Système coopératif de type égal-à-égal pour la recommandation : Application à la gestion et la recommandation de références bibliographiques.

Hajer Karoui

N° attribué par la bibliothèque

Université Paris-Nord

THESE

présentée par

Hajer KAROUI

pour obtenir le titre de

Docteur d'Université

Spécialité Informatique

Sujet:

Système coopératif de type égal-à-égal pour la recommandation : Application à la gestion et la recommandation de références bibliographiques

Soutenue publiquement le 11 Décembre 2007
devant le jury composé de

M. Jean-Paul	BARTHÈS	Rapporteur
Mme Catherine	BERRUT	Rapporteur
Mme Sylvie	DESPRÉS	Invitée
M. Alain	MILLE	Président
M. Rushed	KANAWATI	Invité
Mme Laure	PETRUCCI	Directrice
M. Enric	PLAZA	Examinateur
Mme Laurence	VIGNOLLET	Examinatrice

Remerciements

Je tiens à remercier Rushed Kanawati et Laure Petrucci qui m'ont permis de réaliser cette thèse, pour leurs conseils et la confiance qu'ils m'ont accordée durant toutes ces années.

Je tiens à remercier Alain Mille d'avoir accepter de présider mon jury de thèse.

Je voudrais remercier les rapporteurs Catherine Berrut et Jean-Paul Barthès pour l'intérêt qu'ils ont porté à mon travail.

Je tiens à remercier Enric Plaza d'avoir accepter d'examiner ma thèse et d'avoir fait tout le voyage.

Je remercie Laurence Vignollet d'avoir examiner ma thèse et d'avoir assister à ma soutenance de mi-parcours.

Je remercie Sylvie Després pour sa participation à mon jury de thèse.

Je remercie en particulier Younès Bennani, Catherine Recanati, Christian Codognet et Phillipe Dague pour leurs encouragements.

Je remercie tous les anciens et actuels thésards pour leur amitié et particulièrement Sujeevan Aseervatham, Farida Zerhaoui, Nora Touati, Jalila Sadki, Sana Hamdoun, Sara Boutouhami-Nouioua, Nejla Amara, Hakima Kadri Dahmani, Nicoleta Rogovschi, Farid Nouioua, Idir Khemouj, Djamel Berrabeh, Fayçal Djerourou, Lahcène Dehni, Hassine Moungla, Mourad Hakem, Lionel Falempe, Tsiriniaina Andriamampianina, Nistor Grozavu, Sébastien Guérif.

Je tiens à remercier tous les membres du laboratoire LIPN pour la bonne ambiance qui y règne, pour leur accueil, leur soutien, leur conseils et les discussions intéressantes que j'ai pu avoir avec eux.

Ma plus profonde gratitude va à mes parents qui m'ont fait confiance et qui m'ont donné toutes les chances de réussir. Qu'ils trouvent dans la réalisation de ce travail, l'aboutissement de leurs grands efforts.

Je remercie ma chère sœur Imen et mes frères Marouene et Mohamed Ali pour leur soutien moral et leur réconfort dans les moments difficiles ainsi que leur encouragement de près ou de loin.

Un remerciement particulier à mon fiancé et mon futur mari qui n'a cessé de m'encourager et de me soutenir.

Je remercie enfin, toute ma grande famille ainsi que mes amis pour leur encouragement malgré la distance qui nous sépare.

À ma très chère Maman MONIA

À mon très cher Papa YOUSSEF

Résumé

Dans cette thèse, nous explorons la réutilisation et le partage automatique des expériences passées des utilisateurs dans des tâches de recherche d'information. Le but est de proposer des recommandations pertinentes à l'utilisateur selon ses centres d'intérêt. Nous utilisons le raisonnement à partir de cas (RàPC) comme une méthodologie d'apprentissage et de modélisation de l'expérience des utilisateurs, et l'architecture de type égal-à-égal (P2P) afin de préserver l'autonomie des utilisateurs.

Pour illustrer notre approche, nous avons développé une application pilote COBRAS qui permet la gestion et la recommandation de références bibliographiques. Les utilisateurs partagent une même hiérarchie de thèmes. Cette hiérarchie est utilisée différemment par les différents utilisateurs. Dans ce système, la coopération entre les utilisateurs se fait via des agents logiciels. Chaque agent collabore avec les autres agents pour offrir des services à son utilisateur, notamment la recommandation implicite et intelligente de références pertinentes. Deux problématiques se présentent : comment obtenir les références pertinentes et comment choisir un ensemble d'agents (comité) avec qui collaborer ? Pour résoudre ces problèmes, nous nous sommes basés sur l'analyse et l'exploitation des historiques des interactions entre les agents. L'utilisation de la méthodologie RàPC a deux objectifs :

- déterminer un ensemble d'agents intéressants à interroger ;
- chercher un ensemble de références pertinentes à proposer à l'utilisateur.

Une expérimentation est réalisée afin de valider le système, utilisant des références bibliographiques extraites de la base de données DBLP et classées par thèmes selon la hiérarchie ACM. Les résultats d'expérimentation obtenus montrent que l'utilisation d'une approche de RàPC coopérative pour la recommandation de comités et de références permet l'amélioration des performances du système (réduction du nombre d'agents à contacter et du nombre de messages échangés) ainsi que la pertinence des recommandations fournies (augmentation de la précision des recommandations proposées).

Mots-clés: Système de recommandation, Formation de comités, Raisonnement à partir de cas, Partage automatique d'expériences, Coopération d'agents, Système d'égal-à-égal.

Abstract

In this thesis, we explore reuse and automatic sharing of user past experiences in information retrieval tasks. The goal is to provide a user with relevant recommendations according to her/his interests. We use case-based reasoning (CBR) as learning and user experience modeling methodology. We also use a peer-to-peer (P2P) architecture in order to protect the user's autonomy. To illustrate our approach, we developed a pilot application COBRAS. It aims at managing and recommending bibliographical references. All users share a same topic hierarchy. However, the same hierarchy will be used differently by different users. In this system, user cooperation is done via personal software agents. Indeed, each agent collaborates with other agents in order to offer services to her/his user, mainly intelligent and implicit recommendation of relevent references. Key problems are how to obtain relevant references and how to choose a set of peer agents (committee) that can provide the most relevant recommendations ? To handle both of these problems, we analyse and exploit the interaction backgrounds and users' experiences. Case-based reasoning methodology is used for two reasons :

- it is used to search for a set of appropriate peers to collaborate with.
- it is also used to search for relevant references from the selected agents.

In order to validate our system, we carried out experimentations using bibliographical references extracted from the DBLP database, and classified by topics according to the ACM hierarchy. Experimental results show that the use of a cooperative CBR approach for committee and reference recommendation improves the system performances (by reducing the numbers of contacted agents and exchanged messages) as well as the recommendation relevance (by increasing the proposed recommendation precision).

Keywords: Recommender system, Committee formation, Case-based Reasoning, Automatic experience sharing, Peer-to-Peer systems.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

Nous explorons dans ce travail l'utilisation de la méthodologie du raisonnement à partir de cas et de système de type égal-à-égal (Peer-to-Peer) pour le partage automatique d'expériences au sein d'un groupe organisé d'utilisateurs. Nous désignons par groupe organisé d'utilisateurs un groupe bien identifié où les utilisateurs appartiennent à une même organisation et qui peuvent donc partager des centres d'intérêt. Par exemple, un groupe de R&D, un groupe de chercheurs dans un laboratoire ou dans une équipe de recherche, etc. L'objectif de ce travail consiste à étudier et à proposer des méthodes de réutilisation d'expériences et de formation de comités dans un environnement distribué. Nos contributions sont validées par le développement et la mise en œuvre d'une application pilote de recommandation de références bibliographiques.

1.1 Contexte

L'évolution grandissante du réseau mondial en termes aussi bien de volume d'information disponible que de services offerts, a rendu nécessaire la proposition d'outils d'aide à la localisation et l'exploitation d'informations pertinentes. Trois grandes approches complémentaires sont explorées dans la littérature afin de faciliter l'accès à l'information :

- La première consiste à renforcer la description sémantique de l'information disponible (i.e. le Web sémantique (T.Berners-Lee, 1999)).
- La deuxième consiste à améliorer l'efficacité des méthodes de recherche d'information par le biais de l'élaboration de modèles des utilisateurs (i.e. profils des utilisateurs) qui peuvent être utilisés par exemple pour la reformulation ou l'expansion des requêtes (Bottraud *et al.*, 2004), la sélection et le filtrage des résultats de recherche (Bottraud *et al.*, 2003b).
- La troisième consiste à fournir des outils de coopération afin de permettre la réutilisation des expériences passées des utilisateurs. Cette collaboration devrait permettre d'améliorer non seulement les compétences individuelles mais également les performances de tout le groupe. La coopération entre les utilisateurs peut se faire d'une manière directe (par

l'utilisation d'outils collecticiels (Sire, 1999; Smith *et al.*, 1998)) ou indirecte (par des agents logiciels interposés (Martin *et al.*, 1999; Kanawati et Malek, 2002; Enembreck, 2003; Stuber *et al.*, 2004)).

