurs définissent trois catégories permettant de grouper les actions de même importance ensemble (1/ consultation de la ressource, 2/ conservation de la ressource, 3/ partage de la ressource). On peut aussi citer [Castagnos, 2008] qui, d'une manière plus générale, propose une méthode permettant de calculer l'intérêt présumé d'un utilisateur pour une ressource selon les types d'actions effectuées par l'utilisateur et leurs importances respectives. Afin de conclure cette section, il est important de préciser que, même si le retour explicite semble plus désirable pour les systèmes de recommandation, il a été montré que ces deux types de retours apportent des informations différentes et qu'il peut justement être intéressant de les prendre tous les deux en compte [Jawaheer et al., 2010].

2.1.4 Discussion et limites des systèmes classiques

Dans les deux approches de recommandation présentées ci-dessus, le problème et la solution sont fondamentalement les mêmes. Le problème peut se formuler ainsi : comment estimer l'intérêt r qu'a un utilisateur u sur une ressource i encore non consultée, en se basant sur les connaissances incomplètes représentées par le profil de l'utilisateur u ? Le profil de l'utilisateur est incomplet car les informations sur ses préférences proviennent uniquement des interactions que ce dernier a eu avec le système. Autrement dit, il n'est pas possible de connaître les facteurs ayant abouti aux actions de l'utilisateur puisque ces dernières sont le fruit des mécanismes de raisonnement interne de l'utilisateur. Il n'est par exemple pas possible de savoir réellement pourquoi un utilisateur a donné la note de 5 sur 5 à un film. Est-ce parce qu'il considère que c'est le meilleur film qu'il a vu de sa vie et qu'il aimerait découvrir plus de films de ce genre ? Ou bien est-ce parce que c'est un film qu'il a vu dans un contexte particulier et que, même s'il l'a beaucoup aimé, il ne souhaiterait pas revoir des films similaires aujourd'hui ? Il existe ainsi probablement une multitude de facteurs jouant dans les actions que nous effectuons et il n'est pas possible de tous les appréhender. Devant ce constat, la solution adoptée par les deux approches est la même : se baser sur les préférences connues de l'utilisateur pour extrapoler ses préférences futures potentielles. Cependant et comme nous l'avons vu dans l'exemple de l'introduction sur les appareils photo, il est aisé de concevoir

des situations où les préférences et actions passées d'un utilisateur ne sont pas forcément les meilleurs indicateurs de ses besoins actuels ou futurs.

2.2 Les facteurs humains à prendre en compte dans la recommandation

Comme nous l'avons vu dans la section précédente, la performance des premiers systèmes de recommandation a été mesurée avec les outils du domaine de la recherche d'information. Parmi ceux-ci on peut citer la précision et le rappel qui sont deux mesures permettant de jauger la pertinence des ressources sélectionnées en les comparant avec l'ensemble des ressources disponibles. La Figure 2.1 présente bien la manière dont ces deux mesures sont calculées [10].

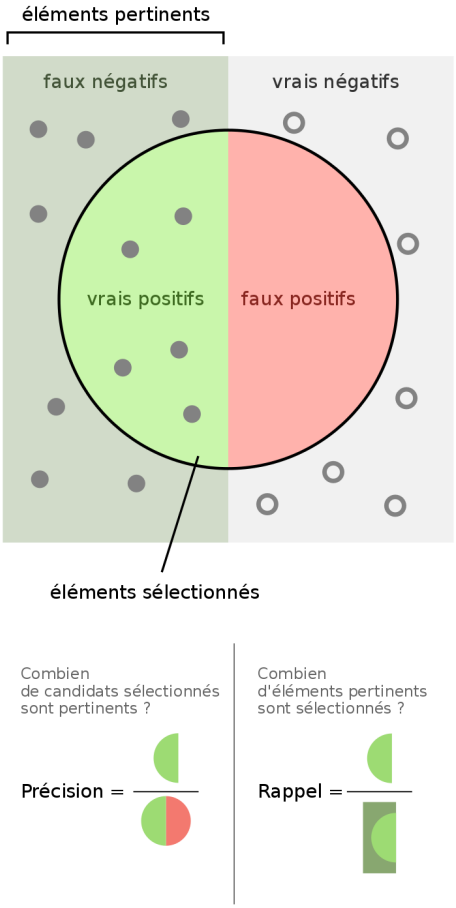

FIGURE 2.1 – Présentation schématisée du calcul de la précision et du rappel en fonction des ressources trouvées par rapport à l'ensemble des ressources disponibles (crédit image : Datamok / User:Walber).

Plusieurs raisons peuvent expliquer pourquoi les systèmes de recommandations ont initialement emprunté des techniques venant de la recherche d'information et en voici trois parmi les plus évidentes. Premièrement, les systèmes de recommandation sont à la croisée de plusieurs

10. https://fr.wikipedia.org/wiki/Précision_et_rappel

disciplines parmi lesquelles, entre autres, la recherche d'information [Adomavicius and Tuzhilin, 2005]. Ce domaine a beaucoup utilisé et utilise toujours les mesures de précision et de rappel [Salton, 1989], il est donc naturel que les premiers systèmes de recommandation aient utilisé ces mesures de performance. Deuxièmement, très tôt dans le domaine des systèmes de recommandation, des corpus de données ont été mis à disposition de tous, tel que, MovieLens [11] grâce à GroupLens [Resnick et al., 1994]. Ces corpus de données ont beaucoup aidé le domaine en permettant à tous les chercheurs de mesurer la performance de leurs algorithmes en les comparant avec les résultats obtenus par d'autres études de la littérature sur les mêmes données. La popularité toujours haute du corpus MovieLens depuis maintenant 25 ans illustre l'intérêt scientifique de partager des données initiales. Cependant, l'utilisation principale des métriques de précision pour évaluer la performance des premiers systèmes de recommandation a mécaniquement poussé les autres auteurs voulant comparer leurs algorithmes à utiliser eux aussi ces mêmes métriques, raréfiant potentiellement le travail sur d'autres métriques. Troisièmement, le mesure de la performance des algorithmes en fonction de la précision mène forcément à la création d'algorithmes de plus en plus performants dans ce domaine. Les chercheurs sont en effet plus enclins à développer des algorithmes optimisant la mesure de précision si la performance de la majorité des algorithmes existants est mesurée via cette métrique. Ce point, même s'il est lié au précédent, n'est pas à négliger et explique en partie la concentration des premiers systèmes sur la métrique de précision. La recherche dans le domaine de l'informatique est compétitive et les revues scientifiques accepteront plus facilement de publier un article montrant qu'un algorithme obtient de meilleures performances sur une métrique bien connue plutôt qu'une étude reposant sur d'autres métriques moins communes.

Après ces explications donnant quelques raisons systémiques de l'utilisation de la précision comme mesure d'évaluation, il est nécessaire de nuancer ce constat. La précision a beaucoup été utilisée car, avant toute chose, elle est au cœur des systèmes de recommandation et doit donc être considérée par tous ces systèmes. Le but de l'utilisation de la précision est d'obtenir des recommandations proches de ce que l'utilisateur a déjà consulté et donc, apprécie déjà. Il est important de préciser aussi ici que, malgré les critiques faites dans le paragraphe précédent sur cette mesure, il est indispensable de la conserver et de l'utiliser lors du processus de recommandation. Un utilisateur souhaite avant tout se voir proposer des ressources pertinentes, et il convient pour un système d'utiliser la précision pour arriver à cet objectif. Il est cependant nécessaire de ne pas uniquement considérer la précision pour produire de bonnes recommandations, et nous allons le voir dans la suite de cette section.

Nous venons de voir quelques-unes des raisons de l'omniprésence de l'évaluation par la précision au début des systèmes de recommandation. Nous allons maintenant étudier les conséquences que la sur-utilisation de la précision a pu avoir sur les utilisateurs, en se basant à la fois sur la littérature et sur des constatations empiriques. Pour comprendre les limites de cette métrique, il est tout d'abord nécessaire de s'intéresser à ce que représente une recommandation "précise" pour l'utilisateur final. Nous l'avons vu précédemment, plus un modèle s'approche des préférences déjà connues d'un utilisateur et plus il obtiendra de bonnes performances dans les métriques mesurant la précision. Dans la pratique, cela se traduit par le système proposant à l'utilisateur final des ressources souvent déjà connues de celui-ci, ou *a minima* des ressources très proches de ce qu'il aime déjà [McNee et al., 2006]. Un utilisateur d'un site de recommandation de films indiquant qu'il a aimé le premier film Star Wars se verra probablement recommander une liste de tous les

11. https://grouplens.org/datasets/movielens/

films Star Wars suivants. Intuitivement, on voit bien que ces recommandations ne seront probablement pas satisfaisantes pour l'utilisateur, bien qu'étant les plus précises possibles par rapport à ses préférences. Ces films sont anciens, connus et il n'y a donc que très peu de chances pour que l'utilisateur n'en ait pas déjà connaissance. On peut imaginer que l'utilisateur aurait souhaité que le système lui propose des films de science-fiction ou de guerre spatiale récents, ou du moins qu'il ne connaissait pas encore. Plus que de ne pas être utiles, ce genre de recommandations peut même dégrader l'intérêt et la confiance que l'utilisateur porte au système. En effet, un utilisateur recevant beaucoup de recommandations inutiles ou déjà connues de la part d'un système finira par ne plus lui accorder de l'importance. Cet argument peut être rapproché du *Technology Acceptance Model* de Fred Davis sur la manière dont les utilisateurs appréhendent une technologie [Davis, 1989], et notamment sur la notion "d'utilité perçue" qui fait référence à la tendance qu'ont les personnes à utiliser une technologie si cela leur permet de réaliser plus efficacement leur tâche en cours (comme par exemple trouver un bon film ou découvrir une nouvelle musique). Pour conclure, il semble évident que de nouvelles manières de concevoir et d'évaluer les systèmes de recommandation sont nécessaires. Dans la suite de cette section, nous allons voir comment l'intégration de facteurs humains supplémentaires dans le processus de recommandation et la mise en place de nouvelles manières d'évaluer la qualité des recommandations finales doivent permettre de palier les problèmes évoqués précédemment.

2.2.1 La similarité

Comme nous l'avons vu dans la section précédente, l'importance de la similarité dans les systèmes de recommandation est évidente tant ces derniers ont longtemps eu comme but principal de proposer des ressources proches, autrement dit similaires, de ce qu'appréciait déjà l'utilisateur. Nous avons vu que la précision a beaucoup été utilisée pour évaluer ces systèmes, mais il était aussi nécessaire de déterminer à quel point des ressources et/ou des utilisateurs étaient proches les uns des autres. En effet, les approches les plus utilisées, comme le filtrage collaboratif ou le filtrage par contenu, nécessitent souvent de déterminer un ensemble d'utilisateurs ou un ensemble de ressources proches des goûts de l'utilisateur. Pour ce faire, ces systèmes vont utiliser des métriques permettant de juger la similarité de deux variables entre elles. On retrouve parmi ces métriques le coefficient de corrélation de Pearson qui calcule la corrélation linéaire r entre deux variables à l'aide de leur covariance. Cette corrélation peut être appliquée à des profils utilisateurs [Shardanand and Maes, 1995] possédant des évaluations communes sur des ressources. L'Équation 2.1 décrit son calcul pour deux profils utilisateurs u_x et u_y.

$$r_{u_x,u_y} = \frac{\sum_1^n (u_{x_i} - \overline{u_x}) \times (u_{y_i} - \overline{u_y})}{\sqrt{\sum_1^n (u_{x_i} - \overline{u_x})^2 \times \sum_1^n (u_{y_i} - \overline{u_y})^2}} \tag{2.1}$$

où n est le nombre de co-consultations des deux profils utilisateurs u_x et u_y, u_{x_i} est une note individuelle du profil de l'utilisateur u_x et $\bar{u_x}$ est la moyenne des notes dans le profil u_x. Classiquement, les évaluations des ressources co-consultées sont considérées pour calculer la corrélation de Pearson dont la valeur $r \in [-1; 1]$ où une valeur proche de 1 signifie une corrélation positive entre les utilisateurs (autrement dit une forte similarité entre les deux), -1 une corrélation négative (autrement dit une forte dissimilarité entre les deux utilisateurs), et 0 une absence de corrélation linéaire entre les deux profils [12].

12. Une valeur de corrélation de Pearson $r = 0$ signifie uniquement une absence de corrélation linéaire entre les deux variables étudiées. Il est cependant possible qu'une corrélation non linéaire existe entre ces deux variables.

On peut aussi citer la très utilisée similarité cosinus, permettant de calculer l'angle formé par deux vecteurs. Cette idée a d'abord été proposée dans le domaine de la recherche d'information afin de comparer deux documents, représentés sous forme de vecteurs d'occurrences de mots [Salton and McGill, 1986]. Cette mesure a ensuite été adaptée pour le filtrage collaboratif en remplacant les documents par les utilisateurs, les mots par les ressources et les occurrences par les votes [Breese et al., 2013] [13]. Le calcul de la similarité cosinus est présenté dans l'Équation 2.2.

$$Cosine\_Similarity(u_1, u_2) = \frac{\sum u_1 u_2}{\sqrt{\sum u_1^2}\sqrt{\sum u_2^2}} \tag{2.2}$$

où, comme pour le coefficient de corrélation de Pearson, on considère uniquement les évaluations des ressources co-consultées par les deux utilisateurs dans le calcul. Beaucoup d'autres mesures de similarité existent dans le cadre de la comparaison de deux variables entre elles, et chacune possède des avantages et des inconvénients. L'ensemble de ces mesures ont cependant pour point commun de mesurer la similarité entre deux ressources uniquement et on peut toutefois apporter quelques critiques à ce genre de mesures. En effet, [McNee et al., 2006] critique justement le fait que les mesures classiques calculent généralement la précision de chaque ressource recommandée une par une. Cependant, l'immense majorité des recommandations sont proposées via des listes à l'utilisateur, et non via une seule ressource. On peut le voir chez tous les grands sites utilisant un système de recommandation tels que Netflix [14], Youtube [15], Amazon [16], Deezer, etc. Cette manière de calculer la précision ressource par ressource peut provoquer l'émergence de listes de recommandations possédant des ressources toutes similaires entre elles. Le site de recommandation de films proposant à un utilisateur la liste de tous les films Star Wars souffre de ce problème. Toutes ces recommandations sont précises et similaires par rapport aux préférences de l'utilisateur, mais les métriques classiques ne donnent aucune indication sur le degré de similarité de ces recommandations entre elles dans la liste proposée à l'utilisateur. Une manière de mitiger ce problème est de calculer la similarité non plus, ressource par ressource, mais pour la liste entière. La similarité intra-liste permet justement de calculer à quel point les ressources d'une liste sont similaires entre elles [Ziegler et al., 2005]. La similarité de chaque paire de ressources est calculée et la moyenne est ensuite calculée pour obtenir à une valeur unique mesurant à quel point la liste dans son ensemble possède des ressources similaires entre elles.

$$ILS(u) = \frac{\sum_{i_j \in L} \sum_{i_k \in L, i_j \neq i_k} sim(i_j, i_k)}{2} \tag{2.3}$$

où ILS_u est la similarité intra-liste d'un utilisateur u, i_j et i_k sont deux ressources de la liste recommandée L et $sim(i_j, i_k)$ est la similarité des deux items i_j et i_k, similarité pouvant être calculée différemment selon la méthode utilisée (similarité cosinus, similarité de Jaccard, etc.). Cette manière de procéder est intéressante car, premièrement, elle dénote avec les mesures classiques et permet de proposer une liste plus homogène plaisante pour l'utilisateur, et non plus uniquement sur les performances pures des systèmes. Deuxièmement, le fait de considérer la liste de recommandations comme un tout, et non plus uniquement comme des ressources alignées les unes à la suite des autres, permet de se rapprocher de la notion de séquence dont nous allons parler dans la suite de cette section. Enfin, l'article de [McNee et al., 2006] dont

13. Il est à noter qu'une variante de la mesure de similarité cosine s'appliquant aux ressources et non aux utilisateurs a aussi été développée.
14. https://www.netflix.com
15. https://www.youtube.com
16. https://www.amazon.com

est tirée cette réflexion marque un tournant dans le domaine des systèmes de recommandation avec la critique de la sur-utilisation de la similarité et l'intégration d'autres facteurs humains. Toutefois, dans le contexte de cette thèse, ce constat doit être nuancé. En effet, la similarité a été la métrique reine pendant plusieurs années dans le domaine de la recommandation pour mesurer la performance des modèles. Cette omniprésence a provoqué l'essor d'un nouveau pan de la recherche portant sur d'autres facteurs humains et d'autres métriques d'évaluation. Néanmoins, la similarité reste une métrique centrale dans la mesure des performances d'un algorithme et dans la qualité perçue des recommandations des utilisateurs comme l'ont montré les multitudes d'études utilisant principalement cette métrique. Il est donc naturel que cette métrique soit intégré dans notre modèle comme un objectif. Notre approche se veut cependant multi-objectifs et nous allons présenter dans la suite de cette section les autres objectifs à considérer.

2.2.2 La diversité

La diversité est une des métriques ayant attiré le plus l'attention de la communauté scientifique des recommandations ces dernières années. La diversité est au cœur du changement de paradigme du domaine et l'article de McNee & al. justement intitulé "*Being accurate is not enough : how accuracy metrics have hurt recommender systems*"[17] représente bien ce changement [McNee et al., 2006]. Dans cet article, les auteurs expliquent que, jusqu'alors, la majorité des recherches s'étaient étroitement concentrées sur l'amélioration de la précision des systèmes. Ils soutiennent cependant que cet état de fait a nui au domaine en n'encourageant pas les recherches allant dans d'autres directions, et a aussi nui aux utilisateurs car les recommandations les plus précises ne sont pas forcément les meilleures pour ces derniers. Même si ce constat a été fait il y a plus d'une décennie, la précision est toujours très utilisée dans le domaine des recommandations. Par conséquent, il n'est pas rare de trouver des études très récentes se démarquant encore en prônant l'utilisation d'autres mesures [Gan et al., 2020].

La diversité et la similarité sont deux mesures extrêmement liées et ont souvent été décrites ensemble. La similarité intra-liste, présentée dans la section précédente, est un exemple de ce lien puisque cette mesure permet de faire varier le taux de similarité, ou inversement de diversité, d'une liste de ressources. Dans [Smyth and McClave, 2001], les auteurs soutiennent l'idée que lorsque plusieurs ressources sont proposées, il est important de mesurer à la fois la similarité de chacune de ces ressources par rapport à la "cible" (pouvant être par exemple un modèle utilisateur), mais aussi la diversité de chacune de ces ressources entre elles. Le défi revient alors à améliorer la diversité des ressources entre elles sans compromettre leur similarité par rapport à la cible. Les auteurs définissent ainsi la diversité d'un ensemble de ressources $(i_1, \ldots, i_n)$ comme la dissimilarité moyenne entre toutes les paires de ressources, illustré dans l'Équation 2.4, où la dissimilarité est définie comme étant l'inverse de la similarité.

$$Diversity(i_1, \ldots, i_n) = \frac{\sum_{i=1..n} \sum_{j=1..n} (1 - Similarity(i_i, i_j))}{\frac{n}{2} * (n-1)} \tag{2.4}$$

Il existe plusieurs interprétations et définitions de la diversité dans les systèmes de recommandation, mais nous n'utiliserons ici que la plus répandue vue ci-dessus : la diversité s'applique à un ensemble de ressources et correspond à la mesure moyenne de la dissimilarité par rapport à chacune d'entre elles. Pour complémenter cette définition qui permet d'introduire une valeur générale de diversité pour une liste de ressources, nous pouvons présenter ici la notion de diversité

17. "Être précis n'est pas suffisant : comment les mesures de précision ont nui aux systèmes de recommandation".

relative RD [18] qui consiste à calculer l'apport en diversité de chaque ressource i à un ensemble de ressources C [19] [Smyth and McClave, 2001]. La RD permet d'observer les changements dans la diversité de la liste au moment même de sa construction à chaque fois que le système ajoute une nouvelle ressource à la liste déjà existante. La manière de la calculer est expliquée dans l'Équation 2.5.

$$RD(i,C) = \begin{cases} 0 \text{ si } C = \{\}, \\ \dfrac{\sum_{i=1..m}(1 - Similarity(i, c_j))}{m}, \text{ sinon.} \end{cases} \tag{2.5}$$

La diversité est aujourd'hui reconnue comme une mesure importante à prendre en compte dans les systèmes de recommandation car permettant d'améliorer la satisfaction de l'utilisateur [Jones, 2010]. Cependant, il est important de savoir utiliser la diversité à bon escient et au bon moment. Comme l'exemple donné dans l'introduction à propos des appareils photos le montrait, l'intérêt de la diversité dans la recommandation dépend de nombreux facteurs, comme l'utilisateur, son ancienneté dans le système, le contexte de sa visite, le moment de la recommandation, etc. Certains travaux se sont ainsi concentrés sur la dimension temporelle de la diversité et sur l'impact que cette dernière avait selon le contexte de la recommandation. On peut citer [McGinty and Smyth, 2003] dont leurs travaux portent sur le rôle de la diversité dans les systèmes de recommandation conversationnels [20] et sur l'intérêt d'apporter de la diversité dans ce genre de systèmes. Les auteurs ont ainsi montré que l'ajout de diversité durant le processus de recommandation était bénéfique pour l'utilisateur. Cependant, ils ont aussi montré que, même si l'intérêt de la diversité est indéniable, il n'est pas forcément justifié d'introduire de la diversité à chaque recommandation sous peine de nuire à l'efficacité du système [Castagnos et al., 2013]. De la même manière, on peut rappeler les travaux de [Castagnos et al., 2010] prouvant qu'il est bénéfique pour l'utilisateur de se voir proposer des recommandations diverses juste avant un achat sur un site d'e-commerce.

2.2.3 La nouveauté

La nouveauté est un facteur humain qui, sous la même impulsion que la diversité, est apparue comme une réponse possible au problème de la sur-utilisation de la précision. Même si les deux mesures de diversité et de nouveauté sont liées, il convient de ne pas les confondre. La nouveauté d'une ressource se définit généralement par rapport aux autres ressources déjà consultées par l'utilisateur. La notion de diversité s'applique, quant à elle, à un ensemble de ressources en déterminant si elles sont diverses les unes par rapport aux autres [Castells et al., 2015]. La nouveauté répond de plus à un besoin bien spécifique de l'utilisateur. En effet, et comme l'expliquent [Vargas and Castells, 2011], dans la majorité des scénarios, le but d'une recommandation est de faire découvrir à l'utilisateur des ressources qu'il n'aurait pas découvertes autrement. Les auteurs précisent aussi que la nouveauté est une caractéristique importante dans un système de recommandation, car elle contribue à diminuer la prévisibilité des recommandations, et donc à maintenir l'intérêt des utilisateurs pour le système. En partant de ce principe, les recommandations évidentes ne sont pas forcément de grande utilité, même si elles possèdent un haut niveau de précision avec les préférences de l'utilisateur. On peut illustrer ce point en

18. RD pour *Relative Diversity*.

19. C pour *Class*, soit classe/ensemble de ressources.

20. Un système de recommandation conversationnel consiste en un système interactif qui va "dialoguer" avec l'utilisateur sous la forme d'échange de propositions et de critiques de ces propositions par ce dernier [Mahmood and Ricci, 2009].

citant le classique exemple du fan de musique rock qui, après avoir écouté quelques musiques de l'album *Dark Side of the Moon* de *Pink Floyd* se voit recommander le reste des musiques de l'album. En terme de précision, la recommandation est quasi parfaite, mais l'utilité d'une telle recommandation peut laisser à désirer. On retrouve le même constat chez [Zhang, 2013] où l'omniprésence de la précision et de ses mesures y est critiquée. Les auteurs proposent ainsi une définition de la nouveauté en définissant les trois caractéristiques qu'une ressource doit posséder pour être considérée comme nouvelle pour un utilisateur :

1. Inconnue : la ressource doit être inconnue de l'utilisateur ;
2. Satisfaisante : la ressource doit être satisfaisante par rapport aux besoins de l'utilisateur ;
3. Dissimilaire : la ressource doit être dissimilaire aux autres ressources que l'utilisateur apprécie.

En utilisant les 3 caractéristiques énoncées ci-dessus, les auteurs ont ensuite proposé d'évaluer la nouveauté d'une ressource i pour un utilisateur u, comme présenté dans l'Équation 3.4.

$$Novelty(i,u) = p(i|unknown,u) \cdot dis(i,pref_u) \cdot p(i|like,u) \tag{2.6}$$

où $p(i|unknown,u)$ est la probabilité que l'utilisateur u ne connaisse pas la ressource i, $dis(i,pref_u)$ est la dissimilarité entre i et l'ensemble des ressources consultées par l'utilisateur et $p(i|like,u)$ est la probabilité que u apprécie i. On retrouve ainsi dans cette manière de mesurer la nouveauté les 3 caractéristiques mises en avant par [Zhang, 2013] : pour que le score de nouveauté d'une ressource soit haut, elle doit être pour l'utilisateur à la fois inconnue, satisfaisante et dissimilaire avec ce qu'il a l'habitude de voir.

De récentes études ont exploré plus avant le lien entre nouveauté et satisfaction de l'utilisateur. Nous pouvons citer ici [Schnabel et al., 2018] où les auteurs ont testé l'impact de nouvelles recommandations par rapport à l'exploitation des préférences déjà connues des utilisateurs. Ils en concluent qu'une petite dose de nouveauté permet d'améliorer la satisfaction de l'utilisateur mais que trop de ressources nouvelles nuisent fortement à sa satisfaction.

2.2.4 Discussion

Au travers de ces quelques exemples, un changement de vision dans le domaine de la recommandation a été illustré. D'un domaine où la majeure partie des études était initialement concentrée sur la maximisation d'une seule métrique, la précision [Ricci et al., 2015], nous avons vu que de plus en plus de travaux s'intéressaient à d'autres manières de proposer des recommandations. Ainsi, on peut désormais trouver pléthore d'études tentant de trouver des compromis entre plusieurs des facteurs discutés ci-dessus, comme la précision et la diversité [Ziegler et al., 2005, L'Huillier et al., 2014], ou encore entre la précision, la sérendipité [21] et la nouveauté [Kaminskas and Bridge, 2016]. Ces nouvelles manières de procéder ont pour but d'identifier les facteurs explicatifs de la satisfaction des utilisateurs afin de les prendre en compte dans des modèles plus complexes et plus généraux.

Il est cependant important de noter que, outre ceux cités dans les sections précédentes, de nombreux autres facteurs humains ont été étudiés dans la littérature pour répondre au problème

21. La notion de sérendipité fait référence à "l'heureuse découverte", soit le fait de découvrir par hasard quelque chose d'intéressant qui n'aurait potentiellement pas été découvert sans aide [Iaquinta et al., 2008].

de la satisfaction utilisateur et de la sur-utilisation de la précision. On peut citer la sérendipité, la couverture, le contexte, la personnalité de l'utilisateur... Nous avons fait le choix de nous baser sur 4 facteurs couramment dans la littérature afin de tester la problématique de cette thèse sur la recommandation multi-objectifs. La multitude de facteurs humains étudiés ces dernières années pour enrichir les recommandations soulignent à quel point il est important de nos jours de penser des modèles génériques capables d'intégrer de nouveaux facteurs humains à la demande.

Cette première grande section de l'état de l'art a montré que le domaine de la recommandation est passé d'un problème d'optimisation mono-objectif à un problème d'optimisation multi-objectifs. Un système de recommandation à l'heure actuelle se doit de prendre en compte les facteurs humains tels que ceux présentés dans les sections précédentes afin d'être performant et capable de satisfaire le plus grand nombre de type d'utilisateurs dans le maximum de situations possibles. C'est dans ce contexte multi-objectif que cette thèse se positionne. Dans la deuxième et dernière grande section de cet état de l'art, nous allons présenter et comparer différentes manières de proposer des recommandations à l'utilisateur : la recommandation unitaire, la recommandation en liste et la recommandation en séquences. Nous allons notamment nous attarder sur les listes et les séquences car elles sont les plus à mêmes de fournir des recommandations complexes intégrant de multiples objectifs.

2.3 La recommandation en séquence

2.3.1 Définition et généralités sur les séquences

Pour comprendre l'intérêt des séquences, il est nécessaire de s'attarder sur les listes. En effet, dès les premiers systèmes de recommandation, les utilisateurs se sont la plupart du temps vu proposer soit des ressources uniques soit des listes de ressources. On peut définir une liste ainsi :

Dans le domaine de la recommandation, une liste est un ensemble de ressources ordonnées selon un ou plusieurs critères, offrant à l'utilisateur le choix de consulter ces ressources indépendamment les unes des autres.

Présenter une liste de recommandations a ainsi pour but de proposer les ressources ayant obtenu le score le plus haut selon un ou plusieurs critères choisis par le système pour produire des recommandations. Comme nous l'avons vu dans la section précédente, un des critères les plus utilisés était originellement la précision. L'autre spécificité d'une liste est d'offrir à un utilisateur plusieurs ressources adaptées à ses préférences à un temps t. Ainsi, le but n'est pas moins de satisfaire l'objectif à moyen/long terme de l'utilisateur que de simplement lui proposer une liste de ressources alternatives adaptées à son prochain choix. Autrement dit, le but d'une liste de recommandations est de satisfaire les besoins d'un utilisateur au temps $t+1$ (soit à sa prochaine action). Le format des listes a donc souvent été utilisé car il permet à la fois de maximiser les chances de satisfaire l'utilisateur tout en tirant parti des mécanismes internes de production de recommandations générant la plupart du temps des listes de ressources, de la plus précise à la moins précise pour chaque utilisateur. Ce mode de représentation, de par sa capacité à proposer des alternatives, se rapproche déjà un peu plus du cadre de notre travail d'optimisation multi-critères contrairement aux recommandations unitaires. Il n'est donc pas question ici de remettre en cause l'intérêt d'utiliser des listes de ressources comme format de recommandations, ces dernières étant adaptées à de nombreux domaines d'application. De très populaires sites

d'e-commerce ou de divertissement utilisent d'ailleurs principalement des recommandations par listes, comme Netflix, Youtube ou encore Amazon. Un client sur un site d'e-commerce pourra par exemple être satisfait de se voir proposer par le système une liste d'articles similaires à ses recherches afin de l'aider dans son choix. Cependant, certains domaines peuvent ne pas bénéficier, voire être desservis, par ce format de recommandation. On peut ici à nouveau rappeler l'exemple directeur de l'introduction à ce sujet, où un apprenant se voyait proposer une liste de ressources éducatives toutes similaires au cours qu'il avait déjà suivi, sans tenir compte de sa progression ni de ses objectifs. De la même manière, un visiteur d'un musée ne sera pas forcément satisfait de se voir proposer une liste de 10 œuvres d'art similaires à ce qu'il vient de voir mais potentiellement éparpillées dans tous le musée. Dans ces deux exemples, on devine bien que d'autres manières de proposer les recommandations seraient bénéfiques. C'est ici que l'on peut définir la notion de séquence qui pourrait permettre de répondre à des situations où une liste de ressources n'est pas suffisante pour satisfaire l'utilisateur et où la composante temporelle entre en ligne de compte :

Dans le domaine de la recommandation, une séquence est une suite de ressources permettant d'atteindre un ou plusieurs objectifs, spécifiés ou non par l'utilisateur (e.g. une compétence spécifique, la sortie d'un musée, un apprentissage, des recommandations diverses pour un parcours éclectique, etc.). Une séquence possède donc un début, une fin et une progression définie sous forme d'un ordonnancement temporel. Une séquence est régie par les objectifs à atteindre, ces derniers étant spécifique au domaine et à l'utilisateur.

Une séquence peut donc être vue comme une liste "améliorée" de ressources, prenant un compte un nombre plus grand de variables et une dimension temporelle, dans le but de mieux satisfaire l'utilisateur. Nous allons ici détailler la notion de progression, qui est importante dans une séquence. Le principe d'une séquence réside dans l'existence d'au moins un objectif à atteindre, et c'est ce qui la différencie d'une liste. Comme nous l'avons vu, une liste de ressources n'a pas fondamentalement d'objectif à atteindre, si ce n'est satisfaire l'utilisateur à $t+1$ en lui proposant des ressources adaptées. De part la notion d'objectif à atteindre, une séquence a pour but de satisfaire les besoins de l'utilisateur, non plus à $t+1$, mais à $t+n$. Il n'est donc plus question d'aider l'utilisateur pour sa prochaine action, mais bien d'aider l'utilisateur à atteindre un objectif plus complexe qui nécessitera n actions ou paliers intermédiaires. Avec ces deux définitions, on se rend bien compte que les notions de liste et de séquence ne sont pas opposées mais complémentaires. Certains domaines ne nécessitent pas forcément l'utilisation de séquences car l'utilisateur n'aura pas d'objectif à moyen/long terme à satisfaire (ex. : la recherche d'un film sur une plateforme de vidéos à la demande) tandis que dans d'autres domaines, une liste sera insuffisante pour satisfaire l'utilisateur (ex. : e-éducation, héritage culturel, playlist musicale). De la même manière, il n'est pas nécessaire de recommander des séquences à un utilisateur si ce dernier n'a pas d'objectif à atteindre et souhaite simplement une aide pour sa prochaine action, et ce, même si le domaine d'application est propice à l'utilisation de séquences.

L'intérêt des séquences n'a pas échappé aux chercheurs du domaine des systèmes de recommandation qui les ont considéré comme étant un moyen supplémentaire permettant de mieux satisfaire l'utilisateur. Cette prise de conscience s'est d'ailleurs démocratisée au même moment que la prise en compte des facteurs humains autre que la précision dans l'article déjà cité précédemment [McNee et al., 2006]. Ainsi, on peut lire en 2004 dans le très complet article sur

l'évaluation des systèmes de recommandation de [Herlocker et al., 2004], une liste des tâches usuelles que doivent réaliser ces systèmes. Dans cette liste se trouve entre autres une réflexion des auteurs sur la recommandation de séquences dans le cadre de l'utilisation d'un site d'écoute de musiques en ligne :

"*[...] Cependant, le défi que représente le fait de passer d'une recommandation d'une seule musique à une recommandation d'une séquence de musiques satisfaisante dans son ensemble nous a intrigué. Cette même tâche peut s'appliquer à la recommandation d'articles de recherche afin d'appréhender un domaine (lire cette introduction, puis lire cette étude, ...). Alors que le domaine de la recherche d'information a exploré le timing et les séquences d'achat de produits, nous n'avons connaissance d'aucun système de recommandation ou de recherche s'attelant directement à la résolution de cette tâche*" (*i.e.* la recommandation en séquences).

Les auteurs expliquent donc clairement ici que la recommandation de séquences, même si elle n'avait pas encore été traitée à l'époque, était un point intéressant à développer et pouvait satisfaire des besoins spécifiques, comme la recommandation de musiques en ligne ou encore l'acquisition de nouvelles connaissances en ligne. La première partie de leur réflexion qui n'a pas été traduite ici sur les spécificités des recommandations musicales sera étudiée plus en détail dans le chapitre suivant. Même si les recommandations de ressources uniques ou de listes de ressources sont restées très largement majoritaires dans le domaine, quelques travaux se sont tout de même attelés au problème des séquences. Il est tout d'abord nécessaire de distinguer deux utilisations de la notion de séquence dans la littérature : l'utilisation de séquences déjà existantes en **entrée** d'un système et la production de séquences en **sortie** d'un système. Dans la suite de cette section, nous allons étudier ces deux manières de procéder ainsi que leurs avantages et inconvénients.

Comme nous l'avons vu dans la majorité des cas, les systèmes de recommandation se basent sur la matrice d'évaluations *utilisateurs* $\times$ *ressources* pour prédire les ressources encore non consultées pouvant intéresser l'utilisateur. Cependant, même si elle permet l'application de nombreuses et puissantes méthodes de prédiction que nous ne détaillerons pas ici[22], l'utilisation d'une matrice ne permet pas en l'état de tenir compte de l'ordre dans lequel les ressources ont été consultées, ni de distinguer les variances à court ou moyen-terme des profils utilisateurs [Quadrana et al., 2018]. De la même manière, une cellule de cette matrice r_{ui} représente uniquement une évaluation r d'un utilisateur u sur une ressource i, pouvant cacher une réalité plus complexe où un utilisateur a consulté plusieurs fois cette ressource au cours du temps, potentiellement dans différents contextes et avec une évaluation différente à chaque fois. Pour remédier à ce problème, des approches à base de tenseurs ont été proposées, permettant d'utiliser plusieurs matrices représentant différents moments ou critères dans le profil de l'utilisateur [Frolov and Oseledets, 2017]. Il est donc clair que les séquences de consultation contiennent des informations riches pouvant être exploitées dans un système de recommandation. L'exploitation de l'ordre des ressources dans des séquences n'est cependant pas nouvelle et provient principalement du domaine de la recherche d'information [Agrawal et al., 1995]. Aujourd'hui, on retrouve l'exploitation de l'ordre des séquences dans de nombreux autres domaines comme la fouille de motifs séquentiels ou encore la reconnaissance de la parole [Han et al., 2007]. L'explosion des services proposés par la démocratisation d'Internet et du numérique en général ces dernières années a amené la recommandation à s'adapter à de nombreux domaines où l'ordre dans lequel les utilisateurs réalisent

22. On peut par exemple citer les techniques de factorisation de matrice qui dominent le domaine [Koren et al., 2009].

des actions a de l'importance. Ces systèmes de recommandation portant une attention particulière aux séquences sont regroupés sous le terme de **systèmes de recommandation sensibles aux séquences** (*sequence-aware recommender system*) [Quadrana et al., 2018]. Dans leur tour d'horizon du domaine, les auteurs proposent une vue d'ensemble de la manière dont un système de recommandation sensible aux séquences fonctionne à l'aide d'un schéma en 3 parties, présenté par la Figure 2.2.

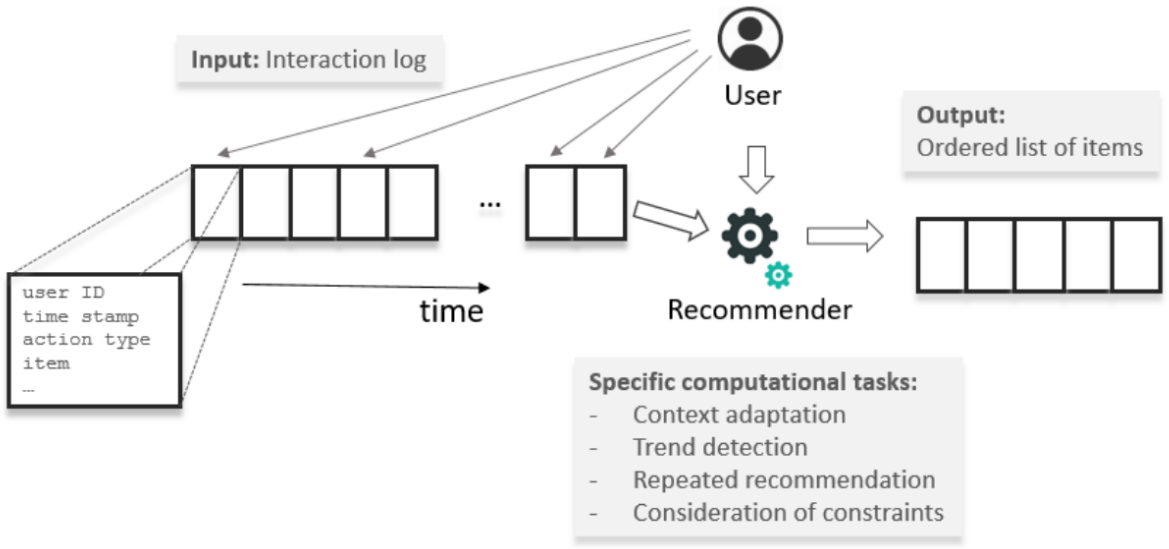

FIGURE 2.2 – Vue d'ensemble du fonctionnement d'un système de recommandation sensible aux séquences (source : [Quadrana et al., 2018]).

Dans la Figure 2.2, on peut donc bien distinguer 3 parties distinctes dans le fonctionnement interne d'un système de recommandation sensible aux séquences :

1. **Les entrées** : les entrées se caractérisent par des données relatives aux actions effectuées par les utilisateurs du système. Ces actions doivent être classées selon l'ordre dans lequel elles ont été effectuées par chaque utilisateur afin de pouvoir exploiter toutes les informations disponibles (si les actions ne sont pas reçues de manière ordonnée, ces dernières doivent au minimum être associées à un *timestamp*, à une date précise, pour pouvoir être ré-ordonnées et traitées dans le bon ordre par le système de recommandation).
2. **Le processus de recommandation** : le processus de recommandation est généralement différent des systèmes de recommandation classiques utilisant des matrices[23], mais le principe général reste le même : utiliser l'ensemble des informations fournies par les entrées pour produire des recommandations adaptées à l'utilisateur final. On retrouve dans cette partie les tâches purement liées à la recommandation : la prise en compte des facteurs humains, du contexte éventuellement, des contraintes s'il y en a, etc. Dans la Figure 2.2, les auteurs distinguent 4 types de tâches liées aux systèmes de recommandation sensibles aux séquences. Nous discuterons de ces 4 tâches dans la suite de ce manuscrit lors de la présentation du modèle et des expérimentations menées pour le tester.

23. Même si quelques modèles utilisant à la fois des séquences et des méthodes de factorisation de matrices ont été proposés ces dernières années [Yu and Riedl, 2012, Zhao et al., 2014, Twardowski, 2016, Chen and Li, 2019].

3. **Les sorties** : les auteurs distinguent ici deux types de sorties possibles. Le premier type de sortie concerne d'abord les listes ordonnées de ressources que nous avons déjà explicitées auparavant dans ce manuscrit. Une liste aura pour but de proposer à l'utilisateur un ensemble de ressources ordonnées de la meilleure à la moins bonne selon une ou plusieurs métriques spécifiques. Une liste propose dans les faits un ensemble d'alternatives, de choix possibles, que le système met à disposition de l'utilisateur afin que ce dernier puisse choisir une (ou éventuellement plusieurs) ressource(s) parmi toutes celles proposées. Ce premier type de sortie est donc identique aux systèmes de recommandation classiques. Le deuxième type de sortie concerne les séquences de ressources. Contrairement aux listes consistant en un ensemble d'alternatives, les ressources d'une séquence ont pour but d'être toutes consultées dans l'ordre donné par le système. Là où pour les listes, une recommandation consiste en une ressource, pour les séquences une recommandation consiste en une suite de ressources à consulter dans l'ordre proposé. Ainsi, une séquence peut être considérée comme une unique recommandation, de la même manière que dans une liste chaque ressource est une recommandation unique.

2.3.2 Domaines d'application des modèles séquentiels

Comme nous l'avons expliqué auparavant, il est évident que certains domaines d'applications sont adaptés aux systèmes de recommandation sensibles aux séquences, comme par exemple la musique ou le tourisme [Nurbakova et al., 2018]. Inversement, certains domaines semblent moins bénéficier de tels systèmes. Le domaine de l'e-commerce est un cas particulier dans le sens où il a vu un grand nombre de ces systèmes y être testés, quand bien même il semble à première vue être moins adaptés aux séquences. Deux raisons peuvent expliquer cet état de fait. Premièrement, il a statistiquement été prouvé que la très grande majorité des clients consultant un site d'e-commerce ne le font en réalité que dans le but de trouver un seul produit spécifique avant de quitter le site [Qiu et al., 2015]. Deuxièmement, le domaine de l'e-commerce est généralement très représenté dans les systèmes de recommandation [Quadrana et al., 2018], potentiellement dû aux forts intérêts financiers et à l'ancienneté de l'utilisation de ce domaine d'application. Dans la suite de cette section, nous allons nous intéresser à quelques exemples de systèmes sensibles aux séquences dans les domaines de la musique et du tourisme afin d'illustrer l'état du domaine ainsi que les directions potentielles pour de futurs travaux.

Dans le domaine musical

Le domaine de la musique est un débouché naturel des systèmes de recommandation s'intéressant aux séquences, autant quand il s'agit d'exploiter les séquences créées par les utilisateurs que lorsqu'il s'agit d'en produire en sortie d'un système. Cet état de fait s'explique par la spécificité du domaine musical où les utilisateurs écoutent très souvent plusieurs musiques les unes à la suite des autres. Ces utilisateurs produisent des séquences significatives de par leurs écoutes et peuvent être demandeurs de séquences de musiques, ou playlists, adaptées à leurs préférences. Il n'est donc pas étonnant de trouver dès le début des années 2000 dans la littérature des travaux portant sur ce sujet. Les travaux de l'époque souffraient cependant d'un certain nombre de problèmes, comme un manque de personnalisation et d'expressivité ou encore des difficultés pour gérer le grand nombre de musiques des catalogues en ligne, problèmes qui ont par la suite été résolus avec la production de nouveaux systèmes et l'augmentation de la puissance de calcul des machines [Aucouturier and Pachet, 2002]. On peut cependant aussi trouver des études dont le

but était d'exploiter les caractéristiques spécifiques du domaine musical pour proposer des adaptations des techniques classiques de recommandation utilisées dans l'e-commerce [Logan, 2002]. Plus récemment, le domaine des recommandations musicales a beaucoup gagné en popularité, comme l'illustre le célèbre *ACM RecSys Challenge* dont l'édition 2018, en partenariat avec Spotify [24], s'est portée sur la musique avec comme challenge de créer un système capable de prolonger des playlists déjà existantes [25]. Au total, plus de 100 équipes ont participé à ce défi et plus de 15 articles ont été publiés. Voici un court résumé des 4 approches ayant été les plus performantes :

— [Volkovs et al., 2018] ont proposé un modèle en deux étapes basé sur des méthodes de filtrage collaboratif utilisant la factorisation de matrice (*Weighted Regularized Matrix Factorization*[Hu et al., 2008]) et de renforcement de gradient (*gradient boosting*). La première étape est optimisée pour récupérer rapidement une liste de musiques candidates tandis que le deuxième stage réordonne ces musiques en maximisant la précision au début de la liste ainsi créée. [Rubtsov et al., 2018] ont proposé une approche relativement similaire avec un modèle possédant aussi deux étapes, la première utilisant des techniques de filtrage collaboratif pour sélectionner des candidats potentiels et la deuxième utilisant une playlist avec ces candidats à l'aide de techniques de renforcement de gradient ;
— [Yang et al., 2018] ont proposé un modèle multimodal de filtrage collaboratif en deux parties utilisant chacune des réseaux de neurones (*autoencoder* pour la première partie et réseau neuronal convolutif pour la deuxième partie). Le but des auteurs était de prendre en compte 3 des problèmes les plus communs aux systèmes de recommandation musicaux, à savoir : (1) le problème du démarrage à froid ; (2) les biais de popularité où des musiques se retrouvent soit dans la plupart des playlists soit dans très peu d'entre elles ; (3) et enfin le contexte de la playlist originale (autrement dit, le fait d'utiliser les préférences à court terme de l'utilisateur dans la session d'écoute en cours pour continuer du mieux possible la playlist) ;
— [Antenucci et al., 2018] ont proposé une approche mêlant à la fois des techniques de filtrage collaboratif et de filtrage par contenu. Les prédictions de ces différentes techniques sont ensuite agrégées ensemble dans le but de maximiser la qualité de la recommandation.

On peut remarquer que ces 4 approches, ayant obtenu les 4 premières places de la compétition, possèdent des points communs et offrent des perspectives intéressantes sur la manière de concevoir des systèmes de recommandation performants dans ce domaine. Premièrement, on remarque que toutes ces approches sont relativement complexes et utilisent des méthodes variées pour la recommandation (filtrage collaboratif, filtrage par contenu, factorisation de matrice, réseaux neuronaux, gradient boosting, etc.), prouvant que les systèmes de recommandation performants utilisent plus que jamais des techniques hybrides. Deuxièmement, chacune de ces 4 approches propose un modèle en deux parties. La première partie est généralement utilisée pour sélectionner un certain nombre de candidats parmi les 2,2 millions de musiques disponibles qui avaient été mises à disposition, et ce afin de limiter l'espace de recherche et donc le temps de calcul. La deuxième partie consiste en la création de la playlist finale basée sur les candidats déjà sélectionnés. Cette partie peut potentiellement être plus complexe d'un point de vue algorithmique car l'espace de recherche est beaucoup plus restreint qu'initialement.

Par ailleurs, chacune de ces 4 approches a aussi tenté de prendre en compte un maximum

24. https ://www.spotify.com
25. http ://www.recsyschallenge.com/2018/

d'informations afin de satisfaire au mieux l'utilisateur (caractéristiques des musiques, précision, diversité, contexte de la playlist initiale, utilisation de la session d'écoute en cours...), validant ainsi l'approche développée depuis le début de ce manuscrit qui consiste à prendre en compte un maximum d'objectifs associés à des facteurs humains dans la recommandation. Cette constatation se retrouve aussi chez [Jannach et al., 2016] où les auteurs expliquent que "*le but des systèmes de recommandation dans le domaine de la musique est d'acquérir et d'exploiter des informations supplémentaires à propos des musiques afin de comprendre le thème sous-jacent des playlists créées par les utilisateurs [...]. Le but ultime serait ainsi de concevoir de nouvelles approches algorithmiques qui seraient explicitement conçues pour créer des playlists correspondant à une ou plusieurs caractéristiques significatives pour l'utilisateur*". En dehors des facteurs humains déjà développés au début de cette section (similarité, diversité, nouveauté), les auteurs explicitent ces "caractéristiques significatives" comme pouvant aussi être l'homogénéité de la playlist dans son ensemble ou encore les transitions entre chacune des musiques d'une playlist. Le but serait donc ici de réussir à créer des playlists semblables à ce qu'un être humain pourrait faire tout en tenant compte au maximum des besoins en similarité diversité et nouveauté. [Quadrana et al., 2018] développent aussi cette idée en expliquant qu'un système de recommandation musical peut soit proposer des musiques correspondant d'une manière générale au thème de la playlist, soit proposer une playlist avec un ordre spécifique permettant de garantir des transitions douces entre chaque musique, comme par exemple une augmentation graduelle du tempo. Les notions de progression et de transitions douces dans une playlist correspondent à la définition d'une séquence donnée plus haut dans le manuscrit et sont donc particulièrement intéressantes dans le cadre de cette thèse. Il est cependant à noter que, dans le domaine des recommandations musicales, un débat est récemment apparu sur l'importance de l'ordre des musiques dans les recommandations de playlist. En effet, une étude exploratoire récente a montré que, si les utilisateurs jugent positivement un système recommandant des musiques diverses et nouvelles, ces derniers ne portaient que peu d'intérêt à l'ordre dans lequel les musiques étaient proposées (voire dans certains cas ne percevaient même pas d'ordre spécifique dans les recommandations) [Tintarev et al., 2017]. Même si cette étude n'a porté que sur 20 participants, les résultats obtenus sont intéressants et soulignent à quel point chaque domaine à des particularités propres dont il est nécessaire de tenir compte pour la conception d'un système de recommandation sensible aux séquences. Pour renforcer encore cette dernière constatation, nous allons nous intéresser dans la prochaine sous-section aux systèmes de recommandation touristiques.

Dans le domaine touristique

Le domaine du tourisme a lui aussi vu sa popularité croître dans la littérature grâce à l'explosion des téléphones portables et tablettes offrant un nouveau support pour les recommandations. De la même manière que pour le domaine de la musique, le tourisme a vu ses premières applications de recommandation apparaître au début des années 2000. Dans [Cheverst et al., 2000] par exemple, les auteurs ont développé un guide mobile permettant d'aider les visiteurs de la ville de Lancaster en leur fournissant des informations contextuelles (localisation, date, heure de la journée, ouvertures et fermetures d'attractions, heures de visites, etc.) ainsi qu'en proposant au visiteur plusieurs services, comme avoir plus d'informations sur un monument ou générer un parcours prenant en compte ces informations contextuelles. Cette étude a permis de proposer un standard dans la recommandation touristique, comme la prise en compte du contexte, et était très complète pour son époque, même si l'on peut critiquer l'absence d'un modèle utilisateur ne permettant pas de prendre en compte les besoins spécifiques de chaque visiteur. Toujours au début des années 2000, nous pouvons aussi citer [Chou et al., 2005] dont les auteurs ont modélisé,

à l'aide d'un système de représentation de connaissances (OWL pour *Web Ontology Language*), les intérêts d'un visiteur ainsi que les œuvres d'un musée dans un système de recommandation proposant des parcours. Il est à noter que, dans cette étude, les auteurs ont pu prouver que le temps passé par un visiteur devant un objet était positivement corrélé avec son appréciation positive sur cet objet. Le terme "objet" est utilisé ici car les auteurs ont réalisé cette expérience sur un site proposant des vêtements, et non dans le musée comme le reste de leur application.

Plus récemment, les systèmes de recommandation touristiques de ces dernières années ont profité des nombreuses avancées technologiques, leur donnant plus de possibilités quant à la localisation de l'utilisateur et à l'aide personnalisée dans des lieux autrefois complexes à atteindre. En effet, la démocratisation des smartphones, recevant tous un signal GPS de bonne qualité, permet désormais à des applications de connaître relativement précisément la position d'un visiteur tant qu'il est dehors. D'une manière générale, les systèmes utilisant des données géographiques dans le processus de recommandation sont regroupés sous le terme "systèmes de recommandation sensibles à la position" (*Location-Aware Recommender Systems (LARS)*). Pour les systèmes de recommandation destinés à être utilisés à l'intérieur d'un bâtiment où le signal GPS ne peut pas passer (comme un musée par exemple), il existe maintenant plusieurs solutions étudiées et développées dans le domaine de recherche des systèmes de localisation en interne, comme par exemple le Wi-Fi [Tesoriero et al., 2014], les puces RFID [26] [Tesoriero et al., 2008], les senseurs de proximité Bluetooth [Yoshimura et al., 2014] et même le géomagnétisme [27]. Les moyens techniques permettant de localiser précisément un visiteur ou de lui fournir des recommandations à distance via une application ne seront pas détaillés ici. Cependant, il est important de prendre en compte ces problèmes lorsque l'on passe d'un domaine à un autre. Même si l'objectif est le même, à savoir recommander les meilleures séquences à un utilisateur d'un système, les moyens techniques à mettre en œuvre ainsi que les caractéristiques à prendre en compte pour produire de bonnes recommandations peuvent drastiquement changer. Ainsi, alors que certains systèmes de recommandations musicaux commencent depuis quelques années à prendre en compte le contexte de l'utilisateur quand cela est possible [Wang et al., 2012], les systèmes déployés dans le domaine du tourisme doivent nécessairement tenir compte du contexte de l'utilisateur, au minimum pour pouvoir le guider activement [Adomavicius and Tuzhilin, 2011, Gavalas et al., 2014]. De la même manière, les modèles utilisateurs requis dans les recommandations touristiques diffèrent de ceux des autres domaines. Il est par exemple important de prendre en compte la manière dont les visiteurs se déplacent physiquement, à la fois pour proposer des recommandations satisfaisantes d'un point de vue individuel, mais aussi pour avoir une vision globale de la foule et de ses effets sur les individus. Ainsi, [Véron and Levasseur, 1989] ont réalisé une étude ethnographique au centre Pompidou montrant que les visiteurs pouvaient être classés en 4 catégories selon leur manière de se déplacer tandis que [Yoshimura et al., 2014] ont observé pendant 3 semaines le comportement de plus de 27.000 visiteurs au musée du Louvre, permettant de discerner 2 types de visiteurs selon la durée de leur visite ainsi que l'influence de la foule sur les comportements.

Ces deux études, toujours très utilisées aujourd'hui, révèlent que le domaine des recommandations touristiques possède des spécificités qu'il est important de prendre en compte afin de correctement modéliser le comportement de l'utilisateur et de lui fournir des recommandations pertinentes. D'une manière générale, et peut-être encore plus que dans d'autres domaines, la

26. Acronyme de Radio-Frequency IDentification, désignant l'utilisation de radio-étiquettes adhésives permettant de recevoir et de répondre à des signaux à courte distance.

27. IndoorAtlas.

recommandation de points d'intérêts physiques nécessite une attention toute particulière à un certain nombre de caractéristiques quant à la modélisation de l'utilisateur, parmi lesquelles on peut citer [Osche et al., 2016] :

- **Le style de visite** : représente le style de visite de l'utilisateur, en rapport avec les deux études citées ci-dessus [Véron and Levasseur, 1989, Yoshimura et al., 2014] ;
- **Le niveau de fatigue** : représente à quel point l'utilisateur est mentalement et/ou physiquement fatigué ;
- **Les préférences implicites** : représente les préférences sur les ressources déduites du comportement de l'utilisateur ;
- **La tolérance à la distance** : représente à quel point l'utilisateur souhaite se déplacer ;
- **La tolérance à la foule** : représente à quel point l'utilisateur est dérangé par les autres visiteurs autour de lui ;
- **La tolérance à la précision** : représente le souhait de l'utilisateur concernant la balance entre recommandations précises et recommandations diverses ;
- **La tolérance à l'intrusion technologique** : représente à quelle fréquence l'utilisateur souhaite recevoir des recommandations du système ;
- **Le niveau de contrôle** : représente le niveau de choix que l'utilisateur attend, c'est-à-dire le nombre d'alternatives proposées.

Outre ces points relatifs à la modélisation de l'utilisateur, il est aussi important de prendre en compte l'environnement dans lequel il se trouve, comme par exemple la géographie du lieu et ses contraintes, les distances physiques entre les points d'intérêts, les données météorologiques, les horaires d'ouvertures et de fermetures s'il y en a, etc.

Même si à première vue le domaine touristique semble différent du domaine musical, de nombreux points communs peuvent être trouvés, comme par exemple la gestion de la distance (distance physique entre deux points d'intérêts ou distance mathématiques entre deux ressources, les préférences implicites, les divers niveaux de tolérances, etc.). Dans la prochaine section nous allons discuter des points communs entre ces différents domaines et des caractéristiques qu'un modèle se voulant générique et multi-domaines se doit de posséder.

2.3.3 Discussion

Cette section a eu pour but de présenter un certain nombre de travaux liés de près ou de loin à la notion de séquence. Nous avons vu que, dans le domaine des recommandations musicales, la création ou la continuation de playlists sont des objectifs de plus en plus considérés par les chercheurs. Ce domaine s'y prête en effet très bien car la majorité des utilisateurs écoutent plusieurs musiques les unes à la suite des autres, permettant d'exploiter l'information contenue dans ces séquences déjà existantes et de tenter d'en créer de nouvelles. De la même manière, le rapide tour d'horizon du domaine des recommandations touristiques a permis de mettre en exergue, même s'il existe des différences notables avec le domaine musical dont il faut tenir compte, un objectif principal commun : considérer le maximum de données pertinentes disponibles (comportements précédents des utilisateurs, informations sur les ressources, caractéristiques personnelles de chaque utilisateur, etc.) dans le but de produire des séquences adaptées à chaque utilisateur. Les études de ces domaines ont permis de faire ressortir 5 directions majeures dans la conception d'un tel système de recommandation :

1. Les approches les plus performantes actuellement semblent généralement composées de

deux parties distinctes. La première partie consiste en une pré-sélection avec pour but de diminuer l'espace de recherche en sélectionnant une liste de candidats potentiels, tandis que dans la deuxième partie, une séquence est produite en utilisant les meilleurs candidats pré-sélectionnés dans l'étape 1.

2. Ces approches ne se limitent plus uniquement à l'utilisation de techniques provenant des systèmes de recommandation classiques mais mêlent désormais différentes techniques provenant de divers domaines de la recherche en informatique.
3. Les séquences déjà créées en amont par les utilisateurs peuvent être utilisées afin d'exploiter les informations qu'elles contiennent et produire en aval des séquences plus conformes à ce qu'un être humain ferait.
4. Le mouvement général des études produites ces dernières années va dans le sens d'une prise en compte conjointe et toujours plus importante de facteurs et caractéristiques humains.
5. Un système générique permettant de produire des séquences personnalisées quelque soit le domaine reste possible, mais ce dernier devra être suffisamment modulable pour s'adapter aux différences bien réelles entre tous les domaines concernés.

Dans la suite de cette section, nous allons nous concentrer sur un domaine de l'informatique dont les méthodes s'accordent avec l'ensemble des points évoqués ci-dessus : les systèmes multi-agents, et plus particulièrement les algorithmes de colonies de fourmis. C'est dans le cadre de ce paradigme que nous répondrons aux problématiques de cette thèse.

2.4 Les systèmes multi-agents

2.4.1 Définition

Les systèmes multi-agents désignent l'ensemble des systèmes où de nombreux agents coopèrent dans le but de résoudre un problème trop complexe pour un agent seul [Ferber and Weiss, 1999, Weiss, 2013]. Un agent, dans sa définition la plus large, représente toute entité pouvant percevoir son environnement à travers ses senseurs et pouvant agir dans cet environnement avec ses actuateurs [Russell and Norvig, 2016], comme le présente la Figure 2.3.

Selon cette définition, l'immense majorité des êtres vivants, robots, machines et programmes informatiques peuvent être considérés comme des agents. Nous allons affiner petit à petit cette définition afin de rentrer dans le cadre spécifique des systèmes multi-agents. Un premier élément pouvant être ajouté est la notion de rationalité. Un agent rationnel est un agent dont le comportement "est le bon", étant donné un environnement spécifique dans lequel il évolue. Avoir le bon comportement, autrement dit effectuer une bonne séquence d'actions, implique de définir un but précis, ainsi qu'une manière d'évaluer les séquences d'actions que l'agent effectue, comme par exemple la maximisation d'une fonction d'utilité. Afin de réaliser des actions dans son environnement, un agent a besoin de pouvoir prendre des décisions de lui-même. Il existe dans la littérature deux types d'agents, représentant deux conceptions différentes pour la résolution d'un problème [Ferber, 1995] :

— Le premier type d'agent est appelé **agent cognitif** et fait référence à des agents possédant une représentation symbolique du monde avec laquelle ils peuvent appliquer des raisonnements. Ces agents possèdent donc une représentation interne de leur environnement et sont capables d'interagir entre eux de manière sophistiquée. Ce type d'agent

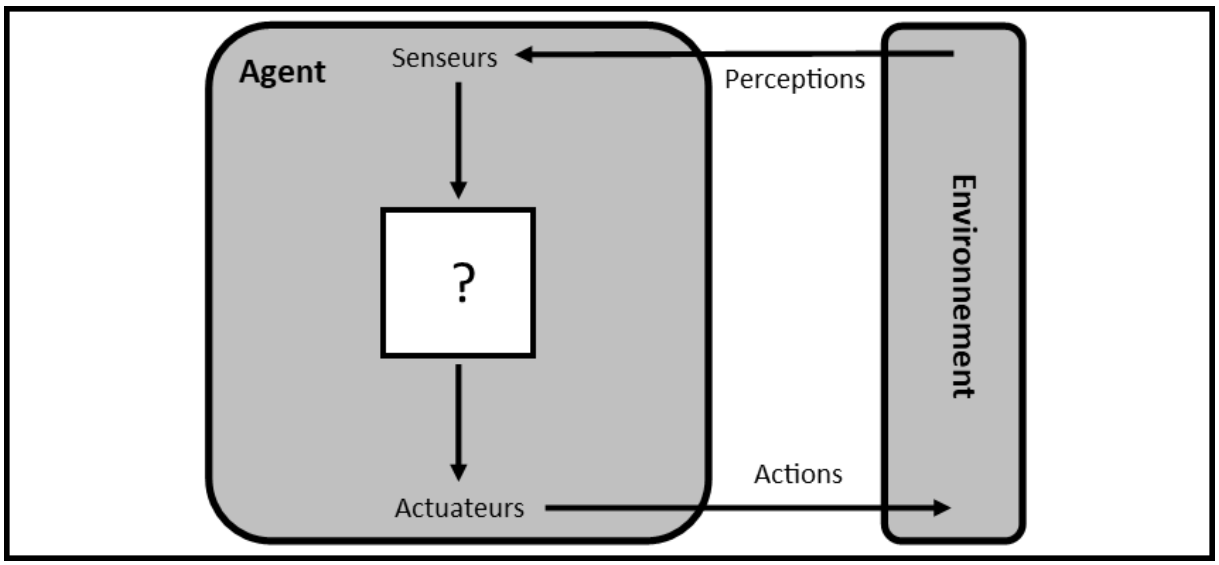

FIGURE 2.3 – Schéma représentant l'interaction entre un agent et l'environnement dans lequel il évolue [Russell and Norvig, 2016].

correspond plus généralement à une approche cognitive qui "*a pour but de faire communiquer et coopérer des systèmes experts classiques [où] chaque agent dispose d'une base de connaissance comprenant l'ensemble des informations et des savoir-faire nécessaires à la réalisation de sa tâche et à la gestion des interactions avec les autres agents et avec son environnement*" [Ferber and Weiss, 1999] ;

— Le deuxième type d'agent, nommé **agent réactif**, ne possède pas de représentation symbolique du monde et est donc limité par ses propres perceptions. Ces agents et leurs interactions sont généralement moins complexes que les agents cognitifs. Ce type d'agent correspond plus généralement à une approche réactive qui, quant à elle, "*prétend qu'il n'est pas nécessaire que les agents soient intelligents individuellement pour que le système ait un comportement global intelligent. Des mécanismes de réaction aux événements, ne prenant en compte ni une explicitation des buts, ni des mécanismes de planification, peuvent alors résoudre des problèmes qualifiés de complexes*" [Ferber and Weiss, 1999].

En plus de ces deux types d'agents différenciés par leur représentation du monde, Ferber propose un autre type de distinction sur les comportements : ces derniers peuvent être **téléonomiques**, c'est-à-dire dirigés vers des buts explicites, ou bien **réflexes**, c'est-à-dire régis par les perceptions. Ces 4 différents types d'agents peuvent se combiner comme illustré dans le Tableau 2.1.

Nous allons plus particulièrement nous intéresser ici aux agents réactifs et à leurs différences avec les agents cognitifs. Malgré leur relative simplicité, les interactions entre agents réactifs existent bien. Mais contrairement aux agents cognitifs, elles s'effectuent au travers de l'environnement, généralement par des traces laissées dans ce dernier comme des données visuelles, olfactives, auditives, etc. Là où les agents cognitifs raisonnent avant de prendre une décision, les agents réactifs ne font que réagir à des stimuli externes. Ils possèdent bien des senseurs afin de détecter leur environnement et des actuateurs pour agir, mais leur fonctionnement reste très

	Agents cognitifs	Agents réactifs
Téléonomiques	Agents intentionnels	Agents pulsionnels
Réflexes	Agents "modules"	Agents tropiques

TABLE 2.1 – Description des différents types d'agents selon leur relation au monde et selon leurs conduites [Ferber, 1995].

simple par rapport aux agents cognitifs. La relative simplicité de ces agents est intéressante car elle s'inscrit dans un changement de paradigme par rapport à la vision classique des l'intelligence artificielle dans le domaine de l'informatique. En effet, le domaine de l'intelligence artificielle existe depuis près de 60 ans maintenant et, à l'origine, les systèmes intelligents créés dans ce domaine pour résoudre des problèmes l'ont été selon une approche dite *top-down*, c'est-à-dire de haut en bas, avec les notions de "pensée" et de "raison" au centre de ces systèmes. Ainsi ces systèmes manipulaient les connaissances à l'aide de symboles et de règles et n'avaient aucune interaction ni avec leur environnement ni avec d'autres systèmes intelligents. C'est ce qu'on appelle communément la "*Good Old Fashioned Artificial Intelligence*". Depuis les années 1990, une autre approche est apparue pour étudier l'intelligence et créer des systèmes intelligents : l'approche *bottom-up*, soit de bas en haut, dont certains systèmes très populaires s'inspirent de systèmes biologiques incarnés dans un environnement et pouvant communiquer entre eux. C'est dans ce mouvement que s'inscrivent les systèmes multi-agents et, comme le dit Rodney A. Brooks dans son article justement intitulé "*Intelligence without Reason*" [Brooks, 1991] :

— "*Les comportements intelligents peuvent être générés sans l'aide de systèmes de raisonnement explicites.*"
— "*L'intelligence est une propriété émergente de certains systèmes complexes, apparaissant parfois sans qu'elle n'ait de raison spécifique d'apparaître.*"

Une première implication de la simplicité des agents réactifs est qu'il est nécessaire de faire interagir un certain nombre de ces agents pour pouvoir solutionner des problèmes. Là où un agent cognitif peut potentiellement être très complexe et capable de solutionner des problèmes principalement par lui-même (en déléguant ou coopérant parfois avec d'autres agents), un agent réactif seul n'en sera pas capable de par sa simplicité. Cependant, cette simplicité offre aussi des avantages dans le sens où il est aisé de comprendre, de programmer ou de modifier ces agents. Après cette partie théorique, nous allons voir dans la sous-section suivante deux modèles multi-agents afin d'illustrer la manière dont ils fonctionnent réellement. Ces deux systèmes nous permettrons par la suite de tirer quelques conclusions sur leur utilité dans le cadre de cette thèse.

2.4.2 Présentation de deux systèmes multi-agents réactifs

Nous avons pour le moment seulement décrit le principe des modèles multi-agents et les agents pouvant les composer. Cependant, ces systèmes ne puisent pas leur origine uniquement dans l'informatique. Beaucoup de systèmes multi-agents s'inspirent à l'origine de comportements d'animaux ou d'insectes observés en milieu naturel par des éthologues. En effet, certaines espèces animales semblent agir en groupe de manière très intelligente sans qu'il n'y ait de meneur ou sans qu'un individu pris séparément ne soit spécialement intelligent. On parle alors d'auto-organisation et d'émergence de l'intelligence quand une masse d'individus peu intelligents se coordonne ainsi dans la nature. Ce phénomène a été étudié par le biologiste français Pierre-Paul Grassé en 1959

et a été nommé "stigmergie" [Grassé, 1959]. La stigmergie est le mécanisme par lequel des agents vont se synchroniser et travailler ensemble dans un but commun, sans qu'aucun n'ait l'idée à l'avance de l'action à faire ou n'ait le plan final de la construction à réaliser (une fourmilière ou un nid de termites par exemple). Pour décrire le phénomène de stigmergie à l'œuvre chez certains animaux, Pierre-Paul Grassé explique ainsi : "L'ouvrier ne dirige pas son travail, mais il est guidé par lui". Cette phrase signifie qu'un agent n'a pas d'intention et d'idée de ce qu'il doit faire, mais qu'il sait cependant exactement quoi faire au moment où il est présent devant le travail. Cette capacité s'explique par le fait que les individus sont génétiquement préprogrammés pour effectuer certaines tâches [Oster, 1981]. C'est de cette manière que des espèces ne possédant qu'une très faible intelligence individuelle, comme les fourmis ou les termites, sont capables de produire des constructions extrêmement grandes et sophistiquées. On peut par exemple citer une espèce nommée la fourmi coupe-feuille du Texas, aussi appelée "fourmi champignonniste" dont les individus vivent dans de grandes colonies de plusieurs millions d'individus et cultivent des champignons dans des pièces souterraines dotées d'une structure particulière et d'une aération spécifique destinée à favoriser la pousse du champignon [Hölldobler et al., 1990]. La clé de la capacité de ces animaux à agir de la bonne manière face à tel ou tel stimulus réside dans l'ADN même de l'espèce, façonné par des millions d'années d'évolution. C'est ainsi qu'une ouvrière vaquera à ses tâches jusqu'à ce qu'un pan de mur de son nid s'effondre et que, devant ce nouveau stimulus, elle déploie une nouvelle panoplie de comportements appropriés à la situation à laquelle elle fait soudainement face. Ces exemples démontrent que l'intelligence est bien présente dans ces systèmes biologiques vivants et, dans la suite de cette section, nous allons donner deux exemples de modélisation de ces systèmes biologiques en systèmes multi-agents dans l'informatique.

Murmuration d'étourneaux : le modèle des boids

Les étourneaux représentent une famille d'une quinzaine d'espèces d'oiseaux et vivent sur les 5 continents. Ces oiseaux se regroupent parfois dans la nature sous forme de nuées pouvant contenir des dizaines de milliers d'individus. La raison de ces murmurations n'est pas encore connue avec certitude mais des théories existent, expliquant par exemple que ces regroupements permettraient une plus grande chance de survie face aux prédateurs [King and Sumpter, 2012], ou bien que le regroupement et la haute densité d'individus leur permettraient de se réchauffer durant les périodes les plus froides de l'année [Goodenough et al., 2017]. Ce phénomène est beaucoup étudié car tous les oiseaux semblent se déplacer dans la même direction et modifient leur trajectoire en même temps sans qu'aucun oiseau ne se touche en vol, ce qui est remarquable étant donné le nombre d'individus et la vitesse à laquelle ils volent. Malgré cette complexité, il n'y a pas d'oiseau meneur décidant de la direction à prendre et un individu seul n'a pas la capacité intellectuelle suffisante pour planifier ce genre de déplacements complexes dans un environnement en 3 dimensions. On retrouve bien là les caractéristiques d'un système multi-agents, où des agents simples provoquent, de par leurs interactions, un comportement intelligent et complexe dépassant de loin les capacités d'un individu seul. Le domaine des systèmes multi-agents s'est donc intéressé à créer un modèle permettant à l'aide de simulations informatiques de recréer les mumurations d'étourneaux. Ce modèle, où les agents sont appelés des *boids*[28], permet d'obtenir informatiquement des murmurations d'agents dans un environnement simulé grâce à trois règles implantées dans chaque agent, données ici par ordre de priorité [Reynolds, 1987] :

28. Le terme *boid* provient de la contraction de l'expression *bird-oid object* et signifiant "objet ressemblant à un oiseau."

1. **Évitement des collisions** : éviter les collisions avec les agents se trouvant à proximité. Si un agent est trop proche, s'éloigner dans une autre direction ;
2. **Coordination de la vitesse** : aller à la même vitesse que les agents se trouvant à proximité ;
3. **Centrage** : rester le plus proche possible des autres agents se trouvant à proximité, sans déclencher la règle d'évitement de collision.

Ces trois règles sont donc exécutées par les agents en permanence, de la plus prioritaire à la moins prioritaire, et permettent d'obtenir virtuellement un comportement de murmuration où tous les agents restent proches les uns des autres et changent de direction ensemble, comme montré dans la Figure 2.4. Ce modèle a par exemple été utilisé dans l'industrie du divertissement pour simuler des foules cohérentes dans des jeux vidéos ou des films [Silva et al., 2009].

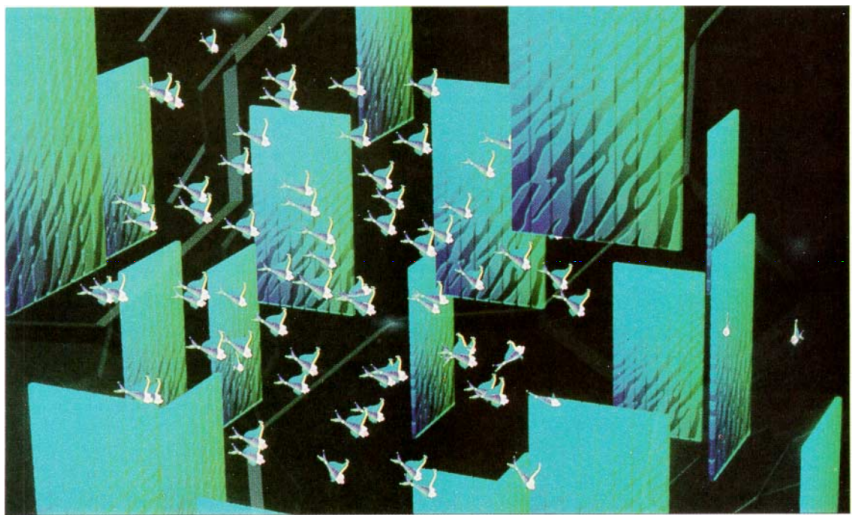

FIGURE 2.4 – Simulation de boids où l'on peut voir les agents se déplaçant dans la même direction à une certaine distance les uns des autres dans un environnement simulé [Reynolds, 1987].

Le fourragement des fourmis : algorithme de colonies de fourmis

Un autre exemple de système biologique complexe où l'on peut observer l'émergence d'une intelligence collective est le comportement des fourmis dans la nature. Dans la section précédente, nous avons déjà vu que par un mécanisme de stigmergie certains insectes sociaux, comme les fourmis ou les termites, étaient capables de construire des structures très complexes. Un autre phénomène d'auto-organisation a été observé lors du fourragement [29] des fourmis de l'espèce de la fourmi d'Argentine. En effet, les ouvrières de cette espèce se déplacent de manière aléatoire aux alentours de la fourmilière jusqu'à trouver une source de nourriture. Une fois que des fourmis ont trouvé de la nourriture, elles retournent à la fourmilière. Après un certain temps, de plus en plus d'ouvrières se dirigent vers cette source de nourriture jusqu'à ce qu'un chemin de fourmis allant de la fourmilière jusqu'à la nourriture se forme. Ce chemin a cependant la particularité d'être optimisé en termes de distance à parcourir et d'obstacles à éviter, de sorte que les ouvrières empruntent effectivement le plus court chemin possible entre leur nid et la nourriture [Deneubourg

29. Le fourragement désigne le comportement des fourmis recherchant de la nourriture et revenant avec celle-ci à la fourmilière.

et al., 1990]. La Figure 2.5 donne un exemple de ce à quoi peut ressembler un de ces chemins une fois emprunté par beaucoup d'individus. Ce comportement est remarquable quand on le replace dans le contexte des individus qui en sont à l'origine. Les ouvrières ne font pas plus de 1 ou 2 centimètre(s) de long et peuvent couvrir des distances jusqu'à 100 mètres afin de trouver de la nourriture. L'environnement dans lequel elles se déplacent est complexe, jonché d'obstacles et évolue en permanence alors qu'elles n'ont qu'un sens de l'orientation limité et une faible capacité de traitement dûe à leur très petit nombre de neurones (environ 250.000).

FIGURE 2.5 – Illustration d'un chemin créé par des fourmis appartenant à l'espèce des fourmis légionnaires (crédit photo : Mehmet Karatay).

Afin de comprendre les mécanismes permettant l'émergence de ce comportement intelligent, des expérimentations ont été menées en laboratoire avec de petites colonies de fourmis dans un environnement contrôlé [Goss et al., 1989]. Voici donc le principe de fonctionnement de la création de chemins optimisés chez les fourmis :

1. Des fourmis se déplacent aléatoirement autour de la colonie à la recherche de nourriture ;
2. Quand une fourmi trouve de la nourriture, elle retourne du mieux possible à la fourmilière en déposant des phéromones ;
3. D'autres fourmis, attirées par ces phéromones, vont avoir tendance à suivre ce chemin. Les nouvelles fourmis atteignant la nourriture vont, elles aussi, revenir à la fourmilière en déposant des phéromones ;
4. Les fourmis réussissant à revenir plus rapidement avec des chemins plus courts à la fourmilière vont redéposer plus rapidement de nouvelles phéromones, étant donné qu'elles mettront moins de temps pour faire des allers-retours entre la nourriture et la fourmilière ;
5. À l'inverse, les fourmis ayant emprunté des chemins plus longs vont déposer de nouvelles phéromones moins rapidement, puisqu'elles mettront plus de temps à faire leurs allers-retours ;
6. Petit à petit, le chemin le plus court va être emprunté par de plus en plus de fourmis déposant chacune des phéromones, renforçant de plus en plus ce même chemin ;

7. Les chemins les plus longs sont peu à peu abandonnés, d'autant plus que les phéromones s'évaporent avec le temps, renforçant encore plus les chemins fortement empruntés ;
8. Au bout d'un certain temps, seul le chemin le plus court est emprunté par la majorité des fourmis ;
9. Quelques fourmis ne prennent cependant pas ce chemin, permettant la recherche de nouvelles sources de nourriture ou de nouveaux chemins. En effet, si jamais le chemin emprunté par la majorité des individus se voit perturbé par un nouvel obstacle (un arbre venant de tomber par exemple), alors ces quelques fourmis pourront éventuellement trouver le nouveau meilleur chemin le plus court et le cycle de renforcement recommencera.

Comme pour les murmurations d'étourneaux, on retrouve bien les caractéristiques de l'auto-organisation dans un système multi-agents : aucun meneur n'est présent, les agents suivent seulement quelques règles simples permettant l'émergence d'un comportement dépassant les capacités cognitives de l'espèce. Ces règles ont été décrites de manière formelle par Marco Dorigo durant sa thèse dans un modèle appelé *Ant System* (AS) permettant de recréer le comportement des fourmis au sein d'une simulation informatique [Dorigo et al., 1996]. Dans ce modèle, l'environnement est représenté par un graphe dont les sommets sont les points d'intérêts où l'agent peut se rendre et dont les arêtes sont les chemins disponibles permettant de se déplacer de point en point. À chaque arête (i, j) est associée deux valeurs numériques : la première représentant la distance à parcourir pour se déplacer du sommet i au sommet j ; la deuxième représentant la quantité de phéromones présente sur cette arête. Un agent se trouvant à un sommet i a connaissance de l'ensemble des sommets V_i accessibles directement à partir de i. Pour décider quelle arête emprunter, l'agent prend en compte la distance et la quantité de phéromones associées à chaque arête. Une fois que l'agent a atteint un sommet spécifique désigné à l'avance comme objectif, il dépose des phéromones sur chacune des arêtes empruntées. La quantité de phéromones déposée est dépendante de la longueur du chemin emprunté de sorte qu'un agent ayant trouvé un chemin court déposera plus de phéromones qu'un agent ayant trouvé un chemin plus long. Ce cycle peut ensuite recommencer avec une multitude d'agents jusqu'à ce qu'une solution émerge, de la même manière que pour les fourmis dans la nature. Marco Dorigo a, dans la suite de ses travaux, développé d'autres modèles ayant tous comme base le modèle AS mais possédant quelques variations ou ajouts, offrant une plus grande variété d'outils pour résoudre les nombreux problèmes de la recherche en informatique. Dans les faits, le modèle original AS a aujourd'hui pratiquement disparu au profit de ses descendants et c'est pour cette raison que nous n'allons pas ici détailler plus avant le modèle AS [Dorigo and Birattari, 2010]. Tous ces différents modèles sont maintenant regroupés sous le nom de *Ant Colony Optimization* (ACO) [Dorigo and Birattari, 2011]. Ces derniers ont été exploités dans de nombreux domaines de la recherche scientifique pour résoudre des problèmes variés. On peut par exemple citer trois de ces problèmes, très connus et étudiés en informatique de par leur complexité, qui ont chacun vu des solutions leur être trouvées grâce à ACO :

— Le problème du voyageur de commerce [Dorigo and Gambardella, 1997] ;
— Le problème des tournées de véhicules [Bell and McMullen, 2004] ;
— Le problème de gestion de projet à contraintes de ressources [Merkle et al., 2002].

Dans la prochaine section, nous allons nous intéresser plus particulièrement à l'un des modèles de la famille ACO nommé *Ant Colony System* (ACS). Nous allons voir son fonctionnement détaillé et expliquer les spécificités qui le rendent particulièrement intéressant dans le cadre de cette thèse.

2.4.3 Ant Colony System

Le modèle ACS possède plusieurs modifications majeures par rapport au modèle initial AS, notamment par rapport aux mécanismes liés aux phéromones et à la prise de décision des agents [Dorigo and Birattari, 2010]. Premièrement, seul l'agent ayant trouvé le meilleur chemin est autorisé à déposer des phéromones sur les arêtes qu'il a emprunté, alors que dans AS tous les agents déposaient des phéromones. Deuxièmement, et pour contrer la relatif manque de phéromones engendré par cette première modification, les agents possèdent un nouveau comportement appelé "mise à jour locale de phéromones" (*locale pheromone update*), consistant en la mise à jour des phéromones par chaque agent sur la dernière arête empruntée durant la construction d'un chemin. Cette mise à jour s'articule autour d'un seuil de phéromones spécifié comme métavariable avant l'exécution du programme. Si une arête traversée par un agent possède une quantité de phéromones inférieure à ce seuil, alors l'agent dépose des phéromones. À l'inverse, si la quantité de phéromone sur cette arête est supérieure au seuil, alors l'agent retire une certaine quantité de phéromones. En d'autres termes, dans ACS dès qu'un agent emprunte une arête, il met à jour la quantité de phéromones sur cette arête, et ce, quelle que soit la qualité du chemin qu'il est en train de construire. Marco Dorigo explique lui-même les avantages que possède ACS par rapport à d'autres versions existantes [Dorigo and Birattari, 2011] :

> *"ACS permet de diversifier l'exploration effectuée par les agents pendant une itération de l'algorithme : en diminuant la quantité des phéromones sur les arêtes les plus utilisées, les agents suivants sont encourager à se déplacer sur d'autres arêtes moins explorées, permettant de diversifier les solutions trouvées à la fin d'une itération. Cette mécanique rend moins probable le fait que plusieurs agents puissent produire des solutions identiques durant une itération."*

Une autre modification du modèle ACS par rapport au modèle AS concerne le déplacement des agents. Il est maintenant possible, grâce à une nouvelle métavariable, de donner l'importance souhaitée entre les phéromones et la distance lors du processus de choix de la prochaine arête à emprunter par un agent. Dans le reste de cette sous-section, nous allons expliquer les principales formules régissant le comportement des agents et la mise à jour des phéromones dans ACS.

Règle de transition d'état - la règle de transition (*state transition rule*) d'état utilise la règle proportionnelle pseudo-aléatoire (*pseudo-random proportional rule*) où une variable aléatoire $q \in [0; 1]$ est comparée à une métavariable q_0 afin de décider si l'agent courant va explorer le graphe ou s'il va exploiter la connaissance produite par les autres agents avant lui. En d'autres termes, il est possible d'équilibrer avec cette variable l'importance des phéromones et de la distance lors du choix des arêtes. $q_0 = 0$ entraîne l'exploration du graphe en tenant uniquement compte de la distance associée aux arêtes tandis que $q_0 = 1$ correspond à un comportement de renforcement pur via les phéromones sans aucune exploration. L'Équation 2.7 permet de choisir entre l'exploitation de la connaissance ou l'exploration du graphe, donnant plus de latitude au modèle ACS par rapport à AS.

$$\begin{cases} \text{Exploitation (Equation 2.8) si } q \leq q_0 \\ \text{Exploration biaisée (Equation 2.9) si } q \geq q_0 \end{cases} \tag{2.7}$$

L'Équation 2.8 représente l'exploitation directe du graphe où la meilleure arête est toujours choisie

$$\arg\max_{l \in V_i} \{\tau_{il}^{\alpha} \cdot \eta_{il}^{\beta}\}, \tag{2.8}$$

où V_i est l'ensemble des sommets accessibles depuis le sommet i, $\tau_{il} \in [0;1]$ est la quantité de phéromones déposée sur l'arête (i,l) par les agents précédents, η_{il} est l'information heuristique de l'arête (i,l), α et β sont deux métavariables représentant respectivement le poids des phéromones et celui de l'heuristique dans le calcul. L'heuristique η_{il} représente classiquement l'information relative à la distance séparant les sommets i et l et sera décrite après.