Notre travail s'insère dans la troisième approche fondée sur le partage et la réutilisation des expériences passées du groupe et dont la collaboration se fait d'une manière indirecte entre les utilisateurs via un agent logiciel assistant.

1.2 Problématique

Le partage d'expériences pose plusieurs problèmes qui peuvent être liés :

1. ***à l'expérience elle-même*** tels que :
   - sa modélisation : c'est-à-dire sa représentation et sa formalisation,
   - son acquisition : en trouvant un moyen approprié pour la collecte des nouvelles expériences,
   - sa réutilisation : qui englobe la sélection, l'évaluation ainsi que l'adaptation de l'expérience au contexte courant afin de pouvoir la réutiliser.
2. ***aux acteurs*** : c'est-à-dire le choix des personnes avec qui les expériences seront partagées. Le but est de former des comités pertinents au sein desquels se fait le partage et la réutilisation des expériences.

Dans ce travail, nous nous intéressons essentiellement aux deux problèmes liés à la réutilisation de l'expérience et à la problématique de formation de comités.

Pour illustrer notre approche, nous avons mis en œuvre un système de gestion et de recommandation de références bibliographiques nommé COBRAS (COoperative Bibliographical Recommender Agent System). Il s'agit d'une application pilote où le problème de modélisation de l'expérience est simple et va donc nous permettre de nous concentrer sur les moyens de partage et de réutilisation de l'expérience.

Dans COBRAS, la coopération entre les utilisateurs se fait via des agents logiciels. En effet, chaque utilisateur est assisté par un agent logiciel personnel qui l'aide à gérer sa base bibliographique. Pour que chaque utilisateur garde son autonomie et le contrôle de ses propres données, nous avons choisi une architecture de type égal-à-égal (P2P). Chaque agent collabore avec les autres agents afin d'offrir des services à son utilisateur, notamment la recommandation de références pertinentes qui se fait d'une manière implicite.

Chaque agent assistant doit être capable de :

1. détecter les centres d'intérêt de l'utilisateur associé,
2. associer un comité d'agents à chaque thème d'intérêt de l'utilisateur,
3. retourner les documents locaux les plus pertinents à une requête de recommandation adressée par un autre agent.

L'approche que nous proposons consiste à doter chaque agent de la capacité d'apprendre à effectuer ses tâches en observant les actions de l'utilisateur (détection des centres d'intérêt) et en explorant les traces d'interaction entre les agents. La méthodologie du raisonnement à partir de cas (RàPC) (Schank, 1983; Sycara, 1988; Riesbeck et Schank, 1989) est choisie comme un moyen d'apprentissage incrémental et de modélisation de l'expérience des utilisateurs (Bergmann, 2002).

L'application présente donc un environnement favorable pour tester et valider notre approche de partage d'expériences et de coopération. L'application traite des données semi-structurées (XML) afin de faciliter la représentation des données. Le but de ce travail ne se focalise pas sur la représentation et la modélisation de l'expérience mais se concentre plutôt sur la proposition d'approches pour la faire réutiliser et partager au sein d'un groupe d'utilisateurs. Ce système possède en particulier les caractéristiques suivantes :

- système distribué dans lequel chaque agent est autonome ;
- architecture d'égal-à-égal où chaque utilisateur garde localement ses données et contrôle les schémas de partage avec autrui ;
- les informations (par exemple, centres d'intérêt, évaluations) sont déduites et interprétées par l'agent personnel, à partir de l'observation du comportement de l'utilisateur associé : aucune demande d'information n'est faite à l'utilisateur ;
- la recommandation de références bibliographiques et d'agents selon les centres d'intérêt de l'utilisateur se fait d'une manière implicite.

1.3 Contribution

Nous avons abordé dans ce travail deux problèmes :

- Partage et recommandation de documents au sein d'un groupe d'utilisateurs. Il s'agit d'un groupe organisé d'utilisateurs partageant des centres d'intérêt. Tous les membres du groupe partagent une hiérarchie commune de thèmes afin de classifier leurs documents. Cependant, l'utilisation de la hiérarchie est personnalisée. Le calcul des recommandations se base sur la gestion des correspondances entre les utilisations individuelles de la hiérarchie pour trouver les documents pertinents dans les bases locales des utilisateurs.
- La formation de comités dans un système P2P. Le but est de trouver un comité d'agents pertinents par rapport à une requête de recommandation. Le calcul des comités se base sur l'exploitation des traces des interactions entre les agents.

1.4 Organisation du mémoire

Le mémoire est organisé en six chapitres :

- Le premier chapitre introduit le contexte et la problématique de notre travail.

- Le deuxième chapitre présente un état de l'art sur les systèmes RàPC pour l'assistance et la recommandation en recherche d'information.
- Le troisième chapitre est consacré à la formation de comités dans les systèmes égal-à-égal.
- Dans le quatrième chapitre, nous détaillons le système COBRAS : le module de la formation de comités et le module de la recommandation de références.
- Dans le cinquème chapitre, nous décrivons la mise en œuvre et la validation de notre système.
- Dans le sixième chapitre, nous concluons et nous donnons quelques perspectives à notre travail.

Chapitre 2

Systèmes RàPC pour l'assistance et la recommandation en recherche d'information

2.1 Introduction

L'objectif d'un système de recommandation est d'aider les utilisateurs à faire leurs choix dans un domaine où ils disposent de peu d'informations, pour trier et évaluer les alternatives possibles (Resnick et Varian, 1997). Un des défis majeurs dans le domaine de la conception de systèmes de recommandation est de produire des recommandations pertinentes et personnalisées tout en minimisant l'effort et l'implication de l'utilisateur, requis pour leur calcul. Les évaluations individuelles sont valorisées dans les systèmes de recommandation puisqu'elles sont considérées dans le calcul des recommandations.

L'expérience d'un utilisateur est importante pour le calcul des recommandations. Cette expérience est acquise suite aux interactions entre les différents utilisateurs à la fois producteurs et consommateurs de recommandations. Elle aidera les utilisateurs à trouver leurs collaborateurs pertinents par rapport à un centre d'intérêt donné et à trouver les références pertinentes tout en minimisant leur effort.

Les attentes de la réutilisation de l'expérience consistent à faire des améliorations tout au long du processus de résolution de problèmes. Ces améliorations sont justifiées comme suit :

- Réduction du temps de résolution de problèmes : la réutilisation mène en général à une réduction du temps de résolution de problèmes car ces derniers n'ont plus besoin d'être résolus plusieurs fois, si les problèmes sont similaires et le coût d'adaptation nécessaire est beaucoup moins important que le coût de la résolution complète d'un problème.
- Amélioration de la qualité de la solution : la qualité de la solution peut être améliorée car l'expérience aide à trouver de bonnes solutions et/ou à éviter les mauvaises.

- Moins de compétences exigées : les solveurs de problèmes ont moins besoin de compétences et de leurs propres expériences car l'accès aux expériences des autres est disponible. Par conséquant, les solveurs d'expériences peuvent supporter la résolution de problèmes qu'ils ne peuvent pas résoudre par eux-même autrement.
- Enrichissement de l'expérience individuelle : l'expérience individuelle de chaque agent s'améliore au fur et à mesure de la réutilisation de sa propre expérience et surtout de la réutilisation des expériences des autres.
- Apprentissage : la réutilisation de l'expérience permet d'apprendre mieux et plus rapidement.

Nous commençons par définir la notion d'expérience et les étapes nécessaires à sa réutilisation. Ensuite, nous définissons le raisonnement à partir de cas comme une méthodologie permettant la gestion d'expériences et l'apprentissage. Finalement nous étudions quelques systèmes assistants pour la recherche d'information et nous terminons par une discussion.

2.2 Partage d'expériences

« Nous payons cher une expérience que nous pourrions trouver à bon marché chez le voisin », *(Aristophane, $V^{ème}$ siècle avant J. C.).*

La gestion d'expériences s'occupe de la collecte, la modélisation, le stockage, la réutilisation, l'évaluation et la maintenance d'expériences. Pour partager l'expérience, elle doit être définie, modélisée et acquise.