Il est à noter qu'on ne parle pas ici de pure exploration mais d'exploration biaisée, dans le sens où les "meilleures arêtes" ont une probabilité plus élevée d'être sélectionnées. L'expression "meilleure arête" est ici définie comme le produit de l'heuristique avec la quantité de phéromones associées à cette arête. L'Équation 2.9 correspond à l'exploration biaisée du graphe.

$$p_{ij} = \begin{cases} \dfrac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum\limits_{l \in V_i} \tau_{il}^{\alpha} \cdot \eta_{il}^{\beta}}, & \text{si } j \in V_i, \\ 0, & \text{sinon} \end{cases} \tag{2.9}$$

Mise à jour globale des phéromones - Après chaque itération, c'est-à-dire après que tous les agents aient terminé leur chemin, seul l'agent qui a trouvé le meilleur chemin est autorisé à mettre à jour le niveau de phéromones τ_{ij} sur chaque arête (i,j) qu'il a emprunté :

$$\tau_{ij} = \begin{cases} (1-\rho) \cdot \tau_{ij} + \rho \cdot \Delta\tau_{ij} & \text{si } (i,j) \text{ appartient au meilleur chemin,} \\ \tau_{ij} & \text{sinon} \end{cases} \tag{2.10}$$

où ρ est le taux d'évaporation des phéromones, et $\Delta\tau_{ij} = 1/L_{best}$ où L_{best} est la longueur du meilleur chemin.

Mise à jour locale des phéromones - Un autre ajout du modèle ACS par rapport au modèle AS est la mise jour locale des phéromones. Étant donné que seul l'agent ayant trouvé le meilleur chemin dépose des phéromones dans ACS (contrairement à tous les agents proportionnellement à la qualité de leur solution dans AS), très peu de phéromones sont déposés avec la règle de mise à jour globale des phéromones. Pour contrer ce problème, tous les agents mettent à jour la quantité de phéromones à chaque arête empruntée lors de la construction du chemin, comme décrit dans l'Équation 2.11.

$$\tau_{ij} = (1-\rho) \cdot \tau_{ij} + \rho \cdot \tau_0, \tag{2.11}$$

où τ_0 est le niveau de phéromones placé sur chaque arête du graphe à l'initialisation du modèle.

Information heuristique - l'information heuristique η_{ij} représente l'information que la fourmi possède *a priori* sur une arête (i,j). Dans l'algorithme ACS, l'information heuristique est calculée en se basant sur la distance entre les deux sommets i et j de l'arête (i,j) : plus les deux sommets sont éloignés l'un de l'autre et plus la valeur de η_{ij} sera basse, indiquant à un agent situé sur le sommet i qu'il n'est *a priori* pas intéressant pour lui d'emprunter l'arête (i,j). L'Équation 2.12 décrit comment obtenir la valeur de l'information heuristique η_{ij} de l'arête (i,j).

$$\eta_{ij} = \frac{1}{d_{ij}}, \tag{2.12}$$

où d_{ij} est la distance entre les sommets i et j.

2.4.4 Discussion sur le modèle ACS

Dans cette sous-section, nous allons revenir sur un des principes de l'algorithme ACS : la règle proportionnelle pseudo-aléatoire et son importance danc le fonctionnement du modèle. Cette règle, comme détaillée ci-dessus, permet de gérer l'équilibre dans l'exploration et l'exploitation de manière semi-aléatoire (*i.e.* avec une variable aléatoire servant à fixer le seuil de basculement d'un comportement vers l'autre). Cette dualité exploration/exploitation est bien connue des systèmes de recherche et de création de connaissances [March, 1991] ; en voici leur définition.

L'exploration est définie comme étant l'expérimentation avec de nouvelles alternatives sans tenir compte de la connaissance du domaine. Les résultats obtenus via cette tactique sont incertains mais peuvent aussi aboutir à de nouveaux, et potentiellement meilleurs, résultats que ceux déjà découverts.

L'exploitation, quant à elle, est définie comme le perfectionnement et l'extension de résultats déjà connus. Les résultats obtenus via cette tactique sont plutôt certains et prévisibles. Cependant, un processus trop axé sur l'exploitation aura des difficultés à trouver de très bonnes solutions à un problème et aura tendance à développer la première solution trouvée, même si d'autres solutions de meilleures qualités pourraient être découvertes en explorant plus avant l'espace de recherche.

Il est donc important de trouver un équilibre entre l'exploration de l'espace de recherche pour trouver de nouvelles solutions et l'exploitation permettant d'affiner ces solutions. D'autres méthodes d'exploration et de découverte de solutions dans un espace de recherche proposent des techniques spécifiques pour solutionner ce problème. On peut par exemple citer la méthode de la descente du gradient, consistant à minimiser une fonction réelle dans un espace de recherche. Cette méthode procède par itérations successives et se déplace d'abord de manière importante dans l'espace de recherche afin d'explorer un maximum de solutions. Puis, une fois qu'une bonne solution est trouvée, la méthode va diminuer son déplacement dans l'espace de recherche afin d'affiner la solution trouvée. La descente de gradient utilise donc une approche où l'exploration est d'abord privilégiée puis, petit à petit, se dirige vers de l'exploitation, comme illustré dans la Figure 2.6.

On pourrait penser à première vue que l'exploitation, autrement dit le fait que l'agent prenne la meilleure arête disponible à chaque choix s'offrant à lui, permettrait d'obtenir les meilleurs résultats. En effet, le but est ici de trouver le meilleur chemin, soit le plus court chemin dans le cadre d'une distance euclidienne, d'un sommet a à un sommet b. Le fait de permettre à l'agent de ne pas forcément prendre la meilleure arête lors de la construction du chemin est donc relativement contre-intuitif. On peut cependant lever cette intuition avec une série d'expérimentations provenant de la littérature et des observations menées sur les fourmis dans la nature. Jean-Louis Deneubourg, dont nous avons déjà parlé au début de cette section, a beaucoup étudié le comportement des fourmis en milieu réel et la modélisation de leur processus décisionnel collectif. Il a ainsi proposé en 1990 une expérience appelée le pont binaire [Deneubourg et al., 1990] pouvant être résumée ainsi : le nid d'une colonie de fourmis est séparée d'une source de nourriture par deux chemins de longueurs identiques. Pour ramener la nourriture au nid, les fourmis peuvent donc emprunter l'une de ces deux voies sans conséquence sur la longueur du chemin total. Au départ, les fourmis empruntent les deux voies sans distinctions puis, une fois un certain temps passé,

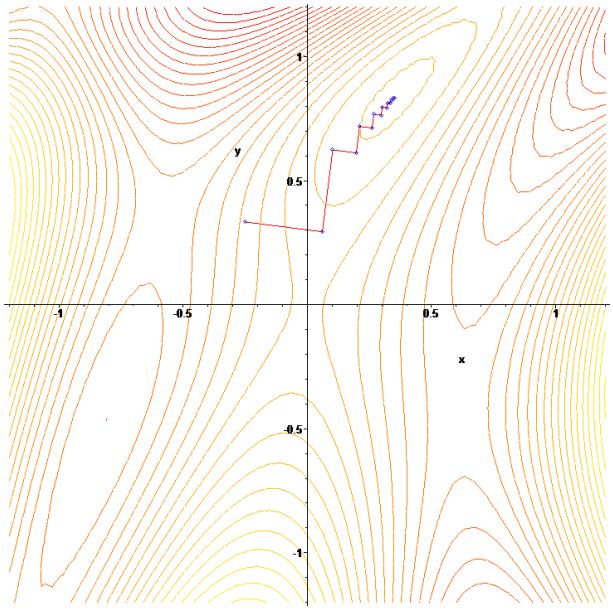

FIGURE 2.6 – Exemple de descente de gradient où l'on peut voir l'exploration au début avec un déplacement rapide puis l'exploitation ensuite vers la solution trouvée. (crédit image : Joris Gillis

une des voies est privilégiée par rapport à l'autre et la majorité des fourmis l'empruntent. Ce comportement est la conséquence des phéromones déposées par les fourmis : au bout de quelques temps, une des deux voies est empruntée par plus de fourmis que l'autre et, comme les fourmis déposent des phéromones et sont attirées par celles-ci, cette différence s'accentue jusqu'à ce qu'une voie soit majoritairement empruntée. Il est à noter que ce comportement de renforcement n'est pas forcément vrai pour toutes les espèce de fourmis dans la nature. Cependant, c'est sur ce principe que Deneubourg a modélisé les équations déterminant le choix effectuée par chaque fourmi. Ces équations sont à la base même des modèles ACO développés par Dorigo quelques années plus tard. Le dispositif de l'expérience des ponts binaires est présenté par la Figure 2.7.

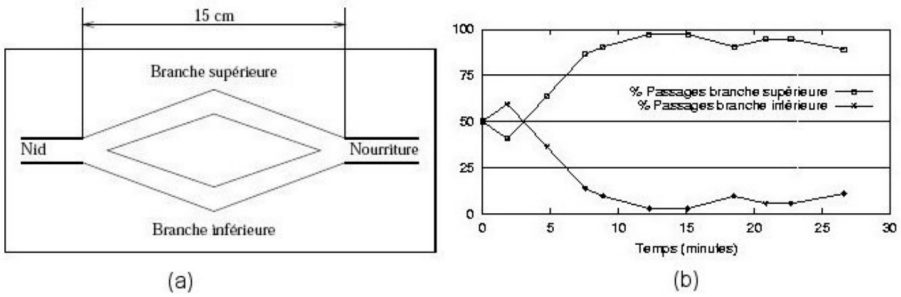

FIGURE 2.7 – Illustration de l'expérience des ponts binaires par Jean-Louis Deneubourg (a). Après un certain temps, les fourmis choisissent majoritairement une des deux voies possibles (b) [Deneubourg et al., 1990].

Une autre expérimentation du même genre a été réalisée par la même équipe de chercheurs où les fourmis doivent se déplacer d'un point A à un point B en choisissant parmi différentes voies possibles [Goss et al., 1989]. Le dispositif expérimental contient cependant une différence notable

avec l'expérience initiale dans le sens où les différentes voies ne font pas la même longueur. Comme le montre la Figure 2.8, pour aller du nid jusqu'à la nourriture, les fourmis disposent d'un chemin où deux embranchements similaires sont proposés. Chacun de ces embranchements possède une voie courte et une voie longue. La fourmi a donc 4 choix à effectuer lors d'un aller-retour pour partir de son nid et ramener de la nourriture. Au début de l'expérimentation, les fourmis ont besoin de 5 à 10 minutes pour explorer leur environnement et trouver la nourriture. Tous les chemins sont d'abord choisis de manière aléatoire puis, quelques minutes plus tard et de manière abrupte, le mécanisme de renforcement rentre en jeu et les voies les plus courtes sont privilégiées. Une trentaine de minutes après le début de l'expérimentation, les fourmis empruntent majoritairement le chemin le plus court.

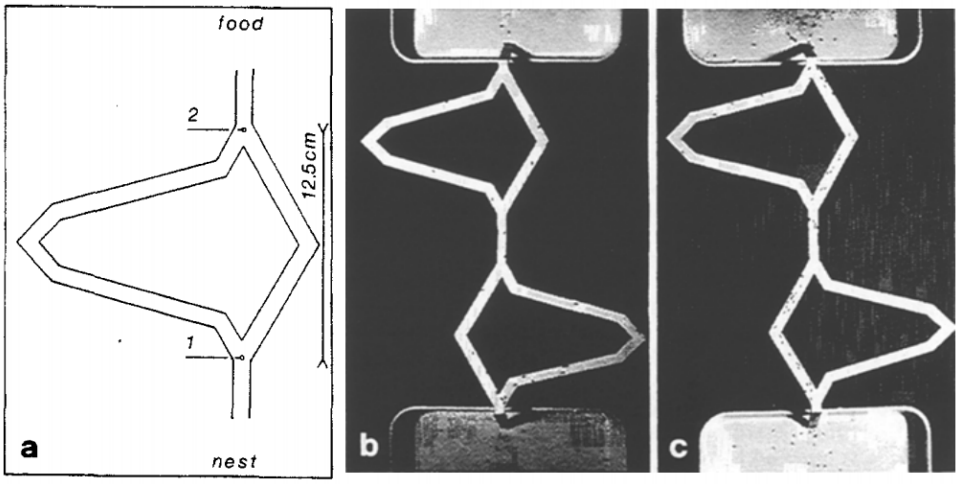

FIGURE 2.8 – Schéma du dispositif expérimental utilisé pour vérifier si les fourmis sont bien capables d'emprunter rapidement la voie la plus courte lorsque leur sont présentés des choix qualitativement différents [Goss et al., 1989].

Alors que la première expérience a montré que les fourmis possèdent bien un mécanisme d'exploration aléatoire menant au choix puis au renforcement d'une des deux voies possibles, la deuxième expérience montre que les fourmis sont bien capables de choisir le chemin le plus court lorsqu'elles sont confrontées à deux voies possibles ayant des distances différentes. Une critique peut être formulée dans le sens où ces deux expérimentations se sont déroulées en laboratoire avec un dispositif expérimental relativement petit (quelques dizaines de centimètres pour aller du nid à la nourriture). De par la taille réduite des dispositifs, les expérimentations ne duraient généralement pas plus d'une trentaine de minutes. Durant ce laps de temps relativement court, le mécanisme d'évaporation des phéromones ne pouvait pas entrer en jeu car les phéromones n'avaient pas le temps de s'évaporer.

Ces deux expériences sont intéressantes et illustrent chacune des capacités fondamentales des fourmis dans la nature quant à leur déplacement et à leur survie. Jean-Louis Deneubourg [Deneubourg et al., 1990] et Marco Dorigo [Dorigo, 2001] ont montré comment il était possible de reproduire ces comportements via des simulations informatiques. Dans la tradition des systèmes multi-agents, les principes permettant de modéliser ces comportements sont relativement simples et, comme montré dans la section précédente, seules quelques équations suffisent à obtenir une modélisation de la réalité et l'émergence de comportements intelligents. Dans la suite de cette section, nous allons voir quelques exemples d'utilisation d'algorithmes provenant des modèles

ACO dans les systèmes de recommandation.

2.5 Utilisation et évolution récentes des algorithmes de colonies de fourmis

Nous avons vu dans les sections précédentes que les algorithmes de colonies de fourmis ont été utilisées dans des problèmes connus comme le voyageur de commerce, les tournées de véhicules ou la gestion de projet à contraintes. Chacun de ces exemples a la particularité d'être un problème d'optimisation où le but est de minimiser une distance à parcourir ou un temps d'exécution avec certaines contraintes, ce qui correspond bien au but premier des ACO. La versatilité des systèmes multi-agents leur a permis de s'adapter dans des domaines comme les systèmes de recommandation, et ils ont particulièrement été exploité pour l'e-éducation, Dans la suite de cette section, nous allons étudier quelques applications des algorithmes de la famille des ACO dans ce domaine particulier.

2.5.1 Les algorithmes de colonies de fourmis dans les systèmes de recommandation

L'une des premières applications des ACO dans le domaine des systèmes de recommandation a été proposée par [Semet et al., 2003] dans l'e-education. Le but de l'étude était de recommander des cours et des exercices à des lycéens avec l'objectif pédagogique d'améliorer leur réussite scolaire. Cette étude est intéressante car, en plus de proposer une modification des modèles ACO pour la recommandation, elle a été réalisée en partenariat avec Paraschool [30], une entreprise dédiée à l'apprentissage en ligne pour les lycéens français, permettant aux auteurs de tester leur modèle. Dans ce dernier, le domaine pédagogique est représenté par un graphe valué où les sommets sont des items pédagogiques et les arcs sont les liens hypertextes où des probabilités de passage sont associées. Les étudiants sont représentés par des agents se déplaçant dans le graphe en laissant des phéromones sur les arcs qu'ils suivent. Chaque page de cours ou d'exercice (représenté par un sommet unique dans le graphe) est doté d'un bouton $NEXT$ amenant l'étudiant vers un nouveau cours selon une arête choisie par l'algorithme selon les facteurs suivants :

— **Poids pédagogique** W : c'est la valeur principale de chaque arête. Il est implémenté comme une variable globale accessible à tous les agents. W est manuellement défini en amont par les professeurs et reflète l'importance des arêtes venant d'un sommet particulier (*i.e.* "cet exercice est important après la leçon x, donc $W = 0,8$; la leçon y n'est pas très importante après la leçon x, donc $W = 0,3$") ;
— **Phéromones** S **and** F : S représente la phéromone de succès tandis que F représente la phéromone d'échec. S est incrémentée quand les étudiants ont réussi à compléter l'exercice correspondant, et inversement pour F. Ces phéromones sont associées à l'arête qui a mené l'agent au sommet courant mais aussi aux précédents sommets empruntés avec une quantité de plus en plus réduite, limitée à 4 sommets en arrière. Le but est de représenter le fait que le résultat d'un exercice dépend de l'ensemble du parcours emprunté et non du dernier sommet uniquement [31]. Les phéromones s'évaporent aussi au fur et à mesure

30. http ://www.paraschool.com/
31. Cette méthode est similaire dans son fonctionnement à la technique de rétropropagation du gradient dans l'apprentissage des réseaux de neurones.

chaque jour ;

— **Historique personnel** H : c'est une variable propre à chaque sommet visité pour chaque agent. Quand un étudiant valide le module pédagogique associé au sommet, alors la valeur h associée à ce sommet particulier est égale à $0,5$; quand un étudiant échoue le module pédagogique, alors $h = 0,75$. Cette valeur est ensuite utilisée comme facteur multiplicatif pour réduire la probabilité de visiter le sommet à nouveau. Si le sommet est revisité, H est à nouveau multiplié par ces mêmes valeurs. H s'évapore avec le temps pour tendre vers 1, symbolisant le fait que plus le temps passe et plus un étudiant aura tendance à oublier le module pédagogique visité.

Pour chaque arc a, une *fitness value* est calculée selon l'Équation : $f(a) = H(\omega_1 W + \omega_2 S - \omega_3 F)$ où ω_1, ω_2 et ω_3 sont les poids permettant de moduler l'importance des 3 facteurs pris en compte. Plus la valeur de $f(a)$ est haute et plus l'arc sera considéré attractif. Un arc sera donc attractif quand :

— son nœud d'arrivée n'a jamais été visité ou n'a pas été visité depuis longtemps ($H \to 1$) ;
— il est plébiscité par les professeurs (W important) ;
— il y a une atmosphère de succès autour de cet arc (S important) ;
— peu d'échecs ont eu lieu près de cet arc (F faible).

Afin de choisir le prochain arc pour l'étudiant, le système sélectionne aléatoirement un certain nombre d'arcs possibles et garde celui possédant la plus haute valeur $f(a)$. La sélection d'un sous-ensemble d'arcs permet de varier un minimum les recommandations et d'éviter de proposer tout le temps les arcs les plus attractifs.

Les auteurs proposent donc ici une adaptation complète des ACO pour un système de recommandation dans l'e-éducation tout en intégrant des principes pédagogiques importants. L'utilisation de plusieurs types de phéromones et de différentes valuations sur les arêtes du graphe permettent d'entrevoir le potentiel et l'adaptabilité des algorithmes ACO. Quelques critiques peuvent cependant être proposées.

Une première critique possible est qu'il n'existe pas de modèle utilisateur prenant en compte les caractéristiques personnelles de chaque étudiant (notes, niveau, style d'apprentissage...). Posséder et utiliser un modèle par étudiant pourrait permettre à l'algorithme de mieux s'adapter aux cas particuliers, nombreux dans le domaine des systèmes de recommandation [Gras et al., 2016]. Dans cet article, les auteurs montrent que les techniques classiques de filtrage collaboratif n'arrivent pas à bien détecter les utilisateurs atypiques [32] alors qu'il est possible de détecter et de satisfaire ces derniers avec des techniques spécifiques.

Une autre critique possible concerne le fait que les professeurs doivent au préalable évaluer les chemins possibles à partir de chaque sommet sous la forme du poids pédagogique W. Ce fait révèle que l'expérimentation n'a pas utilisé un grand nombre de ressources pédagogiques (ce qui est le cas car le graphe utilisé par les auteurs ne possédait que 20 nœuds et 47 arêtes, soit 20 ressources pédagogiques). Dans la pratique, cette méthode n'est pas adaptée pour trois raisons majeures. Premièrement, il est difficilement concevable d'évaluer manuellement la probabilité de

32. Les utilisateurs atypiques (*Grey Sheep Users*) sont des utilisateurs possédant des préférences différentes de la majorité des autres utilisateurs du système.

passage de chaque arête dans des domaines dépassant les quelques dizaines de ressources. Deuxièmement, l'utilisation d'experts du domaine, comme les professeurs dans l'e-éducation, n'est pas forcément possible selon le domaine considéré. Qui peut par exemple juger de la qualité des transitions à partir d'une musique spécifique ? Troisièmement, un tel modèle n'est que difficilement transférable en l'état d'un domaine applicatif à un autre pour les raisons évoquées ci-dessus.

Intéressons-nous maintenant à une autre étude dans l'e-éducation qui propose une solution pour permettre une plus grande personnalisation des recommandations. Dans leur article, [Kurilovas et al., 2015] utilisent eux aussi les ACO avec comme but de fournir des séquences composées de modules d'apprentissage aux étudiants en fonction de leur style d'apprentissage. Des recherches ont montré qu'il existait de grands styles d'apprentissage et que les apprenants possédaient des préférences pour certains styles plus que d'autres. Les auteurs utilisent ici la catégorisation en 4 grands styles proposée par [Honey et al., 1992] [33]. Ainsi, les agents ne sont désormais plus tous identiques mais possèdent un ou plusieurs style(s) d'apprentissage préféré(s) et la mise à jour des phéromones est elle aussi modifiée afin de prendre en compte ces préférences. Ainsi, un agent préférera un chemin composé de modules d'apprentissage correspondant à son ou ses propre(s) style(s) d'apprentissage et déposera plus de phéromones le cas échéant. Ces changements reflètent une volonté de personnalisation des recommandations s'inscrivant dans le récent changement de paradigme du domaine des systèmes de recommandation où les préférences des utilisateurs sont de plus en plus prises en compte. Il est néanmoins à noter pour cette étude qu'il est nécessaire de faire passer un long questionnaire pour définir les styles d'apprentissage des apprenants avant de pouvoir utiliser le système. De plus, ces styles d'apprentissage sont fixes et ne peuvent plus être modifiés après le démarrage du système, empêchant potentiellement un apprenant d'évoluer dans sa manière d'apprendre.

L'idée de modifier le modèle ACO pour prendre en compte plusieurs types d'utilisateurs a aussi été exploitée dans une autre étude en e-éducation par [Kardan et al., 2014]. Cette étude a le même but que la précédente, à savoir recommander des séquences personnalisées éducatives avec ACO, mais les auteurs proposent une autre solution tout aussi intéressante. Ils se basent ainsi sur la théorie de David Ausubel, selon laquelle l'apprentissage est facilité lorsque l'apprenant peut mettre en relation les nouveaux concepts qu'il est en train de découvrir avec ce qu'il connaît déjà [Ausubel, 1963]. Les auteurs proposent un algorithme en deux parties :

1. Les apprenants sont d'abord évalués avec des items de tests correspondant chacun à un ou plusieurs concepts devant être appris. Afin de déterminer le degré de familiarité à un concept par un apprenant, le score qu'il a obtenu à chaque item de test est utilisé en combinaison avec le degré de lien entre les concepts et chaque item pour créer une matrice $learners \times concepts$. L'algorithme des k-means est ensuite utilisé afin de diviser les apprenants en groupes ayant les mêmes familiarités avec les concepts.
2. l'algorithme ACO est appliqué avec autant de groupes d'agents spécifiques qu'il y a de groupes d'apprenants découverts précédemment avec les k-means. Ainsi, un chemin optimal est trouvé pour chaque groupe, permettant de naviguer parmi tous les concepts dans un ordre précis et spécifique au degrés de familiarité de chaque groupe. Autrement dit, le système recommande pour chaque groupe d'apprenants une séquence éducative personnalisée en fonction de ses connaissances initiales.

33. *Activist, Theorist, Pragmatist, Reflector.*

On retrouve donc ici l'idée de créer différents types d'agents pour résoudre différents problèmes. Cette constatation n'est pas étonnante dans le sens où, lorsqu'il s'agit d'optimisation, un seul groupe d'agents peut suffire à optimiser une variable unique, comme par exemple la distance. En revanche, dans le domaine des systèmes de recommandation, il n'est pas envisageable de proposer la même recommandation à tous les utilisateurs. Au contraire, l'approche à privilégier est de pouvoir s'adapter au cas par cas à chaque utilisateur et de prendre en compte ses préférences pour lui proposer une solution adaptée. On peut d'ailleurs émettre deux critiques de cette étude à la lumière de cette analyse. Premièrement, même si le modèle propose un certain degré de personnalisation, sa granularité n'est encore pas assez fine puisqu'il traite des groupes d'individus et non un individu unique. Ensuite, il n'y a encore une fois pas de moyen pour un apprenant de changer dynamiquement de groupe étant donné que ces derniers doivent être créés en amont du processus de recommandation.

Toujours dans la famille des algorithmes ACO, nous pouvons également citer le modèle *Attribute-based Ant Colony System* qui est similaire aux derniers modèles décrits ci-dessus tout en proposant un degrés de personnalisation plus fin [Yang and Wu, 2009]. Les auteurs utilisent le modèle d'apprentissage de Kolb [Kolb and Fry, 1974] afin d'assigner à chaque ressource éducative et à chaque apprenant deux attributs les décrivant (type d'apprentissage de la ressource/de l'apprenant et niveau de compétence de la ressource/de l'apprenant). Les agents possèdent aussi ces attributs puisqu'ils correspondent aux apprenants. Les auteurs calculent ensuite un niveau de correspondance entre l'apprenant et la ressource éducative, permettant de juger de la qualité du chemin trouvé. Finalement, la séquence, dont les ressources éducatives la composant correspondent le plus aux caractéristiques de l'agent, est recommandée à l'apprenant. Ce travail est intéressant car il pousse encore un peu plus la personnalisation des recommandations. Cependant, on commence à remarquer ici que la nécessité de personnaliser toujours plus les recommandations commence à complexifier de manière importante les systèmes proposés. Nous discuterons plus en détail de ce point dans la conclusion de cette section.

Prenons enfin une dernière étude dans le domaine de l'e-éducation qui s'est particulièrement intéressée à la notion de séquence et d'objectif à atteindre pour l'apprenant [Van den Berg et al., 2005]. En effet, les auteurs expliquent que les plateformes d'éducation en ligne proposent de plus en plus d'atteindre des objectifs pédagogiques de haut niveau (e.g. acquérir une compétence spécifique, posséder un certain niveau de connaissance dans un domaine,...) tout en laissant les apprenants libres de suivre les modules qu'ils souhaitent. Cependant, cette flexibilité de fonctionnement complexifie les programmes de ces plateformes et peut amener à l'abandon des apprenants. Pour remédier à ce problème, les auteurs proposent un système de guidage des apprenants utilisant ACO basé sur la spécification au préalable de plusieurs concepts pour chaque apprenant :

— Le but de l'apprenant : c'est la description du niveau de compétence que l'apprenant veut atteindre) ;
— Le parcours : le plan pour atteindre le but de l'apprenant, décrit comme une série de séquences de ressources éducatives. Les plateformes d'apprentissage en ligne proposent souvent elles-mêmes des parcours via lesquels les apprenants peuvent atteindre leur but ;
— La position de l'apprenant : représente l'ensemble des ressources éducatives ayant été complétées par l'apprenant.

Autrement dit, afin de pouvoir satisfaire un apprenant, il est nécessaire de savoir quel est

le but qu'il souhaite atteindre, la manière dont il veut l'atteindre et sa position actuelle. Ces trois informations sont primordiales pour comprendre la trajectoire d'un utilisateur et pour être capable de lui recommander une séquence adaptée à ses besoins.

Nous avons vu dans cette section le potentiel des algorithmes de colonies de fourmis pour la recommandation. Néanmoins, la première partie de l'état de l'art a montré à quel point il était important pour les systèmes actuels de pouvoir prendre en compte plusieurs facteurs humains afin de satisfaire au mieux les utilisateurs. Dans la section suivante, nous allons nous intéresser à la manière dont les algorithmes de colonies de fourmis sont capables de minimiser non plus un seul objectif (la distance), mais bien plusieurs objectifs pouvant être opposés les uns aux autres.

2.5.2 Les algorithmes de colonies de fourmis multi-objectifs

En parallèle du développement des algorithmes de colonies de fourmis mono-objectif dans la littérature, une partie de la communauté scientifique s'est intéressée aux algorithmes de colonies de fourmis multi-objectifs dans le but de résoudre de nouvelles classes de problèmes. En effet, les algorithmes ACO ont à l'origine été conçus pour minimiser une valeur, souvent représentée par la notion de distance. Cependant, de nombreux problèmes nécessitent la minimisation de multiples valeurs en parallèle afin d'obtenir une solution. On peut par exemple citer les variantes multi-objectifs de problèmes vus dans la section précédente :

— Le problème du voyageur de commerce multi-objectifs (*multi-objective travelling salesman problem*) : où la distance à minimiser ainsi que le temps de voyage sont les deux objectifs à minimiser (avec l'introduction de contraintes de circulation rendant certains chemins plus ou moins longs à emprunter) ;
— Le problème de la tournée de véhicules avec fenêtre de temps (*vehicle routing problem with time windows*) : où, en plus de minimiser le nombre de véhicules utilisés et le trajet total parcouru par ceux-ci, les clients doivent être livrés dans une certaine fenêtre de temps.

Dans le cas de ces deux problèmes, ainsi que pour beaucoup de problèmes multi-objectifs traités dans la littérature scientifique, les objectifs à minimiser sont souvent incompatibles entre eux, dans le sens où minimiser un objectif spécifique tend à pénaliser un ou plusieurs des autres objectifs. Prenons comme exemple le problème du voyageur de commerce multi-objectifs. Le but est de trouver le plus court chemin d'un point A à un point B non seulement d'un point de vue de la distance parcourue mais aussi du temps nécessaire pour faire le chemin. Cependant, minimiser la distance parcourue impliquera potentiellement d'emprunter des chemins encombrés, rallongeant le temps de voyage. En d'autres termes, minimiser l'objectif de distance parcourue peut dégrader l'objectif de temps. Inversement, optimiser l'objectif de temps demanderait potentiellement d'utiliser des chemins moins empruntés mais plus longs, dégradant ainsi l'objectif de minimisation de la distance totale parcourue. Ces problèmes, où l'optimisation d'un objectif affecte les performances des autres objectifs, ne possèdent souvent pas de solution optimale. En effet, dans un problème mono-objectif, plus un algorithme réussira à minimiser cet objectif et meilleure sera la solution. Un problème mono-objectif possède donc une solution optimale où l'objectif à optimiser possède une valeur minimale et, corollairement, il est possible de classer les solutions trouvées en fonction de cette valeur. Autrement dit, pour deux solutions différentes s_1, s_2 devant minimiser la valeur v pour le même problème, si $v_{s_1} < v_{s_2}$ alors s_1 est strictement une meilleure solution que s_2. Pour les problèmes multi-objectifs, il n'est pas directement possible de déterminer si une solution est meilleure qu'une autre. L'exemple du voyageur de commerce

multi-objectifs avec la distance d et le temps t illustre bien ce fait : comment déterminer si une solution $s_1 = \begin{cases} d = 13\ km \\ t = 23\ mn \end{cases}$ est meilleure qu'une autre solution $s_2 = \begin{cases} d = 11\ km \\ t = 26\ mn \end{cases}$? Devant la multitude de solutions possibles sans moyen direct de les classer, il est utile de considérer le concept d'optimum de Pareto. Le principe de Pareto a d'abord été inventé dans le domaine socio-économique pour décrire la distribution de pouvoir et de richesses au sein d'une population. Il a ensuite été étendu avec la notion d'optimum de Pareto, décrivant l'ensemble des solutions à un problème multi-objectifs où il n'est pas possible d'améliorer un des objectifs sans dégrader les autres [34]. Toutes les solutions Pareto-optimales à un problème sont donc équivalentes entre elles (même si elles sont bien toutes différentes les unes des autres) et sont strictement meilleures que toutes les autres solutions non optimales à ce même problème (aussi appelées solutions dominées). La Figure 2.9 illustre l'espace des solutions sur un problème de minimisation de deux variables. On remarque la présence d'une frontière, aussi appelé front de Pareto, où toutes les solutions sont optimales.

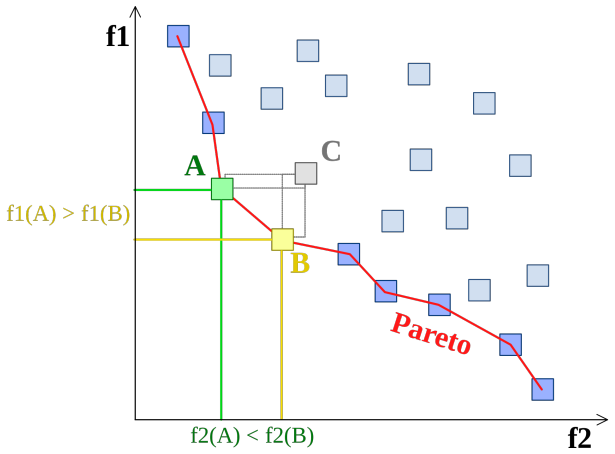

FIGURE 2.9 – Front de Pareto d'un problème de minimisation de deux variables $f1$ et $f2$. Toutes les solutions sur cette frontière sont optimales (source Wikipedia).

On peut trouver dans la littérature des exemples de résolution de problèmes multi-objectifs avec ACO utilisant le concept de Pareto. [Alaya et al., 2007] proposent un algorithme générique basé sur le *MAX-MIN Ant System* [35] où les nombres de colonies et de structures de phéromones varient en fonction du nombre d'objectifs m à optimiser. Les auteurs proposent ainsi plusieurs variantes de leur algorithme ayant des paramètres différents (nombres de colonies, de phéromones, de fourmis, de cycles et valeur des meta-paramètres). Ces différentes versions sont ensuite testées avec le problème du sac-à-dos multi-objectifs [36] et montrent que la version possédant 1 seule colonie et m phéromones obtient les meilleurs résultats. Outre les résultats obtenus, ce travail est intéressant car il propose plusieurs stratégies différentes pour pouvoir adapter les algorithmes

34. On peut décliner ce concept en socio-économie où le but est d'étudier les richesses d'une société afin qu'elles soient réparties selon une solution optimale de Pareto entre toutes les personnes. Ce concept a cependant des limites que nous verrons par la suite.

35. *Le MAX-MIN Ant System* [Stützle and Hoos, 2000] est une version différente de l'algorithme initial de Marco Dorigo possédant deux différences majeures avec ce dernier : seule la fourmi ayant réalisée le meilleur parcours peut mettre à jour les phéromones du chemin emprunté et la quantité de phéromones est limité dans un intervalle spécifique sur chaque arc.

36. Le plus souvent référencé dans la littérature en anglais via l'acronyme MOKP : *Multi-Objective Knapsack Problem*.

ACO aux problèmes multi-objectifs, à savoir :

- Traces de phéromones : 2 stratégies existent lorsqu'un problème possède plusieurs objectifs. La première est de n'utiliser qu'un seul type de phéromones. Dans ce cas, la quantité de phéromones déposée par les fourmis est définie par une agrégation des différents objectifs. La deuxième stratégie consiste à utiliser plusieurs phéromones. Dans ce cas, chaque objectif est généralement associé à une colonie de fourmis unique, chacune possédant son propre type de phéromones.
- Définition des heuristiques : encore une fois, 2 stratégies existent pour la partie heuristique. Une première solution consiste à agréger tous les objectifs en une seule heuristique. Une deuxième solution consiste à considérer chaque objectif séparément. Dans ce cas, il y a généralement une colonie par objectif.
- Mise-à-jour des phéromones : lors de la mise à jour des phéromones, il est nécessaire de décider sur quelle solution des phéromones doivent être déposées. Une première possibilité est de récompenser les meilleures solutions à chaque objectif. Une seconde possibilité est de récompenser toutes les solutions appartenant à l'ensemble des solutions Pareto-optimales.

Le concept d'optimisation multi-objectifs est aussi utilisé pour résoudre des problèmes sans utiliser des algorithmes de type ACO. Ces travaux sont intéressants dans le cadre de cette thèse car ils peuvent offrir un point de comparaison aux modèles développés dans le chapitre suivant où le but est le même mais la technique utilisée pour y parvenir est différente. Nous allons maintenant voir plus en détails le travail de [Ribeiro et al., 2014] où les auteurs ont proposé deux méthodes de recommandation multi-objectifs sans utiliser d'algorithmes de type ACO :

- ***Pareto-efficient ranking*** : dans cette première méthode, le principe est ici de représenter chaque ressource recommandable à un point dans un espace à n dimensions, appelé le *user-interest space*. Un point dans cet espace est représenté par n coordonnées $[c_1, c_2, \ldots, c_n]$ où chacune représente un score de pertinence de la ressource pour l'utilisateur estimé par différents algorithmes. Les points situés à la frontière de Pareto correspondent aux cas où aucune Pareto-amélioration n'est possible dans les différentes dimensions.
- ***Pareto-Efficient Hybridization (PEH)*** : dans cette deuxième méthode, le principe est d'associer chaque ressource avec une valeur de pertinence unique pour l'utilisateur. Cette valeur est elle-même estimée par une combinaison linéaire des scores de pertinence obtenus par n différents algorithmes de recommandation existants (*i.e.* $\alpha \times c_1 + \beta \times c_2 + \ldots + \theta \times c_n$). Dans cette tactique, l'espace, appelé *objective space*, comporte 3 dimensions où chaque point correspond aux niveaux de précision, de diversité et de nouveauté atteint par une technique d'hybridation possible. Le but est de chercher les poids (*i.e.* $\alpha, \beta, \ldots, \theta$) pour lesquels l'hybridation des différents algorithmes permet d'obtenir le meilleur résultat pour l'utilisateur. Il est ensuite possible de moduler cette hybridation afin de correspondre au mieux aux priorités de l'utilisateur au moment de la recommandation (*e.g.* pour un utilisateur récent, le système pourrait proposer des ressources pertinentes et classiques pour lui prouver l'intérêt du système de recommandation, tandis que pour un utilisateur ancien, il pourrait être intéressant de privilégier la diversité et la nouveauté afin de lui apporter de nouvelles ressources qu'il ne connaît pas encore).

Selon les auteurs, la première méthode permet de trouver un ordonnancement partiel entre

les ressources tout en évitant les ressources positionnées aux positions extrêmes de la frontière de Pareto. La deuxième méthode, quant à elle, utilise la frontière de Pareto pour trouver des hybridations capable de s'adapter aux demandes et d'offrir des recommandations étant simultanément pertinentes en termes de précision, de diversité et de nouveauté. Dans le cadre de cette thèse, la deuxième technique, *pareto-efficient hybridization* (PEH), est particulièrement intéressante car elle correspond aux problématiques posées au chapitre précédent. De plus, cet article est à la fois récent est beaucoup référencé dans la littérature scientifique de par la qualité des solutions proposées. Ainsi, nous préciserons le fonctionnement de l'algorithme PEH proposé par [Ribeiro et al., 2014] et nous utiliserons ses résultats comme comparaison pour évaluer les performances de notre modèle dans la chapitre présentant les expérimentations menées et les résultats obtenus.

2.5.3 Discussion sur l'utilisation de l'optimum de Pareto

Dans ces dernières pages, nous nous sommes attardés sur l'utilisation du concept d'optimum de Pareto dans la littérature. Cependant, même si l'utilisation de cette technique est courante pour résoudre des problèmes multi-objectifs, elle n'est pas forcément la plus adaptée selon les situations rencontrées et les solutions attendues. On peut dénombrer au moins quatre problèmes engendrés par l'utilisation du concept de Pareto :

1. Comme le souligne [Iredi et al., 2001], l'utilisation du front de Pareto est utile lorsqu'il n'est pas possible d'ordonner l'importance des différents objectifs du problème. Lorsque cela est possible, des poids sont attribués à chacun des objectifs et il devient possible de classer les différentes solutions obtenues selon ces poids.
2. Il est nécessaire d'être capable de générer beaucoup de solutions pour un problème donné afin de pouvoir déterminer les solutions Pareto-optimales et les solutions dominées.
3. Une solution Pareto-optimale n'est pas forcément une solution intéressante pour le problème. Un exemple connu illustre ce fait : une société où un seul homme possède l'ensemble des richesses est optimale car partager ces richesses entraînerait la réduction du bien-être d'au moins un individu.
4. Si le problème exige le choix d'une solution, le front de solutions Pareto-optimales ne permet pas, *a priori*, de faire un choix étant donné que toutes les solutions du front sont équivalentes les unes aux autres. Le problème dispose alors d'un ensemble de solutions le résolvant et non d'une seule solution.

Le dernier point ci-dessus est également évoqué dans [Dorigo and Birattari, 2010] où les auteurs expliquent que deux manières de résoudre les problèmes multi-objectifs existent dans la littérature : soit il est possible de classer les objectifs par ordre d'importance, ou éventuellement de combiner tous les objectifs en un seul grâce à une somme pondérée ; soit les préférences ou les poids *a priori* ne sont pas connus et, dans ce cas, il est intéressant de déterminer un ensemble de solutions Pareto optimales. Dans le cadre d'un système de recommandation, et comme nous l'avons illustré dans la Section 2.2, il est possible d'estimer les besoins des utilisateurs dans chacune des dimensions évaluées (diversité, nouveauté, précision, ...) et donc de moduler leur importance en amont de la recommandation (*e.g.* en leur attribuant des poids par exemple).

2.6 Discussion générale

Dans cet état de l'art, nous avons fait un tour d'horizon du domaine de la recommandation et de son évolution au cours de ces dernière années. Avec la démocratisation d'Internet, de plus

en plus d'utilisateurs ont été amenés à utiliser ces systèmes et, aujourd'hui, ils accompagnent l'immense majorité des grands sites web et prennent une place de plus en plus importante. Cette démocratisation a permis la mise en relief des besoins toujours plus variés des utilisateurs et a montré que les systèmes mono-objectif basés uniquement sur la précision des recommandations n'étaient plus suffisants. À la lumière de ce constat, nous nous sommes particulièrement intéressés aux techniques permettant de prendre en compte le plus de dimensions possibles dans les besoins des utilisateurs, que nous avons appelés "facteurs humains". Les travaux de la littérature du domaine présentés dans cette section ont permis de mettre en exergue un certain nombre de points intéressants qui seront utilisés pour répondre aux problématiques de cette thèse, ils sont résumés dans la liste suivante :

— [Dorigo and Birattari, 2011] : les auteurs ont présenté différents modèles d'algorithmes de fourmis permettant de résoudre un grand nombre de problèmes. Ces modèles sont intéressants notamment de par leur capacité de résoudre des problèmes multi-objectifs complexes grâce aux notions d'intelligence collective et d'émergence. Ils sont utilisés dans différents domaines d'application (optimisation, recommandation,...), ils sont très modulables et offrent de nombreuses possibilités pour les problématiques de cette thèse ;
— [Semet et al., 2003] : cette étude est l'une des premières à proposer l'utilisation de plusieurs phéromones différentes dans les ACO dans le cadre d'un système de recommandation dans l'e-education. Ces différents phéromones étaient chacun destinés à résoudre une facette du problème. Ce travail a néanmoins aussi montré les limites de la création manuelle d'un graphe sur lequel appliquer les ACO, démontrant l'intérêt d'avoir une méthode automatique de création et de valuation d'un graphe représentant le domaine d'application et ses ressources ;
— [Kurilovas et al., 2015] : les auteurs ont démontré dans cette étude l'intérêt de pouvoir catégoriser les utilisateurs d'un système afin de personnaliser davantage les recommandations, mais elle a aussi montré le besoin de pouvoir modifier en cours de fonctionnement ces catégories pour ne pas enfermer les utilisateurs dans les mêmes recommandations ;
— [Kardan et al., 2014] : les auteurs ont ici montré l'utilisation de plusieurs types d'agents différents, qui chacun cherchaient différentes solutions. Cette technique est proche de l'utilisation des différents types de phéromones de [Semet et al., 2003] et permet de chercher des solutions plus variées à des problèmes complexes ;
— [Yang and Wu, 2009] : cette étude utilise les modèles ACO dans l'e-education en étoffant encore le niveau de personnalisation pour chaque utilisateur. Elle illustre cependant le problème courant où l'amélioration de la personnalisation des recommandations résulte souvent dans l'augmentation de la complexité du modèle ;
— [Van den Berg et al., 2005] : cet article propose de recommander des séquences de ressources éducatives à des apprenants. Il illustre l'intérêt de définir une séquence de recommandation avec des buts pédagogiques à atteindre, un objectif final et un point de départ où l'utilisateur se trouve actuellement ;
— [Ribeiro et al., 2014] : ce dernier article n'utilise pas les ACO mais une hybridation de plusieurs algorithmes permettant de proposer des recommandations de ressources pertinentes pour chaque utilisateur sur 3 dimensions (précision, diversité, nouveauté). Il propose donc de résoudre un problème similaire à celui posé dans cette thèse mais avec une technique différente des travaux présentés précédemment, il sera par conséquent une solide base de comparaison pour mesurer les performances d'un algorithme de type ACO.

D'une manière générale, cette revue des travaux sur les algorithmes de colonies de fourmis

ont montré leur puissance de résolution de problèmes variés et leur modularité. Cependant, nous avons aussi constaté ici une tendance générale ces dernières années à complexifier toujours plus le modèle initial et les agents le composant, illustrant un certain éloignement avec la philosophie des modèles multi-agents. Celle-ci repose sur la résolution de problèmes complexes grâce à la propriété de l'émergence et de l'auto-organisation d'agents simples, régis par des règles simples. Cette complexification découle cependant de la direction prise par le domaine des systèmes de recommandation ces dernières années avec la démocratisation de nombreuses mesures d'évaluation telles que la diversité, la nouveauté, etc. Dans la suite de cette thèse, nous étudierons la manière dont il est possible de combiner la nécessaire complexification des recommandations afin d'être le plus pertinent possible pour l'utilisateur, et la simplicité inhérente aux modèles multi-agents.

En conclusion finale de cet état de l'art, nous pouvons définir deux enjeux qui guideront la suite de ce travail : **la recommandation multi-objectifs** et **la recommandation en séquences**. Le but de cette thèse est donc de s'inscrire à l'intersection de ces deux enjeux grâce à un modèle de colonies de fourmis multi-objectifs qui sera décrit dans le chapitre suivant.

Chapitre 3

Notre modèle de recommandation multi-objectifs : AntRS

Sommaire

3.1 Introduction

Après cet état des lieux de la recherche sur les systèmes de recommandation, il peut être utile de rappeler ici les principaux objectifs que nous nous sommes fixés pour être satisfaits par notre modèle multi-objectifs de recommandation de séquences :

— Être suffisamment générique pour qu'il puisse être utilisé dans plusieurs domaines tout en s'adaptant aux diverses contraintes et évolutions de l'environnement de ces domaines ;
— Permettre de prendre des facteurs humains ;
— Pouvoir gérer d'importantes quantité de données, c'est-à-dire être capable de produire des recommandations satisfaisantes dans un grand espace de recherche ;
— Produire des recommandations sous forme de séquences.

Tous ces objectifs sont nécessaires pour produire un modèle répondant aux problématiques énoncées dans l'introduction de ce manuscrit. Le modèle que nous avons conçu prend inspiration des modèles de colonies de fourmis présentés dans l'état de l'art, et plus particulièrement de la variante ACS (*Ant Colony System*), car cette version possède des qualités intrinsèques importantes, notamment sa capacité supérieure d'exploration de l'environnement, pour répondre aux problématiques posées. Pour rappel, les avantages du modèle ACS ont été détaillés dans la Section 2.4.3. En se basant sur . Dans la suite de ce chapitre, nous détaillerons le fonctionnement général du modèle dans la Section 3.2. Ensuite, nous présenterons la création du graphe dans la Section 3.3 et les facteurs humains considérés et leur intégration dans la Section 3.4. Nous détaillerons par la suite les deux tactiques de fusion des résultats pour la recommandation finale dans la Section 3.5 et, enfin, nous discuterons du modèle dans son ensemble.

3.2 Fonctionnement général du modèle

Dans le chapitre précédent sur l'état de l'art, nous avons présenté dans la Section 2.4.3 le modèle ACS de Dorigo et son fonctionnement [Dorigo and Birattari, 2010]. Notre modèle s'inspire d'ACS dans le sens où nous reprenons les grandes étapes de ce dernier tout en l'adaptant au contexte et aux contraintes des systèmes de recommandation. Pour rappel, voici l'algorithme 1 de haut niveau sur lequel se base le fonctionnement de l'ensemble des algorithmes des ACO.

Algorithm 1 Métaheuristique *Ant Colony Optimization*

```
Initialisation des paramètres et des phéromones
while conditions d'arrêts non atteintes do
    Construction des solutions
    Recherche Locale (optionnelle)
    Mise à jour des phéromones
end while
```

Chaque version existante des ACO a introduit des variations dans une ou plusieurs des étapes centrales de l'algorithme. La variante ACS a par exemple introduit la mise à jour locale des phéromones dans la dernière étape *UpdatePheromones*. Notre modèle suit cette philosophie en prolongeant les travaux de Dorigo à travers une variante multi-colonies et multi-objectifs que nous avons nommé AntRS. Avant de définir plus en détail ces modifications dans le reste du chapitre, voici l'algorithme 2 modifié correspondant au fonctionnement de notre modèle.

Algorithm 2 AntRS

```
Initialisation des paramètres et des phéromones
Création du graphe
Initialisation des différentes colonies
while conditions d'arrêts non atteintes do
    Construction des solutions pour chaque colonie
    Mise à jour locale et globale des phéromones modifiée
end while
Fusion des différentes solutions
Recommandation
```

Dans notre modèle inspiré d'ACS, nous pouvons distinguer un certain nombre d'apports qui seront approfondis dans la suite de ce chapitre. La création du graphe prend ainsi une place prépondérante en amont de l'exécution de l'algorithme afin de pouvoir représenter correctement le domaine d'application et ses ressources. De plus, alors que dans les modèles initiaux de Dorigo, une seule colonie de fourmis avait pour but de trouver des solutions, notre modèle possède plusieurs colonies travaillant chacune à optimiser un objectif (i.e. un aspect correspondant à un facteur humain) de la recommandation finale. Ces différentes colonies constituent le cœur de notre approche et seront expliquées en détail. Enfin, après la convergence d'une solution dans chacune des colonies (i.e. un chemin dans le graphe correspondant à une séquence de ressources dans un ordre précis), un dernier processus de fusion de ces solutions a lieu afin de ne proposer à l'utilisateur final qu'une seule séquence offrant le meilleur compromis entre les différents objectifs de chaque colonie. Après cette dernière étape, la recommandation peut être proposée à l'utilisateur. Dans la suite de ce chapitre, ces différentes étapes seront détaillées afin de mettre en exergue le fonctionnement de notre modèle.

3.3 Graphe

La première étape de fonctionnement du modèle est la création du graphe. Celui-ci représente l'environnement dans lequel les agents évoluent. Dans ce contexte, les nœuds représentent les ressources du domaine (musiques, oeuvres d'art, contenu pédagogique, etc.) tandis que les arêtes sont les chemins permettant d'accéder à ces ressources. Les agents se déplacent ainsi de nœuds en nœuds en empruntant des arêtes. Le processus de création du graphe n'est souvent que peu détaillé dans la littérature pour plusieurs raisons. Premièrement, un certain nombre d'études se focalisent sur un jeu de test composé de peu de ressources, limitant l'intérêt de développer des tactiques élaborées de création d'un graphe. Cela peut être dû à la simplicité d'utiliser un petit jeu de données, aux limites posées par le domaine d'application ou encore à la difficulté de récolter des données. Par exemple, si le but est de développer un système de recommandation dans un musée contenant 100 œuvres d'art, le graphe représentant le domaine sera relativement peu complexes (i.e. 100 nœuds et quelques centaines ou milliers d'arêtes). De la même manière, lorsque le jeu de données est limité en taille, il est possible pour des experts du domaine de créer et de valuer manuellement les arêtes entre les différentes ressources, comme cela a été fait dans [Semet et al., 2003]. Cependant, la conception d'un algorithme permettant de construire la topologie du graphe de manière automatique possède à la fois des avantages et des inconvénients :

Avantages	Inconvénients
• La conception algorithmique du graphe permet de s'adapter rapidement à différents domaines applicatifs ayant différents types et nombres de ressources ; • Il devient possible d'intégrer un très grand nombre de ressources du domaine applicatif si cela est nécessaire (par exemple toutes les musiques d'un site d'écoute de musiques en ligne) ; • L'intervention d'experts humains ne devient plus nécessaire pour créer la topologie du graphe et valuer ses arêtes.	• La qualité d'un graphe produit automatiquement dépend fortement de la qualité de son algorithme et peut potentiellement être moins pertinent qu'un graphe crée par un expert du domaine ; • La prise en compte d'un grand nombre de ressources a pour conséquence d'augmenter significativement le temps de calcul de l'algorithme travaillant sur ce graphe en aval ; • L'algorithme de création automatique du graphe doit capturer les informations d'un domaine et les refléter dans la topologie du graphe (i.e. pourquoi une ressource est liée par une arête à une autre, etc.).

Le but de cette section sera donc de détailler l'algorithme de création du graphe ainsi que les réponses proposées aux inconvénients énoncés ci-dessus.

Avant de présenter la méthode de création du graphe, il est important de discuter de ce que représente ce graphe dans le modèle ACO et de son pendant pour les fourmis. Une des différences principales entre le comportement réel des fourmis dans la nature et les simulations informatiques des algorithmes ACO réside dans la définition de l'espace de recherche. Les fourmis réelles évoluent dans un espace de recherche continu sans aucun point de repère ni chemin pré-établi. Les fourmis sont, en outre, libres d'explorer l'environnement où bon leur semble. Les agents simulés avec ACO sont, quant à eux, limités à un espace de recherche discret (le graphe) et ne peuvent se mouvoir qu'à certains points d'intérêts (les sommets) en empruntant certains chemins déjà établis (arêtes). Cette simplification de l'espace de recherche est nécessaire à la fois pour réduire la complexité de l'algorithme et pour permettre de représenter des domaines d'application possédant des ressources connues, contrairement à un environnement naturel inconnu et potentiellement infini du point de vue d'une fourmi. Cependant, les algorithmes ACO ont avant tout un but de simulation du comportement de ces fourmis. Afin de combiner à la fois les contraintes d'une simulation informatique avec le comportement réel observé des fourmis dans la création du graphe, plusieurs solutions sont possibles.

L'un des moyens les plus simples pour approcher le comportement réel des fourmis dans la nature consiste à créer des arêtes entre chaque sommet du graphe dans le but de créer un environnement où l'agent simulé pourra se déplacer où il veut en empruntant le chemin qu'il désire. Le graphe créé serait ainsi complet. Néanmoins, cette approche n'est pas réaliste pour deux raisons principales. Premièrement, l'environnement peut contenir des obstacles infranchissables pour des fourmis et elles ne peuvent donc pas se déplacer partout. Dans un graphe, cela se traduirait par une absence d'arête entre deux sommets. Deuxièmement, cette solution est aussi peu pratique dès que le nombre de sommets dépasse plusieurs centaines car le nombre d'arêtes n créées se calcule avec $|E| = \frac{n(n-1)}{2}$. Le nombre d'arêtes d'un graphe de 100 sommets est donc de 4950 tandis que pour un graphe de 1000 sommets, ce nombre atteint déjà 499.500, comme le montre la Figure 3.1. Cette augmentation exponentielle n'est évidemment pas envisageable dès lors que l'on traite de domaines pouvant posséder plusieurs dizaines de milliers de ressources à

représenter dans le graphe.

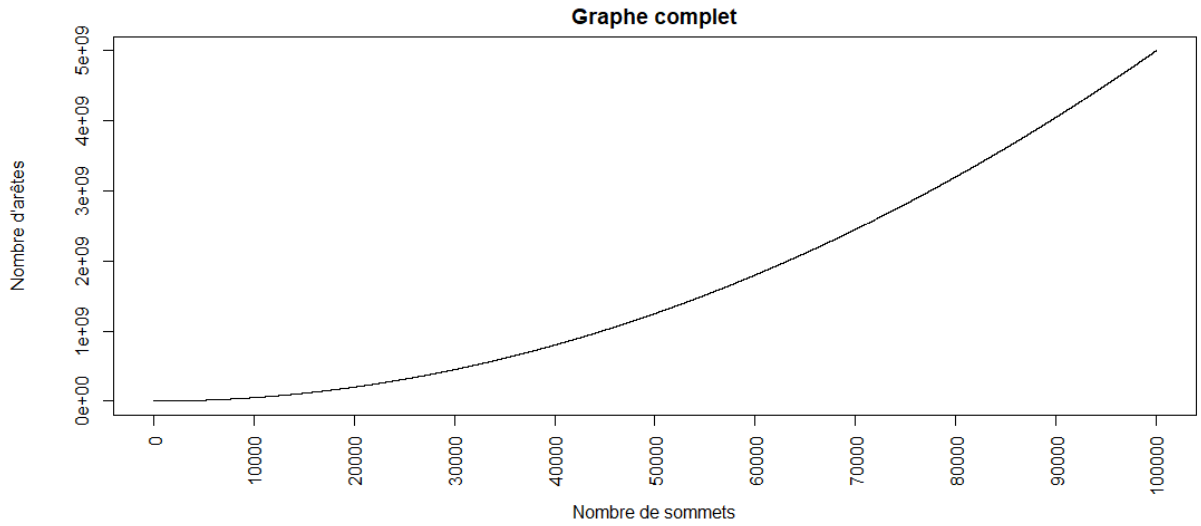

FIGURE 3.1 – Graphe représentant l'évolution exponentielle du nombre d'arêtes selon le nombre de sommets existants dans un graphe complet.

Un des objectifs de cette thèse étant d'appliquer notre modèle à un domaine applicatif réel possédant de très nombreuses ressources, la création d'un graphe complet n'était pas une option. Nous avons donc opté pour une approche plus réaliste dans la création du graphe afin de diminuer sa taille et les temps de calculs potentiels. Pour réaliser cette tâche, nous n'avons que deux solutions possibles : supprimer des sommets et/ou supprimer des arêtes. La première possibilité a rapidement été écartée car supprimer des sommets affecte directement la qualité des solutions trouvées en faisant mécaniquement baisser la mesure de couverture du système de recommandation, c'est-à-dire la capacité qu'a un système à recommander toutes les ressources d'un domaine. De plus, il n'est pas évident de justifier la suppression de certains sommets par rapport à d'autres. Chaque sommet du graphe représentant une ressource unique du domaine d'application, la suppression de certains sommets ne pourrait se faire que par la création de règles générales. Une solution possible pourrait être par exemple de ne pas intégrer dans le graphe les ressources n'ayant jamais été consultées, ou n'ayant pas été consultées lors des 6 derniers mois. Cependant, on voit bien que ces règles ne sont pas forcément adaptées à chaque utilisateur et ont un grand risque de supprimer des ressources que ces derniers auraient pu apprécier. Nous avons donc préféré la deuxième solution : supprimer des arêtes. Cette méthode a l'avantage de garder l'ensemble des ressources du domaine recommandables par le système tout en réduisant grandement la complexité du graphe [37]. La question s'est donc posée d'identifier les arêtes à sélectionner, le but étant de permettre au système de proposer des recommandations à la fois qualitatives, adaptées mais aussi potentiellement nouvelles à l'utilisateur tout en réduisant la taille du graphe. Pour ce faire, nous avons formulé deux hypothèses aidant à la sélection des arêtes :

37. Tant qu'il y a un chemin dans le graphe permettant de se déplacer d'un sommet quelconque vers tous les autres sommets, alors toutes les ressources peuvent théoriquement être recommandées par le système.

1. **Les séquences de ressources déjà consultées par le passé contiennent des informations sur le comportement des utilisateurs qui peuvent être exploitées pour la topologie du graphe ;**
2. **Les séquences de ressources déjà consultées par les utilisateurs auparavant ne sont pas toujours les meilleures possibles et auraient pu être améliorées avec des recommandations pertinentes.**

La première hypothèse H1 se justifie par le fait que l'utilisateur est le premier à connaître ses envies, ses besoins et les ressources qui le satisferaient le plus à un moment t [Jones, 2010]. Il est donc nécessaire de considérer ces séquences de consultations passées afin de comprendre l'utilisation que font les utilisateurs des ressources du domaine. La deuxième hypothèse se justifie quant à elle par le fait qu'un utilisateur ne peut pas, dans la majorité des cas, avoir une vue d'ensemble de toutes les ressources d'un domaine. Il est en effet courant pour un site de e-commerce ou d'écoute de musiques de posséder plusieurs dizaines de milliers de ressources. La deuxième hypothèse H2 se base donc sur le problème fondamental que les systèmes de recommandation essaient de résoudre, à savoir aider les utilisateurs à faire les meilleurs choix possibles dans des environnements toujours plus complexes et vastes. Cette discussion sur les défis amenés par ces deux hypothèses nous amène à considérer le travail d'Herbert Simon qui a été décoré d'un prix nobel d'économie et d'un prix Turing. Ce dernier a théorisé et nommé le "problème de rationalité limitée" : devant des contraintes altérant et limitant les capacités d'un individu (domaine trop vaste, manque d'information, etc.), il est préférable de se diriger vers des solutions satisfaisantes plutôt qu'optimales [Simon, 1997]. En réponse à ce problème, les utilisateurs procèdent donc par essais-erreurs pour répondre à leurs besoins de manière empirique [Jameson et al., 2015]. Les séquences d'interactions créées par les utilisateurs durant ces processus contiennent donc beaucoup d'informations exploitables par un système de recommandation, mais aussi beaucoup de bruits et d'approximations [Kuhlthau, 1991, Castagnos et al., 2010]. La philosophie de notre modèle est les réponses que nous apportons aux problèmes ci-dessus sont inspirées par cette théorie.

Afin de prendre en compte ces deux hypothèses, nous proposons dans la suite de cette section une méthode permettant de créer un graphe orienté en se basant à la fois sur les séquences passées des utilisateurs ainsi que sur la création de nouvelles arêtes. L'objectif est d'exploiter les informations contenues dans les séquences passées des utilisateurs, mais aussi de garantir une certaine sérendipité du système de recommandation via la création de nouvelles arêtes. La première étape de ce processus est de créer un sommet par ressource du domaine. Ensuite, pour construire les arêtes reliant ou non ces sommets, nous calculons les poids associés aux nombre de co-consultations présentes dans l'ensemble des séquences de consultations passées. Autrement dit, quand un utilisateur consulte une ressource A puis une ressource B directement après, nous considérons que la transition $A \rightarrow B$ est une co-consultation. Pour déterminer si une arête est construire entre deux sommets, nous calculons ensuite le poids de la co-consultation, c'est-à-dire le nombre de fois qu'elle apparaît dans les séquences des utilisateurs. En fonction de ce poids, deux opérations sont réalisées :

1. Toutes les co-consultations possédant un poids supérieur à seuil spécifique sont ajoutées comme arêtes orientées dans le graphe reliant les deux sommets concernés. Le seuil est déterminé de manière empirique car il est dépendant de chaque domaine et de la base de données utilisées ;
2. En dessous du seuil, seules certaines co-consultations sont sélectionnées pour devenir des

arêtes selon un processus intégrant une part d'aléatoire que nous détaillerons dans la suite de cette section.

Ces deux opérations permettent de peupler le graphe d'arêtes correspondant à la fois à toutes les transitions les plus effectuées par les utilisateurs ainsi qu'à certaines des transitions moins usitées par ces derniers. Après ces deux opérations, le graphe possède donc un certain nombre d'arêtes reliant ses sommets, mais rien ne garantit que chaque sommet est bien relié au reste du graphe (si par exemple un sommet représente une ressource très peu consultée du domaine). De plus, et malgré la deuxième opération mitigeant cet effet, les transitions très usitées (entre des ressources populaires par exemple) seront sur-représentées avec cette méthode de construction. Cette sur-représentation produira forcément en aval des solutions que l'on pourrait qualifier de "classique", dans le sens où elles reprendraient en majorité ces transitions très utilisées et connues des utilisateurs. Afin de limiter cet effet et de s'assurer que tous les sommets sont bien connectés au graphe, une troisième et dernière opération est effectuée :

3. Pour chaque sommet, si un certain niveau de connectivité n'est pas atteint, autrement dit si le sommet possède moins d'arêtes qu'un seuil fixé à l'avance, alors des arêtes aléatoires sont ajoutées afin d'atteindre ce seuil.

Cette troisième opération permet de s'assurer qu'aucun sommet représentant une ressource peu consultée ne soit laissé sans connexion avec le reste du graphe. Ainsi, chaque sommet possédera un certain niveau de connectivité, aussi appelé degré, avec le reste du graphe. Le graphe ainsi obtenu après ces 3 opérations possède dans sa topologie à la fois les liens les plus usités par les utilisateurs, ainsi que des liens aléatoires permettant la création de recommandations potentiellement nouvelles. L'ensemble du processus de création du graphe est repris dans l'Équation 3.1 ci-dessous.

$$\begin{cases} (1) \text{ si } t_{ij} \geq m, \text{ alors créer } e_{ij} \\ (2) \text{ si } t_{ij} < m \text{ et } q < t_{ij} \text{ où } q \in [min\ t_{il}; \log(max\ t_{il})], \text{ alors créer } e_{ij} \\ (3) \text{ si } deg(i) < d \text{ alors choisir un sommet k aléatoire} \\ \text{ et créer } e_{ik} \text{ jusqu'à } deg(i) = d \end{cases} \tag{3.1}$$

(1) et (2) s'appliquent pour chaque co-consultation tandis que (3) s'applique pour chaque sommet du graphe. e_{ij} représente l'arête entre les sommets i et j, t_{ij} est le poids de la co-consultation de la ressource i vers la ressource j par les utilisateurs dans les séquences de consultations passées, m est le seuil au-dessus duquel les co-consultations sont automatiquement ajoutées au graphe comme arêtes, $q \in [min\ t_{il}; \log(max\ t_{il})]$ est une variable aléatoire uniformément distribuée dans l'intervalle décrit, $min\ t_{il}$ est le nombre minimal de co-consultations entre la ressource i et toutes les autres ressources $l \in V_i$ où au moins une co-consultation a été trouvée, $deg(i)$ est le degré du nœud i dans le graphe et d est un paramètre spécifiant le degré minimal de connectivité que doit posséder chaque sommet dans le graphe final. Il est à noter qu'à ce stade du processus de création du graphe, les arêtes ne sont pas valuées car les valeurs attribuées aux arêtes varient pour chaque utilisateur. Ce processus sera détaillé dans la suite de la section.

Avec cette technique de construction, la taille du graphe dépend désormais principalement de la variable d définissant le degré minimal de connectivité à atteindre (en plus du nombre de sommets comme précédemment). La Figure 3.2 ci-dessous représente l'évolution du nombre d'arêtes uniquement en fonction de la variable d. Dans la pratique, le nombre d'arêtes sera

supérieur aux nombres présentés dans la figure, étant donné que les deux premières étapes de la création du graphe ajoutent des arêtes en fonction des transitions trouvées dans les consultations passées des utilisateurs.

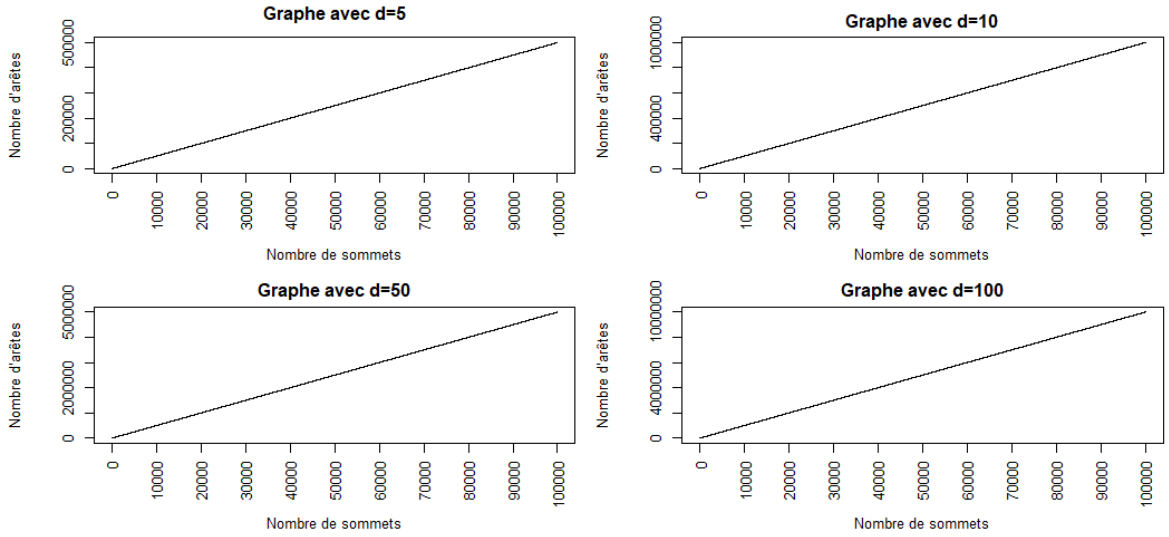

FIGURE 3.2 – Graphe représentant l'évolution linéaire du nombre d'arêtes selon le nombre de sommets voulus dans un graphe créé avec différentes valeurs de d.

Ce processus de construction de graphe a pour double objectif d'exploiter les informations des séquences des utilisateurs et de créer de nouvelles séquences possibles. Il existe potentiellement une infinité de créer un graphe à partir des ressources dont nous disposons, et il est évident que la topologie du graphe a une influence importante sur les résultats possibles, car le graphe représente l'entièreté de l'environnement dans lequel les agents vont évoluer. Le processus de construction que nous avons décrit n'est qu'une manière parmi d'autres d'aboutir à un graphe. Cependant, nous faisons l'hypothèse que la combinaison d'arêtes représentant des co-consultations aux poids importants, d'arêtes représentant certaines co-consultations moins présentes, et d'arêtes créant de nouvelles transitions permet de capturer l'information présente dans le domaine tout en garantissant une certaine sérendipité avec l'émergence possible nouvelles séquences.

3.4 Objectifs

Une fois le graphe créé, le modèle peut être appliqué sur ce dernier afin de produire des recommandations. Dans cette section, nous allons définir comment fonctionne le modèle pour produire des recommandations à partir du graphe ainsi que les facteurs humains, i.e. les objectifs, qu'il cherche à satisfaire.

Comme nous l'avons vu dans le chapitre précédent sur l'état de l'art, il est désormais largement admis que la mesure de précision seule n'est pas suffisante pour produire de bonnes recommandations aux utilisateurs. D'autres facteurs humains doivent être considérés pour satisfaire la plus grande variété d'utilisateurs et de contextes possible. Nous proposons donc de définir un ensemble de 4 facteurs humains parmi les plus étudiés dans la littérature et de les prendre en compte dans notre modèle de recommandation. Ces facteurs humains seront par la suite appe-

lés "objectifs" car, pour notre modèle, ces facteurs humains sont en effet considérés comme des objectifs à atteindre et à maximiser. Les voici accompagnée d'une courte description :

1. **Similarité** : recommander des ressources similaires à ce que l'utilisateur a aimé par le passé ;
2. **Diversité** : recommander des ressources diverses par rapport à ce que l'utilisateur a aimé par le passé ;
3. **Nouveauté** : recommander des ressources nouvelles que l'utilisateur ne connaît pas encore ;
4. **Préférences** : recommander des ressources que l'utilisateur a déjà apprécié dans le passé.

La similarité et les préférences sont proches des notions de précision et de rappel mais diffèrent toutefois dans le sens où l'on ne cherche pas à reproduire exactement les mêmes séquences que l'utilisateur a consulté, mais à lui recommander des séquences pouvant le satisfaire. Ces objectifs représentent des facteurs humains important dans la création d'une bonne recommandation. Ils seront détaillés dans la suite de cette section. À ces 4 objectifs concernant les ressources recommandées, nous ajoutons un cinquième objectif s'intéressant quant à lui à la manière dont sont proposées les recommandations à un utilisateur :

5. **Progressivité** : recommander une séquence de ressources de sorte que la transition entre chacune de ces ressources soit fluide et adaptée aux préférences de l'utilisateur.

Ces 5 objectifs considérés sont tous transposables dans différents domaines applicatifs, garantissant la généricité de cette approche. Cependant, il est évident que d'un domaine applicatif à l'autre, certaines différences peuvent justifier la mise en avant d'un objectif plutôt qu'un autre. De la même manière, les besoins et préférences pour ces objectifs peuvent varier d'un utilisateur à un autre au sein du même domaine. Pour répondre à ces deux problèmes, nous expliquerons aussi dans cette section comment adapter l'importance de chaque objectif. Il est aussi important de considérer le fait que nous nous appuyons sur ces objectifs car ils sont les plus fréquemment cités dans la littérature. L'objectif est de démontrer que notre modèle est capable de produire des séquences satisfaisant les objectifs considérés. Ce faisant, nous montrons par la même occasion que les objectifs peuvent être adaptés, enlevés ou ajoutés. En effet, on pourrait très bien considérer d'autres objectifs (personnalité, point de congestion si l'on est dans un musée, etc.). En d'autres termes, le but est de démontrer la généricité de notre modèle.

Avant de détailler comment calculer ces objectifs, nous allons expliquer ici comment les intégrer dans le modèle. Le but de notre modèle est de proposer des recommandations offrant un bon compromis entre les différents objectifs, où l'importance de ces derniers peut être ajustées à volonté. Pour ce faire, notre modèle doit être capable de transformer ces objectifs listés plus haut en valeurs pouvant être chiffrées et optimisées. Autrement dit, notre modèle a pour but de résoudre un problème de recommandation multi-objectifs. Le choix des algorithmes ACO est ici du sens étant donnés ces besoins spécifiques. En effet, nous avons montré dans l'état de l'art que le modèle ACS, faisant partie des ACO, était capable de résoudre des problèmes possédant plusieurs objectifs, comme par exemple avec le *vehicle routing problem with time windows* [Gambardella et al., 1999] ou encore avec le *multi-objective travelling salesman problem* [Angus, 2007]. Devant la complexité et les dimensions toujours croissantes des problèmes à résoudre en informatique, les approches multi-dimensionnelles des algorithmes ACO se sont multipliées ces dernières années

[Doerner et al., 2004, Alaya et al., 2007, Angus and Woodward, 2009, Dorigo and Stützle, 2019]. Même si chacune de ces approches a pour but de résoudre un problème multi-objectif, la manière d'y arriver est différente tant les algorithmes ACO sont modulaires. Pour rappel, notre problème consiste en la recommandation de séquences à partir d'un graphe et prenant en compte plusieurs objectifs pouvant potentiellement être incompatible entre eux. Il est important d'insister sur ce dernier point afin de comprendre la solution mise en place dans notre modèle. Les objectifs que nous avons décrits ci-dessus peuvent en effet être incompatibles les uns avec les autres, quand bien même le but est de générer une recommandation possédant un certain équilibre parmi chacun d'entre eux. L'objectif de préférences, dont le but est de proposer des ressources déjà connues et appréciées, ainsi que l'objectif de nouveauté, dont le but est de proposer des ressources encore non connues, sont par exemple en contradiction claire. Chaque objectif traité séparément produira potentiellement des recommandations mutuellement exclusives, et pourtant il peut être intéressant de proposer à l'utilisateur une séquence de recommandations possédant ces deux aspects spécifiques. Partant de ce constat, nous avons décidé d'intégrer chaque objectif comme une colonie différente dans le modèle ACS. Ainsi, nos 4 premiers objectifs (similarité, diversité, nouveauté, préférences) seront représentés par 4 colonies dans notre modèle tandis que le cinquième objectif, la progressivité, sera quant à lui traité à part étant donné qu'il concerne la structure de la recommandation et non les ressources recommandées en elles-mêmes.

Intégrer ces objectifs par le biais de plusieurs colonies a nécessité quelques changements dans l'algorithme ACS. Dans le modèle initial, les agents d'une seule colonie se déplacent sur le graphe dont chacune des arêtes est valuée en fonction de la distance séparant les 2 sommets la constituant. Le but des agents est ensuite de se déplacer sur ce graphe afin de trouver le plus court chemin d'un sommet A à un sommet B à l'aide des phéromones qu'ils y déposent. Une fois qu'une solution émerge comme étant la meilleure grâce au mécanisme de stigmergie, l'algorithme peut s'arrêter et la meilleure solution est retenue.

La philosophie derrière cette solution découle des principes même des systèmes multi-agents. L'état de l'art du chapitre précédent l'a montré, l'approche multi-agent pour la résolution de problèmes est à la fois simple, élégante et efficace. D'autres solutions auraient été possibles, comme par exemple complexifier les agents en leur ajoutant de nouveaux comportementS, mais nous avons voulu garder l'essence des systèmes multi-agents réactifs : l'émergence de comportements intelligents de par les interactions entre agents très simples. Afin de rester aussi proche que possible des principes de base de l'algorithme, nous avons voulu converser le fonctionnement général des colonies et de la découverte de solutions sur le graphe tout en les adaptant à notre contexte. Pour ce faire, nous avons jouer sur la manière dont se définit et se calcule la notion de **distance** dans le graphe. Dans les algorithmes ACO, la distance d associée à chaque arête du graphe représente comme son nom l'indique la distance physique nécessaire pour aller d'un sommet à l'autre de l'arête. Cette distance ne demande *a priori* pas de calcul et est simplement déduite, ou mesurée, de l'environnement représenté par le graphe. Elle est ensuite utilisée pour calculer l'heuristique η_{ij} comme montré dans l'Équation 2.12 (pour rappel, l'heuristique se calcule ainsi $\eta_{ij} = \frac{1}{d_{ij}}$). Nous n'avons pas modifié le calcul de η_{ij}, mais nous avons modifié le sens de la distance d qui le compose. Au lieu de représenter la distance physique entre deux sommets, d représente dans notre modèle une distance calculée entre deux ressources, sa valeur étant directement dépendante de l'objectif de la colonie. Prenons un exemple pour illustrer la manière dont cette distance peut varier entre deux sommets pour différentes colonies.

Supposons un système de recommandation dans le domaine muséal dont le but est de proposer des parcours composés de différentes peintures. Le musée possède, entre autres, deux tableaux de Vincent van Gogh appartenant à la série des Tournesols. Ces tableaux sont tous les deux des natures mortes représentant un vase et 15 tournesols ; ils ont tous les deux été peints en 1889 ; les palettes de couleurs et le style sont quasi similaires. Dans le graphe, les sommets A et B représentent ces deux tableaux et (A, B) représente l'arête les reliant. Pour toutes les raisons évoquées ci-dessus, on peut raisonnablement penser que ces deux ressources sont très similaires entre elles. La distance $d_{similarit}$ de la colonie axée sur la similarité sera donc faible, indiquant aux agents que les deux sommets sont effectivement "proches" l'un de l'autre et favorisant le passage par cette arête pour minimiser la distance totale de la solution. À l'inverse, la colonie relative à la diversité aura à traverser une plus longue distance $d_{diversit}$ entre les deux sommets comme ces derniers sont très similaires entre eux. À travers ces exemples sont illustrées les différences possibles dans le calcul de la distance d en fonction de l'objectif considéré. La conséquence à cela est que, même si la topologie du graphe peut rester la même pour tous les objectifs, le calcul des distances et la valuation des arêtes seront quant à eux différents selon l'objectif. Le reste de cette section propose une manière de calculer ces distances pour chaque objectif.

3.4.1 Similarité

C'est l'un des facteurs humains les plus importants des systèmes de recommandation, et il est pris en compte par l'immense majorité de ces systèmes. En effet, un des objectifs majeurs d'un système de recommandation est de proposer des ressources similaires à celles que l'utilisateur a aimé auparavant, ou proche des ressources qu'il consultait récemment. La similarité est une caractéristique bien connue et largement utilisée dans la littérature . Nous adhérons au principe selon lequel un système de recommandation ne peut pas la négliger et doit l'inclure pour produire des recommandations satisfaisantes. Cependant, la similarité ne devrait pas non plus être la pierre angulaire de chaque système de recommandation comme elle l'a été par le passé. Le but de cette colonie est donc de trouver une liste avec des éléments aussi proches que possible de ce que l'utilisateur apprécie ou est en train de consulter. Il existe de nombreuses méthodes pour calculer la similarité entre deux ressources. D'une manière générale, les ressources d'un domaine sont décrites par des métadonnées numériques, textuelles, binaires, etc. Le format de ces métadonnées est entièrement dépendant de chaque domaine applicatif. Nous avons décidé de généraliser ce problème et de le considérer comme une comparaison entre deux vecteurs. Nous avons ensuite décidé d'utiliser la mesure de la similarité cosinus entre ces deux vecteurs de par sa popularité dans la littérature scientifique et sa pertinence quand aux données dont nous disposons [Su and Khoshgoftaar, 2009]. La manière d'implémenter cette similarité d'un point de vue applicatif dépend des métadonnées disponibles. Dans le cas de données non numériques, il est possible d'obtenir des valeurs numériques de similarité à partir de listes ou de valeurs qualitatives [l'Huillier, 2018]. De même, d'autres manière de calculer la similarité peuvent être utilisées en fonction de leur pertinence par rapport au domaine applicatif. Dans notre contexte généraliste, et afin de déterminer la valeur de la distance d associée à chaque arête du graphe, nous avons tout d'abord calculé la similarité cosinus entre les deux ressources représentées par les sommets. Plus formellement, pour une arête (i, j), la similarité de ses deux ressources s_{ij} est calculée avec la similarité cosinus entre les deux vecteurs des caractéristiques descriptives C des ressources i et j, comme montré dans l'Équation 3.2.

$$s_{ij} = \frac{1}{sim(C_i, C_j)} \tag{3.2}$$

où C_i représentent les caractéristiques de la ressource i. Le terme de caractéristiques est générique et peut inclure toutes les métadonnées exploitables de domaine applicatif sur une ressource. Comme expliqué plus haut, ces caractéristiques sont donc dépendantes de chaque base de données et de chaque domaine d'application. Nous avons ici généralisé ces données avec la formalisation suivante : chaque ressource (ou item) i du domaine d'application est décrite par n caractéristiques c comme suit : $C_i = \{c_1, c_2, \ldots, c_n\}$.

Dans l'exemple donné plus haut avec les deux tableaux de van Gogh, nous avons expliqué que la distance séparant deux sommets sur le graphe représentait intuitivement leur "éloignement" selon l'objectif considéré. Ainsi pour la similarité, deux sommets représentant des ressources similaires verront l'arête les reliant posséder une faible distance, et inversement. Le but est ici d'être le plus proche possible du fonctionnement du modèle initial où une petite distance entre deux sommets représentent effectivement une petite distance physique entre ces deux points (par exemple deux points géographiques proches l'un de l'autre). La similarité cosinus produit cependant une valeur comprise entre $[0; 1]$, ne correspondant pas à la valuation des arêtes dans le modèle initial. Afin de convertir cette similarité $s \in [0; 1]$ en distance $d \in [1; +\infty]$, nous avons simplement utilisé l'inverse multiplicatif $d = \frac{1}{s}$. Ainsi, et à l'instar du modèle initial de Dorigo, une distance proche de 1 associée à une arête (i, j) signifie que les deux ressources i et j sont similaires.

3.4.2 Diversité

Ce facteur est souvent présenté en même temps que la similarité étant donné qu'ils sont tous deux liés à la distance, ou à la corrélation, entre les ressources consultées par l'utilisateur et ses recommandations. Mais, contrairement à la similarité, la diversité illustre à quel point deux éléments sont dissemblables l'un par rapport à l'autre. La similarité et la diversité se complètent dans la mesure où elles sont toutes deux nécessaires pour adapter le système de recommandation aux besoins des différents utilisateurs [Jones, 2010]. Nous avons choisi d'utiliser l'interprétation classique consistant à considérer la diversité comme l'inverse de la similarité. La diversité div_{ij} entre deux ressources i et j est donc obtenue en calculant l'inverse de la similarité entre i et j, comme indiqué dans Équation 3.3. De la même manière que pour la similarité, nous avons ensuite utilisé l'inverse multiplicatif de la diversité pour obtenir une distance $d \in [1; +\infty]$.

$$div_{ij} = \frac{1}{1 - sim(C_i, C_j)} \tag{3.3}$$

Toutefois, cette seule mesure ne capture pas l'ensemble de la diversité d'une séquence. Des travaux de la littérature ont été menés sur les séquences et les différentes manières de les caractériser. On peut citer la diversité relative (*RD : Relative Diversity*) qui permet de mesurer l'apport de chaque ressource ajoutée à une séquence existante en cours de construction, ou encore la similarité intra-liste (*ILS : Intra-List Similarity*), qui est la moyenne des similarités cosinus de toutes les ressources composant une séquence [Ziegler et al., 2005]. Nous n'avons pas utilisé ces deux mesures à ce stade de notre modèle, mais nous les exploiteront afin de mesurer les performances de notre modèle.

3.4.3 Nouveauté

Ce facteur humain représente les ressources encore non connues de l'utilisateur. Il peut s'agir de nouvelles ressources récemment ajoutées au système ou de ressources plus anciennes mais

moins populaires, que l'utilisateur n'a pas consultées. Selon le domaine d'application et/ou selon l'ancienneté de l'utilisateur, les ressources non consultées peuvent ne constituer qu'une faible partie des ressources totales (e.g. dans un petit musée ne possédant que quelques dizaines d'œuvres) comme elles peuvent aussi représenter l'immense majorité des ressources totales (e.g. sur des sites d'écoute de musiques en ligne qui ont souvent des catalogues de plusieurs centaines de milliers de musiques disponibles). La nouveauté ne doit cependant pas être confondue avec la diversité, car les ressources considérées comme nouvelles peuvent être soit similaires soit différentes de celles que l'utilisateur aime habituellement. La nouveauté est une caractéristique importante dans un système de recommandation car elle permet d'éviter un désintérêt potentiel des utilisateurs anciens. En effet, après un certain temps passé à produire des recommandations, un système peut avoir tendance à enfermer l'utilisateur dans les mêmes recommandations, à l'instar d'un algorithme d'apprentissage automatique trop bien entraîné[38]. Ce problème dans les systèmes de recommandation résulte en une trop grande prévisibilité des ressources recommandées [Vargas and Castells, 2011]. L'introduction de la nouveauté permet, entre autres, de mitiger ce problème et de produire des recommandations inattendues à l'utilisateur.

Afin de déterminer si une ressource est nouvelle ou pas pour un utilisateur en particulier, nous avons utilisé le travail de Zhang [Zhang, 2013] qui a défini la nouveauté comme une mesure composée de trois caractéristiques :

1. *Unknown* (notion d'inconnue) : représente si la ressource est connue ou non de l'utilisateur ;
2. *Satisfactory* (satisfaction) : représente si la ressource est aimée ou non par l'utilisateur ;
3. *Dissimilarity* (dissimilarité) : représente si la ressource est dissimilaire aux autres ressources connues par l'utilisateur.