Il n'y a pas de consensus sur la définition de l'expérience : cela est dû à ses spécificités. En effet, l'expérience est personnelle, relative, contextualisée et a un caractère cognitif. L'expérience peut être considérée comme un ensemble de leçons apprises pour contribuer à l'amélioration des compétences de l'utilisateur lui-même et des autres utilisateurs du système, si elle est partagée. L'expérience est souvent située dans un contexte bien spécifique de résolution de problèmes. Elle présente une plus value, comme une utilité future. Par exemple, dans les systèmes de gestion de projets, une expérience peut être vue comme un ensemble de décisions prises dans un contexte donné ou un ensemble de leçons apprises lors de la planification et l'exécution du projet (Tautz *et al.*, 2000). Dans le domaine de la recherche d'information, une expérience peut être vue comme une session de recherche (Jéribi *et al.*, 2001). Cette expérience est caractérisée par un contexte de recherche, des objectifs, des moyens mis en œuvre et des résultats obtenus.

La modélisation de l'expérience est la tâche de base pour la sélection d'expériences et leur réutilisation. Modéliser une expérience signifie trouver un moyen approprié pour la représenter et la formaliser si nécessaire. La représentation doit permettre une recherche et une adaptation efficace de l'expérience. D'un point de vue informatique, l'Intelligence Artificielle fournit des méthodes de représentation et de traitement d'expériences. Le RàPC, qui est un domaine de recherche active, s'adresse à ce problème en particulier (Champin *et al.*, 2004).

Il convient comme un principe et une méthodologie pour la gestion d'expériences et une technologie pour soutenir les systèmes d'information à base d'expériences (Bergmann, 2002).

Le partage de l'expérience permet d'avoir accès à l'expérience de différentes personnes, évitant ainsi la résolution de problème à partir de zéro et permet surtout de profiter des expériences précieuses des autres pour enrichir sa propre expérience.

Selon (Bergmann, 2002), toutes les connaissances reliées à la réutilisation d'expériences peuvent être divisées en trois conteneurs de connaissances :

- ***le vocabulaire*** : c'est le vocabulaire du domaine, il présente la base de la représentation d'expérience et de toutes les connaissances. Il définit les entités d'information et les structures pouvant être utilisées. Le vocabulaire spécifie le langage utilisé pour parler du domaine mais n'exprime aucune connaissance sur le domaine.
- ***la base d'expériences*** : c'est le conteneur central de connaissances qui présente la collection d'expériences disponibles pour la réutilisation dans un domaine. La base d'expériences peut être vue comme une collection d'expériences, ayant chacune une signification qui lui est propre. La base d'expériences peut être structurée, reliant ainsi les expériences les unes aux autres. Plusieurs structures sont possibles, par exemple, des structures exprimant une classification d'expériences selon certains critères du domaine, ou des structures pour l'organisation hiérarchique d'expériences. La base d'expériences est donc une collection structurée d'expériences qui sont représentées en utilisant les entités d'information et les structures définies dans le vocabulaire.
- ***les connaissances reliées à la réutilisation*** : afin de soutenir la résolution de problèmes en réutilisant l'expérience, des connaissances sont nécessaires en plus de l'expérience elle-même. Ce sont typiquement des connaissances générales sur l'évaluation des expériences précédentes dans un nouveau contexte de résolution de problème. Cela inclut des connaissances sur l'utilité de la réutilisation d'une certaine expérience pour résoudre un problème particulier et aussi des connaissances sur comment adapter une certaine expérience pour mieux convenir à la situation courante. Les connaissances reliées à la réutilisation sont représentées en utilisant les entités d'information et les structures définies dans le vocabulaire. Elles peuvent avoir différentes formes telles que :
  - des connaissances sur la similarité entre les situations de problèmes,
  - des ontologies de domaines,
  - des connaissances de transformation de l'expérience, ou
  - des modèles de résolution de problèmes dans un domaine.

La réutilisation d'expérience est un des buts fondamentaux de la gestion d'expériences qui exige :

- d'avoir accès à l'expérience et de sélectionner l'expérience appropriée pour la réutiliser,
- d'évaluer l'utilité d'une expérience sélectionnée dans le contexte du nouveau problème devrant être résolu,

– si c'est nécessaire, d'adapter l'expérience au nouveau contexte,
– de résoudre le nouveau problème ou aider l'utilisateur à résoudre le nouveau problème en réutilissant l'expérience.

L'adaptation d'expérience est typiquement une tâche intensive de connaissances. L'expérience peut être changée si la situation courante diffère de la situation dans laquelle l'expérience a eu lieu. Il peut être nécessaire de combiner différentes expériences si chacune d'elles aide seulement à résoudre une petite partie de tout le problème.

Il est nécessaire d'avoir un retour du processus d'adaptation d'expérience au processus de recherche de l'expérience : si l'adaptation échoue ou conduit à un résultat inapproprié, la recherche doit recommencer pour trouver des expériences supplémentaires, plus appropriées, comme source pour l'adaptation (Smyth et Keane, 1993).

Durant la réutilisation d'expérience dans la résolution d'un problème, l'expérience réutilisée peut être évaluée dans le contexte du nouveau problème. L'évaluation peut être en terme de l'appropriation de l'expérience récupérée. Une telle évaluation est importante pour améliorer, d'une manière continue, le processus de réutilisation d'expériences. La réutilisation d'une expérience dans le contexte d'un problème particulier, doit être vue à partir des perspectives de son utilité. Plusieurs expériences peuvent être utilisées avec différentes utilités et une leçon peut être plus utile qu'une autre (Bergmann, 2002). La notion d'utilité est basée sur la théorie économique de (Neumann et Morgenstern, 1947) qui décrit les connexions entre l'utilité, les préférences et la prise de décisions humaines. La préférence est l'expression d'un choix, en raison de critères soit subjectifs (goûts), soit objectifs, basés sur des critères clairement énoncés. Cette formulation de l'utilité a été très généralement adoptée dans la modélisation de choix lorsque les événements sont probabilisables. Dans le contexte de la réutilisation d'expériences, l'utilité fait la connection entre un problème particulier et une leçon particulière capturée dans une expérience. L'utilité est une propriété du domaine de résolution de problème selon la perspective de réutilisation d'expériences. Elle remplace les conditions de vérité traditionnelles (l'exactitude d'une solution à un problème) par une qualité de classement fine. La valeur de l'utilité peut avoir plusieurs significations qui dépendent de la tâche de résolution de problème courante. Mais dans tous les cas, une expérience avec une grande utilité est préférée à une autre avec une basse valeur d'utilité. Selon Bergmann et Wilke (Bergmann et Wilke, 1998), les utilités pour les expériences peuvent être interprétées de diverses manières telles que :

– la qualité de la solution obtenue en réutilisant l'expérience,
– l'effort ou le coût de la résolution du problème gagné par la réutilisation d'expérience,
– le coût monétaire gagné par la résolution plus rapide du problème par la réutilisation d'expérience,
– la probabilité que la solution suggérée par l'expérience soit correcte,
– la satisfaction de l'utilisateur avec l'expérience présentée.

La fonction d'utilité pour un espace de problèmes P et un espace de leçons L possibles

enregistrées en expériences est $U : P \times L \rightarrow R$ qui affecte à chaque paire (problème, leçon), une valeur d'utilité appartenant à l'ensemble des nombres réels. Elle présente l'utilité de la réutilisation de la leçon pour résoudre le problème. Le problème fondamental des fonctions d'utilité est qu'elles peuvent être difficilement déterminées complètement dans un domaine de problèmes complexes. Nous pouvons souvent obtenir seulement des connaissances partielles telles que :

1. Pour un problème, l'utilité d'une leçon est plus élevée, égale ou inférieure à l'utilité d'une autre leçon.
2. L'utilité d'une certaine leçon dépend de certaines caractéristiques de la leçon elle-même ou du problème.

Ainsi, une fonction de préférence est induite par une fonction d'utilité. Dans le contexte d'un problème p, l'action de choisir la leçon l_i pour la réutilisation est préférée à l'action de choisir la leçon l_j, notée $(l_i \succ_p l_j)$ si seulement si $U(p, l_i) > U(p, l_j)$. Cette relation de préférence est souvent seulement partiellement connue. Sont seulement connues les préférences $c_i \succ_p c_j$ pour un certain problème p et certaines leçons l_i et l_j (Bergmann, 2002).

Les expériences disponibles doivent être mises à jour d'une manière continue. À cause des changements de l'environnement, l'expérience peut avoir seulement une durée de vie limitée. Les expériences invalides doivent être identifiées et supprimées ou mises à jour. De plus, la manière de modéliser l'expérience existante peut être sujette à des changements. La maintenance d'expériences peut être declenchée par une évaluation négative d'expérience ou peut être exécutée par précaution.