L'auteur a ensuite proposé d'évaluer formellement la nouveauté d'une ressource i pour l'utilisateur u en proposant l'Équation 3.4 :

$$Novelty(i,u) = p(i|unknown,u) \cdot dis(i, pref_u) \cdot p(i|like,u) \tag{3.4}$$

où $p(i|unknown,u)$ est la probabilité que l'utilisateur u ne connaisse pas la ressource i, $dis(i, pref_u)$ est la dissimilarité entre i et les ressources consultées par l'utilisateur, et $p(i|like,u)$ est la probabilité que l'utilisateur u apprécie la ressource i. Cette équation est intéressante mais, dans le contexte de ce travail, les notions de satisfaction et de dissimilarité de l'utilisateur envers la ressource i sont proches des autres objectifs de notre modèle, à savoir respectivement pour la satisfaction par l'objectif de diversité et pour la dissimilarité par les deux objectifs de similarité et de préférences. Nous avons donc décidé de réduire l'Équation 3.4 à l'Équation 3.5 afin d'éviter une redondance entre les objectifs :

$$p(i|unknown,u) = -log(1 - pop_i) \tag{3.5}$$

où pop_i est la popularité de la ressource i. La notion de popularité est ici volontairement laissée générale car largement dépendante du domaine d'application et des ressources considérées. En effet, la popularité peut prendre la forme d'un nombre de vues, d'un nombre d'écoutes, d'une valeur calculée de manière interne par des sites, d'une comparaison entre les consultations des

38. C'est le problème statistique classique de l'*overfitting*, ou du surentrainement, souvent rencontré dans le domaine de l'apprentissage automatique.

ressources, etc. On peut toutefois considérer que cette valeur sera généralement numérique.

Il est important ici de préciser un point de différence important avec les deux autres objectifs qui ont été vu jusqu'à présent. Dans les équations ci-dessus, on peut en effet remarquer que le calcul de la distance se fait non plus en considérant deux ressources mais uniquement avec une ressource et le profil de l'utilisateur. Ainsi, une arête du graphe ne représente plus la distance entre les deux sommets la composant mais la distance nécessaire pour atteindre le sommet selon chaque utilisateur. Autrement dit, là où jusqu'à maintenant la distance associée à l'arête (A, B) représentait la distance entre le sommet A et le sommet B (la similarité ou la diversité entre A et B), cette distance représente maintenant la distance de l'utilisateur avec ce sommet B. Le sommet d'origine ne rentre plus en compte dans le calcul de la distance d'une arête mais uniquement son sommet de destination. Par conséquent, toutes les arêtes arrivant à un sommet unique posséderont la même valuation. Les calculs de valuation des arêtes du graphe sont donc moins nombreux, mais en contrepartie il est nécessaire de recalculer ces valuations pour chaque utilisateur étant donné qu'elles dépendent maintenant du profil de ce dernier.

3.4.4 Préférences

Le facteur que nous avons appelé préférences correspond aux ressources que l'utilisateur a particulièrement appréciées dans le passé. Cet objectif, bien qu'assez proche de la similarité au premier abord, exprime un autre aspect d'une recommandation satisfaisante que nous pensons important. En effet, le but n'est pas ici de recommander des ressources proches et potentiellement nouvelles de ce que l'utilisateur connaît déjà comme la similarité, mais bien de lui reproposer des ressources qu'il a déjà consultées et aimées par le passé. Autrement dit, cet objectif cherche à recommander à l'utilisateur les ressources les plus en accord avec ses préférences (ses ressources préférées), sans essayer de faire des prédictions. Par conséquent, le but de l'objectif de similarité est de proposer à l'utilisateur des ressources proches de ce que l'utilisateur a consulté ou consulte, mais il n'est pas possible de savoir si ce dernier va réellement apprécier ces recommandations. Il est par exemple commun d'apprécier une musique particulière d'un artiste et de ne pas aimer les autres musiques appartenant pourtant au même album. Toutes les musiques de cet album sont très similaires les unes aux autres (même artiste, même date de sortie, même album, probablement même genre), elles auraient donc de très hautes valeurs de similarité entre elles, mais l'utilisateur qui se verrait proposer le reste de l'album jugerait ces recommandations non satisfaisantes. L'objectif de préférences favorisera quant à lui les ressources connues et aimées par l'utilisateur et pourrait donc recommander dans cette situation uniquement la musique que l'utilisateur apprécie.

Le but des agents de la colonie préférences est donc de trouver un chemin où les ressources sont connues et aimées de l'utilisateur. On peut d'ores et déjà lister deux conséquences à ce fait. Tout d'abord, nous nous retrouvons dans le même cas de figure que pour l'objectif de nouveauté où une arête du graphe ne représente plus la distance entre ses deux sommets mais la distance nécessaire pour atteindre le sommet d'arrivée. Ensuite, il est nécessaire de définir des moyens de calculer les préférences des utilisateurs envers les ressources mises à leur disposition. Comme nous l'avons vu dans l'état de l'art, il y a deux grandes manières de calculer l'intérêt d'un utilisateur envers une ressource : avec des retours explicites (e.g. notes sur les ressources, bouton "j'aime" et "je n'aime pas",...), avec des retours implicites (e.g. temps passé à consulter les ressources, nombre de consultations,...) [39]. La nature des retours va donc énormément varier

39. Ces deux types de retours ne sont pas redondants mais plutôt complémentaires. Lorsque cela est possible,

selon le domaine d'application, mais nous pouvons tout de même formaliser la manière dont ils sont considérés. Chaque information collectée par le système concernant le comportement d'un utilisateur doit être prise en compte. Plus précisément, chaque interaction, directe ou indirecte, qu'un utilisateur peut avoir avec une ressource doit être intégrée dans l'estimation de son intérêt sur cette ressource. Soit C_u l'ensemble de tous les types d'interactions possibles qu'un utilisateur u peut avoir avec les ressources du système. $c_{u,i}$ représente alors la somme des interactions d'un même type c d'un utilisateur u sur une ressource i. Voici un rapide exemple toujours dans le domaine musical afin d'illustrer ces derniers points :

— C_u : écouter une musique, noter une musique, passer une musique avant la fin, mettre une musique dans sa playlist, ... ;
— $c_{u,i}$ pourrait par exemple renseigner sur le fait que : "l'utilisateur u à écouter la musique i 15 fois".

Afin d'agréger tous les types d'interactions possibles dans une unique valeur d'intérêt d'un utilisateur envers une ressource, nous avons utilisé la formule proposée par Castagnos [Castagnos and Boyer, 2006] et décrite dans l'Équation 3.6.

$$presumed\ interest_{u,i} = v_{min} + \frac{\sum_{c \in C} (w(c) \cdot c_{(u,i)})}{\sum_{c \in C} w(c)} \cdot \frac{(v_{max} - v_{min})}{c_{max}} \tag{3.6}$$

où $c(u,i)$ correspond aux valeurs normalisées données à la ressource i par l'utilisateur u à chaque type d'interaction c, $w(c)$ est le poids, ou l'importance, de chaque type d'interaction c, v_{min} et v_{max} sont les valeurs minimales et maximales attendues pour l'intérêt présumé et c_{max} est la valeur maximale que $c(u,i)$ peut prendre, peu importe le critère c considéré.

L'intérêt de cette colonie peut être discuté selon les domaines d'applications auquel le modèle est appliqué. L'exemple précédant portant sur la musique illustre bien l'intérêt d'intégrer un tel facteur humain dans un système de recommandations. En effet, il est rare de n'écouter qu'une seule fois une musique que l'on apprécie. Comme l'ont montré divers travaux, il est beaucoup plus courant d'écouter plusieurs fois ses musiques préférées, parfois de suite, parfois à différents moments de la journée, parfois sur plusieurs semaines, mois ou années, etc. [l'Huillier, 2018][Jones, 2010]. L'idée d'avoir une colonie dédiée à la revisite de ressources déjà consultées peut être étendue à d'autres domaines *a priori* moins évidents que la musique. Cet objectif de préférences peut par exemple être intéressant dans le domaine de l'e-éducation, où une ressource éducative peut avoir été consultée par un utilisateur sans pour avoir été comprise (après avoir passé une évaluation sur les thèmes de cette ressource par exemple). Dans ce cas, il peut être intéressant de recommander une nouvelle fois à l'utilisateur cette ressource afin qu'il puisse mieux la comprendre. En conclusion, l'objectif de préférences est certes plus spécifique à certains domaines que les autres objectifs plus généralistes présentés jusqu'à maintenant, mais nous pensons néanmoins qu'il a sa place dans le modèle, notamment pour les possibilités qu'il apporte dans ces domaines d'application spécifiques.

3.4.5 Progressivité

Le concept de la progressivité fait partie intégrante du but de notre modèle qui est de trouver et de recommander une bonne séquence de ressources. Contrairement à la majorité des systèmes

il est bénéfique de les prendre tous les deux en comptes pour estimer l'intérêt de l'utilisateur.

de recommandation actuels, nous ne souhaitons pas proposer qu'une simple liste à l'utilisateur final mais bien une séquence construite dans un but, possédant un début, une fin et une progression entre ses divers éléments. Afin de bien faire la différence entre une liste et une séquence de recommandation, voici 3 exemples dans 3 domaines d'applications différents :

— **Listes**

1. **Musique** : Les musiques les plus similaires aux dernières écoutes, voire à la toute dernière écoute, de l'utilisateur sans ordre particulier.
2. **Musée** : Des œuvres d'art correspondant à ce que l'utilisateur vient de voir. Si le critère principal de la création de liste est la similarité, les ressources de celles-ci ne seront pas forcément adaptées à la topologie du musée, ou à sa scénographie.
3. **E-éducation** : Des ressources éducatives sans ordre particulier (cours, vidéos explicatives, exercices) tous proches de ce que l'utilisateur était en train de consulter.

— **Séquences**

1. **Musique** : Une playlist de musiques proposant par exemple les titres d'un même artiste selon leur date de sortie, ou bien possédant un tempo de plus en plus rythmé, ou encore proposant une transition douce d'un style de musical à un autre.
2. **Musée** : Un chemin composé de plusieurs œuvres d'art, prenant compte de la distance physique entre chaque œuvre afin d'avoir un parcours adapté au musée, et en intégrant une progression dans les œuvres proposées (par date, styles, auteurs, etc.).
3. **E-éducation** : Une séquence composée de ressources éducatives adaptées au niveau actuel de l'utilisateur et à son objectif, intégrant des cours et des exercices proposant une progression mesurée (ni trop rapide ni trop lente) entre chaque ressources éducatives de la séquences.

Comme le montrent ces exemples, l'idée générale derrière la recommandation de séquences est de proposer une suite de ressources étant toutes en lien les unes avec les autres et possédant une progression sur un ou plusieurs critères. Cette progression se manifeste alors par de légères transitions entre chaque ressource afin de faire avancer petit à petit la séquence vers un but. Le but est donc ici de parvenir à construire une séquence dont la progression n'est ni trop lente, ni trop rapide, mais adaptée à l'utilisateur et à son but. Nous donnons à cet objectif le nom de "progressivité optimale" o. Cette métrique est utilisée afin de s'assurer que les agents de la colonie de progressivité trouvent un chemin dans le graphe où chaque sommet sélectionné offre une bonne progressivité par rapport à son prédécesseur et à son successeur. Pour ce faire, nous avons calculé la valeur de o en prenant en compte une ressource de départ et une ressource d'arrivée, représentée dans le graphe par deux sommets spécifiques.

Colonie de progressivité

Le concept de progressivité est lié à notre objectif de construire une bonne séquence de ressources. Contrairement à la majorité des systèmes de recommandation, nous ne souhaitons pas seulement recommander une liste de ressources, mais une séquence ayant un début, une fin et un ordre spécifique. Comme indiqué précédemment, dans un musée, une bonne séquence recommandée pourrait consister en des œuvres d'art liées d'une manière ou d'une autre (*i.e.* du

même artiste ou de la même période) et situées à proximité les unes des autres. Une telle séquence pourrait ensuite progresser lentement vers un autre artiste ou une section différente du musée. Notre objectif est donc de recommander une séquence qui ne progresse ni trop lentement ni trop rapidement : c'est ce que nous appelons la progressivité optimale o. Cette métrique est utilisée pour s'assurer que chaque ressource offre une bonne progressivité par rapport à ses prédécesseurs et successeurs. Pour ce faire, nous avons calculé o sur la base des premier et dernier éléments d'une séquence donnée pour un utilisateur spécifique comme le montre l'Équation 3.7.

$$o_n = \frac{c_{z_n} - c_{a_n}}{s} \tag{3.7}$$

où o_n est la valeur de progressivité optimale de la n-ième caractéristique d'une séquence spécifique, c_{z_n} est la n-ième caractéristique de la dernière ressource z de la séquence, c_{z_n} est identique pour la première ressource de la séquence et s est le nombre de ressources de la séquence. L'Équation 3.7 sera appliquée pour les caractéristiques de la ressource m, ce qui donne un vecteur de taille m de valeurs de progressivité optimales $(o_1, o_2, \cdots, o_m)$ pour la séquence spécifique. Ce vecteur sera ensuite utilisé dans le calcul de la distance d'un bord (i, j) pour la colonie de progressivité. Cette méthode implique de disposer de caractéristiques numériques pour appliquer l'équation, comme par exemple le tempo d'une piste musicale ou la date d'une œuvre d'art.

La valeur optimale o de la progressivité est alors utilisée pour calculer la distance d entre deux nœuds du graphe dans l'Équation 3.8 :

$$d_{ij} = \frac{\sum_{n}^{m} w_{o_n} \cdot \frac{c_{j_n} - c_{i_n}}{o_n}}{\sum_{n}^{m} w_{o_n}} \tag{3.8}$$

où m est le nombre de caractéristiques de la ressource considérée, o_n est la valeur de progressivité optimale de la n-ième caractéristiques de la ressource, w_{o_n} est le poids de la n-ième caractéristique pour l'utilisateur dépendant de ses préférences et c_{j_n} est la valeur de la n-ième caractéristique de la ressource j.

L'Équation 3.8 produit une valeur de distance $d_{ij} \in [-\infty; +\infty]$. En fonction de l'intervalle de d, nous pouvons déduire que :

- $d_{ij} \in [-\infty; 0]$: les caractéristiques de la ressource j vont dans la direction inverse par rapport à la valeur de progressivité optimale o ;
- $d_{ij} \in [0; 1]$: la progression est trop lente par rapport à la valeur o attendue ;
- $d_{ij} = 1$: la progression est idéale ;
- $d_{ij} \in [1; +\infty]$: la progression trop rapide.

Dans un souci d'homogénéisation des distances entre toutes les colonies et pour respecter le modèle initial ACS où d représentait une distance physique, nous avons souhaité garder la distance $d \in [1; +\infty]$. Pour ce faire, nous avons donc réalisé une opération de transformation lorsque $d_{ij} < 1$, présentée dans l'Équation 3.9 :

$$\text{Si } d_{ij} \in \begin{cases} [-\infty; 0], \text{ alors } d_{ij} = |d_{ij}| \\ [0; 1], \text{ alors } d_{ij} = \dfrac{1}{d_{ij}} \end{cases} \tag{3.9}$$

Intégration complémentaire de la progressivité dans le calcul de la distance parcourue des agents

Dans la section précédente, nous avons décrit le fonctionnement d'une colonie dédiée à la progressivité. Cependant, nous soutenons le fait que la progressivité ne doit pas être l'apanage d'une seule colonie, mais doit être présente dans toutes les colonies. La progressivité doit avoir une place prépondérante dans les séquences que notre modèle construit, car c'est une notion fondamentale dans la différence entre une liste de recommandation et une séquence de recommandation.

Afin de renforcer la progressivité des séquences construites par toutes les colonies, nous proposons une version modifiée de la mise à jour globale des phéromones décrite dans l'Équation 2.10. Dans l'algorithme initial ACS, L_{best} est la longueur du plus court chemin trouvé par un agent. Cette valeur est une simple somme de la distance associée à chaque arête du meilleur chemin. Cette valeur est ensuite utilisée dans le calcul déterminant la quantité de phéromones à ajouter sur chaque arête du chemin. Ainsi, plus un chemin est court, plus les arêtes le composant sont récompensées par une plus grande quantité de phéromones. Ce mécanisme est au cœur du renforcement de bonnes solutions et de l'émergence de comportements intelligents. Afin de maximiser les séquences possédant une progression constante, nous avons modifié le calcul de L_{best} pour intégrer une notion de progressivité. Pour ce faire, nous n'utilisons plus simplement la distance d'une arête, mais aussi sa dérivée, comme montré dans l'Équation 3.10 :

$$L_{best} = \sum_{e \in E} d_e + \frac{\sum_{e \in E} \partial d_e}{|E|} \tag{3.10}$$

où E est l'ensemble des arêtes du meilleur chemin et d_e est la distance associée à l'arête e. Avec cette équation modifiée du calcul de la distance totale d'un chemin, nous pénalisons les grandes différences de distance entre des arêtes successives tout en continuant de favoriser les chemins les plus courts. En d'autres termes, l'objectif des agents est maintenant de trouver des chemins à la fois courts et progressifs les uns par rapport aux autres.

3.5 Tactiques de fusion

Dans les sections précédentes, nous avons décrit 4 facteurs humains, 4 objectifs à optimiser par notre modèle, ainsi qu'un cinquième objectif de contrainte de construction des séquences. Chacun de ces 4 premiers objectifs est associé à une colonie de fourmis spécifique dont le but est de trouver un chemin dans le graphe. Par conséquent, après cette première étape, il existe autant de chemins, ou de séquences de ressources, que de colonies dans le modèle. Le but est cependant de ne proposer à la fin qu'une seule recommandation à l'utilisateur. Chaque séquence représente donc idéalement une partie de la recommandation finale de l'utilisateur. Afin de construire cette séquence finale, il était nécessaire de fusionner ces différentes séquences provenant des colonies en une séquence de recommandation unique. Pour résoudre ce problème, de nombreuses manière de procéder existent. Dans la suite de cette section, nous avons exploré deux tactiques de fusion

que nous allons présenter.

3.5.1 Colonie de fusion

La première tactique pour fusionner les différentes séquences produites précédemment a naturellement été d'utiliser le même modèle que pour la première étape, à savoir une colonie spécifique dont les agents se déplacent sur un graphe pour trouver une solution. Cette colonie est cependant différente des précédentes pour deux raisons principales.

Premièrement, le graphe sur lequel cette colonie opère n'a aucun intérêt à contenir l'ensemble de toutes les ressources existantes du domaine. Nous considérons ici que la première étape du modèle effectue une sélection d'un certain nombre de ressources spécifiques à chaque colonie. Ces ressources peuvent ensuite être utilisées pour créer un second graphe, possédant beaucoup moins de nœuds, sur lequel la colonie de fusion pourra chercher une solution. Nous avons utilisé toutes les ressources sélectionnées dans la première étape en tant que nœuds du nouveau graphe. Nous avons ensuite ajouté des arêtes à chaque paire consécutives de ressources dans les séquences de la première étape afin de conserver les meilleurs chemins trouvés par les agents. Enfin, nous avons ajouté des arêtes aléatoires en suivant la même méthode que décrite au début de ce chapitre dans la création du graphe principal en Équation 3.1. Le but de ces arêtes aléatoires est le même que précédemment, à savoir, permettre aux agents d'explorer de nouveaux chemins potentiels. Cette première tactique reprend donc les points principaux de la création du graphe principal.

Deuxièmement, et contrairement à la première étape où toutes les colonies avaient un but précis, la colonie de fusion n'a pas pour vocation d'optimiser une caractéristique spécifique. Au contraire, cette colonie a pour essence de regrouper au mieux dans une solution unique les résultats spécifiques trouvés auparavant. Pour ce faire, la colonie de fusion utilise l'ensemble des facteurs humains présentés précédemment dans une somme pondérée afin d'attribuer aux arêtes une distance d prenant en compte tous les objectifs. Le calcul de cette distance est présenté ci-dessous dans l'Équation 3.11.

$$d_{ij} = \sum_{col \in colonies} w(col) \cdot d_{ij}(col) \tag{3.11}$$

où $w(col)$ est le poids représentant l'importance estimée de l'objectif de la colonie *col* dans la recommandation finale. Nous proposons ici de calculer les poids moyens de chaque objectif (similarité, diversité, nouveauté et préférences) sur les dernières n sessions d'activité de l'utilisateur. Par exemple, un utilisateur n'ayant écouté que l'album *Wish you were here* des Pink Floyd durant ses dernières sessions d'écoute se verra attribuer des poids forts pour la similarité et les préférences. Inversement, les poids des objectifs de diversité et de nouveauté seront faibles.

Cette méthode permet d'obtenir l'importance relative de chacun des objectifs dans l'historique de consultations récent de l'utilisateur. En faisant varier n, il est aussi possible de faire varier la taille de l'historique pris en compte. Ainsi, une valeur de n faible concernera uniquement les dernières interactions de l'utilisateur avec le système tandis qu'une valeur de n élevée prendra en compte à la fois l'historique récent et plus ancien. La variation de la fenêtre d'historique peut ainsi apporter un comportement différent du modèle pour différents types d'utilisateurs (utilisateur récent ou ancien, peu actif ou très actif, etc.).

3.5.2 Fusion de séquences

Nous avons souhaité implémenter une deuxième tactique de fusion afin d'avoir un point de comparaison avec la colonie de fusion. Dans cette deuxième tactique, nous avons également calculé le poids $w(col)$ de chaque objectif sur l'historique n de l'utilisateur, de la même manière que dans l'Équation 3.11. Ces poids ont ensuite été utilisés dans la construction d'une séquence finale ressource par ressource. Pour ce faire, nous avons réunis toutes les ressources composant les séquences des colonies spécialisées dans un ensemble, à la manière d'un sac de mots. Puis, à chaque itération, l'apport de chacune des ressources restantes à la séquence finale a été mesuré et la meilleure ressource a été ajoutée. Les itérations sont stoppées lorsqu'il n'est plus possible d'améliorer la séquence finale. De la même manière que pour la tactique précédente, l'apport de chaque ressource à la séquence de recommandation peut être mesuré de multiple manières. De la même manière que dans l'Équation 3.11, nous avons choisi de calculer l'importance de chaque objectif de la séquence finale en construction, puis de choisir les ressources permettant de s'approcher des poids correspondant à l'historique n de l'utilisateur.

3.5.3 Conclusion

Dans ce chapitre, nous avons présenté le fonctionnement de notre modèle de recommandation multi-objectifs AntRS. Notre modèle s'inspire de l'algorithme multi-agents ACS de Marco Dorigo. Au lieu de trouver le plus court chemin dans un graphe à l'aide d'une colonie d'agents fourmis, nous utilisons 4 colonies en parallèle qui, chacune, maximise un aspect de la recommandation finale. Comme nous l'avons montré dans l'état de l'art, il est désormais nécessaire que les recommandations prennent en compte de multiples facteurs comme la diversité, la similarité ou la nouveauté. Il est tout aussi important pour les systèmes de recommandation d'être modulaires et génériques afin de s'adapter aux situations toujours plus complexes et aux domaines applicatifs toujours plus variés dans lesquels ils sont utilisés. Le choix de s'inspirer d'un modèle multi-agents est une réponse à ces défis de la recommandation moderne. Les systèmes multi-agents ont une faible complexité algorithmique et sont particulièrement adaptables par nature. Le modèle que nous avons proposé est de plus suffisamment générique pour s'adapter aux domaines applicatifs où il est possible de faire de la recommandation de séquences.

Dans la suite de ce manuscrit, nous allons appliquer notre modèle AntRS a un domaine applicatif et réaliser des expérimentations avec des données d'utilisation concrète afin d'étudier ses performances dans des conditions d'utilisation réelles.

Chapitre 4

Expérimentations et résultats

Sommaire

4.1 Introduction

Dans la section précédente, nous avons détaillé le modèle AntRS que nous avons développé pour répondre aux problématiques de cette thèse. Comme nous l'avons expliqué, AntRS se base sur le travail de [Dorigo and Birattari, 2011] qui a modélisé le comportement de fourragement des fourmis dans la nature. Son modèle, et le notre par extension, ont donc au cœur de leur fonctionnement le même mécanisme permettant à des agents téléonomiques de suivre et de déposer des phéromones sur un graphe. Afin de mettre en exergue le cœur de ces modèles qui sera par la suite enrichi, nous avons en premier lieu développé les modèle *Ant Colony System* de Marco Dorigo [Dorigo and Birattari, 2011] en Java avec une interface visuelle. Cela nous a permis de tester son fonctionnement dans diverses conditions expérimentales contrôlées, ainsi que de tester ses limites. Nous avons ensuite implémenté l'ensemble de notre modèle AntRS en utilisant la base déjà développée précédemment puis nous avons réalisé les expérimentations principales de cette thèse à l'aide d'une base de données décrivant les habitudes de personnes écoutant de la musique.

Ce chapitre est constitué de deux sections principales. Dans la première section, nous allons d'abord nous intéresser à l'implémentation et au fonctionnement expérimental du modèle ACS de Dorigo. Cette section mettra en exergue les points importants permettant le bon fonctionnement du modèle classique. À l'aide d'expérimentations simples, nous reprendrons et expliquerons d'un point du vue pratique les notions introduites de manière théorique dans l'état de l'art lors de la présentation du modèle ACS. Ces premières expérimentations permettront au lecteur de mieux appréhender la suite du chapitre dans laquelle les notions abordées seront ré-utilisées et étendues. Dans la deuxième section de ce chapitre, nous nous intéresserons aux expérimentations principales de cette thèse portant sur notre modèle AntRS. Nous décrirons d'abord la base de données que nous avons utilisé pour tester notre modèle, puis l'ensemble des expérimentations faites et enfin les principaux résultats obtenus.

4.2 Expérimentations préliminaires portant sur le modèle ACS initial

4.2.1 Présentation et intérêts de ces premières expérimentations

La première étape de ce travail expérimental a consisté dans l'implémentation du modèle initial proposé par Dorigo sans modification. Le but était à la fois de vérifier si les résultats obtenus correspondaient à ce qui était attendu dans la littérature, et d'avoir par ailleurs un programme représentant une base fonctionnelle permettant de tester de nouveaux modèles plus complexes par la suite. Nous avons donc commencé par implémenter le modèle *Ant Colony System* décrit par Dorigo dans son article [Dorigo and Birattari, 2011]. Comme nous avons déjà décrit en détail ce modèle dans la Section 2.4.3 de l'état de l'art, nous ne reviendrons pas sur les équations régissant le fonctionnement d'ACS. Cependant, nous allons plutôt nous intéresser ici aux résultats obtenus lors d'expérimentations connues de la littérature scientifique. Le choix du langage de programmation dans lequel a été réalisé l'implémentation et les expérimentations durant toute cette thèse a été *Java*. Ce choix se justifie par 3 principales raisons :

1. La connaissance avancée de ce langage ;
2. les bibliothèques [40] disponibles correspondant aux besoins de la thèse ;

40. Ensemble de ressources (fonctions, algorithmes spécifiques, documentation) permettant la création accélérée de programmes.

3. Le fait que Java soit un langage de programmation orienté objet, qui est un paradigme de programmation particulièrement adapté pour l'implémentation d'agents (chaque agent peut être représenté par un objet Java).

Pour rappel, l'algorithme général des *Ant Colony Optimization* servant de base pour toutes les différentes versions proposées par Dorigo (*Ant System, Ant Colony System, Ant-q, ...*) est synthétisé dans l'Algorithme 3 ci-dessous.

Algorithm 3 Métaheuristique *Ant Colony Optimization*

```
Initialisation des paramètres et des phéromones
while conditions d'arrêts non atteintes do
    Construction des solutions
    Recherche locale (optionnelle)
    Mise à jour des phéromones
end while
```

À partir de l'algorithme général *Ant Colony Optimization*, voici le fonctionnement de sa variante, *Ant Colony System*, synthétisé dans l'Algorithme 4 ci-dessous.

Algorithm 4 Algorithme *Ant Colony System*

```
Initialisation des paramètres et des phéromones
while conditions d'arrêts non atteintes do
    while toutes les fourmis n'ont pas atteint le but do
        while tous les nœuds n'ont pas été pris en compte do
            Appliquer la règle de transition et choisir le prochain nœud à atteindre
            Appliquer la mise à jour locale des phéromones sur l'arête traversée
        end while
    end while
    Mise à jour globale des phéromones
end while
```

Cet algorithme ACS peut aussi être décrit sous la forme d'un diagramme de flux, comme l'a proposé [Schlunz, 2011] dans la Figure 4.1. Le format du diagramme de flux permet de bien visualiser le fonctionnement de l'algorithme et notamment ses boucles itératives imbriquées : *tant qu'un critère de fin n'a pas été trouvé ; tant qu'il reste des fourmis à déplacer ; et tant qu'il reste des nœuds à explorer pour une fourmi.*

Après avoir implémenté en Java l'algorithme *ACS* décrit ci-dessus, nous avons voulu tester son fonctionnement. À cette fin, nous avons repris les expériences des ponts binaires que nous avons présenté dans la section 2.4.4 de l'état de l'art. Pour rappel, ces types d'expérimentations consistent à placer des fourmis dans un dispositif comportant un point de départ (la "fourmilière"), un point d'arrivée (la "nourriture") et deux chemins y menant. Ces chemins peuvent avoir la même distance ou des distances différentes selon le type d'expérimentation. Le but est ensuite d'observer le comportement des fourmis sur quelques dizaines de minutes afin de déterminer si un chemin est plus emprunté qu'un autre. La Figure 4.2 présente deux exemples de configurations possibles de cette expérience utilisée sur de vraies fourmis. Le but étant bien sûr d'observer le comportement de groupes de fourmis et leurs capacités à trouver le chemin le plus court vers

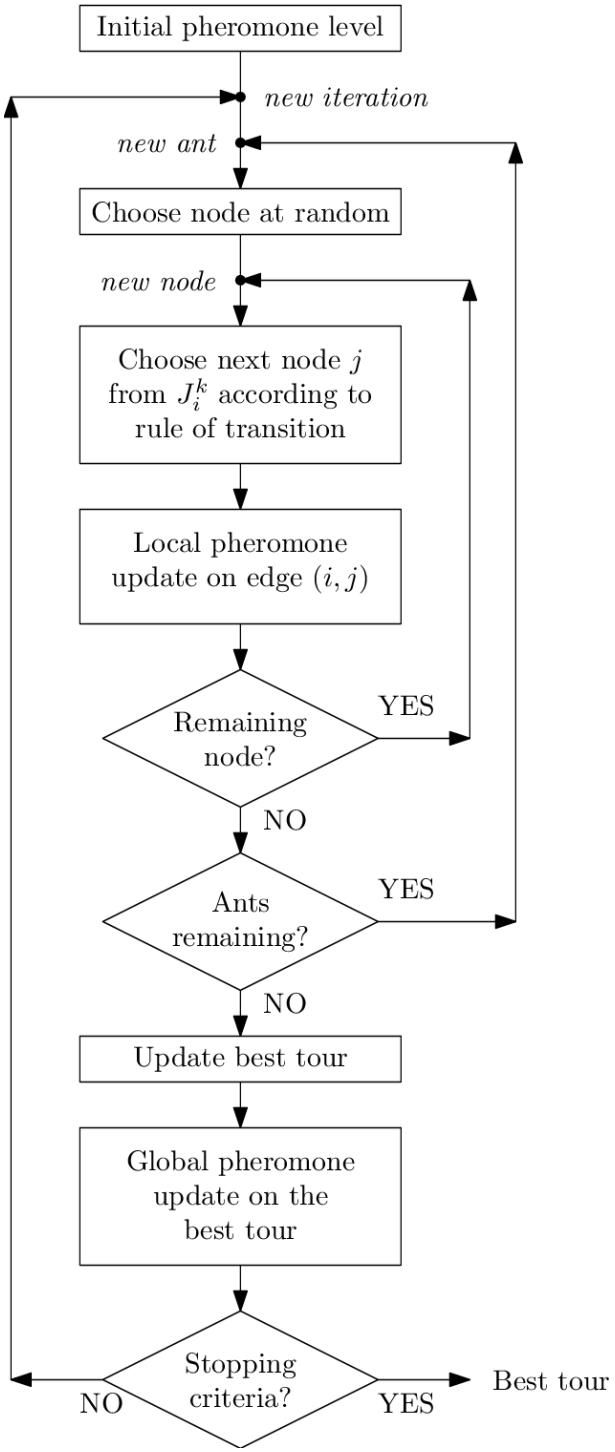

FIGURE 4.1 – Diagramme de flux représentant le déroulement de l'algorithme ACS, proposé par [Schlunz, 2011].

une source de nourriture dans un environnement inconnu.

Nous revenons sur ces expérimentations dans ce chapitre de la thèse, après les avoir déjà

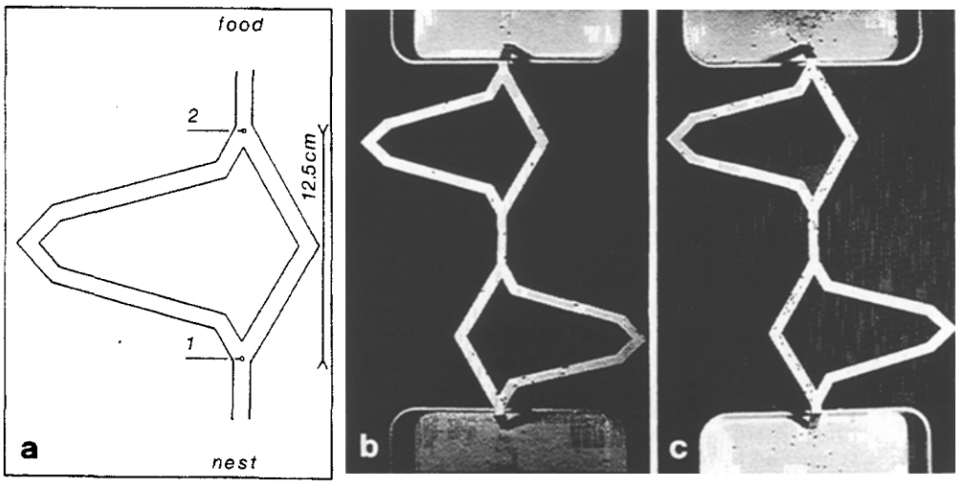

FIGURE 4.2 – Deux exemples de configurations expérimentales différentes permettant chacune de tester la capacité des fourmis à trouver le plus court chemin dans un evironnement [Goss et al., 1989].

mentionnées dans l'état de l'art, car elles peuvent être utilisées pour vérifier le comportement du modèle ACS que nous avons implémenté. En effet, ces expérimentations sont simples à mettre en œuvre et les résultats attendus sont connus. De plus, le modèle ACS possède plusieurs métavariables permettant chacune de moduler le comportement de certaines parties du système. Le paramétrage de ces métavariables influe de manière importante sur les résultats potentiels du modèle, il est donc important de bien comprendre le rôle de chacune d'entre elles et de les tester extensivement pour maximiser les capacités du modèle. Voici toutes les métavariables du modèle ACS :

1. $\tau_0 \in [0;1]$: valeur initiale des phéromones disposées sur chaque arête du graphe avant le premier passage des agents ;
2. $\alpha \in [0;1]$: importance des phéromones lorsque l'agent fourmi choisit d'une nouvelle arête à explorer. Plus la valeur de ce paramètre est élevée, plus l'agent privilégiera les arêtes avec un niveau élevé de phéromones ;
3. $\beta \in [0;1]$: importance de l'heuristique lorsque l'agent fourmi décide d'une nouvelle arête à explorer. Plus la valeur de ce paramètre est élevée, plus l'agent privilégiera les arêtes possédant une bonne heuristique ;
4. $\rho \in [0;1]$: paramètre gérant l'évaporation des phéromones sur les arêtes du graphe. Plus la valeur de ce paramètre est élevée et plus les phéromones s'évaporent rapidement entre chaque passage d'agent ;
5. $Q_0 \in [0;1]$: paramètre permettant d'équilibrer la part d'exploration et d'exploitation biaisée du modèle. Plus la valeur de ce paramètre est élevée et plus les agents auront tendance à exploiter les arêtes où beaucoup d'autres agents sont déjà passés. Inversement, plus la valeur de ce paramètre est basse et plus les agents auront tendance à explorer de nouveaux chemins encore peu visités.

En plus de ces 5 métavariables utilisées dans le modèle théorique et les équations d'ACS, d'autres métavariables sont inhérentes au fonctionnement du modèle en pratique. On peut ainsi dénombrer 2 autres métavariables :

6. a : nombre d'agents fourmis se déplaçant sur le graphe. Plus le nombre d'agents est élevé et plus l'intelligence collective du système basée sur les phéromones se développera. Un grand nombre d'agents allonge cependant la durée de l'exécution du programme ;
7. n : nombre d'itérations du programme. Lors d'une itération, chaque agent se déplace sur le graphe et produit un parcours complet. Plus le nombre d'itérations est grand et plus les agents ont de temps pour renforcer l'émergence d'une solution (un chemin particulier sur le graphe), mais plus l'exécution prendra du temps.

L'intérêt de mettre en place les expérimentations des ponts binaires est donc double :

1. Vérifier le comportement du modèle ACS implémenté en Java sur des expérimentations connues ;
2. Observer l'impact des 7 métavariables listées ci-dessus sur le comportement et les résultats du modèle, dans le but de mieux appréhender les expérimentations principales de cette thèse qui seront présentées dans la suite de cette section.

4.2.2 Expérimentations préliminaires sur le modèles ACS

Expérimentation 1.1 : ponts binaires de même longueur

La première expérimentation avait pour but de tester le graphe le plus simple possible, à l'image du dispositif de l'expérimentation de Deneubourg dont nous avons déjà parlé dans l'état de l'art et qui est rappelé avec la Figure 4.3 : un point de départ, un point d'arrivée et deux chemins possibles strictement équivalents en terme de distance à parcourir.

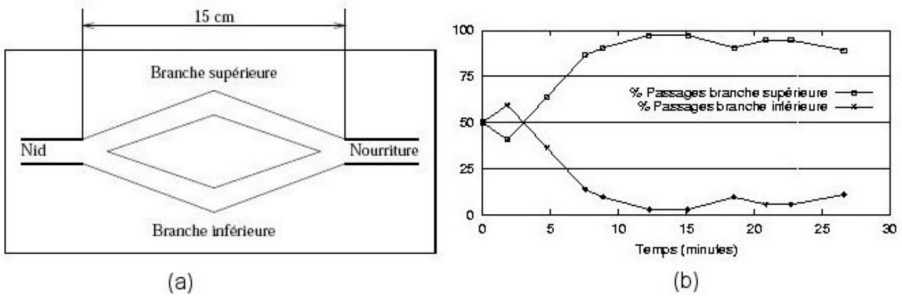

FIGURE 4.3 – Illustration de l'expérience des ponts binaires par Jean-Louis Deneubourg (a). Après un certain temps, les fourmis choisissent majoritairement une des deux voies possibles (b) [Deneubourg et al., 1990].

Avec le passage successif des fourmis, un chemin va petit à petit être privilégié et renforcé via les phéromones déposées par ces dernières. Au bout d'un certain temps, un des deux chemins est emprunté par la majorité des fourmis même si quelques-unes d'entre elles continuent toujours d'emprunter l'autre chemin au cas où celui-ci deviendrait plus avantageux. Comme les chemins restent strictement identiques dans cette expérience, le choix final du chemin n'est pas important.

Afin de pouvoir mieux apprécier le fonctionnement du modèle en temps réel, nous avons implémenté une interface graphique permettant d'afficher le graphe, ses nœuds, ses arêtes et

toutes leurs informations. Cette interface a été développée à l'aide d'une bibliothèque Java gratuite nommée *GraphStream*[41] offrant tout un ensemble de fonctions permettant l'affichage de graphes. Cette couche graphique se rajoute au-dessus de l'implémentation complète du modèle sans toutefois en modifier son fonctionnement d'aucune manière. L'interface graphique récupère simplement les données à afficher à l'initialisation (la disposition du graphe avec ses arêtes et ses nœuds) ainsi qu'à chaque itération du modèle pour mettre à jour les valeurs de phéromones à chaque arête. Une fois l'exécution terminée, le chemin final trouvé par les agents est coloré, permettant une rapide visualisation des résultats du modèle. Pour cette sous-section, les résultats des expérimentations seront présentés à l'aide de GraphStream.

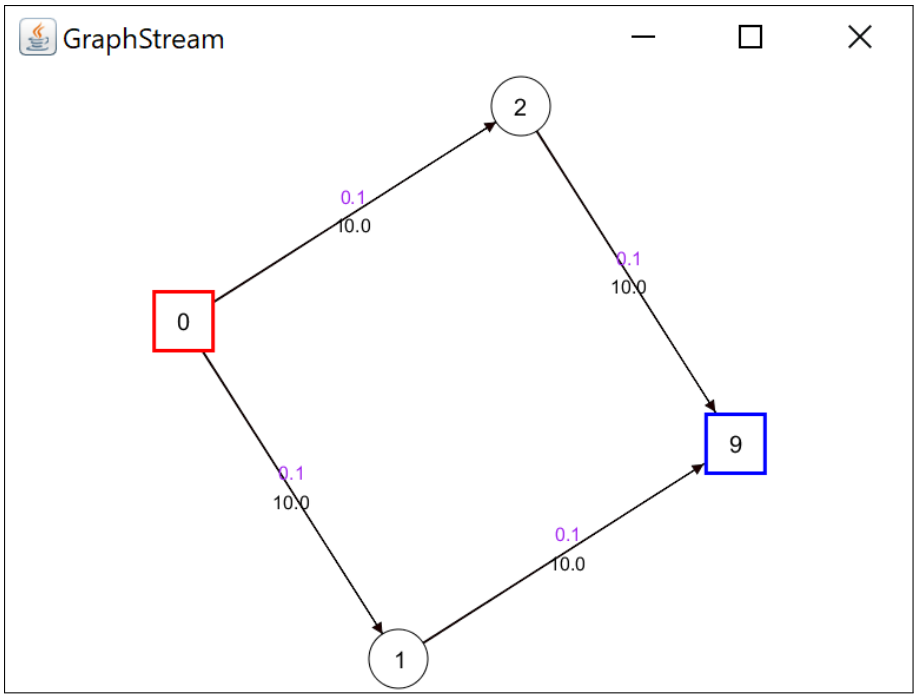

FIGURE 4.4 – Représentation graphique de l'Expérimentation 1.1 menée sur le modèle ACS grâce à la bibliothèque Java *GraphStream*. Le nœud représenté par un carré rouge correspond au départ ; le nœud représenté par un carré bleu représente l'arrivée ; les nœuds ronds noirs représentent des points de passage possible ; les nombres noirs accolés aux arêtes représentent la distance pour parcourir l'arête ; les nombres violets représentent le taux de phéromones déposées sur l'arête.

La Figure 4.4 est une capture d'écran de la fenêtre graphique construite avec les fonctionnalités offertes par *GraphStream* montrant la configuration de l'Expérimentation 1.1. On y voit notamment l'ensemble du graphe avec ses 4 nœuds distinguables par leur numéro d'identification $\{0, 1, 2, 9\}$. Le nœud 0 représente le point de départ où les agents sont initialement placés, tandis que le nœud 9 représente le nœud d'arrivée où les agents atteignent leur but et cessent de se déplacer. Au niveau des arêtes du graphe, on peut distinguer deux informations : en violet, la quantité de phéromones τ présente sur l'arête ; en noir, la distance d à parcourir pour traverser l'arête. Dans cette configuration, l'heuristique η_{ij} associée à chaque arête i, j est calculée de manière classique uniquement avec la distance d_{ij}, à savoir $\eta_{ij} = \frac{1}{d_{ij}}$. L'heuristique est donc ici

41. http://graphstream-project.org.

équivalente à la distance et n'est pas affichée dans la fenêtre graphique. La configuration représentée par la Figure 4.4 correspond à l'état d'initialisation du système, avant que les agents ne parcourent le graphe. Toutes les arêtes ont donc les mêmes valeurs : le taux de phéromones τ correspond au taux de phéromones initial τ_0, ici égal à $0,1$; et la distance est fixé à 10 unités. Les deux chemins possibles permettant aux agents de se déplacer du nœud de départ 0 au nœud d'arrivée 9 sont effectivement strictement identiques.