2.3 Le raisonnement à partir de cas

Inspiré des travaux de M. Minsky sur les cadres de données (Minsky, 1975), le paradigme du Raisonnement à Partir de Cas (RàPC) a été défini par R. Schank (Schank, 1983) à travers le modèle de « mémoire dynamique ».

Les expériences qui servent à résoudre de nouveaux problèmes, sont constituées de problèmes déjà résolus appelés *cas sources*. Un cas est composé de deux parties décrivant un problème et la solution qui lui est appliquée. Lorsqu'un nouveau problème se présente, il est intégré dans un cas, appelé *cas cible*, dont la partie solution est inconnue et sera apportée par le raisonnement. Le cas cible est résolu en trouvant un ou plusieurs cas sources qui lui sont similaires et en adaptant leurs solutions à la nouvelle situation. Une caractéristique du RàPC est qu'il s'agit d'une approche incrémentale, c'est-à-dire qu'une nouvelle expérience peut être mémorisée à chaque fois qu'un problème est résolu, ce qui la rend disponible pour la résolution de problèmes futurs. Cela entraîne la rétention d'une mémoire d'anciens problèmes et de leurs solutions pour la résolution de nouveaux problèmes (Main *et al.*, 2000). Le principe consiste à chercher dans

la base de cas et à trouver un cas ayant des spécifications similaires du problème que le cas courant.
Un cycle de RàPC se compose essentiellement de quatre phases (Aamodt et Plaza, 1994), auxquelles est ajoutée une phase préliminaire d'élaboration au début du cycle (Mille, 2006b). Le cycle comporte donc les étapes suivantes :

- L'élaboration du problème cible,
- La recherche des cas sources dans la base de cas,
- La réutilisation des solutions trouvées pour construire la solution au problème cible,
- La révision de la solution proposée,
- L'apprentissage de la nouvelle expérience afin d'enrichir la mémoire du système.

2.3.1 Cycle du raisonnement à partir de cas

Le cycle de RàPC est illustré dans la figure 4.2.2.

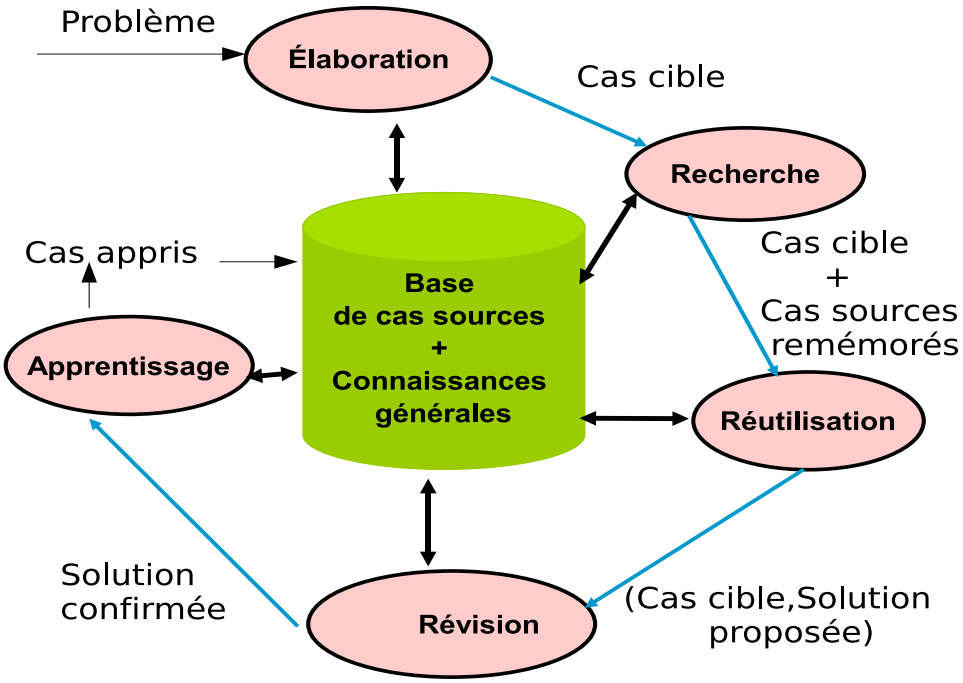

FIG. *2.1: Cycle du raisonnement à partir de cas*

1. ***L'élaboration*** : avant d'exécuter un cycle RàPC, le cas cible doit être d'abord élaboré. Cette tâche est importante puisqu'elle va influer sur tout le processus de recherche de la solution. C'est dans ce cadre que Mille (Mille, 2006b) propose une version légèrement complétée du cycle RàPC en ajoutant explicitement cette phase, qui se révèle être un aspect important d'ingénierie de connaissances. L'élaboration d'un nouveau cas consiste à

faciliter la description du problème pour permettre la recherche d'un cas dont la solution sera la plus facilement adaptable. L'adaptabilité d'un cas se mesure à « l'effort » de l'adaptation qui sera nécessaire pour que la solution du cas source puisse servir de base à la solution courante du cas cible. La méthode générale consiste à compléter ou filtrer la description d'un problème en se fondant sur la connaissance du domaine pour inférer tout ce qui est possible à partir d'une description éventuellement incomplète, et pondérer les descripteurs en fonction des dépendances identifiées entre les descripteurs du problème cible et les descripteurs de la solution recherchée. C'est ainsi que l'ontologie du domaine va être utilisée pour compléter les descripteurs importants et que d'une manière générale, l'ensemble des règles définies sur le domaine vont être considérées. Le but de cette phase est d'orienter la recherche d'une solution adaptable. Élaborer un cas cible consiste donc à affecter des descripteurs au nouveau cas et construire des descripteurs possédant une sémantique liée au problème pour anticiper au maximum l'adaptabilité des cas qui seront remémorés.

2. ***La recherche*** : cette phase permet de remémorer un ou plusieurs cas sources de la base de cas, à partir de la description de la partie problème du cas cible. Les méthodes de remémoration dépendent de la façon dont la mémoire est organisée. Il est souhaitable que la remémoration soit la plus correcte possible, c'est-à-dire que l'expérience rappelée soit, dans la mesure du possible, celle qui permet d'obtenir la meilleure solution à partir des cas présents dans la base de cas. Les tâches effectuées durant cette phase sont : l'identification des caractéristiques pertinentes du problème, la remémoration et la sélection des meilleurs cas parmi les cas sources. La remémoration des cas sources est basée sur des mesures de similarité entre les cas sources et le cas cible (McSherry, 2002). La similarité permet de mesurer le degré d'appariement entre deux cas. Elle peut dépendre, en plus de la notion de similarité, de celle de la diversité (Smyth et McClave, 2001). Le but consiste alors à remémorer des cas similaires au cas cible et de choisir parmi ces cas ceux qui ne sont pas très similaires entre eux. La remémoration des cas sources se base aussi sur des connaissances d'adaptation (Smyth et Keane, 1995). Plusieurs recherches sur l'adaptation ont été menées : les démarches unificatrices, les catalogues et les méthodes d'acquisition de connaissances d'adaptation. Les démarches unificatrices visent à proposer des modèles généraux de l'adaptation sous différents angles (principes, algorithmes, etc.) et peuvent aider un concepteur de RàPC à mieux appréhender l'adaptation dans sa globalité (par exemple, (Fuchs *et al.*, 2000)). D'autres travaux visent à mettre en évidence des catalogues de stratégies d'adaptation qui sont susceptibles de s'appliquer à plusieurs domaines (Riesbeck et Schank, 1989; Lieber, 2002). Les démarches d'acquisition de connaissances d'adaptation considèrent l'adaptation dans le cadre d'un domaine d'application donné et visent à mettre en évidence des principes généraux destinés non à l'adaptation elle-même, mais aux moyens de l'expliciter dans le domaine d'application considéré (Hanney et Keane, 1996;

Fuchs et Mille, 2000; Lieber *et al.*, 2001).