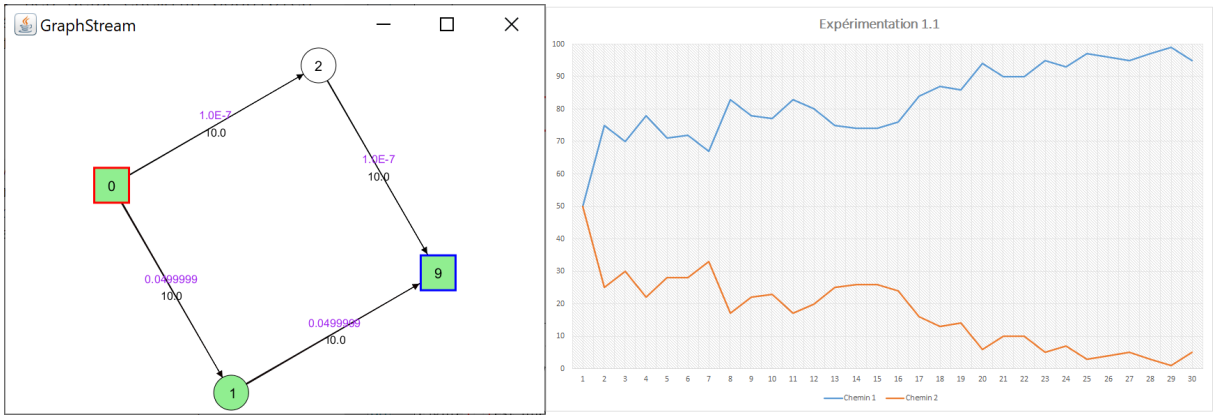

FIGURE 4.5 – Résultats obtenus après 30 itérations sur l'Expérimentation 1.1. À gauche, la configuration finale du graphe ; à droite, le pourcentage d'agents fourmis ayant empruntés les deux chemins.

L'Expérimentation 1.1 se déroule donc ainsi : 100 agents se déplaceront à chaque fois et chercheront un chemin du nœud 0 au nœud 9 durant 30 itérations. La Figure 4.5 présente les résultats de cette expérimentation. À gauche, la fenêtre graphique montre l'état du graphe à la fin des 30 itérations, on peut voir le chemin sélectionné par les agents passant par les nœuds $< 0, 1, 9 >$. À droite, le graphique présentant l'évolution du pourcentage d'agents prenant soit le chemin 1 ($< 0, 1, 9 >$) ou le chemin 2 ($< 0, 2, 9 >$). L'évolution des pourcentages est similaire aux résultats trouvés par Deneubourg avec de vraies fourmis (Figure 4.3). Une fois qu'un chemin est privilégié par rapport à un autre, entre 80 et 95% des agents y passent. Cet écart s'accentue de plus en plus au fil des itérations. Il est toutefois important de noter qu'il subsiste un faible pourcentage d'agents décidant quand même de prendre le chemin 2, permettant au système de s'adapter à un potentiel changement d'environnement. On voit ici l'intérêt de la métavariable Q_0, qui équilibre l'exploitation du chemin 1 en majorité mais qui autorise aussi quelques agents à explorer d'autres chemins du graphe.

Expérimentation 1.2 : ponts binaires de même longueur et influence des métavariables

L'Expérimentation 1.2 illustre l'importance des métavariables sur le comportement du système. La même topologie du graphe est conservée mais certaines métavariables sont modifiées : Q_0 et β. La première gère l'équilibre entre exploration et exploitation, tandis que la deuxième concerne l'importance de l'heuristique dans l'exploration. Contrairement à l'Expérimentation 1.1, Q_0 est ici biaisé pour privilégier l'exploration et β est augmenté afin de donner plus d'importance à l'heuristique par rapport au phéromones lors de l'exploration. Les agents ont donc plus

de probabilité d'explorer de nouvelles arêtes plutôt que d'emprunter des arêtes où beaucoup de phéromones sont déjà déposés. Dans cette configuration, l'exploration se base principalement sur l'heuristique qui est ici strictement identique quelle que soit l'arête puisque basée sur la distance. Les agents n'ont donc en théorie pas vraiment de direction claire quant au chemin à prendre. On peut voir le résultat de l'Expérimentation 1.2 sur la Figure 4.6 : les agents oscillent d'un chemin à l'autre selon l'itération sans qu'aucun des deux ne soit vraiment privilégié. Quand bien même un chemin serait plus emprunté que l'autre pendant une itération, les phéromones supplémentaires déposées sur les arêtes n'ont que peu d'influence sur la décision lors des prochaines itérations.

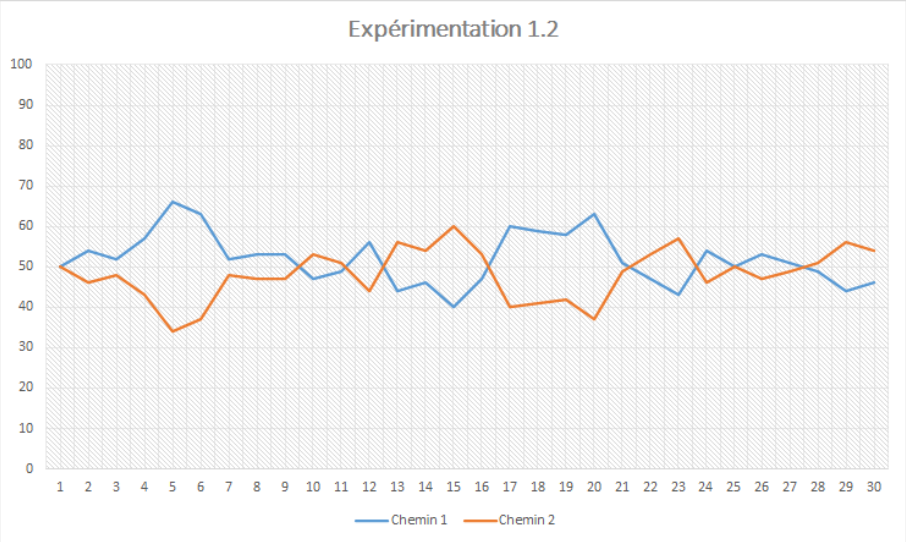

FIGURE 4.6 – Résultats obtenus après 30 itérations sur l'Expérimentation 1.2. Comme les agents se basent principalement sur l'heuristique qui est partout égale, ils oscillent entre les deux chemins sans en privilégier un.

Expérimentation 1.3 : ponts binaires de même longueur et influence des métavariables

L'Expérimentation 1.3, dont les résultats sont présentés par la Figure 4.7 montre qu'il est aussi possible d'obtenir l'inverse du comportement de l'Expérimentation 1.2. En donnant une forte importance aux phéromones et en favorisant grandement l'exploitation par rapport à l'exploration, les agents choisissent très rapidement un chemin et n'en changent plus. Au bout de quelques itérations seulement, 100% des agents prennent le même chemin. L'autre chemin abandonné finit par voir toutes ses phéromones s'évaporer et les agents n'ont plus aucun moyen d'y retourner.

Expérimentation 2 : ponts binaires avec chemins de longueurs différentes

Après avoir testé le comportement du modèle ACS sur un graphe où les chemins sont identiques dans l'Expérimentation 1, nous allons maintenant créer un graphe ayant la même configuration que dans la Figure 4.2. Le but est ici de tester la capacité des agents à imiter les fourmis dans la nature qui sont capables après quelques dizaines de minutes de privilégier les 2

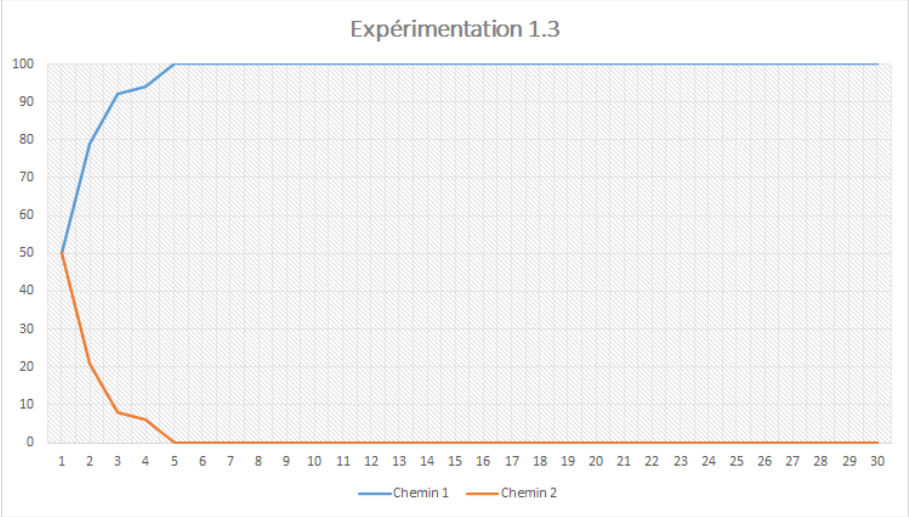

FIGURE 4.7 – Résultats obtenus après 30 itérations sur l'Expérimentation 1.3. Les agents exploitent ici les arêtes ayant le plus de phéromones. Très rapidement, tous les agents utilisent le même chemin.

arêtes courtes par rapport aux 2 arêtes longues. Nous allons de plus garder les mêmes valeurs de métavariables que pour l'Expérimentation 1.1, sauf pour la quantité initiale de phéromone τ_0 qui sera augmentée afin de rester au-dessus de 0. La Figure 4.8 montre la configuration du graphe directement après 30 itérations. Toutes les arêtes courtes avec une distance de 10 ont été sélectionnées tandis que toutes les arêtes longues avec une distance de 20 n'ont pas été utilisées.

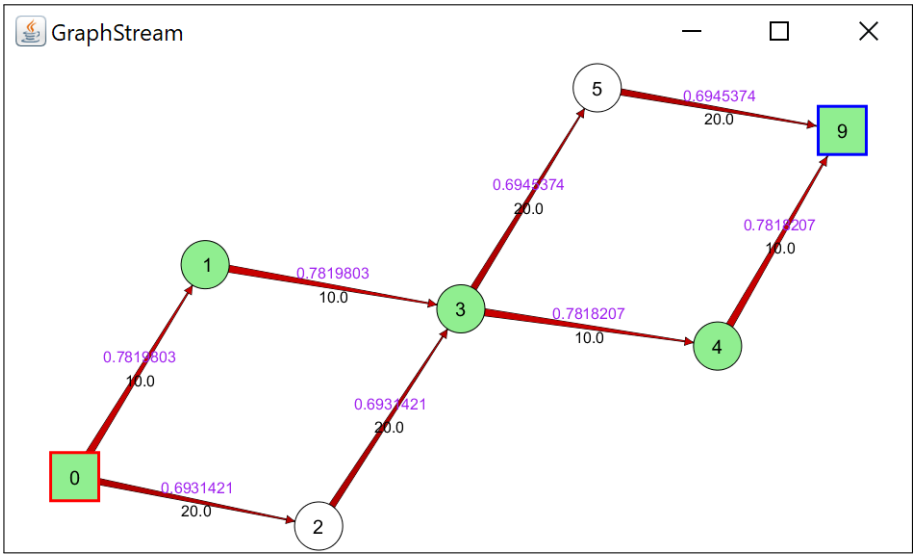

FIGURE 4.8 – Configuration finale du graphe pour l'Expérimentation 2 après 30 itérations. Les agents ont en majorité bien sélectionné le chemin le plus court $< 0, 1, 3, 4, 9 >$.

Expérimentation 3 : graphes aléatoires plus complexes

Dans les précédentes expérimentations, le graphe était le plus simple possible pour pouvoir jouer avec le comportement du modèle. Avant de passer à la section suivante, nous souhaitions tester les capacités du modèle sur des topologies de graphes plus complexes. Nous avons donc créé aléatoirement 10 graphes possédant chacun 20 nœuds et des arêtes aléatoires selon un degré moyen de 5, sur lesquels 100 agents se déplaceront sur 50 itérations. La Figure 4.9 présente le premier de ces 10 graphes après 30 itérations. On peut voir que le but est de trouver un chemin du nœud 0 au nœud 19 et les agents ont trouvé le chemin $< 0, 1, 4, 2, 19 >$. Ce chemin est effectivement le plus court avec une distance de 284, le deuxième meilleur chemin étant $< 0, 5, 6, 19 >$ avec une distance de 315.

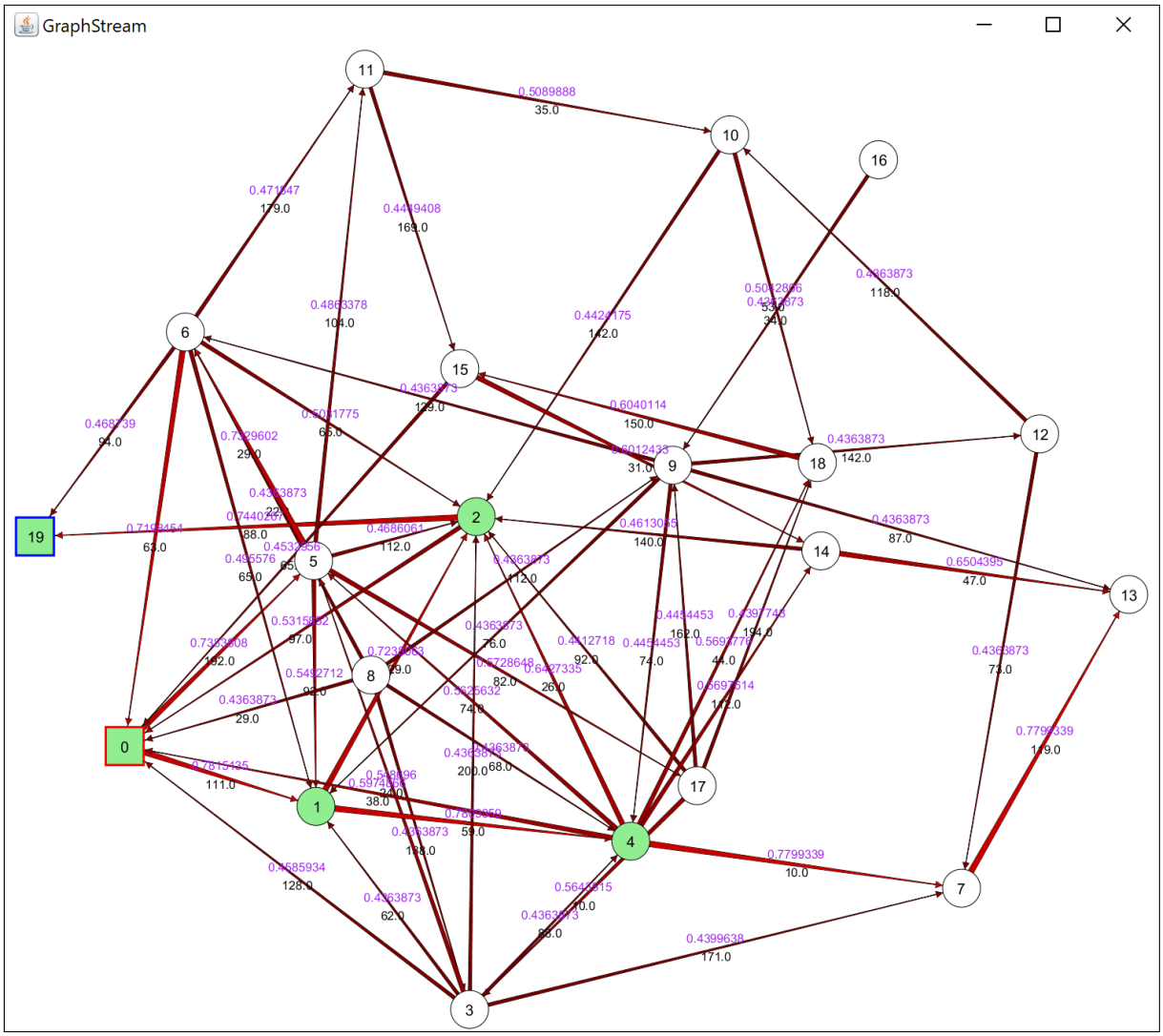

FIGURE 4.9 – Exemple de graphe de l'Expérimentation 3 à la fin des 50 itérations. Ces graphes sont générés aléatoirement.

Parmi les 10 graphes aléatoires, le modèle a trouvé le chemin le plus court pour 9 d'entre eux car, dans l'un des graphes, il n'était pas possible de rejoindre le nœud de fin 19. Autrement dit,

les agents ont trouvé le chemin le plus optimisé à chaque fois que cela était possible.

4.2.3 Conclusion

Dans cette première section, nous avons implémenté et testé le modèle ACS proposé par Dorigo. Malgré la relative simplicité des expérimentations réalisées, nous avons pu comparer et vérifier les résultats que nous avons obtenus avec ceux de la littérature scientifique du domaine. Outre ce point, nous avons aussi pu déduire quelques informations cruciales pour la suite de cette section :

— Les **métavariables jouent un rôle central** dans le comportement, les performances et les résultats du modèle ;
— **Les métavariables peuvent donner de bons résultats avec un type de graphe mais pas forcément avec un autre.** Il sera donc important de pouvoir les adapter aux graphes des expérimentations principales ;
— **L'équilibre de ces métavariables est relativement complexe à obtenir.** Même avec ces configurations simples de graphes et l'aide de l'interface graphique, chaque expérimentation a demandé un certain nombre d'essais avant de trouver des valeurs concluantes. De plus, comme ces métavariables sont propres à chaque expérimentation, l'état de l'art ne contient que très peu d'indications sur les valeurs à leur donner. Il sera donc important d'étudier ce problème pour les expérimentations principales de cette thèse.

Il est important de noter que, comme notre modèle AntRS est une extension du modèle ACS de Dorigo et reprend ses principes de fonctionnement, toutes les conclusions ci-dessus pourront être appliquées dans la section suivante où nous réaliserons plusieurs expérimentations avec le modèle AntRS.

4.3 Expérimentations sur le modèle AntRS

Dans cette section, nous allons maintenant décrire les expérimentations menées sur le modèle AntRS que nous avons conçu dans le cadre de cette thèse et décrit dans le chapitre 3 précédent. Nous n'allons pas re-détailler la théorie du modèle AntRS ici comme cela a déjà été fait dans le chapitre précédent, mais nous allons plutôt nous concentrer sur son application et sur ses comportements d'un point de vue expérimental. De la même manière que pour le modèle ACS dans la section précédente, il est important de procéder à quelques tests pour jauger notre modèle avant de lancer des expérimentations pouvant être longues. Toutefois, et contrairement à la section précédente, toutes les expérimentations qui vont être présentées ici ont été réalisées sur une base de données réelles. Avant de s'intéresser à ces expérimentations, nous allons donc présenter la base de données utilisée.

4.3.1 Base de données utilisée

Tout d'abord, et avant de discuter du choix des données, il est important de préciser pourquoi nous n'avons pas eu recours, comme dans la section précédente pour le modèle ACS, à des données simulées. L'explication de cette décision réside dans l'objectif visé par ces expérimentations. Les expérimentations réalisées précédemment avaient seulement pour but de tester le bon fonctionnement du modèle, son comportement et l'influence de ses métavariables. L'objectif des expérimentations que nous allons présenter maintenant est différent : proposer des séquences

de ressources correspondant aux préférences de l'utilisateur. Il serait théoriquement possible de simuler un grand nombre de ressources aléatoires possédant des caractéristiques elles aussi aléatoires. Cependant, nous souhaitons principalement évaluer la pertinence des séquences produites par notre modèle. Pour ce faire, il est nécessaire de disposer de données d'usage pour vérifier, *a posteriori*, si ces séquences sont susceptibles d'intéresser les utilisateurs. L'exploitation d'une base de données réelle, possédant ces données d'usages, était essentielle.

Choix du domaine applicatif

Le choix du domaine applicatif a été un point crucial pour pouvoir tester notre modèle dans de bonnes conditions. La recommandation de séquences restreint le choix du domaine en imposant certaines contraintes :

- Les utilisateurs doivent consulter régulièrement les ressources du domaine applicatif ;
- Lors d'une période d'activité, les utilisateurs doivent consulter un nombre suffisant (au moins une dizaine) de ressources dans un court intervalle de temps ;
- Par conséquent, les ressources doivent pouvoir être consultées et "consommées" rapidement. Ces ressources doivent aussi être disponibles en nombre suffisant.

En se basant sur les études de l'état de l'art explorant les algorithmes de fourmis ou la recommandation de séquences, deux domaines applicatifs sont majoritairement utilisés : les ressources éducatives ou les ressources multimédias. Le domaine éducatif permet de générer des séquences destinées aux apprenants comme le ferait un vrai professeur tandis que le domaine du divertissement multimédia est en plein essor depuis quelques années grâce à la démocratisation d'Internet haut débit et des smartphones permettant à tout un chacun de regarder des films ou d'écouter de la musique par exemple. Un troisième domaine a aussi été considéré car il a été le sujet de mon stage de fin de Master en collaboration avec le projet européen CrossCult [42] : le domaine muséal, autrement dit les séquences que créent les visiteurs d'un musée en se déplaçant physiquement dans un musée [Osche et al., 2016, Osche et al., 2019b].

Toutefois, même si ces trois domaines sont adaptés à notre application, la récupération d'une base de données assez importante pour réaliser des expérimentations ne pose pas les mêmes problèmes selon le domaine considéré :

- **Domaine éducatif** : dans ce domaine, les ressources sont des supports pédagogiques de natures différentes [43] et le but est de créer des séquences permettant à des utilisateurs d'atteindre des connaissances spécifiées. Il existe des bases de données publiques provenant d'universités ouvertes comme par exemple l'Université Ouverte des Humanités [44] mais ces dernières n'ont pas de profils utilisateurs assez fournis pour pouvoir les utiliser ici, notamment dûe à une connexion avec une session unique et anonyme sans profil. Comme il est préférable d'avoir plusieurs dizaines (voire centaines) d'entrées pré-existantes pour modéliser le graphe et les profils utilisateurs, il est nécessaire d'utiliser les données d'un site permettant la connexion de l'utilisateur et l'enregistrement de ses habitudes pendant une certaine durée ;

42. http://www.crosscult.eu : projet européen ayant pour but de modifier la manière dont les citoyens européens voient et s'approprient l'Histoire en créant de nouvelles interconnexions entre l'héritage culturel, des lieux de culture (musées, villes, etc.) et le point de vue des citoyens eux-mêmes.
43. Les supports pédagogiques peuvent inclure des cours écrits, oraux, des vidéos, des exercices, des tests, etc.
44. http://www.uoh.fr

— **Domaine muséal** : ce domaine est unique car il est le seul des trois à nécessiter de récolter des données de manière physique et non sur ordinateur. Comme nous l'avons vu dans l'état de l'art, des études existent dans ce domaine et ont récolté les positions des utilisateurs avec plus ou moins de précision, mais il n'existe à notre connaissance pas de base de données publique permettant de savoir ce que le visiteur regarde précisemment [45]. Or cette information est essentielle à notre modèle, car la position seule ne permet pas de déterminer ce que fait précisément l'utilisateur (par exemple, il peut s'assoir en face d'une œuvre mais regarder derrière lui, ou encore discuter avec quelqu'un sans prêter attention à l'œuvre devant lui, etc.). Autrement dit, sans savoir ce que regarde l'utilisateur, il n'est pas possible de savoir les ressources qu'il a consulté. Étant donné le sujet du stage effectué avant cette thèse, c'est sur ce problème précis que nous nous sommes d'abord concentrés. Cependant, devant le coût en temps de la mise en place d'une telle expérimentation, nous avons cherché des alternatives plus réalisables. Il est à noter qu'il existe aussi la possibilité d'un musée virtuel, où les visiteurs utilisent un ordinateur et un casque de réalité virtuelle pour se déplacer d'œuvre en œuvre dans un espace recréé pour simuler un musée. La création de l'ensemble des composants permettant cette expérimentation ainsi que le recrutement des visiteurs virtuels est cependant complexe et très coûteux en temps, ne changeant pas le problème principal de la récolte de données dans notre cas ;

— **Domaine du divertissement multimédia** : depuis le tout début des systèmes de recommandations, le domaine multimédia (musique, films, etc.) occupe une place majeure dans l'état de l'art. On peut citer par exemple la base de données publique MovieLens qui, depuis 1998, a été utilisée dans des milliers d'études sur la recommandation de films [Harper and Konstan, 2015]. La raison de cette omniprésence est la grande quantité de sites web permettant, souvent de manière gratuite, de regarder des vidéos, d'écouter de la musique, etc. Les données d'utilisation de ce genre de sites sont aussi relativement simples à récupérer et à exploiter. L'écoute de musique en ligne est spécifiquement intéressante dans le cadre de cette thèse car elle est propice à la consommation de multiples ressources à la suite dans une même session, offrant ainsi la possibilité de travailler avec des séquences. De plus, il existe plusieurs sites de diffusion de musique en ligne offrant la possibilité d'accéder à leur API et à certaines de leurs données, comme Deezer [46] ou encore Spotify [47].

Le choix s'est finalement porté sur le domaine de l'écoute de musique en ligne, d'une part pour la possibilité d'exploiter et de recommander des séquences de musiques, et d'autre part pour la possibilité d'obtenir une base de données de la part de nos partenaires dans l'industrie musicale. Notre modèle AntRS ayant la capacité de s'adapter à de nombreux domaines applicatifs, nous avons pu nous diriger vers une base de données existante afin de commencer les expérimentations. Amaury L'Huillier, ancien doctorant de l'équipe KIWI, a récupéré et exploité durant sa thèse une base de données musicale provenant de Deezer [l'Huillier, 2018]. Dans la section suivante, nous allons décrire les caractéristiques de cette base de données.

45. Bien que des solutions soient en cours d'études[Mokatren et al., 2018].
46. https://www.deezer.com
47. https://www.spotify.com

Description de la base de données

La base de données que nous avons utilisé se base sur le comportement d'utilisateurs de Deezer pendant un mois, entre le 5 décembre 2016 et le 5 janvier 2017. Durant ce mois, nous avons pu identifier :

— 3.561 utilisateurs uniques ;
— 1.871.919 écoutes enregistrées ;
— 178.910 musiques uniques écoutées.

En plus de ces données, nous avons calculé le nombre de sessions uniques durant ce mois. Une session est déterminée par un utilisateur se connectant sur le site, écoutant un certain nombre de musiques puis se déconnectant. Une session se poursuit tant que l'utilisateur écoute des musiques sans pause plus longues que 15 minutes. Nous avons ainsi pu identifier 91.468 sessions uniques composées en moyenne de 18,3 morceaux de musique. Ces sessions seront la base de travail nous permettant de construire le graphe de notre modèle AntRS et de proposer des recommandations. La Figure 4.10 présente en schéma la détection de sessions dans la base de données.

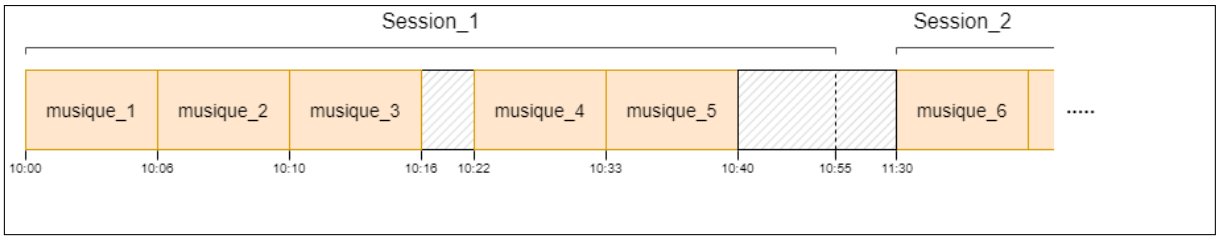

FIGURE 4.10 – Exemple de détection de sessions dans la base de données. L'heure à laquelle est lancée chaque musique (ici en bas dans le schéma) ainsi que sa durée sont enregistrées. Si l'utilisateur ne relance pas de musique pendant 15 minutes après la fin de sa dernière écoute, alors la session en cours est terminée et une nouvelle session est commencée à la prochaine musique lancée.

Deezer possède des métadonnées sur chaque musique qu'il est possible de récupérer et de coupler à d'autres API [48] (comme par exemple l'API de Spotify [49]) afin de les compléter si nécessaire. Sans présenter la base de données dans l'ensemble de ses détails, voici ci-dessous les caractéristiques potentiellement exploitables enregistrées pour chaque musique, que nous pouvons regrouper en trois catégories. Tout d'abord, nous disposons d'informations textuelles classiques :

— ***id*** : identifiant unique de la musique ;
— ***title*** : titre de la musique ;
— ***album*** : nom de l'album auquel appartient la musique ;
— ***artist*** : nom de l'artiste (ou du groupe) qui a créé la musique ;
— ***artistGenre*** : genre musical de l'artiste.

Ensuite, les API de descriptions de métadonnées musicales disposent d'un ensemble de valeurs numériques correspondant aux caractéristiques de chaque musique :

48. API : *Application Programming Interface*
49. https://developer.spotify.com/documentation/web-api/

- ***duration*** : durée de la musique en secondes ;
- ***acousticness*** : $\in [0; 1]$, indique à quel point la musique est "acoustique", c'est-à-dire si la musique est plutôt de type électronique ou plutôt classique (i.e. composée avec des instruments de musique à corde ou à vents par exemple) ;
- ***danceability*** : $\in [0; 1]$, indique à quel point la musique est adaptée à la danse ;
- ***energy*** : $\in [0; 1]$, indique à quel point la musique est énergique ;
- ***instrumentalness*** : $\in [0; 1]$, indique à quel point la musique possède ou non des parties vocales ;
- ***liveness*** : $\in [0; 1]$, indique si la musique a été enregistrée avec ou sans la présence d'un public ;
- ***loudness*** : $\in [0; 1]$, mesure la présence et la puissance des basses dans la musique ;
- ***speechiness*** : $\in [0; 1]$, mesure l'importance des paroles dans la musique s'il y en a ;
- ***tempo*** : rythme de la musique ;
- ***valence*** : indique si l'émotion générale de la musique est plutôt heureuse ou triste.

Enfin, Deezer étant un service de musique en ligne, il dispose des données de fréquentations des musiques les unes par rapport aux autres. Il est donc possible de connaître pour chaque musique et artiste sa popularité :

- ***trackRankDeezer*** : représente la popularité de la musique sur le site Deezer ;
- ***artistRankDeezer*** : représente la popularité de l'artiste sur le site Deezer.

La base de données possède aussi une table concernant les utilisateurs. Ces derniers sont bien sûr anonymisés avec un identifiant généré automatiquement. De plus, pour chaque chanson, Deezer peut connaître quelles actions l'utilisateur a effectué durant son écoute (s'il en a effectué) :

- ***idUser*** : identifiant unique de l'utilisateur ;
- ***idTrack*** : identifiant unique de la musique écoutée ;
- ***timestamp*** : date précise à laquelle l'utilisateur a lancé la musique ;
- ***skip*** : indique si l'utilisateur est passé à la musique suivante durant son écoute ;
- ***ban*** : indique si l'utilisateur a banni la musique écoutée de ses futures recommandations ;
- ***love*** : indique si l'utilisateur a aimé la musique durant son écoute.

Ces trois dernières informations sont importantes car elles représentent l'avis de l'utilisateur sur la musique qu'il écoute. On peut par exemple déduire d'un utilisateur qui passe la musique en cours (*skip*) qu'il ne l'apprécie pas, ou alors qu'il l'aime mais que ce n'est pas le bon moment pour l'écouter. Un utilisateur supprimant une musique de ses futures écoutes (*ban*) indique qu'il ne souhaite plus écouter cette musique, et ce quelles que soient les circonstances. Le Tableau 4.1 présente la fréquence à laquelle les utilisateurs se sont servis de ces fonctionnalités dans notre base de données.

Discussion sur la base de données

Dans cette section, nous avons justifié le choix d'une base de données musicale et décrit son contenu. Même si, initialement, nous nous sommes plutôt dirigés vers l'espace muséal et l'e-éducation, devant le temps et la complexité de la récupération de données dans ces deux domaines, nous avons finalement procédé au choix plus pragmatique du domaine musical dont

	Nombre d'occurrences	Pourcentage
"skip"	603773	32,25%
"ban"	3417	0,18%
"love"	14174	0,76%
Total	621364	33,19%

TABLE 4.1 – Tableau représentant le nombre de fois où un utilisateur a cliqué sur les boutons pour passer, bannir ou aimer une musique durant son écoute. On remarque qu'au total dans notre base de données, près d'un tiers des utilisateurs ont passé la musique qu'ils écoutaient.

une base de données était déjà disponible et prête à l'emploi. Le travail préliminaire sur les deux domaines applicatifs non sélectionnés n'a cependant pas été vain, car des projets en parallèle sur la création d'une base de données de visiteurs se déplaçant dans le musée des Beaux-arts de Nancy sont en cours de réalisation [Osche et al., 2019b][Castagnos et al., 2019].

Un autre objectif derrière la prise en compte de plusieurs domaines sur lesquels appliquer notre modèle est la généricité de notre approche. Ces trois domaines possèdent de nombreuses différences les uns par rapports aux autres. Le domaine des musées est un environnement physique où les distances entre les œuvres d'art et la foule sont des notions importantes à prendre en compte et unique à ce domaine. Le domaine de l'e-éducation, quant à lui, nécessite de prendre en compte l'objectif pédagogique de l'apprenant, son niveau ou encore sa vitesse d'apprentissage plutôt que les seules préférences qu'il est d'usage d'exploiter dans les systèmes de recommandation. Notre modèle AntRS a été conçu dans le but de pouvoir s'adapter à ces différentes spécificités. Ainsi, le fonctionnement du cœur du modèle reste le même quel que soit le domaine et seuls les objectifs des colonies, ainsi que le type des colonies, nécessite d'être adapté en fonction des objectifs importants du domaine et des caractéristiques de ses ressources. Pour illustrer ce dernier point très concrètement, les expérimentations qui suivent n'incluent pas la colonie de progressivité. Nous avons réalisé de nombreux tests avec cette colonie avec le corpus de données mais les résultats obtenus n'ont jamais été concluants. Cela s'explique par le fait que les séquences d'écoutes utilisateurs que nous avons recueilli ne possédaient elles-mêmes aucun motif détectable de progressivité selon les critères définis dans notre modèle. Nous avons donc décidé d'utiliser la modularité qu'offre notre modèle en enlevant cette colonie.

4.3.2 AntRS mono-objectif : expérimentations sur chaque colonie isolée

Après avoir détaillé la base de données, nous allons maintenant aborder la première expérimentation que nous avons réalisé avec celle-ci. Comme nous l'avons fait au début de ce chapitre avec le modèle ACS classique, nous avons jugé important de réaliser quelques expérimentations de tests afin d'apprécier le fonctionnement en pratique de notre modèle qui, jusqu'à maintenant dans ce manuscrit, n'a été présenté que de manière théorique. Pour rappel, notre modèle simule en parallèle plusieurs colonies de fourmis ayant chacune un objectif unique. Ensuite, un algorithme se charge de fusionner les différents résultats de l'ensemble des colonies pour ne proposer à l'utilisateur qu'une seule séquence de ressources. Avant de tester l'ensemble de notre modèle, nous avons voulu tester les colonies séparément afin d'observer si les résultats qu'elles produisent sont bien adaptés à leurs objectifs respectifs. Voici les 4 colonies que nous avons présentés dans le chapitre 3 et que nous allons mesurer ici :

1. Similarité ;

2. Diversité ;
3. Nouveauté ;
4. Préférences.

Afin de mesurer les performances de ces 4 colonies, nous allons les isoler et les lancer sur un graphe identique en tous points. La modularité d'AntRS, nous permettant de changer le nombre de colonies utilisées au vol à chaque exécution, est ici mis à profit pour lancer notre modèle avec uniquement une colonie au lieu des 4 habituelles. La deuxième phase de notre modèle, la fusion des séquences provenant des différentes colonies, n'est ici pas utilisée étant donné que nous n'aurons que les résultats d'une seule colonie, autrement dit nous n'aurons à la fin qu'une seule séquence unique par utilisateur et par expérimentation (soit 4 séquences par utilisateur pour l'ensemble des 4 expérimentations). Nous allons donc procéder à 4 tests différents pour chaque colonie avec des conditions de départ strictement identiques. Le but est ici de voir si les résultats obtenus correspondent bien à l'objectif de la colonie testée. Autrement dit, pour un utilisateur donné, les agents de la colonie de similarité devraient favoriser les musiques similaires au goût de l'utilisateur en question. De son côté, la colonie de nouveauté devrait, dans les mêmes conditions, produire des séquences de musiques moins familières pour l'utilisateur.

Avant de présenter les résultats de ces premières expérimentations, nous allons décrire de quelle manière nous les avons réalisées. Nous avons déjà détaillé la base de données avec laquelle nous travaillons dans la section précédente, mais nous n'avons pas encore expliqué comment nous avons utilisé ces données. Comme nous l'avons précisé, nous avons à notre disposition plus de 3.500 utilisateurs et plus de 91.400 sessions. Parmi l'ensemble total des utilisateurs, nous en avons sélectionné 500 au hasard afin de travailler uniquement sur eux. Cette réduction du nombre d'utilisateurs nous a permis de réaliser beaucoup plus rapidement des expérimentations tout en gardant une diversité importante. Pour chacun de ces 500 utilisateurs, nous avons récupéré leurs sessions d'écoute et gardé toutes celles contenant 5 musiques ou plus. Ces 500 utilisateurs ont ensuite été divisés aléatoirement en 2 groupes de 400 et de 100 : les 400 premiers utilisateurs ont servi pour l'entraînement du modèle (dans notre cas la création du graphe) et c'est sur les 100 utilisateurs restants que nous avons mené les tests. Dans le cadre de cette expérimentation précise, les utilisateurs de la base d'entraînement ont écouté 10.621 sessions uniques de plus de 5 musiques et les utilisateurs de la base de test ont quant à eux écouté 2.569 sessions. Pour chacune des sessions d'écoute des utilisateurs de la base de test, notre modèle a produit une séquence de recommandations en partant de la première musique écoutée. Autrement dit, pour chaque session, les agents de notre modèle ont été lancés sur le graphe en prenant comme point de départ la première musique que l'utilisateur avait écouté. Par exemple, pour la session d'un utilisateur consistant en l'écoute de l'album entier *Dark Side of the Moon* des Pink Floyd contenant 9 musiques, les agents étaient placés sur le nœud du graphe correspondant à la première musique de l'album, *Speak to me / Breathe*, puis notre modèle s'exécute et les agents trouvent un chemin dans le graphe. Les agents possèdent 3 conditions de parcours de graphe : une condition de taille minimale de la séquence recommandée et deux conditions d'arrêts. La condition de taille minimale de la séquence recommandée est fixée à la moitié de la longueur de la session initiale de l'utilisateur. Dans l'exemple de l'album de Pink Floyd, un agent sera donc forcé de parcourir au moins 5 nœuds du graphe avant de pouvoir s'arrêter. Les deux conditions d'arrêts étaient quant à elles imposées dans notre modèle afin de limiter la taille maximale de la séquence recommandée des agents. Premièrement, la dernière musique de la session initiale écoutée par l'utilisateur n'était pas donnée en amont mais, si jamais les agents arrivaient à cette musique, ils

arrêtaient ici leur chemin. Deuxièmement, dans le cas où les agents n'atteignaient pas la dernière musique de la session initiale de l'utilisateur, ces derniers s'arrêtaient lorsqu'ils avaient parcouru deux fois la taille de la session initiale. Dans notre exemple avec l'album de Pink Floyd, les agents pourraient donc parcourir au maximum 18 nœuds du graphe, et produire ainsi une séquence de recommandations de 18 musiques. En prenant en compte le nombre de nœuds des graphes sur lesquels nous avons travaillé (entre plusieurs dizaines de milliers jusqu'à plusieurs centaines de milliers), c'est cette deuxième condition d'arrêt qui était très largement atteinte. Ces trois conditions son nécessaires et ont été mises en place afin d'obtenir des séquences de recommandation faisant sens dans le contexte du domaine applicatif musical. Sans ces 3 conditions, il est possible d'obtenir des séquences ne faisant que deux musiques ou possédant des centaines voire des milliers de musiques. Comme le reste des paramètres de notre modèle, nous avons fait en sorte qu'il soit aisé de les changer au vol en fonction du domaine applicatif, voire même en fonction de chaque utilisateur au sein d'un même domaine.

Nous allons maintenant nous intéresser à la création du graphe. Nous avons décrit la manière dont nous créons le graphe sur lequel les agents se déplacent dans la Section 3.3 du chapitre précédent. Nous avons utilisé un sous-ensemble de 500 utilisateurs choisis aléatoirement parmi l'ensemble des utilisateurs de notre base de données Deezer pour réaliser cette expérimentation. Après cette sélection aléatoire, nous avons séparé ces 500 utilisateurs en 5 blocs de 100 utilisateurs afin de réaliser une validation croisée à 5 blocs. 4 blocs ont constitué la base d'apprentissage tandis que le dernier bloc restant a constitué la base de test. Ce processus a été répété 5 fois pour réaliser l'expérimentation avec toutes les combinaisons possibles. Le choix spécifique d'utiliser l'ensemble des connaissances disponibles de 80% des utilisateurs et d'essayer de prévoir les préférences des derniers 20% est délibérément voulu. Il nous permet de simuler au mieux le fonctionnement d'un service de recommandations en cours d'utilisation. En effet, un nouvel utilisateur arrivant sur le site sera inconnu du système de recommandation. La tâche de prédiction est plus ardue et s'aidera initialement des préférences des autres utilisateurs déjà connus du modèle. Au fur et à mesure que le nouvel utilisateur écoute des musiques et apporte ses retours sur ces dernières, le modèle peut intégrer ses préférences spécifiques et, de ce fait, améliorer les recommandations qu'il lui propose. Pour simuler cette évolution des connaissances du modèle, nous avons intégré chaque session de l'utilisateur au graphe après avoir proposer une recommandation pour celle-ci. Autrement dit, une fois que le modèle a proposé une recommandation de séquence en se basant sur une session d'écoute de l'utilisateur, le graphe est reconstruit en intégrant les informations de cette nouvelle session d'écoute. Ainsi, à chaque nouvelle session d'écoute pour laquelle notre modèle propose une recommandation, le graphe gagne en information et est plus à même de prédire correctement les préférences de l'utilisateur. Cependant, l'inconvénient majeur de cette méthode est de devoir reconstruire le graphe après chaque recommandation afin d'intégrer les nouvelles informations. Après cette première expérimentation, nous proposerons une méthode plus classique de répartition des connaissances apportées par les utilisateurs.