3. ***La réutilisation*** : cette phase consiste à adapter la solution du cas source remémoré pour résoudre le cas cible. L'adaptation consiste à effectuer un raisonnement par analogie (Lieber *et al.*, 2004) : *sachant que la solution du cas cible est à la solution du cas source ce que le cas cible est au cas source, connaissant le cas source et sa solution ainsi que le cas cible, que vaut la solution du cas cible ?* D'après (Fuchs *et al.*, 1999), l'adaptation peut être considérée comme un plan dont l'état initial est la solution de départ et l'état final est la solution adaptée. Le fait de modéliser l'adaptation de cette façon permet d'envisager de façon théorique et pratique la combinaison de l'adaptation avec la remémoration, pour être en mesure de retrouver un cas adaptable lors de la remémoration et de guider l'adaptation d'un tel cas.
4. ***La révision*** : cette phase consiste à évaluer la solution proposée par l'étape de réutilisation. Si le résultat de la révision est satisfaisant, nous passons à l'étape suivante qui est celle d'apprentissage. Par contre, si le résultat de l'évaluation n'est pas bon, la solution est révisée en utilisant les connaissances du domaine ou bien en consultant un expert humain. Un point de départ pour corriger le système peut s'appuyer sur :
   - Un expert humain : qui présente un expert du domaine ; par exemple, le travail de (Cordier *et al.*, 2007).
   - Un processus automatique : par le système lui-même en utilisant la base de cas ; par exemple, le travail de (Malek et Kanawati, 2004).
   - L'utilisateur : qui donne son évaluation par rapport à la solution retournée par le système RàPC ; par exemple, le travail de (Karoui *et al.*, 2006).
5. ***L'apprentissage*** : c'est le processus qui permet d'incorporer ce qui est utile à retenir, à partir de l'étape de résolution d'un problème, dans la base de cas. L'apprentissage peut s'effectuer, par exemple, à partir du succès ou de l'échec dans la résolution du cas cible. La mémorisation du nouveau cas résolu (par exemple selon sa qualité) est l'occasion d'enrichir la base de cas et les connaissances (Mille, 2006b) :
   - naturellement, un cas supplémentaire résolu augmente l'expérience du système,
   - en cas de révision, la connaissance générale peut être modifiée et en particulier les connaissances duales liées aux tâches de recherche et d'adaptation,
     - les connaissances d'influences peuvent être raffinées pour piloter l'adaptation. Ces connaissances d'influence sont duales des connaissances de similarité ; elles sont directement liées aux poids utilisés pour pondérer la mesure globale de similarité.
     - des dépendances nouvelles peuvent être découvertes, ...
   - il peut être utile de garder la « trace » de l'ensemble du cycle avec le détail des corrections faites. Cette trace pourra être utilisée pour considérer ce cas comme un modèle pour « corriger » en s'inspirant de cette correction, une nouvelle adaptation qui se ferait avec le même type de similarité.

Ce cas est organisé et indexé dans la base et de nouvelles connaissances pourront être synthétisées.

2.3.2 Structure d'un cas

Il existe de nombreux travaux sur la représentation des cas, mais la représentation la plus communément utilisée est celle structurée en liste de descripteurs (qui peuvent être des objets complexes). Un cas, son problème et sa solution sont donc décrits par un ensemble de descripteurs. Un descripteur d est défini par une paire $d = (a, v)$ où a est un attribut et v est la valeur qui lui est associée dans ce cas. Conformément à ce vocabulaire, source et cible sont définis de la manière suivante :

- source = $d_1^s..d_n^s$ où les d_i^s sont des descripteurs du problème source.
- Sol(source) = $D_1^s..D_m^s$ où les D_i^s sont des descripteurs de la solution source.
- cible = $d_1^c..d_n^c$ où les d_i^c sont des descripteurs du problème cible.
- Sol(cible) = $D_1^c..D_m^c$ où les D_i^c sont des descripteurs de la solution cible à retourner par le cycle RàPC.

Les informations contenues dans un cas dépendent du domaine d'application et des objectifs pour lesquels ce cas est construit.

La base de cas dans un système de RàPC est la mémoire qui contient tous les cas précédemment retenus. Lors de la création d'une base de cas, trois points principaux doivent être considérés (Main *et al.*, 2000) :

- La structure et la représentation des cas.
- Le modèle de la mémoire utilisée pour organiser la base de cas.
- La sélection des indices qui sont utilisés pour identifier chaque cas.

Afin de permettre la remémoration d'un cas, sa partie problème est décrite par un ensemble de caractéristiques pertinentes appelées « indices ». Ces indices vont déterminer dans quelles situations et dans quels contextes les cas seront remémorés. Quand un cas est ajouté à la base de cas, le problème qui se pose est de savoir quels indices générer pour ce cas. Le problème du choix des indices dépend du domaine d'application, mais ces indices doivent vérifier certaines propriétés (Kolodner, 1996). En effet, les indices sont construits de façon à être :

- *Prédictifs* afin de jouer un rôle dans la détermination d'une solution pour un nouveau problème.
- *Suffisamment abstraits* de manière qu'un cas puisse être utilisé plusieurs fois pour la résolution de différents problèmes.
- *Suffisamment concrèts* afin de pouvoir être reconnus le plus rapidement possible pour la résolution d'un nouveau problème.

Nous avons adopté le RàPC comme un moyen de gestion d'expériences et d'apprentissage, pour les raisons suivantes :

1. Nos connaissances du domaine sont insuffisantes pour construire un modèle causal, surtout quand nous nous basons uniquement sur l'observation du comportement de l'utilisateur et la déduction de ses centres d'intérêts : le RàPC peut fonctionner avec seulement un ensemble de cas de domaine (Main *et al.*, 2000).
2. Initialement, notre système ne dispose pas d'expériences. Celles-ci seront créées au fur et à mesure : le RàPC peut être créé avec un ensemble limité d'expériences developpées d'une façon incrémentale et peut fonctionner avec un ensemble de cas.
3. Les agents apprennent et acquièrent petit à petit de l'expérience en coopérant les uns avec les autres et en évaluant leur apport : le RàPC effectue un apprentissage incrémental de cas durant son utilisation. L'ajout de cas pertinents permet de couvrir plus de problèmes et permet d'améliorer les résultats qu'il fournit (Main *et al.*, 2000).
4. Notre but consiste à faire partager les expériences et d'accélérer le processus de recherche : le RàPC permet d'éviter la répétition de toutes les étapes nécessaires pour arriver à la solution en réutilisant une solution précédente.

2.3.3 Conditions d'application du RàPC

Le RàPC présente de nombreux atouts dans le domaine de l'intelligence artificielle. Toutefois, l'application de ce type de raisonnement repose sur des hypothèses concernant le type d'application ou le domaine d'étude (Kolodner, 1996) :

- Similarité : deux problèmes similaires acceptent des solutions similaires.
- Régularité : les mêmes actions menées dans les mêmes conditions ont des effets identiques ou proches.
- Typicalité : les problèmes se répètent et les expériences apprises peuvent être utiles dans le futur.
- Cohérence : de petits changements dans l'univers nécessitent seulement de petits changements dans la façon dont nous l'interprétons et dans les solutions.
- Facilité d'adaptation : les différences entre les descriptions des problèmes sont faibles et peuvent être prises en compte facilement.

2.3.4 Du RàPC au RàPE

Dans notre contexte de travail, un cas est une connaissance qui représente une expérience individuelle. La base de cas d'un agent présente l'ensemble de ses expériences acquises lors de ses activités de recherche d'information. Notre but est de faire partager les expériences des différents agents du système et de mettre l'expérience du groupe au profit de tous. Le

RàPE présente une extension du RàPC. Ce nouveau type de raisonnement a été étudié par Mille (Mille, 2006b) et par Bergmann (Bergmann, 2002). Cependant, plusieurs chercheurs de RàPC ont confondu le RàPE avec le RàPC. Le RàPC est une sorte de RàPE se basant sur la réutilisation d'expériences antérieures encapsulées comme la base pour traiter de nouvelles situations similaires.

Étant donné que dans le RàPC, la structure de cas est définie a priori et qu'il y a de faibles possibilités d'adaptation au contexte et à l'utilisateur, Mille et Prié (Mille et Prié, 2006) proposent une extension du paradigme RàPC en RàPE dont l'idée consiste à se donner la possibilité de définir des cas en fonction des besoins et des contextes d'utilisation, dont la liste n'est jamais figée. Pour cela, il faut enregister une trace correspondant à une tâche générique d'utilisation du système. Les traces offrent la possibilité de construire dynamiquement de nouvelles structures de cas et d'étendre le contexte des cas si c'est nécessaire (Mille, 2006b). Dans le RàPE, un cas est un épisode découpé dans la trace. Un problème s'exprime en fonction de ce qu'a déjà fait l'utilisateur. L'assistance dans ce cas peut être fondée sur des épisodes précédents. Un épisode est une sous-partie de la trace correspondant à la signature d'une tâche particulière. La notion de signature de tâche désigne un motif récurrent dans les traces, qui est un événement marquant relié à au moins une situation. L'expérience de l'utilisateur est récupérée à partir de la trace primitive.

L'idée principale consiste à apprendre progressivement des contextes pratiques à partir des interactions des utilisateurs avec leur système. Mille (Mille, 2006a) propose d'exploiter les traces d'un environnement de calcul comme un enregistrement possible et indirecte de connaissances qui émerge quand l'utilisateur exécute ses tâches avec l'aide du système. Un cycle RàPE élabore dynamiquement des épisodes pouvant être potentiellement utiles dans des traces disponibles selon des « signatures de tâche ». L'épisode cible est construit avec l'aide d'autres épisodes proposés sous le contrôle de l'utilisateur. Les traces stockées sont des contenaires d'épisodes potentiels pouvant être révélés dans de nouvelles situations. La représentation d'expérience doit permettre une recherche et une adaptation efficaces de l'expérience. Le RàPC fournit un moyen approprié pour la représentation d'expérience. Selon Kolodner (Kolodner, 1993) : « Un cas est une partie de connaissances contextualisées représentant une expérience qui apprend une leçon fondamentale pour la réalisation des buts du raisonneur ». Un cas structure une expérience d'une certaine façon : le choix particulier de cette structure dépend fortement du domaine de gestion d'expériences et de la tâche qui doit être supportée par la réutilisation d'expérience. Les composants d'un cas sont :

1. Une partie problème (caractérisation) : décrivant l'expérience d'une façon qui permet d'évaluer son utilité dans une situation particulière. Cette partie de caractérisation fonctionne comme un type d'index pour la partie solution (leçon). Cependant, comparée à un index régulier tel que les mots d'un dictionnaire, la caractérisation d'un cas peut être plus complexe puisque :

- elle doit décrire le contexte global dans lequel l'expérience s'est produite, aussi bien que la situation spécifique à laquelle elle est reliée,
- le degré de détail utilisé pour caractériser l'expérience détermine la précision de l'évaluation de sa réutilisation.

2. Une partie solution (leçon) : décrivant la leçon particulière composée par des éléments d'expérience tels qu'une solution à un problème, des décisions ou étapes de résolution de problèmes prises, justifications des décisions, etc.. Originellement, le terme « leçon » est utilisé pour les grandes lignes, des checklists, etc. de ce qui se passe bien ou mal dans un événement particulier (Stewart *et al.*, 1998). Ce terme est également utilisé dans un sens plus large (Weber *et al.*, 2001). « Une leçon apprise est une connaissance ou compétence gagnée par expérience » (Secchi, 1999).

2.4 Les systèmes assistants pour la recherche d'information

Avec l'essor du Web, les problèmes de recherche d'information sont accentués. Plusieurs systèmes d'aide dans le domaine de la recherche d'information sont étudiés dans la littérature scientifique. La plupart de ces systèmes reposent sur un processus actif d'adaptation à l'utilisateur et sont assimilables à des « assistants personnels » de recherche d'information. Différents types d'aide peuvent être offerts à l'utilisateur tels que :

- recommander où chercher (i.e. sélection des moteurs de recherche les plus adéquats par rapport au thème de la requête (Dreilinger et Howe, 1997)),
- l'aide à la navigation : parcourir un ensemble de liens hypertextuels sur le Web (Trousse *et al.*, 1999; Malek et Kanawati, 2001),
- la reformulation de requêtes utilisateur : affiner une requête dans un moteur de recherche (Kanawati *et al.*, 1999a; Jéribi *et al.*, 2001),
- la gestion des signets : organiser et gérer les signets Web des utilisateurs (Kanawati et Malek, 2000),
- le tri des résultats issus des moteurs de recherche : filtrer et réordonner la liste des documents retournés par ces moteurs (Chidlovski, 2000).

L'aide à la recherche d'information repose essentiellement sur deux aspects. Le premier concerne l'intégration du contexte et le deuxième concerne la représentation des besoins de l'utilisateur en informations à travers son profil.

2.4.1 Définition

Nous commençons par définir les notions clés de contexte et de profil utilisés dans le domaine de la recherche d'information :

- **Contexte** : cette notion a été initialement introduite par les travaux de Saracevic (Saracevic, 1997), (Belkin, 1996) et Ingerwersen (Ingwersen, 1996). Le contexte y est défini comme l'ensemble des facteurs cognitifs et sociaux ainsi que les buts et intentions de l'utilisateur au cours d'une session de recherche. (Allan, 1997) et (Sonnenwald, 1999), précisent que le contexte couvre des aspects larges tels que l'environnement cognitif, social et professionnel dans lesquels s'inscrivent des situations liées à des facteurs tels que le lieu, le temps et l'application en cours. Même si les auteurs ne convergent pas vers une même définition, des dimensions descriptives sont communes telles que l'environnement cognitif, le besoin mental en information et l'interaction liée à la recherche d'information. Par exemple, d'après Ingerwersen, parmi les éléments significatifs du contexte pour la recherche d'information, nous citons :
  - les caractéristiques d'intérêt,
  - les caractéristiques du chercheur,
  - les caractéristiques d'interaction,
  - les caractéristiques du système,
  - les caractéristiques des documents,
  - les caractéristiques temporelles.
- **Profil** : un profil utilisateur doit représenter l'ensemble des centres d'intérêt, des préférences, des connaissances ou des habitudes de l'utilisateur. Ce profil est généralement défini par un ensemble de mots-clés avec éventuellement un poids associé à chaque mot. Il existe plusieurs représentations de profils, par exemple :
  - ***La représentation Vectorielle*** : le profil est généralement formalisé comme des vecteurs de termes pondérés (Budzik et Hamond, 2000; Dumais *et al.*, 2003) ou classes de vecteurs (Gowan, 2003). Un profil utilisateur s'appuie généralement sur le Vector Space Model (Salton, 1971). Il est constitué d'un ou de plusieurs vecteurs définis dans un espace de termes ; les coordonnées correspondent aux poids associés aux termes retenus dans un profil. Par exemple le profil utilisateur suivant :
    $\{(photographie, 0.8), (art, 0.5), (couleur, 0.3), (caméra, 0.7)\}$ exprime que l'utilisateur s'intéresse à la photographie à 80%, à l'art à 50%, à la couleur à 30% et à la caméra à 70%.
  - ***Sémantique*** : la représentation du profil met en évidence, dans ce cas, les relations sémantiques entre informations le contenant. La représentation est essentiellement basée sur l'utilisation d'ontologies (Gauch *et al.*, 2003; Challam, 2004; Nanas *et al.*, 2003; Liu et Croft, 2004) ou réseaux sémantiques probabilistes (Lin, 2005; Wen *et al.*, 2004). Elle permet d'exprimer des relations sémantiques entre les unités d'information décrivant le profil.
  - ***Connexionniste*** : ce type de représentation est basé sur l'interconnexion de nœuds représentant les termes, préférences ou documents (Jenning, 1993). Les avantages de ce

type de représentation sont la structuration et la représentation associative permettant de considérer l'ensemble des aspects caractéristiques du profil.

- ***Multidimensionnelle*** : le profil est structuré selon un ensemble de dimensions, représentées selon divers formalismes (Amato et Straccia, 1999; Bouzeghoub et Kostadinov, 2005; Zemirli *et al.*, 2005). Par exemple Tamine considère trois dimensions qui sont les préférences, les données personnelles et les données de l'environnement :

  1. Les préférences : se divisant en deux sous-catégories :
     - le domaine d'intérêt : contenant des informations caractérisant les centres d'intérêt de l'utilisateur,
     - les préférences de recherche : ce sont l'ensemble de préférences liées aux informations recherchées portant sur le processus de recherche (temps de réponse), sur les documents recherchés (contenu et contenant) et sur la personnalisation des résultats (mise en page).
  2. Les données personnelles : c'est l'ensemble des informations personnelles de l'utilisateur (identité, profession).
  3. Les données de l'environnement : qui comportent l'emplacement géographique, la configuration matérielle et logicielle, les liens avec le contexte physique.

La construction de profils peut se faire selon trois approches :

1. Définition explicite où est l'utilisateur définit son propre profil.
2. Définition supervisée qui demande à l'utilisateur son degré de satisfaction pour chaque réponse générée par le système.
3. Définition implicite qui crèe le profil par observation de l'utilisateur.

Le profil peut être intégré dans le processus de recherche d'information, essentiellement pour :

- ***L'exécution des requêtes*** : qui traduit la succession éventuelle des opérations de sélection des sources d'information, la reformulation et le calcul du score de pertinence. Le profil peut servir pour enrichir le contenu de la requête. La sélection personnalisée de sources d'information est une pratique courante dans les méta-moteurs de recherche (Chau *et al.*, 2001; Glover *et al.*, 1999).
- ***Calcul des résultats*** : qui englobe le filtrage et le trie des résultats avant de les présenter à l'utilisateur. L'ordre final des résultats à présenter à l'utilisateur est une combinaison de l'ordre produit par le processus de sélection et celui donné par le contexte de l'utilisateur via un calcul de similarité (Challam, 2004; Gowan, 2003).