Le Tableau 4.2 présente les résultats de cette expérience pour les 4 colonies choisies. Ce format de tableau de résultats et les métriques d'évaluations qui le compose seront utilisés pour les prochaines expérimentations, nous allons donc les expliquer ici avant de discuter des résultats obtenus. Afin de mesurer les performances de notre modèle, nous avons utilisé diverses mesures nous permettant de capturer différents aspects que nous considérons être importants dans une recommandation de séquence. Nous allons décrire ces différentes mesures ci-dessous :

— **Précision, Rappel et F-mesure** : les mesures classiques dans le domaine de la recherche

	Précision	Rappel	F-mesure	Similarité	SIL	Diversité	DR	Nouveauté	Préférences
Colonie de Similarité	0,317	0,126	0,165	**0,952**	**0,923**	0,048 ***	0,049 ***	0,72 ***	0,447 ***
Colonie de Diversité	0,211	0,104	0,127	0,862 ***	0,77 ***	**0,138**	**0,19**	0,806 ***	0,393 ***
Colonie de Nouveauté	0,132	0,112	0,114	0,895 ***	0,837 ***	0,105 ***	0,144 ***	**0,909**	0,79 ***
Colonie de Préférences	0,296	0,159	0,191	0,946 ***	0,91 ***	0,054 ***	0,08 ***	0,695 ***	**0,804**

TABLE 4.2 – Expérimentations avec le modèle AntRS mono-objectif pour les 4 objectifs de similarité, diversité, nouveauté, préférences.
Le code couleur fait référence à la significativité statistique de la p-valeur avec $p \in [0; 1]$:
0 '***' 0,001 '**' 0,01 '*' 0,05 '.' 0,1 ' ' 1

d'information [Baeza-Yates et al., 1999]. Elles permettent de comparer directement les séquences de recommandation produites par notre modèle avec les sessions d'écoute initiales des utilisateurs. Dans un contexte d'utilisation réel où un utilisateur se connecte et le système propose une recommandation, il ne serait pas possible d'utiliser ces mesures. Cependant, dans le cadre d'une évaluation hors-ligne, elles apportent des informations majeures sur la qualité des recommandations, car nous considérons que les sessions d'écoute que nous possédons sont de bons modèles pour nos recommandations ;

— **Similarité et SIL (Similarité Intra-Liste)** : bien qu'elles soient sur-utilisées, les mesures de similarité, et notamment la similarité intra-liste, sont de bons indicateurs de la cohérence de nos séquences de recommandations. La mesure de similarité intra-liste provient des travaux de Ziegler & al.[Ziegler et al., 2005] ;

— **Diversité et DR (Diversité Relative)** : comme nous l'avons expliqué plus tôt dans ce manuscrit, la diversité est une caractéristique cruciale des systèmes de recommandations en plus de la similarité. Il était donc important de mesurer à quel point les musiques recommandées ainsi que les séquences en elles-mêmes étaient diverses [McGinty and Smyth, 2003, Vargas and Castells, 2011] ;

— **Nouveauté et Préférences** : pour mesurer ces deux dernières caractéristiques, nous avons utilisé les deux mesures proposées dans l'article de[Vargas and Castells, 2011], que nous avons aussi reproduites précédemment dans les Équations 3.5 et 3.6.

Après avoir expliqué en détail l'expérimentation et le processus d'évaluation, nous pouvons maintenant nous intéresser aux résultats à proprement parler. Le Tableau 4.2 présente les résultats de notre première expérimentation mono-objectif avec 500 utilisateurs pour chacune des 4 colonies isolées.

Afin de s'assurer que les résultats obtenus et présentés ci-dessus sont significatifs, nous avons réalisé, pour chacune des cases du tableau, un test de significativité statistique. Afin de savoir si nos données suivaient la loi normale, nous avons réalisé un test de Shapiro-Wilk et ce dernier s'est révélé négatif. Nous avons ainsi dû utiliser le test non paramétrique de Wilcoxon qui nous a permis de comparer les moyennes des deux échantillons connexes (même utilisateurs mais différentes colonies). Les résultats pour cette expérimentation se sont tous révélés être très significatifs, avec un p maximal inférieur à $p < 0,001$. Nous avons présentés la significativité de chaque case de résultat avec un système de symboles colorées où les trois premiers symboles (0 '***' 0,001 '**' 0,01 '*') sont significatifs. Ces tests de significativité ont été systématiquement effectués à chaque expérimentation et, par conséquent, tous les résultats que nous présenterons

dans la suite de cette section seront accompagnés de leur significativité.

Nous pouvons maintenant revenir sur les résultats présentés dans le Tableau 4.2. Comme il était prévu, on peut observer que chaque colonie a produit des recommandations orientées vers son objectif spécifique. Ainsi, la colonie de similarité a produit des séquences de musiques ayant obtenu les plus hauts scores de précision, similarité et similarité intra-liste ; la colonie de diversité a produit les séquences avec le plus haut score de diversité et de diversité relative ; tandis que les colonies de nouveauté et de préférences ont respectivement produit des séquences avec le plus haut score de nouveauté et de préférences. On peut cependant remarquer que, pour certaines mesures (la précision, la similarité ou encore la diversité par exemple), les écarts sont relativement faibles entre les différentes colonies. Nous pouvons répondre à cette constatation en deux points. Tout d'abord, et comme nous l'avons précisé plus haut, toutes les cases où 3 étoiles vertes ont été apposées représentent des résultats très significatifs d'un point de vue statistique. Ensuite, l'objectif principal de cette expérimentation est de prouver que notre modèle est bien capable de produire des recommandations diverses et complètes suivant plusieurs dimensions. La deuxième partie de notre modèle, où les résultats de chaque colonie seront fusionnés, devra produire des séquences équilibrant au mieux chacune des dimensions. Les premières séquences produites par les colonies mono-objectifs prouvent que cela est possible, mais qu'il est nécessaire d'avoir cette deuxième étape afin de mieux équilibrer les séquences finales qui seront proposées à l'utilisateur.

4.3.3 AntRS multi-objectifs : expérimentations sur l'ensemble des colonies simultanément et sur les tactiques de fusion

Objectifs de l'expérimentation

Après avoir montré que toutes les colonies étaient capables de produire des séquences de musiques spécialisées dans une dimension spécifique, nous pouvons maintenant réaliser des expérimentations sur notre modèle AntRS dans son ensemble, c'est-à-dire avec les 4 colonies en parallèle suivi de l'étape de fusion pour ne produire qu'une séquence unique possédant toutes les qualités de chaque dimension à l'utilisateur final. Autrement dit, avec la première expérimentation nous avons obtenu 4 séquences excellant chacune dans une dimension spécifique (similarité, diversité, nouveauté et préférences), et dorénavant l'objectif est d'obtenir une seule séquence réunissant ces 4 dimensions.

Nous n'allons pas réexpliquer en détail ici la construction du graphe ni le fonctionnement des colonies étant donné que nous l'avons déjà fait précédemment, mais les différences entre cette expérimentation et les précédentes seront cependant soulignées. Contrairement à l'expérimentation précédente, les 4 colonies ont été exécutées en parallèle en exploitant les *threads* de Java. La deuxième et principale différence se trouve dans l'exploitation des séquences générées par les colonies. Au lieu de les présenter directement comme nous l'avons fait dans l'expérimentation mono-objectif, ces séquences ont été fusionnées avec la deuxième partie du modèle présentée dans le chapitre précédent dans la Section 3.5. Le but est ici d'obtenir une seule séquence combinant au mieux tous les points forts des 4 colonies. Les métavariables utilisées sont les mêmes que pour l'expérience précédente.

Comparaison de performances intra-modèle

Maintenant que nous avons décrit l'objectif de cette expérimentation, nous allons décrire la manière dont nous avons mesuré et comparé les résultats obtenus. Tout d'abord, nous avons testé 3 versions de notre modèle, les voici décrites ci-dessous :

— AntRS avec les 4 colonies en parallèle et le **fusion de séquences** : Comme nous l'avons expliqué dans le chapitre précédent, Section 3.5, nous avons proposé deux tactiques de fusion pour les séquences trouvées par les différentes colonies lors de la première étape du modèle. La fusion de séquences est la première de ces deux tactiques. Pour rappel, cette tactique consiste à générer une liste de toutes les ressources de chacune des 4 meilleures séquences trouvées par les colonies lors de la première étape du modèle, puis de les réarranger afin de créer la séquence finale.

— AntRS avec les 4 colonies en parallèle et la **colonie de fusion** : Le deuxième modèle testé est identique au premier, sauf en ce qui concerne la tactique de fusion. Nous avons utilisé ici la deuxième tactique de fusion présentée dans la Section 3.5. Pour rappel, cette tactique consiste à construire un nouveau graphe constitué uniquement des ressources qui composent les séquences des colonies de la première étape, cette dernière agissant donc comme une sorte de sélection pour récupérer les ressources les plus intéressantes.

— **Colonie de fusion uniquement** : La troisième et dernière version de cette expérimentation sur le modèle AntRS est différente des deux premières. Au lieu d'effectuer séquentiellement la première étape (les 4 colonies) puis la deuxième étape avec les résultats de la première (la fusion), nous avons voulu tester si le fait d'exécuter la colonie de fusion directement avec l'ensemble des ressources de tous les utilisateurs pouvait produire des résultats intéressants. En effet, dans le modèle AntRS classique, cette colonie est exécutée avec quelques dizaines de ressources tout au plus qui ont été sélectionnées par les autres colonies de la première étape, mais rien n'empêche de lancer directement cette colonie avec le même graphe qu'utilisent les premières colonies et qui contient quelques dizaines de milliers de sommets. Notre hypothèse est que la séquence finale obtenue par ce biais obtiendra de moins bons résultats que le processus complet de notre modèle AntRS avec notamment la pré-sélection de ressources potentiellement intéressantes par les premières colonies spécialisées.

Enfin, les mêmes métriques ont été utilisées pour mesurer les performances de chaque version testée du modèle que pour l'expérimentation mono-objectif précédente afin de garder la consistance des résultats entre toutes les expérimentations menées.

Comparaison de performances inter-modèles

En plus de mesurer les performances des différentes versions de notre modèle, nous avons voulu comparer nos résultats obtenus avec d'autres systèmes de recommandation déjà existants. Le but est ici de confirmer que notre modèle produit des recommandations compétitives par rapport à d'autres algorithmes souvent utilisés dans l'état de l'art. Pour ce faire, nous avons utilisé deux types d'algorithmes : (1) trois algorithmes classiques mono-objectif capables de produire des listes de recommandations et (2) un algorithme spécifiquement multi-objectifs afin de d'avoir un point de comparaison provenant d'un modèle de la même classe que le nôtre. Nous

avons choisi ces 2 familles d'algorithmes afin de comparer au mieux notre modèle. Les algorithmes mono-objectif permettent d'établir une base de comparaison afin de déterminer si notre modèle est compétitif face à des systèmes n'optimisant qu'un seul de nos objectifs. De plus, ces algorithmes sont classiques dans le domaine de la recommandation et sont très souvent utilisés comme base de comparaison. L'algorithme multi-objectifs de comparaison permet quant à lui de mesurer nos performances sur l'ensemble des objectifs que nous optimisons. Nous allons décrire succinctement le principe de ces algorithmes sans entrer dans les détails étant donné que nous avons déjà décrit leur fonctionnement général précédemment dans l'état de l'art.

Algorithmes classiques mono-objectifs :

1. **UserKNN** [Breese et al., 2013] : algorithme classique de filtrage collaboratif où les k plus proches voisins de l'utilisateur actif basés sur la similarité des préférences sont sélectionnés afin de participer au processus de recommandation ;
2. **TrustMF** [Yang et al., 2017] : dans cette méthode plus récente de filtrage collaboratif, les auteurs proposent des améliorations du modèle classique de factorisation de matrices afin d'adresser les problèmes de démarrage à froid et de manque de données. Même s'ils utilisent la méthode classique de la factorisation de matrice, ils se démarquent en utilisant la confiance qu'ont les utilisateurs les uns envers les autres afin de refléter leur influence réciproque sur leurs opinions ;
3. **SVD++** [Koren, 2008] : dans cet article, les auteurs proposent des améliorations aux techniques des facteurs latents et des plus proches voisins, qui sont les deux approches du filtrage collaboratif obtenant les meilleurs résultats. Les auteurs combinent ces deux approches ensemble et utilisent à la fois les retours explicites et implicites afin d'obtenir une précision plus importante. Ce modèle a été crée dans le cadre du *Netflix Prize* de 2008, une compétition organisée par Netflix où le but était de créer un algorithme de filtrage collaboratif afin de prédire au mieux les préférences des utilisateurs [Bennett et al., 2007].

Ces trois algorithmes ont été sélectionnés car ils représentent des techniques connues et bien utilisées du domaine des systèmes de recommandation. Ils ont prouvé leur efficacité de par les nombreuses études de l'état de l'art les utilisant comme nous l'avons montré dans le chapitre 2 de ce manuscrit. De plus, ces algorithmes sont mis à disposition dans la librairie Java **Librec** [50] [Guo et al., 2015]. Cette librairie propose plus de 70 algorithmes de systèmes de recommandation en open source et a deux missions principales : la prédiction de préférences (via des notes) et le classement de ressources. L'utilisation de Librec nous a permis de travailler directement sur des algorithmes à jour et sans erreur.

Algorithme multi-objectifs :

4. **PEH** [Ribeiro et al., 2014] : Nous avons souhaité ajouter à cette liste un quatrième algorithme qui, contrairement aux trois premiers, est multi-objectifs. Cet algorithme, nommé PEH pour *Pareto-Efficient Hybridization*, a été présenté dans la Section 2.5.2 de l'état de l'art de ce manuscrit et nous allons ici discuter de son implémentation. PEH est un modèle où plusieurs algorithmes de recommandation sont exécutés en même temps. Le but est d'obtenir une valeur de pertinence pour chaque paire $ressource \times utilisateur$ indiquant à quel point il est pertinent de recommander la ressource à l'utilisateur. Chaque algorithme produit un score et, afin de n'obtenir qu'une seule valeur unique par paire,

50. Librec, la librarie Java pour les systèmes de recommandation

TABLE 4.3 – Expérimentations avec 3 versions du modèle AntRS et 4 algorithmes de comparaison : 3 algorithmes mono-objectifs classiques (UserKNN, TrustMF, SVD++) et 1 algorithme multi-objectifs (Pareto-Efficient Hybridization).

	Précision	Rappel	F-mesure	Similarité	SIL	Diversité	DR	Nouveauté	Préférences	Couverture
AntRS (4 colonies + fusion de séquences)	**0,344**	0,131	**0,1838**	0,947	0,908	0,053	0,069	**0,741**	**0,61**	**96,92%**
AntRS (4 colonies + colonie de fusion)	**0,288** ***	**0,151** ***	**0,1836** **	0,945	0,905 ***	0,055	**0,082** ***	0,702 ***	**0,612** **	**96,92%**
AntRS (colonie de fusion uniquement)	0,197 ***	0,118 ***	0,141 ***	0,953 ***	0,93 ***	0,047 ***	0,068	0,324 ***	0,389 ***	96,92%
UserKNN	0,224 ***	0,138	0,162 ***	0,95 ***	0,905 ***	0,05 ***	0,081 ***	0,68 ***	0,541	61,39%
TrustMF	0,195 ***	0,117 ***	0,14 ***	0,95 ***	0,894 ***	0,058 ***	0,09 ***	0,697 ***	0,458 ***	68,17%
SVD++	0,195 ***	0,12 ***	0,14 ***	0,941 ***	0,892 ***	0,06 ***	0,093 ***	0,70 ***	0,455 ***	68,17%
PEH	0,193 ***	0,118 ***	0,14 ***	0,941 ***	0,892 ***	0,059 ***	0,096 ***	0,697 ***	0,452 ***	97,52%

Significance codes (compared to AntRS with 4 colonies + lists merging) : 0 '***' 0,001 '**' 0,01 '*' 0,05 '.' 0,1 ' ' 1

le modèle PEH associe chacun de ces scores dans une combinaison linéaire selon une hybridation spécifique (*i.e.* des poids différents attribués à chaque algorithme, mitigeant ou augmentant ainsi leur importance sur la valeur de pertinence finale). Le résultat de cette combinaison linéaire est la valeur de pertinence de la recommandation de la ressource à l'utilisateur. Plusieurs hybridations sont ainsi testées et représentées dans un espace en 3 dimensions où chaque point correspond aux niveaux de précision, de diversité et de nouveauté atteint par une hybridation. Comme ces 3 dimensions sont aussi importantes les unes que les autres, les auteurs utilisent le concept d'Optimum de Pareto afin de sélectionner les meilleurs hybridations, c'est-à-dire les hybridations les plus proches ou sur la frontière de Pareto. Dans leur article, les auteurs utilisent différents algorithmes classique de recommandation. Ayant à notre disposition Librec ainsi que les 3 algorithmes présentés ci-dessus représentant plusieurs techniques de recommandation à la fois différentes les unes des autres et très utilisées dans la littérature, nous les avons inclus dans le processus d'hybridation. Le reste du modèle a été implémenté à partir de l'article des auteurs. Ce modèle est très intéressant pour notre comparaison de performance car il est multi-objectifs comme le nôtre, mais utilise une technique tout à fait différente de AntRS, non basée sur des modèles multi-agents et utilisant le concept d'Optimum de Pareto.

Pour cette expérimentation, nous avons donc un total de 4 algorithmes différents qui vont être exécutés sur exactement la même base de données et le même nombre d'utilisateurs que pour les trois versions de notre modèle. Les résultats de cette expérimentation sont présentés dans le Tableau 4.3.

Résultats

Comme pour l'expérimentation mono-objectif, nous avons mesuré la significativité statistique avec le test des rangs signés de Wilcoxon afin de s'assurer que les données obtenues pouvaient être interprétée. De plus, les valeurs en gras pour chaque colonne montrent les meilleurs résultats obtenus.

Les résultats de cette expérimentation nous révèlent plusieurs informations importantes. Tout d'abord, nous allons nous intéresser aux performances de notre modèle AntRS. Comme expliqué

ci-dessus, nous avons testé deux tactiques de fusion pour combiner les résultats de nos quatre objectifs. Nous avons également exécuté directement la colonie de fusion sur le graphe complet et sans la première étape de notre modèle.

Les deux premières lignes du Tableau 4.3 présentent les résultats de notre modèle AntRS complet avec ses deux variations : la fusion de séquences et la colonie de fusion. Nous pouvons voir que, d'une manière générale, notre modèle obtient de très bonnes performances par rapport aux 4 algorithmes de comparaison. Les deux variations de notre modèle obtiennent la meilleure précision (jusqu'à +78,23%), de rappel (jusqu'à +29,05%) et F-mesure (jusqu'à +31,28%) de tous les modèles testés dans cette expérimentation, ce qui signifie que notre modèle est le meilleur pour modéliser les préférences à partir des sessions initiales écoutées par les utilisateurs. AntRS a également surpassé les autres modèles pour les préférences et les métriques de nouveauté, tout en parvenant à maintenir un niveau correct de similitude et de diversité. Il est utile de rappeler ici que nous avons délibérément choisi des objectifs non compatibles entre eux (Similarité vs Diversité, Nouveauté vs Préférences), ce qui rend le processus de recommandations plus complexe pour les algorithmes multi-objectifs (AntRS et PEH) par rapport aux algorithmes mono-objectif. Malgré une diversité un peu plus faible par rapport aux algorithmes de comparaison, AntRS offre un meilleur compromis entre tous les objectifs que PEH, et dans un temps d'exécution beaucoup plus court. Enfin, nous pouvons remarquer que les deux modèles multi-objectifs, AntRS et PEH, ont une bien meilleure couverture par rapport aux autres algorithmes.

Les résultats montrent que notre modèle AntRS a obtenu de meilleurs résultats que les autres algorithmes de comparaison. Cependant, un autre objectif de cette expérimentation était de comparer les deux versions de la deuxième phase de notre modèle : la fusion de séquences et la colonie de fusion. Ces deux tactiques, permettant d'obtenir une séquence de recommandation finale multi-objectifs à partir de plusieurs séquences mono-objectif, représentent deux philosophies différentes de construction de séquences et il est intéressant de pouvoir observer leurs différences de performance. D'après les résultats obtenus, ces deux méthodes sont relativement proches l'une de l'autre avec l'exception majeure de la précision qui est plus élevée pour la fusion de séquences $(0,344)$ que pour la colonie de fusion $(0,288)$. On peut noter aussi une légère différence au niveau de la diversité relative (respectivement $0,069$ contre $0,082$) et de la nouveauté ($0,741$ contre $0,702$). L'importance de la précision dans la littérature et la différence entre les deux tactiques concernant cette métrique nous permettent de dire que la fusion de séquence est la meilleure des deux tactiques pour cette expérimentation.

Nous allons maintenant nous intéresser à la troisième ligne du tableau de résultats représentant l'exécution de la 2ème partie de notre modèle seule. Pour rappel, l'hypothèse de cette troisième version de notre modèle est de vérifier si la première étape de notre modèle permet bien d'améliorer la qualité des résultats en pré-sélectionnant les ressources correspondant le mieux aux quater objectifs à maximiser. On peut observer en effet avec les résultats que l'exécution directe de la colonie de fusion sans la première étape du modèle dégrade bien la qualité des solutions finales trouvées. Nous observons une nette diminution de la précision, du rappel, de la F-mesure, ainsi que de la diversité, de la nouveauté et de préférences. Ces résultats nous permettent donc de confirmer que la première étape de notre modèle, où les 4 colonies travaillent simultanément à obtenir des séquences spécialisées dans leur objectif, améliore bien la qualité finale des recommandations d'une manière significative. Autrement dit, les deux tactiques de fusion que nous avons développées (fusion de séquences et colonie de fusion) permettent de trouver un compromis entre tous les objectifs de manière plus efficace quand elles se basent sur les résultats de la

première étape de notre modèle.

Expérimentation alternative : techniques de validation croisée et de création de graphe alternative

Dans l'expérimentation précédente, nous avons décidé de séparer les 500 utilisateurs en 5 blocs de 100, où à tour de rôle 400 utilisateurs étaient utilisés pour la construction du graphe et 100 utilisateurs correspondaient à la base de test. Ainsi, le modèle avait pour tâche de produire des séquences de recommandations basées sur les sessions d'écoute de ces 100 utilisateurs. Une fois que le modèle avait produit sa séquence recommandée, la session d'écoute initiale sur laquelle était basée la recommandation était dévoilée au modèle et le graphe était reconstruit pour en tenir compte. Le but de ce fonctionnement était de simuler un fonctionnement réel d'un service d'écoute de musique. Cependant, un certain nombre d'études du domaine de la recommandation de la littérature scientifique utilisent une méthode différente lors de la séparation des informations des utilisateurs pour réaliser une validation croisée à 5 blocs. Cette méthode consiste à peupler les 5 blocs non plus à partir d'utilisateurs mais à partir de sessions d'écoute dans chaque utilisateur. Autrement dit, au lieu d'avoir une base d'apprentissage contenant 80% des utilisateurs et une base de test contenant les derniers 20% d'utilisateurs, la base d'apprentissage consiste en 80% des sessions d'écoute de chaque utilisateur et la base de test correspond aux 20% des sessions restantes. Le but du modèle est ici de recommander pour chaque utilisateur des séquences correspondant à la base de test. Par rapport à la première méthode, le graphe possède donc en permanence 80% des sessions de tous les utilisateurs. Cependant, le graphe ne contient pas les connaissances provenant de profils complets d'utilisateurs. Ces deux méthodes sont toutes les deux valides mais proposent deux approches différentes concernant l'exploitation des données des utilisateurs et il est intéressant de les comparer.

Par ailleurs, et comme nous l'avons expliqué dans le chapitre 3 décrivant notre modèle AntRS, la construction du graphe exploite les transitions trouvées dans les sessions d'écoute des utilisateurs pour créer un réseau de nœuds et d'arêtes représentant l'ensemble des données de la base d'apprentissage. Aucune emphase n'est portée spécifiquement sur l'utilisateur actif pour lequel le système est en train de construire des recommandations. Cependant, nous avons proposé dans le chapitre 3 une autre méthode de construction du graphe portant une emphase particulière sur l'utilisateur actif. À chaque nouvel utilisateur traité, nous calculons ses k plus proches voisins parmi les 499 utilisateurs restants. Pour faire cela, nous avons calculé la similarité cosine à partir des préférences existantes de ces utilisateurs. Autrement dit, pour chaque utilisateur actif, nous avons comparé ses 80% des sessions d'écoute de la base d'apprentissage avec les 80% des sessions d'écoute des 499 autres utilisateurs. Les k utilisateurs les plus similaires sont ensuite utilisés pour construire le graphe avec la même technique décrite dans le chapitre 3. L'avantage est ici de réduire la taille du graphe, d'accélérer la durée d'exécution et de potentiellement augmenter la qualité des recommandations finales, notamment en terme de similarité puisque c'est le critère de sélection ici. Cette variante de construction du graphe à partir des n plus proches voisins n'est possible qu'avec la technique de validation croisée décrite dans le paragraphe précédent, puisqu'il est nécessaire d'avoir accès aux sessions d'écoute des utilisateurs actifs, ce qui n'est pas le cas avec la première expérimentation de cette section.

Dans la suite de la section, nous allons réaliser la même expérimentation que précédemment avec ces deux modifications (le changement du processus de validation croisée et le graphe construit à partir de n plus proches utilisateurs en termes de similarité.), puis nous allons pré-

TABLE 4.4 – Expérimentation AntRS multi-objectifs avec les modifications de validation croisée et de création de graphe

	Précision	Rappel	F-mesure	Similarité	SIL	Diversité	DR	Nouveauté	Préférences	Couverture
AntRS graphe classique (fusion de séquences)	**0,379**	0,147	0,205	0,942	0,902	0,058	0,073	**0,748**	0,631	**71,68%**
AntRS graphe classique (colonie de fusion)	0,339	**0,176**	**0,215**	0,943	0,9	0,057	**0,085**	0,72	**0,718**	**71,68%**
AntRS graphe k voisins (fusion de séquences)	0,369	0,149	0,203	0,941	0,901	**0,059**	0,075	0,737	0,633	**71,68%**
AntRS graphe k voisins (colonie de fusion)	0,329	0,167	0,206	0,944	0,905	0,056	0,081	0,707	0,68	**71,68%**

senter les résultats obtenus.

Le Tableau 4.4 présente les résultats de cette deuxième expérimentation. Nous avons testé 4 versions de notre modèle. Pour chacune de ces 4 versions, la nouvelle technique de validation croisée a été utilisée. Comme pour l'expérimentation suivante, nous avons également testé les deux tactiques de fusion de notre modèle. Quant aux modifications portant sur la création du graphe, nous avons exécuté notre modèle selon 2 modalités : avec la même construction de graphe que l'expérimentation précédente ("graphe classique"), et avec la prise en compte des k voisins les plus similaires ("graphe k voisins"). On remarque déjà que les résultats sont, d'une manière générale, relativement similaires à la première expérimentation. Les valeurs de précision obtenues sont plus élevées que pour l'expérimentation précédente, suggérant que le modèle produit des recommandations plus proches des sessions d'écoute des utilisateurs. Cette augmentation de précision est la conséquence logique du fait que 80% des sessions d'écoute initiales sont connues du modèle. Ces résultats sont cependant à nuancer avec la couverture qui est significativement plus basse dans cette version de notre modèle (71,68% par rapport à 96,92%).

La tactique de fusion de séquence obtient une nouvelle fois la meilleure précision par rapport à la tactique de colonie de fusion. Cependant, nous n'observons pas de nette différence entre les deux méthodes de construction du graphe. Cette expérimentation ne nous permet donc pas de valider l'hypothèse selon laquelle un graphe moins grand et construit à partir d'utilisateurs similaires à l'utilisateur actif pourrait améliorer les recommandations finales. Cependant, l'autre objectif de cette expérimentation, qui était de démontrer que notre modèle obtient également des résultats satisfaisants avec une méthode plus classique de validation croisée, est validée. La technique classique de création du graphe, sans connaître par avance 80% des séquences d'écoutes des utilisateurs, reste cependant plus proche d'un fonctionnement en conditions réelles de notre modèle.

4.3.4 Optimisation des métavariables de AntRS

Influence du nombre d'agents et d'itérations sur la précision des recommandations

Comme tous les systèmes multi-agents, AntRS est dépendant du nombre d'agents utilisés et des itérations effectuées. Un nombre trop faible d'agents ou d'itérations peut empêcher l'émergence d'un comportement intelligent en raison d'un manque d'interaction entre les agents. Dans le même temps, il est également crucial de ne pas augmenter aveuglément ces paramètres car ils

affectent linéairement le temps d'exécution de l'algorithme. Nous avons ainsi réalisé des expérimentations sur notre modèle AntRS afin de déterminer un bon compromis entre performance et temps d'exécution. Dans la figure 4.11, la précision moyenne des recommandations obtenues est présentée en fonction du nombre d'agents et d'itérations par incrémentation de 50, allant de 50 à 300. On observe comme attendu une légère augmentation de la précision suivant l'accroissement des agents et des itérations, sans toutefois que celle-ci ne soit majeure. On remarque visuellement que la précision obtenue atteint rapidement un plateau à partir de 100 agents et itérations. Cette observation a été confirmée par un test de significativité statistique avec le test de Wilcoxon. Dans un contexte de fonctionnement en ligne où les utilisateurs s'attendent à des recommandations quasi immédiates, l'augmentation importante du temps d'exécution liée à un nombre élevé d'agents et d'itérations pour un gain minime de précision n'est pas recommandable. Ce sont aussi ces valeurs de 100 agents et itérations qui ont été utilisées dans les expérimentations de ce chapitre.

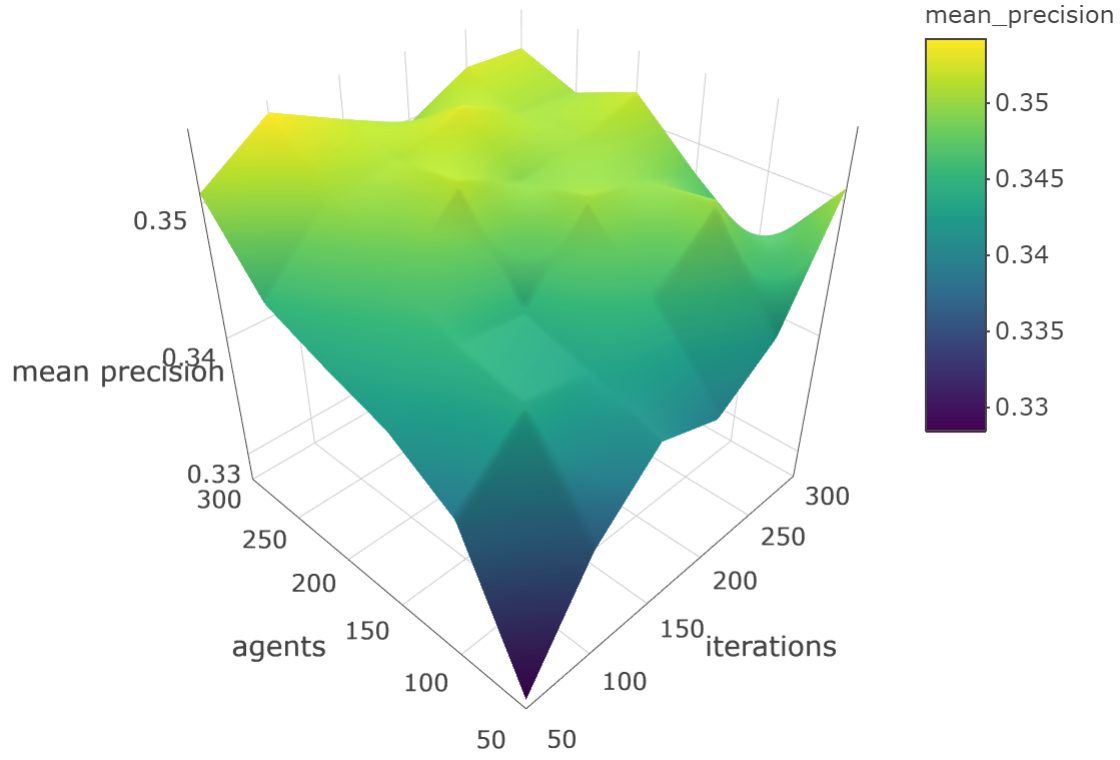

FIGURE 4.11 – Influence du nombre d'agents et d'itérations sur la précision moyenne obtenue par notre modèle.

Principe de fonctionnement et application d'un algorithme génétique

Les métavariables des algorithmes multi-agents ont une place importante dans la qualité des résultats obtenus. Pour rappel, et en plus du nombre d'itérations et d'agents dont nous avons déjà discuté dans la section précédente, on peut lister :

— $\tau_0 \in [0; 1]$: quantité de phéromones déposée sur les arêtes à l'initialisation du modèle ;
— $q_0 \in [0; 1]$: gère la part d'exploration et d'exploitation des agents ;
— $\alpha \in [0; 1]$: importance des phéromones dans la prise de décision des agents ;

— $\beta \in [0;1]$: importance de l'heuristique dans la prise de décision des agents ;
— $\rho \in [0;1]$: taux d'évaporation des phéromones.

Les valeurs de ces 5 métavariables ont été fixées à l'aide des expérimentations préliminaires présentées au début de ce chapitre dans la Section 4.2.2. Les valeurs de ces métavariables ont permis d'obtenir des résultats convaincants à la fois sur le modèle ACS classique de Dorigo et sur notre modèle ACS. Cependant, et afin de maximiser le potentiel de notre modèle AntRS, nous avons voulu tester une optimisation automatique de ces métavariables à l'aide d'un algorithme génétique. Ces derniers font partie de la famille des algorithmes évolutionnaires s'inspirant de la théorie de l'évolution. Le principe de ces algorithmes est de créer des individus possédant des gènes, c'est-à-dire des valeurs à des caractéristiques pré-déterminées ; puis de tester ces individus sur le problème à optimiser. À la fin d'une itération, les meilleurs individus transmettent une partie de leurs gènes à leurs descendants. Ainsi, de génération en génération, les gènes sélectionnés provenant des meilleurs individus permettent d'augmenter la qualité des résultats. Appliqué à notre modèle, un gène correspond à des valeurs pour les 5 métavariables à optimiser. Ces valeurs sont ensuite appliquées à notre modèle et testées durant une exécution complète, dans les mêmes conditions que les autres expérimentations de ce chapitre. Les résultats sont ensuite évalués selon les 4 métriques correspondant à nos objectifs : similarité, diversité, nouveauté et préférences. L'utilisation d'un algorithme génétique sur notre modèle a cependant posé un problème majeur de temps d'exécution. En effet, chaque individu de chaque génération a dû être testé sur une exécution complète de notre modèle. Pour rappel, une exécution de notre modèle correspond à 100 itérations où, dans chacune d'entre elles, 100 agents se déplacent pour trouver une séquence sur un graphe de plusieurs dizaines de milliers de nœuds et de plusieurs centaines de milliers d'arêtes en moyenne. Même si la complexité algorithmique est faible dûe à la nature même des modèles multi-agents, la quantité d'opérations à résoudre est conséquente. En plus de l'utilisation de serveurs de calculs dédiés, une optimisation importante du code de AntRS a été nécessaire pour pouvoir effectuer l'optimisation des métavariables avec un algorithme génétique dans des temps raisonnables.

Choix de l'algorithme génétique d'optimisation

De nombreuses familles d'algorithmes génétiques ont été développées depuis leur création, à l'instar des problèmes algorithmiques complexes nécessitant des techniques d'optimisation et de résolution avancées. Il est donc nécessaire de correctement définir le problème à optimiser afin de choisir l'algorithme génétique le plus adapté à la tâche. AntRS est un modèle multi-objectifs et les paramètres à optimiser sont les 4 objectifs de similarité, diversité, nouveauté et préférences. Il est ainsi nécessaire d'utiliser une technique parmi la famille des algorithmes évolutionnaires multi-objectifs (MOEA [51]). Nous avons choisi l'algorithme NSGA-II [52] pour sa rapidité et son efficacité pour trouver de meilleurs solutions non-dominées à des problèmes multi-objectifs classiques de la littérature par rapport à d'autres MOEA [Deb et al., 2002].

Pour les mêmes raisons que pour l'implémentation des algorithmes de comparaison utilisés dans cette section, nous avons utilisé une librairie Java nommée jMetal [53] [Durillo and Nebro, 2011]. Cette librairie est spécialisée dans l'implémentation et la mise à disposition d'une vingtaine

51. MOEA : *Multi-objective evolutionary algorithms*
52. *NSGA-II : Non-dominated Sorting Genetic Algorithm 2.*
53. *jMetal : Metaheuristic Algorithms in Java.*

TABLE 4.5 – Expérimentations présentant AntRS avec les métavariables empiriques et avec les métavariables optimisées via NSGA-II).

	Précision	Rappel	F-mesure	Similarité	SIL	Diversité	DR	Nouveauté	Préférences	Couverture
AntRS métavariables empiriques (fusion de séquences)	0,344	0,131	0,1838	0,947	0,908	**0,053**	0,069	0,741	0,61	96,92%
AntRS métavariables empiriques (colonie de fusion)	0,288	0,151	0,1836	0,945	0,905	0,055	0,082	0,702	0,612	96,92%
AntRS métavariables NSGA-II (fusion de séquences)	**0,382** ***	**0,153** ***	**0,221** ***	**0,953** ***	0,91 ,	0,047	0,07	**0,762** ***	**0,744** ***	96,92%
AntRS métavariables NSGA-II (colonie de fusion)	**0,312** ***	**0,174** ***	**0,226** ***	0,946	0,906	0,054	0,082	0,703	**0,802** ***	96,92%

Code de significativité statistique : 0 '***' 0,001 '**' 0,01 '*' 0,05 '.' 0,1 ' ' 1

de MOEA, permettant de les adapter à tous types de problèmes multi-objectifs.

Résultats

La version du modèle AntRS utilisée est la même que présentée dans le Chapitre 3 et dans l'expérimentation 4.3.3. L'algorithme d'optimisation NSGA-II a, quant à lui, été lancé sur un serveur de calculs dédié de l'équipe KIWI et arrêté à partir du moment où 10 générations de suite n'amélioraient plus les résultats obtenus. Les valeurs obtenues pour les 5 métavariables ont ensuite été utilisées dans une exécution complète de AntRS afin de mesurer l'impact de l'optimisation. L'algorithme de comparaison PEH [Ribeiro et al., 2014] inclut déjà dans son fonctionnement un MOEA nommé SPEA2 [54] [Zitzler et al., 2001] dont l'objectif est le même que NSGA-II : optimiser les recommandations en termes de similarité, diversité et nouveauté. Nous n'inclurons donc pas PEH dans les résultats de cette dernière expérimentation. Outre les métavariables, l'exécution de notre modèle s'est déroulée sous les mêmes conditions que pour les expérimentations précédentes.

Le Tableau 4.5 présente les résultats obtenus. Les deux premières lignes du tableau sont reprises de l'expérimentation principale AntRS multi-objectifs Section 4.3.3 pour plus de clarté. Pour rappel, cette expérimentation avait utilisé les métavariables empiriques. Les deux dernières lignes du tableau correspondent à AntRS avec les métavariables optimisées via le NSGA-II. La significativité statistique de la différence des résultats obtenus a été mesurée entre chaque paire ayant la même tactique de fusion (*i.e.* AntRS métavariables empiriques (fusion de séquences) vs AntRS métavariables NSGA-II (fusion de séquences) ; et AntRS métavariables empiriques (colonie de fusion) vs AntRS métavariables NSGA-II (colonie de fusion)). Les valeurs en gras indiquent les différences les plus notables entre les mêmes versions de AntRS sans et avec optimisation.

Ces résultats mettent principalement en exergue un net gain en précision et rappel obtenu avec l'optimisation des métavariables. Les valeurs obtenues pour ces métriques sont similaires à l'expérimentation 4.4 où le modèle avait connaissance de 80% des séquences d'écoutes des utilisateurs. On note également des améliorations pour les métriques de nouveauté et de préférences, montrant que les séquences recommandées étaient à la fois plus proches des préférences de l'uti-

54. *SPEA2 : Strenght Pareto Evolutionary Algorithm 2.*

lisateur tout en introduisant parfois des musiques qu'il ne connaissait pas.

Cependant, il est important de nuancer ces résultats. Les gains obtenus, même s'ils sont statistiquement significatifs, restent relativement peu élevés par rapport au temps nécessaire demandé par l'algorithme d'optimisation. De plus, certaines métriques comme la similarité ou la diversité n'ont que peu été affectées par l'optimisation. En conclusion, cette expérimentation nous a montré qu'il était possible d'optimiser les métavariables de notre modèle afin d'améliorer certains aspects des séquences recommandées, comme la précision, la nouveauté ou les préférences. Toutefois, le temps d'exécution d'un tel processus d'optimisation peut être prohibitif dans un contexte d'utilisation en ligne et en temps réel.

4.4 Conclusion

Dans ce Chapitre, nous avons mis en pratique le modèle multi-agents et multi-objectifs AntRS. Nous avons utilisé une base de données musicale provenant du site Deezer représentant un mois complet d'écoutes en ligne. Nous avons ensuite réalisé plusieurs expérimentations portant chacune sur un aspect théorique de AntRS. La première expérimentation présentée dans la Section 4.3.2 nous a permis de vérifier que les colonies de notre modèle permettent bien d'obtenir des recommandations de séquences spécialisées dans les facteurs humains identifiés dans les chapitres précédents. Dans le cadre de ce travail, nous avons testé de manière isolée 4 colonies correspondant à 4 facteurs humains spécifiques : la similarité, la diversité, la nouveauté et les préférences. Les résultats obtenus ont permis de prouver qu'il était possible de lier des facteurs humains à la recherche de chemins dans un graphe par des agents téléonomiques.

La deuxième expérimentation, présentée dans la Section 4.3.3, s'est appuyée sur ces conclusions pour tester le modèle AntRS dans son entièreté. Le but de cette expérimentation était triple :

1. Prouver que notre modèle est capable de produire une séquence unique possédant des aspects de chacun des 4 facteurs humains en utilisant les résultats des colonies spécialisées ;
2. Tester les performances des deux tactiques de notre modèle (fusion de séquences et colonie de fusion) permettant de fusionner les séquences des colonies spécialisées ;
3. Comparer les performances de AntRS avec d'autres systèmes de recommandations mono-objectif et multi-objectifs de la littérature du domaine.

Avec cette expérimentation, nous avons montré que AntRS était bien capable de synthétiser dans une même séquence les différents aspects humains considérés. De plus, les séquences de recommandations produites par AntRS ont obtenu de meilleurs résultats par rapport aux algorithmes de comparaison de l'état de l'art. La première tactique de fusion de séquences s'est révélée être plus efficace que la colonie de fusion pour obtenir un compromis entre les différents objectifs.

L'expérimentation suivante, présentée dans la Section 4.3.3, avait pour but de tester des variantes dans le fonctionnement de AntRS. Au cours de cette thèse, nous avons réalisé et testé de multiples modifications et variantes de notre modèle de manière empirique, tant le fonctionnement du modèle initial ACS de Dorigo est modulable. Pour mettre en exergue un exemple de cette modularité, nous avons testé les performances de notre modèle en modifiant la base d'apprentissage et la manière dont le graphe était créé. Les résultats obtenus ont mis en évidence

le potentiel d'évolution et d'adaptabilité de AntRS à des travaux futurs.

Enfin, nous nous sommes intéressés dans la Section 4.3.4 à l'optimisation automatique des métavariables de notre modèle. Comme nous l'avons illustré au début de ce chapitre avec les expérimentations préliminaires sur le modèle ACS, les métavariables jouent un rôle essentiel dans la qualité des résultats. Ces dernières gouvernent le comportement des agents, comme par exemple leurs tendances à explorer l'environnement, ou au contraire à exploiter des chemins déjà connus. Afin de maximiser le potentiel de notre modèle dans ce cadre applicatif, nous avons utilisé l'algorithme génétique d'optimisation de problèmes multi-objectifs NSGA-II sur 5 métavariables de AntRS. Le résultat de cette optimisation a permis d'améliorer de manière statistiquement significative les performances de notre modèle en termes de précision, rappel, nouveauté et préférences. Cependant, la longueur du processus d'optimisation limite pour le moment son application dans le contexte d'une utilisation en temps réel du modèle.

Le choix de la base de données musicale provenant de Deezer a permis de réaliser de nombreuses expérimentations et de tester différentes facettes de notre modèle. Cependant, nous avons aussi rencontré beaucoup de problèmes quant à la notion de progressivité avec cette base de données, et les résultats obtenus n'ont pas été concluants concernant cette métrique. Nous discuterons de ce point et d'autres perspectives d'améliorations dans la conclusion générale de cette thèse.