Un profil regroupe l'ensemble des connaissances nécessaires à une évaluation des requêtes et à une production d'une information pertinente adaptée à chaque utilisateur. Il convient donc de distinguer la notion de profil de la notion de requête. Un profil peut être défini comme un modèle personnalisé d'accès à l'information alors qu'une requête est l'expression d'un besoin

circonstancié que l'utilisateur souhaite voir satisfait en tenant compte de son profil. Un profil a donc un caractère plus stable que les requêtes même si le centre d'intérêt et les préférences de l'utilisateur peuvent évoluer (Bouzeghoub et Kostadinov, 2005).

2.4.2 Le filtrage d'information

Les systèmes assistants utilisent différents types de filtrage d'information dont le but est de fournir l'information adéquate aux personnes qui en ont besoin (Belkin et Croft, 1992).

Le filtrage d'information se décline essentiellement selon deux grandes familles :

1. **Filtrage basé sur le contenu** : appelé aussi filtrage cognitif. Le choix des documents proposés est basé sur une comparaison des thèmes abordés dans les documents par rapport aux thèmes intéressant l'utilisateur. Il peut être vu comme un système de recherche d'information dont la fonction de correspondance entre une requête et un corpus de documents joue le rôle d'un filtre permanent entre un profil (sorte de requête à long terme et évolutive) et le flot de documents entrant (sorte de corpus évolutif). Deux fonctionnalités centrales ressortent : la sélection des documents pertinents vis-à-vis du profil et la mise à jour du profil en fonction du retour de pertinence fourni par l'utilisateur sur les documents qu'il a reçu. Cette mise à jour se fait par intégration des thèmes abordés dans les documents jugés pertinents.
2. **Filtrage collaboratif** : se base sur l'hypothèse que des utilisateurs à la recherche d'information devraient pouvoir se servir de ce que d'autres personnes avec des intérêts similaires ont déjà trouvé et évalué (Resnick et Varian, 1997; Berrut et Denos, 2003). Le choix des documents proposés est donc basé sur les évaluations d'utilisateurs sur ces documents. Pour chaque utilisateur d'un système de filtrage collaboratif, un ensemble de proches voisins est identifié, et la décision de proposer ou non un document à un utilisateur dépendra des appréciations des membres de son voisinage. En général, les systèmes de filtrage collaboratif demandent aux utilisateurs de fournir des jugements sur des documents de leur choix, exprimés sous la forme d'une note. L'objectif d'un système est alors de comparer les goûts des utilisateurs afin de prédire les notes qui n'ont pas été fournies au système.

   Deux classes d'algorithmes de filtrage collaboratif, exploitant ensuite ces données pour aider les utilisateurs dans leurs futures recherches, ont été distinguées (Breese *et al.*, 1998) :
   - les algorithmes basés sur la mémoire utilisent l'intégralité de la base de données utilisateur pour faire des prédictions. L'idée de base est la suivante :
     (a) Le système construit un profil utilisateur, c'est-à-dire un enregistrement des intérêts de l'utilisateur pour certains documents.
     (b) Puis il compare ce profil avec les profils des autres utilisateurs, et évalue chaque profil en fonction de son degré de similarité avec le profil de l'utilisateur considéré.

(c) Enfin, il considère un ensemble des profils les plus similaires (ou les plus opposés), et utilise l'information qu'ils contiennent pour recommander à l'utilisateur (ou mettre en garde contre) des documents qu'il n'a pas encore évalués.

Pour prédire la pertinence d'un document pour un utilisateur, nous calculons donc la moyenne des notes données aux documents par les utilisateurs ayant les mêmes goûts (ou des goûts opposés), en utilisant des poids différents selon la mesure de similarité entre utilisateurs.

– les algorithmes à base de modèles utilisent la base de données utilisateur pour estimer ou apprendre un modèle, qui sera ensuite utilisé pour les prédictions. D'un point de vue probabiliste, la tâche du filtrage collaboratif peut être vue comme le calcul de la valeur la plus probable d'un vote, étant donné ce que nous savons à propos de l'utilisateur, et le modèle que nous aurons construit à partir de la base de données utilisateur.

Ces deux grandes approches peuvent être combinées en ce que nous appelons **filtrage hybride**. Ce filtrage est basé sur l'analyse du contenu visité et centré fouille de données. Il exploite les avantages des deux approches afin de fournir une assistance pertinente aux utilisateurs.

Les systèmes de filtrage utilisent différentes techniques pour extraire des connaissances nécessaires pour le calcul des recommandations. Parmi ces techniques, nous trouvons la catégorisation, le RàPC, les réseaux bayésiens, les réseaux de neurones, les chaînes de Markov. Nous citons ci-dessous certaines de ces techniques :

1. **La catégorisation (ou clustering)** : présente une technique d'apprentissage non supervisé. Le but consiste à déterminer des groupements intrinsèques dans un ensemble de données non-étiquetées. Par exemple, dans le travail de (Yan *et al.*, 1996), l'approche consiste à catégoriser les navigations passées. Une catégorie est assimilée à une navigation type. Une fonction d'appariement est appliquée à la navigation courante afin de déterminer la catégorie la plus proche et le système recommande des pages citées dans la catégorie identifiée, qui ne sont pas encore visitées.

   Un exemple dans la recherche documentaire consiste à regrouper (en clusters) les utilisateurs ayant les mêmes goûts, et à regrouper (en clusters) les documents portant sur les mêmes sujets, ou qui tendent à plaire aux mêmes personnes. Ainsi, pour prédire la note qu'un utilisateur donnera à un document, nous pourrons utiliser les avis des utilisateurs qui appartiennent à son groupe. En d'autres termes, nous voulons associer une classe (ou plusieurs) à chacun des utilisateurs, ainsi qu'à chacun des documents. Mais ces classes étant a priori inconnues, elles doivent être dérivées du processus d'estimation du modèle. Typiquement, les systèmes de clustering se différencient par la fonction « objectif », choisie pour évaluer la qualité du clustering, et la stratégie de contrôle pour parcourir l'espace des clusters possibles. Mais tous suivent le principe général traditionnel de clustering qui consiste à maximiser la similarité des observations à l'intérieur d'un cluster, et minimiser

la similarité des observations entre clusters, pour arriver à une partition de la base aussi pertinente que possible. En entrée, nous avons un ensemble d'enregistrements de qui a apprécié quoi, représenté par une matrice.

L'idée du clustering est alors de former des blocs les plus pertinents et significatifs possibles, à partir de cette matrice, et former ainsi des clusters d'utilisateurs et des clusters de documents, tels que les utilisateurs considérés comme similaires, tendent à noter de la même façon les documents considérés comme similaires.

2. **Les réseaux bayésiens** : un réseau de Bayes (Heckerman, 1999) est un graphe acyclique orienté qui représente une distribution de probabilités de dépendance entre un ensemble de variables. Chaque nœud dans le graphe représente une variable, et chaque arc représente une dépendance directe entre variables. Les réseaux bayésiens sont à la fois des modèles de représentation des connaissances et des « machines à calculer » les probabilités conditionnelles. L'intérêt particulier des réseaux bayésiens est de tenir compte simultanément de connaissances a priori d'experts (dans le graphe) et de l'expérience contenue dans les données. Par exemple, dans le travail de (Hibou, 2006), les réseaux bayésiens sont utilisés pour la modélisation de l'utilisateur dans un environnement informatique pour l'apprentissage humain (EIAH).
3. **Règles d'association** : les règles d'association permettent de découvrir des informations utiles, cachées dans des grandes bases de données. C'est un outil d'aide à la décision dans le domaine commercial, scientifique et industriel. Dans le domaine des Multimédia et d'Internet, les règles d'association extraites à partir des historiques d'accès des internautes ont été utilisées pour l'aide à la conception et l'organisation des sites web. Des sites de E-commerce comme Amazon.fr, utilisent les règles d'association pour faire des suggestions à leurs clients. Par exemple, dans le domaine commercial, suite à l'analyse du panier d'une ménagère dans un supermarché, il s'est avéré que 80% des clients qui achètent des couches pour bébé achètent du lait. Cela pourra aider l'établissement pour faire progresser ses ventes en approchant l'emplacement des couches de celui du lait par exemple.
4. **Les chaînes de Markov** : une chaîne de Markov est un processus stochastique possédant la propriété markovienne (S un ensemble d'états $\{S_1, S_2, S_3, ..., S_n\}$), une matrice M telle que M_{ij} présente la probabilité de passer d'un état S_i à un état S_j et $\sum_{i \in [1..n]} M_{ij} = 1$. Dans un tel processus, la prédiction du futur à partir du présent ne nécessite pas la connaissance du passé.