Chapitre 5

Conclusion et perspectives

5.1 Conclusion

Le domaine de la recommandation est en plein essor depuis maintenant une vingtaine d'années. Durant cette période, la recommandation n'a cessé d'évoluer afin de s'adapter à l'explosion du nombre d'utilisateurs et de la quantité de données disponibles. Nous pouvons distinguer trois aspects des recommandations modernes qui ont servi de fil directeur dans cette thèse. Premièrement, il y a encore 10 ans seulement, la majorité de la recherche avait pour but de maximiser la métrique de précision des résultats des algorithmes, il est aujourd'hui admis qu'une bonne recommandation nécessite aussi de satisfaire d'autres métriques comme la diversité ou la nouveauté. Même si la précision reste centrale, il est important de considérer ces autres métriques car elles permettent de répondre à une plus grande diversité de situation chez les utilisateurs demandant des recommandations. Deuxièmement, la quantité de données disponibles sur Internet a explosé tandis que l'utilisation des systèmes de recommandation s'est très largement démocratisée. La diversité des domaines applicatifs utilisant des systèmes de recommandations est telle que l'hyper-spécialisation d'un système pour un seule domaine fait moins de sens qu'auparavant. Les systèmes de recommandations capables de s'adapter rapidement à de multiples domaines applicatifs sont donc aujourd'hui particulièrement intéressants. Troisièmement, la croissance et la complexification des données décrivant les ressources et les utilisateurs permettent de proposer à ces derniers des recommandations plus complexes qu'auparavant. Comme nous l'avons dit, d'autres aspects que la précision sont déjà exploités, mais il est aussi possible de complexifier les recommandations en les proposant par séquences et non plus par ressource unique. En effet, la norme était avant de ne proposer à l'utilisateur qu'une seule recommandation, ou au mieux une liste de recommandations au même niveau les unes que les autres. Or de nombreux domaines, comme l'e-éducation, la musique ou encore le tourisme, peuvent bénéficier de systèmes capables de recommander une séquence de ressources dans un ordre précis. L'utilisateur d'un tel système dispose donc d'une séquence de ressources, permettant d'atteindre potentiellement des objectifs qui ne seraient pas atteignables avec des recommandations simples ou en liste. Ces trois points évoqués ci-dessus constituent le cœur du travail de cette thèse et nous allons maintenant rappeler les apports de notre travail sur ces trois points :

1. **Modèle multi-objectifs** : En se basant sur l'architecture des colonies de fourmis, notre modèle AntRS permet d'intégrer autant de métriques que voulues dans le processus de génération de recommandations. Dans ce travail, nous en avons considéré 4 : la similarité, la diversité, la nouveauté et les préférences passées. Ces 4 métriques sont intégrées en

tant que colonies dans notre modèle et participent toutes à la construction de la séquence de recommandations. Leur importance dans le processus de recommandation peut être modulée afin de mieux correspondre à chaque profil d'utilisateur.

2. **Généricité inter-domaine** : Dans le point précédent, nous avons discuté de l'intégration de différentes métriques dans les recommandations. Comme nous l'avons présenté dans l'état de l'art, un certain nombre de modèles prennent désormais en compte différentes métriques. Cependant, un des points forts de notre modèle est la modularité avec laquelle nous pouvons l'exécuter. Il est en effet possible d'enlever ou de rajouter des colonies à la demande. Autrement dit, nous sommes capable de changer à la volée les facteurs humains à considérer pour construire les recommandations finales. Outre ce point, il est relativement simple de passer d'un domaine applicatif à un autre. Nous avons construit notre modèle de sorte qu'il soit le plus générique possible étant donné que, lors de sa conception, nous n'avions pas encore décidé quelle base de données utiliser. Ainsi, la seule modification nécessaire lors d'un changement de domaine est d'adapter le calcul de la distance d entre chaque ressource lors de la construction du graphe. Après cette modification le reste de l'algorithme est le même.

3. **Séquences de recommandation** : Étant donné que notre modèle AntRS se base sur le modèle des colonies de fourmis de Marco Dorigo, les résultats que nous obtenons se présentent sous le même format que pour ce dernier. Initialement, dans les modèles proposés par Dorigo, les fourmis ont comme but de chercher un chemin parmi plusieurs possibles, dans un environnement représenté par un graphe et en empruntant des points de passage représentés par les nœuds du graphe. Le résultat obtenu est donc une suite de noeud avec un point de départ, un objectif de fin et une série de points de passage dans un ordre précis entre les deux ; autrement dit une séquence. Nous avons souhaité garder ce principe de production naturelle de séquences et de l'adapter à la recommandation. Ainsi, en construisant avec attention le graphe afin qu'il représente de la manière la plus fidèle possible les liens présents entre les ressources du domaine applicatif ; et en adaptant le calcul de la distance d séparant deux nœuds, notre modèle AntRS est capable de produire des séquences avec des ressources variées.

Les deux premiers apports énumérés ci-dessus permettent de s'adapter non seulement à de nombreux profils utilisateurs grâce aux différentes métriques, mais aussi à différents domaines applicatifs grâce à la simplicité des changements à faire pour passer d'un domaine à un autre. Le troisième apport énuméré permet quant à lui de proposer non plus des ressources uniques ou des listes de recommandations à l'utilisateur final, mais bien des séquences possédant un sens car basées sur les habitudes de consommation des ressources des utilisateurs passés.

5.2 Perspectives

Dans ce travail, nous avons répondu aux problématiques que nous avons définies au début du manuscrit, mais nous pouvons cependant distinguer des pistes d'améliorations futures.

Premièrement, les séquences peuvent bénéficier d'améliorations, notamment au niveau de la notion de progressivité. Pour le moment dans notre modèle, les séquences sont créées via

les agents fourmis parcourant le graphe à partir d'un point de départ jusqu'à un objectif. La progressivité des séquences est garantie par deux processus :

1. La topologie du graphe qui prend en compte les séquences créées naturellement par les utilisateurs lors de précédentes visites ;
2. La formule du calcul du meilleur chemin une fois les agents ayant terminé de se déplacer sur le graphe.

Nous avons exploré d'autres pistes pour offrir des séquences plus progressives aux utilisateurs. Par exemple en incluant une colonie dédiée uniquement à la progressivité, ou encore en modifiant les calculs de distance totale de chemin. Les tests que nous avons réalisés n'ont cependant pas abouti à des résultats concluants. Nous attribuons ces résultats au fait que le domaine de la musique n'est pas optimal pour utiliser ce genre de technique. Quand bien même il est courant pour les utilisateurs d'écouter des musiques sous formes de playlists, de récentes études ont montré que l'ordre dans lequel les musiques sont proposées n'est pas une caractéristique prioritaire dans la satisfaction de l'utilisateur. On peut par exemple citer les études suivantes [Vall et al., 2018, Vall et al., 2019] où les auteurs ont cherché à mesurer l'importance de l'ordre dans lequel étaient présentées les musiques d'une playlist. Les auteurs n'obtiennent pas de différences de résultats entre plusieurs expérimentations où l'ordre des musiques était maintenu ou lorsqu'il était rompu. Les auteurs concluent que l'ordre des musiques n'est pas une variable cruciale pour la génération automatique de playlists. D'autres domaines sont ainsi potentiellement plus à même d'être le sujet d'études plus approfondies sur la progressivité, comme par exemple les recommandations de parcours dans des musées ou les recommandations de séries de cours et d'exercices dans l'e-éducation.

Une autre perspective d'amélioration découlant du paragraphe précédent est l'application de notre modèle à un ou plusieurs autres domaines applicatifs. Nous avons déjà mentionné dans ce manuscrit que nous avions considéré les domaines muséal et de l'e-éducation en plus du domaine musical. Ces deux autres domaines proposent chacun des défis uniques et pourraient profiter de séquences de recommandations répondant aux besoins spécifiques de leurs utilisateurs. Il a malheureusement était trop complexe de récupérer ou de créer une base de données exploitable sur ces deux domaines durant cette thèse. Une expérimentation est toutefois menée actuellement en partenariat avec le Musée des Beaux Arts de Nancy afin de construire une base de données des chemins empruntés par les visiteurs du musée, ce qui permettra alors d'appliquer notre modèle afin de proposer des parcours personnalisés dans le musée.

Une dernière perspective d'amélioration réside dans l'évaluation des séquences générées par notre modèle. Notre but était de considérer plusieurs facteurs humains pour générer et évaluer nos recommandations tout en unifiant l'évaluation avec une seule métrique de satisfaction. Si nous avons réussi à intégrer plusieurs facteurs dans notre modèle, nous avons cependant rencontré des problèmes quant à l'unification des évaluations. L'objectif était de proposer une mesure de progressivité permettant de mesurer la qualité de la séquence recommandée mais, comme nous l'avons expliqué, nous n'avons pas obtenu de résultats concluants au regard de la progressivité dans notre base de données. Ainsi, la mesure de progressivité optimale o_n proposée et détaillée dans la Section 3.4.5 n'a pas pu être testée dans de bonnes conditions. Même si les expérimentations menées dans ce travail n'ont pas pu exploiter cette mesure de progressivité optimale, nous pensons qu'elle pourrait à termes donner une indication générale de la qualité d'une séquence de ressources dans d'autres domaines comme l'e-éducation ou le tourisme.

D'un point de vue plus général sur le fonctionnement de notre modèle AntRS, certaines améliorations sont possibles si l'on considère le fonctionnement du système dans des conditions réelles et non plus pour une expérimentation hors-ligne comme nous l'avons fait dans ce travail. Dans le cadre d'une utilisation en conditions réelles, quelques modifications du fonctionnement de notre algorithme pourraient améliorer la satisfaction de l'utilisateur. Prenons l'exemple d'un utilisateur se connectant sur un site de diffusion de musique et commençant la lecture d'une musique en particulier. Notre système sera capable, à partir de ce point de départ et des préférences passées de l'utilisateur, de créer une séquence de musiques. Cependant, si l'utilisateur ne clique pas sur la recommandation de playlist, ou écoute une deuxième chanson complètement différente des recommandations du système, il serait intéressant de pouvoir considérer et intégrer ces informations dans le modèle afin que ce dernier puisse s'adapter et satisfaire au mieux l'utilisateur. De plus, on voit bien l'intérêt de ces adaptations en temps réel dans les domaines muséal (re-calcul d'un nouvel itinéraire de visite si l'utilisateur ne suit pas la recommandation, à l'instar d'un GPS par exemple) et de l'e-éducation (adapter le contenu pédagogique en fonction du chemin de l'apprenant ou de ses résultats). Une dernière remarque générale concernant notre modèle peut être faite sur le temps d'exécution. Malgré les optimisations réalisées dans la dernière section du chapitre précédent, le temps d'exécution était relativement élevé lors des expérimentations. Cependant, les expérimentations concernaient 100 utilisateurs à la fois et étaient lancées majoritairement sur un ordinateur portable. Ainsi, la recommandation en temps réel d'une séquence de ressources à un seul utilisateur peut être faite en temps réel en utilisant un serveur dédié.

Le domaine de la recommandation s'élargit et se complexifie chaque année, à l'instar du monde dans lequel nous vivons. Devant la myriade de contenus accessibles sur Internet, les utilisateurs peuvent plus que jamais bénéficier de recommandations intelligentes, personnalisées, pertinentes, et répondant à leurs besoins. Dans le futur, les systèmes d'assistance aux utilisateurs devront être à la fois génériques, adaptables à tout types de contextes et toujours plus efficaces. C'est dans cette direction que s'inscrit le travail de cette thèse et le modèle AntRS.

Bibliographie

[Adomavicius and Tuzhilin, 2005] Adomavicius, G. and Tuzhilin, A. (2005). Toward the next generation of recommender systems : A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge & Data Engineering*, (6) :734–749.

[Adomavicius and Tuzhilin, 2011] Adomavicius, G. and Tuzhilin, A. (2011). Context-aware recommender systems. In *Recommender systems handbook*, pages 217–253. Springer.

[Aggarwal et al., 2016] Aggarwal, C. C. et al. (2016). *Recommender systems*. Springer.

[Agrawal et al., 1995] Agrawal, R., Srikant, R., et al. (1995). Mining sequential patterns. In *icde*, volume 95, pages 3–14.

[Alaya et al., 2007] Alaya, I., Solnon, C., and Ghedira, K. (2007). Ant colony optimization for multi-objective optimization problems. In *Tools with Artificial Intelligence, 2007. ICTAI 2007. 19th IEEE International Conference on*, volume 1, pages 450–457. IEEE.

[Angus, 2007] Angus, D. (2007). Crowding population-based ant colony optimisation for the multi-objective travelling salesman problem. In *2007 IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making*, pages 333–340. IEEE.

[Angus and Woodward, 2009] Angus, D. and Woodward, C. (2009). Multiple objective ant colony optimisation. *Swarm intelligence*, 3(1) :69–85.

[Antenucci et al., 2018] Antenucci, S., Boglio, S., Chioso, E., Dervishaj, E., Kang, S., Scarlatti, T., and Dacrema, M. F. (2018). Artist-driven layering and user's behaviour impact on recommendations in a playlist continuation scenario. In *Proceedings of the ACM Recommender Systems Challenge 2018*, page 4. ACM.

[Aucouturier and Pachet, 2002] Aucouturier, J.-J. and Pachet, F. (2002). Scaling up music playlist generation. In *Proceedings. IEEE International Conference on Multimedia and Expo*, volume 1, pages 105–108. IEEE.

[Ausubel, 1963] Ausubel, D. P. (1963). The psychology of meaningful verbal learning.

[Baeza-Yates et al., 1999] Baeza-Yates, R., Ribeiro-Neto, B., et al. (1999). *Modern information retrieval*, volume 463. ACM press New York.

[Belkin and Croft, 1992] Belkin, N. J. and Croft, W. B. (1992). Information filtering and information retrieval : two sides of the same coin. In *Communications of the ACM*. Citeseer.

[Bell and McMullen, 2004] Bell, J. E. and McMullen, P. R. (2004). Ant colony optimization techniques for the vehicle routing problem. *Advanced engineering informatics*, 18(1) :41–48.

[Benbasat and Wang, 2005] Benbasat, I. and Wang, W. (2005). Trust in and adoption of online recommendation agents. *Journal of the association for information systems*, 6(3) :4.

[Bennett et al., 2007] Bennett, J., Lanning, S., et al. (2007). The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35. New York.

[Bonabeau et al., 1999] Bonabeau, E., Marco, D. d. R. D. F., Dorigo, M., Théraulaz, G., Theraulaz, G., et al. (1999). *Swarm intelligence : from natural to artificial systems*. Number 1. Oxford university press.

[Borràs et al., 2014] Borràs, J., Moreno, A., and Valls, A. (2014). Intelligent tourism recommender systems : A survey. *Expert Systems with Applications*, 41(16) :7370–7389.

[Boyer et al., 2015] Boyer, A., Casanova, G., Roussanaly, A., and Ducreau, F. (2015). Pericles or how to assess and to improve quality in higher education.

[Breese et al., 2013] Breese, J. S., Heckerman, D., and Kadie, C. (2013). Empirical analysis of predictive algorithms for collaborative filtering. *arXiv preprint arXiv :1301.7363*.

[Brooks, 1991] Brooks, R. A. (1991). Intelligence without reason.

[Castagnos, 2008] Castagnos, S. (2008). *Modélisation de comportements et apprentissage stochastique non supervisé de stratégies d'interactions sociales au sein de systèmes temps réel de recherche et d'accès à l'information*. PhD thesis, Université Nancy II.

[Castagnos and Boyer, 2006] Castagnos, S. and Boyer, A. (2006). A client/server user-based collaborative filtering algorithm : Model and implementation. In *17th European Conference on Artificial Intelligence (ECAI 2006)*, pages 617–621.

[Castagnos et al., 2013] Castagnos, S., Brun, A., and Boyer, A. (2013). Utilité et perception de la diversité dans les systèmes de recommandation.

[Castagnos et al., 2010] Castagnos, S., Jones, N., and Pu, P. (2010). Eye-tracking product recommenders' usage. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 29–36. ACM.

[Castagnos et al., 2019] Castagnos, S., Marchal, F., Bertrand, A., Colle, M., and Mahmoudi, D. (2019). Inferring art preferences from gaze exploration in a museum. In *Adjunct Publication of the 27th Conference on User Modeling, Adaptation and Personalization*, pages 425–430.

[Castells et al., 2015] Castells, P., Hurley, N. J., and Vargas, S. (2015). Novelty and diversity in recommender systems. In *Recommender Systems Handbook*, pages 881–918. Springer.

[Chen and Li, 2019] Chen, H. and Li, J. (2019). Adversarial tensor factorization for context-aware recommendation. In *Proceedings of the 13th ACM Conference on Recommender Systems*, pages 363–367.

[Cheverst et al., 2000] Cheverst, K., Davies, N., Mitchell, K., Friday, A., and Efstratiou, C. (2000). Developing a context-aware electronic tourist guide : some issues and experiences. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 17–24. ACM.

[Chou et al., 2005] Chou, S.-C., Hsieh, W.-T., Gandon, F. L., and Sadeh, N. M. (2005). Semantic web technologies for context-aware museum tour guide applications. In *Advanced Information Networking and Applications, 2005. AINA 2005. 19th International Conference on*, volume 2, pages 709–714. IEEE.

[Davis, 1989] Davis, F. D. (1989). Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS quarterly*, pages 319–340.

[Deb et al., 2002] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm : Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2) :182–197.

[Deneubourg et al., 1990] Deneubourg, J.-L., Aron, S., Goss, S., and Pasteels, J. M. (1990). The self-organizing exploratory pattern of the argentine ant. *Journal of insect behavior*, 3(2) :159–168.

[Doerner et al., 2004] Doerner, K., Gutjahr, W. J., Hartl, R. F., Strauss, C., and Stummer, C. (2004). Pareto ant colony optimization : A metaheuristic approach to multiobjective portfolio selection. *Annals of operations research*, 131(1) :79–99.

[Dorigo, 2001] Dorigo, M. (2001). Ant algorithms solve difficult optimization problems. In *European Conference on Artificial Life*, pages 11–22. Springer.

[Dorigo and Birattari, 2010] Dorigo, M. and Birattari, M. (2010). *Ant colony optimization*. Springer.

[Dorigo and Birattari, 2011] Dorigo, M. and Birattari, M. (2011). Ant colony optimization. In *Encyclopedia of machine learning*, pages 36–39. Springer.

[Dorigo et al., 2006] Dorigo, M., Birattari, M., and Stutzle, T. (2006). Ant colony optimization. *IEEE computational intelligence magazine*, 1(4) :28–39.

[Dorigo and Gambardella, 1997] Dorigo, M. and Gambardella, L. M. (1997). Ant colonies for the travelling salesman problem. *biosystems*, 43(2) :73–81.

[Dorigo et al., 1996] Dorigo, M., Maniezzo, V., Colorni, A., et al. (1996). Ant system : optimization by a colony of cooperating agents. *IEEE Transactions on Systems, man, and cybernetics, Part B : Cybernetics*, 26(1) :29–41.

[Dorigo and Stützle, 2019] Dorigo, M. and Stützle, T. (2019). Ant colony optimization : overview and recent advances. In *Handbook of metaheuristics*, pages 311–351. Springer.

[Drachsler et al., 2015] Drachsler, H., Verbert, K., Santos, O. C., and Manouselis, N. (2015). Panorama of recommender systems to support learning. In *Recommender systems handbook*, pages 421–451. Springer.

[Durillo and Nebro, 2011] Durillo, J. J. and Nebro, A. J. (2011). jmetal : A java framework for multi-objective optimization. *Advances in Engineering Software*, 42(10) :760–771.

[Ekstrand et al., 2014] Ekstrand, M. D., Harper, F. M., Willemsen, M. C., and Konstan, J. A. (2014). User perception of differences in recommender algorithms. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 161–168. ACM.

[Ferber, 1995] Ferber, J. (1995). Les systèmes multi-agents : Vers une intelligence collective, intereditions, 1995. *Connaissances et compétences requises Cette thèse s' adresse à des étudiants ayant un profil Intelligence Artificielle/Base de Données et elle nécessite d'avoir des compétences théoriques dans les domaines*, 1.

[Ferber and Weiss, 1999] Ferber, J. and Weiss, G. (1999). *Multi-agent systems : an introduction to distributed artificial intelligence*, volume 1. Addison-Wesley Reading.

[Frolov and Oseledets, 2017] Frolov, E. and Oseledets, I. (2017). Tensor methods and recommender systems. *Wiley Interdisciplinary Reviews : Data Mining and Knowledge Discovery*, 7(3) :e1201.

[Gambardella et al., 1999] Gambardella, L. M., Taillard, É., and Agazzi, G. (1999). Macs-vrptw : A multiple ant colony system for vehicle routing problems with time windows.

[Gan et al., 2020] Gan, L., Nurbakova, D., Laporte, L., and Calabretto, S. (2020). Enhancing recommendation diversity using determinantal point processes on knowledge graphs. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2001–2004.

[Gavalas et al., 2014] Gavalas, D., Konstantopoulos, C., Mastakas, K., and Pantziou, G. (2014). Mobile recommender systems in tourism. *Journal of network and computer applications*, 39 :319–333.

[Goodenough et al., 2017] Goodenough, A. E., Little, N., Carpenter, W. S., and Hart, A. G. (2017). Birds of a feather flock together : Insights into starling murmuration behaviour revealed using citizen science. *PloS one*, 12(6) :e0179277.

[Goss et al., 1989] Goss, S., Aron, S., Deneubourg, J.-L., and Pasteels, J. M. (1989). Self-organized shortcuts in the argentine ant. *Naturwissenschaften*, 76(12) :579–581.

[Gras et al., 2016] Gras, B., Brun, A., and Boyer, A. (2016). Identifying grey sheep users in collaborative filtering : a distribution-based technique. In *Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization*, pages 17–26. ACM.

[Grassé, 1959] Grassé, P.-P. (1959). La reconstruction du nid et les coordinations interindividuelles chez bellicositermes natalensis etcubitermes sp. la théorie de la stigmergie : Essai d'interprétation du comportement des termites constructeurs. *Insectes sociaux*, 6(1) :41–80.

[Guo et al., 2015] Guo, G., Zhang, J., Sun, Z., and Yorke-Smith, N. (2015). Librec : A java library for recommender systems. In *UMAP Workshops*.

[Han et al., 2007] Han, J., Cheng, H., Xin, D., and Yan, X. (2007). Frequent pattern mining : current status and future directions. *Data mining and knowledge discovery*, 15(1) :55–86.

[Harper and Konstan, 2015] Harper, F. M. and Konstan, J. A. (2015). The movielens datasets : History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4) :1–19.

[Herlocker et al., 2004] Herlocker, J. L., Konstan, J. A., Terveen, L. G., and Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1) :5–53.

[Hölldobler et al., 1990] Hölldobler, B., Wilson, E. O., et al. (1990). *The ants*. Harvard University Press.

[Honey et al., 1992] Honey, P., Mumford, A., et al. (1992). The manual of learning styles.

[Hu et al., 2008] Hu, Y., Koren, Y., and Volinsky, C. (2008). Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining*, pages 263–272. Ieee.

[Iaquinta et al., 2008] Iaquinta, L., De Gemmis, M., Lops, P., Semeraro, G., Filannino, M., and Molino, P. (2008). Introducing serendipity in a content-based recommender system. In *2008 Eighth International Conference on Hybrid Intelligent Systems*, pages 168–173. IEEE.

[Iredi et al., 2001] Iredi, S., Merkle, D., and Middendorf, M. (2001). Bi-criterion optimization with multi colony ant algorithms. In *Evolutionary Multi-Criterion Optimization*, pages 359–372. Springer.

[Jameson et al., 2015] Jameson, A., Willemsen, M. C., Felfernig, A., de Gemmis, M., Lops, P., Semeraro, G., and Chen, L. (2015). Human decision making and recommender systems. In *Recommender Systems Handbook*, pages 611–648. Springer.

[Jannach et al., 2016] Jannach, D., Kamehkhosh, I., and Bonnin, G. (2016). Biases in automated music playlist generation : A comparison of next-track recommending techniques. In *Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization*, pages 281–285. ACM.

[Jawaheer et al., 2010] Jawaheer, G., Szomszor, M., and Kostkova, P. (2010). Comparison of implicit and explicit feedback from an online music recommendation service. In *proceedings of the 1st international workshop on information heterogeneity and fusion in recommender systems*, pages 47–51. ACM.

[Jones, 2010] Jones, N. (2010). *User Perceived Qualities and Acceptance of Recommender Systems : The Role of Diversity*. PhD thesis, EPFL.

[Kaminskas and Bridge, 2016] Kaminskas, M. and Bridge, D. (2016). Diversity, serendipity, novelty, and coverage : A survey and empirical analysis of beyond-accuracy objectives in recommender systems. *ACM Trans. Interact. Intell. Syst.*, 7(1) :2 :1–2 :42.

[Kardan et al., 2014] Kardan, A. A., Ebrahim, M. A., and Imani, M. B. (2014). A new personalized learning path generation method : Aco-map. *Indian Journal of Scientific Research*, 5(1) :17.

[King and Sumpter, 2012] King, A. J. and Sumpter, D. J. (2012). Murmurations. *Current Biology*, 22(4) :R112–R114.

[Kolb and Fry, 1974] Kolb, D. A. and Fry, R. E. (1974). *Toward an applied theory of experiential learning*. MIT Alfred P. Sloan School of Management.

[Konstan et al., 1998] Konstan, J. A., Riedl, J., Borchers, A., and Herlocker, J. L. (1998). Recommender systems : A grouplens perspective. In *Recommender Systems : Papers from the 1998 Workshop (AAAI Technical Report WS-98-08)*, pages 60–64. AAAI Press.

[Koren, 2008] Koren, Y. (2008). Factorization meets the neighborhood : a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434. ACM.

[Koren et al., 2009] Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, (8) :30–37.

[Kuhlthau, 1991] Kuhlthau, C. C. (1991). Inside the search process : Information seeking from the user's perspective. *Journal of the American society for information science*, 42(5) :361–371.

[Kurilovas et al., 2015] Kurilovas, E., Zilinskiene, I., and Dagiene, V. (2015). Recommending suitable learning paths according to learners' preferences : Experimental research results. *Computers in Human Behavior*, 51 :945–951.

[l'Huillier, 2018] l'Huillier, A. (2018). *Modéliser la diversité au cours du temps pour comprendre le contexte de l'utilisateur dans les systèmes de recommandation*. PhD thesis.

[L'Huillier et al., 2014] L'Huillier, A., Castagnos, S., and Boyer, A. (2014). Understanding usages by modeling diversity over time. In *22nd Conference on User Modeling, Adaptation, and Personalization*, volume 1181.

[Logan, 2002] Logan, B. (2002). Content-based playlist generation : Exploratory experiments. In *ISMIR*.

[Lops et al., 2011] Lops, P., De Gemmis, M., and Semeraro, G. (2011). Content-based recommender systems : State of the art and trends. In *Recommender systems handbook*, pages 73–105. Springer.

[Mahmood and Ricci, 2009] Mahmood, T. and Ricci, F. (2009). Improving recommender systems with adaptive conversational strategies. In *Proceedings of the 20th ACM conference on Hypertext and hypermedia*, pages 73–82. ACM.

[March, 1991] March, J. G. (1991). Exploration and exploitation in organizational learning. *Organization science*, 2(1) :71–87.

[McGinty and Smyth, 2003] McGinty, L. and Smyth, B. (2003). On the role of diversity in conversational recommender systems. In *Proceedings of the 5th International Conference on Case-based Reasoning : Research and Development*, ICCBR'03, pages 276–290.

[McNee et al., 2006] McNee, S. M., Riedl, J., and Konstan, J. A. (2006). Being accurate is not enough : how accuracy metrics have hurt recommender systems. In *CHI'06 extended abstracts on Human factors in computing systems*, pages 1097–1101. ACM.

[Merkle et al., 2002] Merkle, D., Middendorf, M., and Schmeck, H. (2002). Ant colony optimization for resource-constrained project scheduling. *IEEE transactions on evolutionary computation*, 6(4) :333–346.

[Mokatren et al., 2018] Mokatren, M., Kuflik, T., and Shimshoni, I. (2018). Exploring the potential of a mobile eye tracker as an intuitive indoor pointing device : A case study in cultural heritage. *Future Generation Computer Systems*, 81 :528–541.

[Nurbakova et al., 2018] Nurbakova, D., Laporte, L., Calabretto, S., and Gensel, J. (2018). Recommandation de séquences d'activités lors d'évènements distribués.

[Oard et al., 1998] Oard, D. W., Kim, J., et al. (1998). Implicit feedback for recommender systems. In *Proceedings of the AAAI workshop on recommender systems*, pages 81–83.

[Osche et al., 2019a] Osche, P.-E., Castagnos, S., and Boyer, A. (2019a). Antrs : Recommending lists through a multi-objective ant colony system. In *European Conference on Information Retrieval*, pages 229–243. Springer.

[Osche et al., 2019b] Osche, P.-E., Castagnos, S., and Boyer, A. (2019b). From music to museum : Applications of multi-objective ant colony systems to real world problems. In *Adaptive and Learning Agents Workshop at AAMAS (ALA 2019)*.

[Osche et al., 2016] Osche, P.-E., Castagnos, S., Napoli, A., and Naudet, Y. (2016). Walk the line : toward an efficient user model for recommendations in museums. In *Semantic and Social Media Adaptation and Personalization (SMAP), 2016 11th International Workshop*. IEEE.

[Oster, 1981] Oster, G. (1981). E. 0. wilson. 1978. caste and ecology in the social insects. *Monographs in Population Biology*, 12 :1–352.

[Qiu et al., 2015] Qiu, J., Lin, Z., and Li, Y. (2015). Predicting customer purchase behavior in the e-commerce context. *Electronic commerce research*, 15(4) :427–452.

[Quadrana et al., 2018] Quadrana, M., Cremonesi, P., and Jannach, D. (2018). Sequence-aware recommender systems. *ACM Computing Surveys (CSUR)*, 51(4) :1–36.

[Resnick et al., 1994] Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J. (1994). Grouplens : an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186. ACM.

[Reynolds, 1987] Reynolds, C. W. (1987). *Flocks, herds and schools : A distributed behavioral model*, volume 21. ACM.

[Ribeiro et al., 2014] Ribeiro, M. T., Ziviani, N., Moura, E. S. D., Hata, I., Lacerda, A., and Veloso, A. (2014). Multiobjective pareto-efficient approaches for recommender systems. *ACM Trans. Intell. Syst. Technol.*, 5(4) :53 :1–53 :20.

[Ricci et al., 2015] Ricci, F., Rokach, L., and Shapira, B. (2015). *Recommender Systems Handbook*. Springer, US.

[RISK, 2002] RISK, U. (2002). Draft standard for learning object metadata. *IEEE standard*, 1484(1).

[Rubtsov et al., 2018] Rubtsov, V., Kamenshchikov, M., Valyaev, I., Leksin, V., and Ignatov, D. I. (2018). A hybrid two-stage recommender system for automatic playlist continuation. In *Proceedings of the ACM Recommender Systems Challenge 2018*, page 16. ACM.

[Russell and Norvig, 2016] Russell, S. J. and Norvig, P. (2016). *Artificial intelligence : a modern approach*. Malaysia ; Pearson Education Limited,.

[Salton, 1989] Salton, G. (1989). Automatic text processing : The transformation, analysis, and retrieval of. *Reading : Addison-Wesley*, 169.

[Salton and McGill, 1986] Salton, G. and McGill, M. J. (1986). Introduction to modern information retrieval.

[Sammut and Webb, 2010] Sammut, C. and Webb, G. I., editors (2010). *Mean Absolute Error*, pages 652–652. Springer US, Boston, MA.

[Schlunz, 2011] Schlunz, E. B. (2011). *Decision support for generator maintenance scheduling in the energy sector*. PhD thesis, Stellenbosch : Stellenbosch University.

[Schnabel et al., 2018] Schnabel, T., Bennett, P. N., Dumais, S. T., and Joachims, T. (2018). Short-term satisfaction and long-term coverage : Understanding how users tolerate algorithmic exploration. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 513–521.

[Semet et al., 2003] Semet, Y., Lutton, E., and Collet, P. (2003). Ant colony optimisation for e-learning : Observing the emergence of pedagogic suggestions. In *Swarm Intelligence Symposium, 2003. SIS'03. Proceedings of the 2003 IEEE*, pages 46–52. IEEE.

[Shani and Gunawardana, 2011] Shani, G. and Gunawardana, A. (2011). Evaluating recommendation systems. In *Recommender systems handbook*, pages 257–297. Springer.

[Shardanand and Maes, 1995] Shardanand, U. and Maes, P. (1995). Social information filtering : Algorithms for automating" word of mouth". In *Chi*, volume 95, pages 210–217. Citeseer.

[Silva et al., 2009] Silva, A. R. D., Lages, W. S., and Chaimowicz, L. (2009). Boids that see : Using self-occlusion for simulating large groups on gpus. *Computers in Entertainment (CIE)*, 7(4) :51.

[Simon, 1997] Simon, H. A. (1997). *Models of bounded rationality : Empirically grounded economic reason*, volume 3. MIT press.

[Smyth and McClave, 2001] Smyth, B. and McClave, P. (2001). Similarity vs. diversity. In *International Conference on Case-Based Reasoning*, pages 347–361. Springer.

[Stützle and Hoos, 2000] Stützle, T. and Hoos, H. H. (2000). Max–min ant system. *Future generation computer systems*, 16(8) :889–914.

[Su and Khoshgoftaar, 2009] Su, X. and Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009 :4.

[Tesoriero et al., 2008] Tesoriero, R., Gallud, J. A., Lozano, M., and Penichet, V. M. R. (2008). A location-aware system using rfid and mobile devices for art museums. In *Fourth International Conference on Autonomic and Autonomous Systems (ICAS'08)*, pages 76–81. IEEE.

[Tesoriero et al., 2014] Tesoriero, R., Gallud, J. A., Lozano, M., and Penichet, V. M. R. (2014). Enhancing visitors' experience in art museums using mobile technologies. *Information Systems Frontiers*, 16(2) :303–327.

[Tintarev et al., 2017] Tintarev, N., Lofi, C., and Liem, C. (2017). Sequences of diverse song recommendations : An exploratory study in a commercial system. In *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization*, pages 391–392. ACM.

[Twardowski, 2016] Twardowski, B. (2016). Modelling contextual information in session-aware recommender systems with neural networks. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 273–276. ACM.

[Vall et al., 2018] Vall, A., Quadrana, M., Schedl, M., and Widmer, G. (2018). The importance of song context and song order in automated music playlist generation. *arXiv preprint arXiv :1807.04690*.

[Vall et al., 2019] Vall, A., Quadrana, M., Schedl, M., and Widmer, G. (2019). Order, context and popularity bias in next-song recommendations. *International Journal of Multimedia Information Retrieval*, 8(2) :101–113.

[Van den Berg et al., 2005] Van den Berg, B., Van Es, R., Tattersall, C., Janssen, J., Manderveld, J., Brouns, F., Kurvers, H., and Koper, R. (2005). Swarm-based sequencing recommendations in e-learning. In *Intelligent Systems Design and Applications, 2005. ISDA'05. Proceedings. 5th International Conference on*, pages 488–493. IEEE.

[Vargas and Castells, 2011] Vargas, S. and Castells, P. (2011). Rank and relevance in novelty and diversity metrics for recommender systems. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 109–116. ACM.

[Véron and Levasseur, 1989] Véron, E. and Levasseur, M. (1989). *Ethnographie de l'exposition : l'espace, le corps et le sens.* Centre Georges Pompidou, Bibliothèque publique d'information.

[Volkovs et al., 2018] Volkovs, M., Rai, H., Cheng, Z., Wu, G., Lu, Y., and Sanner, S. (2018). Two-stage model for automatic playlist continuation at scale. In *Proceedings of the ACM Recommender Systems Challenge 2018*, page 9. ACM.

[Wang et al., 2012] Wang, X., Rosenblum, D., and Wang, Y. (2012). Context-aware mobile music recommendation for daily activities. In *Proceedings of the 20th ACM international conference on Multimedia*, pages 99–108. ACM.

[Weiss, 2013] Weiss, G. (2013). *Multiagent systems.* MIT press.

[Yang et al., 2017] Yang, B., Lei, Y., Liu, J., and Li, W. (2017). Social collaborative filtering by trust. *IEEE transactions on pattern analysis and machine intelligence*, 39(8) :1633–1647.

[Yang et al., 2018] Yang, H., Jeong, Y., Choi, M., and Lee, J. (2018). Mmcf : Multimodal collaborative filtering for automatic playlist continuation. In *Proceedings of the ACM Recommender Systems Challenge 2018*, page 11. ACM.

[Yang and Wu, 2009] Yang, Y. J. and Wu, C. (2009). An attribute-based ant colony system for adaptive learning object recommendation. *Expert Systems with Applications*, 36(2) :3034–3047.

[Yoshimura et al., 2014] Yoshimura, Y., Sobolevsky, S., Ratti, C., Girardin, F., Carrascal, J. P., Blat, J., and Sinatra, R. (2014). An analysis of visitors' behavior in the louvre museum : a study using bluetooth data. *Environment and Planning B : Planning and Design*, 41(6) :1113–1131.

[Yu and Riedl, 2012] Yu, H. and Riedl, M. O. (2012). A sequential recommendation approach for interactive personalized story generation. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 71–78. International Foundation for Autonomous Agents and Multiagent Systems.

[Zhang, 2013] Zhang, L. (2013). The definition of novelty in recommendation system. *Journal of Engineering Science & Technology Review*, 6(3).

[Zhao et al., 2014] Zhao, G., Lee, M. L., and Wynne, H. (2014). Utilizing purchase intervals in latent clusters for product recommendation. In *Proceedings of the 8th Workshop on Social Network Mining and Analysis*, page 4. ACM.

[Ziegler et al., 2005] Ziegler, C.-N., McNee, S. M., Konstan, J. A., and Lausen, G. (2005). Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web*, pages 22–32. ACM.

[Zitzler et al., 2001] Zitzler, E., Laumanns, M., and Thiele, L. (2001). Spea2 : Improving the strength pareto evolutionary algorithm. *TIK-report*, 103.

Résumé

Les systèmes de recommandations représentent un thème de recherche fondamental à l'intersection entre plusieurs grandes disciplines telles que l'apprentissage automatique, l'interaction homme-machine et les sciences cognitives. Ils constituent en outre un cadre applicatif ambitieux pour la communauté des chercheurs en Intelligence Artificielle de par leur très grande complexité et les nombreuses contraintes qu'ils génèrent.

L'objectif de ces systèmes est d'améliorer les interactions entre le grand public et les systèmes de recherche et d'accès à l'information. Il est en effet devenu difficile, dans un contexte de masse de données hétérogènes, d'identifier les ressources les plus pertinentes. Il s'agit d'assister les utilisateurs dans leurs explorations (que cela soit dans un environnement virtuel ou physique), mais également de proposer des ressources susceptibles de les intéresser mais qu'ils n'auraient pas consultées spontanément.

Les systèmes actuels ont largement prouvé leur valeur ajoutée et reposent sur des techniques d'apprentissage variées (numérique ou symbolique, supervisé ou non, etc.) [Castagnos, 2008]. Néanmoins, ils souffrent encore de limitations lorsqu'il s'agit de faire des recommandations de séquences (recommandations d'items dans un ordre précis pouvant dépendre des pré-requis, de la progressivité requise, du contexte, des contraintes de temps, etc.). Certains modèles, tels que le modèle DANCE [Castagnos, 2015], intègrent cette dimension temporelle en suivant en temps réel l'évolution en diversité des ressources consultées par les utilisateurs pour mieux comprendre le contexte d'exploration. Dans [Bonnin, 2010], l'auteur propose également un modèle temporel capable de détecter des motifs de consultations fréquents dans un historique de consultations, afin de fournir des recommandations de ressources a priori liées à un même contexte. Néanmoins, si la modélisation temporelle et spatiale a été rendue possible [Zheng, 2015], les modèles de l'état de l'art s'intéressant à l'ordre dans lequel les recommandations doivent être faites ou à la qualité globale d'une séquence de recommandations sont encore trop rares.

Dans le cadre de cette thèse, nous nous attacherons à définir un nouveau formalisme et un cadre méthodologique permettant : (1) la définition de facteurs humains menant à la prise de décision et à la satisfaction des utilisateurs ; (2) la construction d'un modèle générique et multi-critères (contraintes physiques ou temporelles, diversité, progressivité, etc.), intégrant ces facteurs humains dans le but de recommander des ressources pertinentes s'inscrivant dans une séquence cohérente ; (3) une évaluation holistique de la satisfaction utilisateur vis-à-vis de son parcours de recommandations. L'évaluation des recommandations, tous domaines confondus, se fait pour l'heure recommandation par recommandation, chaque métrique d'évaluation prise indépendamment (précision, diversité, nouveauté, couverture, ...). Il s'agira de proposer un cadre plus complet mesurant l'évolutivité et la complétude du parcours.

Un tel modèle de recommandations multi-critères présente de nombreux cadres applicatifs. À titre d'exemple, il peut être utilisé dans le cadre de l'écoute de musique en ligne avec la recommandation de playlists adaptatives (recommandation de séquences de musique pour faire évoluer l'ambiance dans un lieu tel qu'un bar, pour susciter une certaine progressivité de l'émotion res-

sentie par le public, ou pour s'adapter aux attentes complémentaires/similaires/différentes d'un groupe). Il peut également être utile pour adapter le parcours de recommandation aux progrès de l'apprenant et au scénario pédagogique de l'enseignant dans un contexte d'e-éducation. Citons aussi le domaine touristique où ce modèle pourrait intégrer les contraintes spatiales et temporelles d'un environnement physique (villes, musées, etc.).

Mots-clés: Systèmes de recommandation, Systèmes multi-agents, Modélisation utilisateur.

Abstract

Recommender systems are a fundamental research topic at the intersection of several major disciplines such as machine learning, human-computer interaction and cognitive sciences. They also constitute an ambitious application framework for the community of researchers in Artificial Intelligence by their great complexity and the numerous constraints they generate.

The purpose of these systems is to improve the interaction between the general audience and the systems of search and access to information. It has become difficult to identify the most relevant items in the context of big data. The goal is thus to assist users in their explorations (whether in a virtual or physical environment), but also to propose items that may interest them but that they would not consult spontaneously.

Current systems have largely proven their added value and are based on various machine learning techniques (numerical or symbolic, supervised or not, etc.) [Castagnos, 2008]. Nevertheless, they still suffer from limitations when making recommendations of sequences (recommending items in a specific order may depend on requirements, progressiveness, context, time constraints, etc.). Some models, such as the DANCE model [Castagnos, 2015], integrate this temporal dimension by following in real time the evolution in diversity of resources consulted by users to better understand the exploration context. In [Bonnin, 2010], the author also proposes a temporal model capable of detecting frequent consultation patterns in a history of consultations, in order to provide a priori resource recommendations related to the same context. Nevertheless, while temporal and spatial modeling have been made possible [Zheng, 2015], state-of-the-art models that focus on sequence recommendations or on the overall quality of the sequence are still too rare.

In the framework of this thesis, we will focus on defining a new formalism and a methodological framework allowing : (1) the definition of human factors leading to decision making and user satisfaction ; (2) the construction of a generic and multi-criteria model (physical or temporal constraints, diversity, progressiveness, etc.), integrating these human factors in order to recommend relevant resources in a coherent sequence ; (3) a holistic evaluation of user satisfaction with its recommendation path. The evaluation of recommendations, all domains included, is currently done recommendation by recommendation with each evaluation metric taken independently (accuracy, diversity, novelty, coverage, ...). Thus, we expect a more comprehensive evaluation framework, measuring the progressiveness and the completeness of the path.

Such a multi-criteria recommendation model has many application frameworks. As an example, it can be used in the context of online music listening with the recommendation of adaptive playlists (recommendation of music sequences to change the atmosphere in a place such as a bar, to raise or lower the emotion felt by the audience progressively, or to adapt to the complementary/similar/different expectations of a group). It can also be useful to adapt the recommendation path to the learner's progress and the teacher's pedagogical scenario in an e-education context. Let us also mention the tourism field where this model could integrate the spatial and temporal constraints of a physical environment (cities, museums, etc.).

Keywords: Recommender systems, Multi-agent systems, User modeling.