   L'idée de base des approches, fondées sur l'exploration de la structure du Web, est d'inférer l'importance d'une page Web en fonction du nombre de pages qui pointent (avec un lien hypertexte) vers cette page et en fonction de l'importance des pages pointées par cette même page. L'exemple le plus connu de cette approche est l'algorithme *PageRank* (Brin et Page, 1998) employé par le moteur de recherche Google. Les pages les plus importantes

sont placées en tête de la liste des réponses du moteur de recherche.

5. **Les réseaux de neurones** : Les réseaux de neurones sont généralement optimisés par des méthodes d'apprentissage de type statistique, si bien qu'ils sont placés : d'une part, dans la famille des approches statistiques qu'ils enrichissent avec un ensemble de paradigmes permettant de générer de vastes espaces fonctionnels, souples et partiellement structurés ; d'autre part, dans la famille des méthodes de l'intelligence artificielle qu'ils enrichissent en permettant de prendre des décisions s'appuyant davantage sur la perception que sur le raisonnement logique formel. Un des travaux s'appuyant sur cette technique pour l'analyse (Classification et Visualisation) des données issues d'usages d'Internet, est présenté dans le travail de (Benabdeslem et Bennani, 2007).

2.4.3 Classification des systèmes existants

Nous présentons dans cette partie une classification de quelques systèmes d'aide existants selon certains critères notamment le type de filtrage utilisé, la technique de filtrage utilisée, le type de l'approche, etc.
Afin de simplifier la lecture, nous organisons ces systèmes selon le type de filtrage.

2.4.3.1 Filtrage basé sur le contenu

Dans cette section, nous présentons quelques systèmes qui ont utilisé le filtrage basé sur le contenu pour assister leurs utilisateurs.

Dans le système AIRA (Adaptive Information Research Assistant) (Bottraud *et al.*, 2003a), les agents de recherche d'information apprennent les profils en utilisant les références documentaires rassemblées par l'utilisateur. Ils utilisent également des informations obtenues en observant les activités en cours de l'utilisateur afin de restreindre l'utilisation du profil aux parties les plus pertinentes. AIRA est un système à base d'agents dont le fonctionnement est basé sur trois tâches :

- l'observation continue de l'activité de l'utilisateur (les observateurs d'activités sont chargés de collecter de façon autonome les sous-produits textuels des activités de l'utilisateur), par interaction avec les composants logiciels qu'il utilise (contenu du presse-papier, pages Web visualisées, documents en cours d'édition, etc.) pour garder à jour le journal des actions,
- la gestion de l'évolution du profil (agent gestionnaire de profils),
- la gestion de la recherche d'information déclenchée à l'initiative de l'utilisateur lorsqu'il soumet une requête.

Les composants utilisés pour la recherche correspondent :

- Au gestionnaire de contexte, qui est chargé d'exploiter le journal des actions et le profil opérationnel pour identifier le contexte de travail actuel, qui est représenté sous forme

d'un vecteur de poids associés à des termes,
– Au gestionnaire de requêtes, qui utilise ce contexte pour interpréter la requête, la soumettre aux moteurs de recherche sélectionnés et enfin afficher les résultats collectés, après qu'ils aient été analysés et reclassés.

Le fonctionnement du système AIRA repose sur l'hypothèse que l'utilisateur maintient à jour un ensemble de documents, ou de « bibliothèques personnelles », qui est représentatif de ses centres d'intérêts. C'est à partir de cet ensemble qu'AIRA extrait et structure les connaissances représentatives de l'utilisateur. Le système est construit à partir de trois grands blocs : le profil utilisateur, les observateurs d'activités et les composants intervenant pour gérer une requête. Le profil utilisateur est construit à partir de la bibliothèque de l'utilisateur et est vu comme une mémoire « à long terme » associée à l'utilisateur. Il est organisé grâce à plusieurs composants ayant chacun un rôle particulier : observateur de la bibliothèque, profil conceptuel et profil opérationnel. Le profil conceptuel organise les documents de la bibliothèque en une hiérarchie stricte de classes. Il peut être donné ou construit automatiquement par un mécanisme de classification incrémentale. Le profil opérationnel est dérivé du profil conceptuel (Bottraud *et al.*, 2003b). Pour chaque classe, il construit une description intentionnelle sous forme de liste de termes, directement exploitable lors de la recherche. Ainsi le profil utilisé est hiérarchisé plutôt que des combinaisons de vecteurs. Ce qui permet de jouer sur le niveau de généralité et sur le nombre de termes utilisés en fonction de la requête. L'identification et la description du contexte de travail dans lequel l'utilisateur se trouve lorsqu'il soument une requête revient à déterminer la ou les parties de profil opérationnel qui sont concernées. Cette étape s'appuie sur l'observation des activités en cours.
Afin d'assister l'utilisateur, deux stratégies sont définies : la première utilise le contexte pour filtrer et réordonner les résultats retournés par le ou les moteurs de recherche auxquels la requête a été soumise. L'inconvénient de cette stratégie est qu'elle n'explore pas l'espace en dehors du sous-espace initialement défini par la requête. La seconde stratégie est plus élaborée et utilise un mécanisme « d'expansion de requêtes » exploitant les informations disponibles dans le contexte. L'espace exploré sera alors plus important en générant des requêtes dérivées de la requête initiale, ce qui accroît les chances d'obtenir des documents pertinents. Dans AIRA, l'expansion de requêtes repose sur un double mécanisme itératif d'hypothèses candidates. À chaque itération, les hypothèses retenues sont spécialisées. L'évaluation des documents résultats est faite par l'utilisateur. La comparaison des notes fournies par les agents avec les évaluations de l'utilisateur (évaluation de excellent (++) à très mauvais (- -)) permet de classer les différentes configurations en fonction de leur capacité à prévoir la pertinence d'un document pour l'utilisateur. L'évaluation donnée par l'utilisateur est aussi utilisée pour comparer les agents à travers la qualité de leurs documents proposés. La qualité d'un document est directement liée à la valeur perçue par l'utilisateur des documents obtenus.

Le système WEBMATE (Chen et Sycara, 1998) est un agent logiciel personnel qui accompagne l'utilisateur lors de sa navigation et sa recherche et lui fournit une aide intelligente. Il fournit une assistance à la formulation de requêtes soit en suggérant a priori des mots complémentaires, soit par « bouclage de pertinence » (c'est-à-dire en reformulant une nouvelle requête à partir des documents sélectionnés par l'utilisateur dans les résultats). Les capacités d'un tel agent sont essentiellement :

- apprendre les intérêts de l'utilisateur d'une façon incrémentale avec des mises à jour continues et une fourniture automatique de documents (par exemple, un journal personnel),
- aider l'utilisateur à raffiner sa recherche pour améliorer la recherche de documents pertinents.

WEBMATE est composé d'un proxy HTTP (situé entre le navigateur Web de l'utilisateur et le World-Wide Web). Ainsi toutes les transactions HTTP passent parWEBMATE qui peut contrôler la navigation et les activités de recherche de l'utilisateur. WEBMATE comporte aussi une applet (qui présente une interface entre l'utilisateur et le proxy) qui va interagir avec l'utilisateur. À travers cette applet, l'utilisateur peut exprimer ses intérêts et fournir un retour intéressant lors d'une recherche. De plus, à travers ce contrôleur, l'utilisateur reçoit de l'aide intelligente de WEBMATE. Le système se base sur la notion de profil utilisateur qui peut avoir plusieurs sous-catégories d'intérêt.WEBMATE apprend automatiquement et d'une façon incrémentale ces catégories et le profil de son utilisateur. À chaque fois qu'il y a un nouvel exemple positif, le système met à jour le profil. Les exemples positifs sont des documents HTML que l'utilisateur a marqué « I Like it ». Une approche d'apprentissage du profil d'utilisateur est utilisée pour compiler et recommander un journal personnel (ensemble de pages Web). La compilation du journal personnel peut se faire de deux façons :

1. Une première façon consiste à fixer une liste d'URLs que l'utilisateur veut avoir. Il permet de plus la recherche automatique de documents intéressants pour créer un journal personnel. Il procède par extraction d'URLs dans une page identifiée (par exemple, sommaire d'un magazine Web), fréquemment modifiée, puis récupération des pages correspondantes. Ces pages sont indexées et leur intérêt est évalué en mesurant leur similarité avec les domaines du profil utilisateur. Les pages retenues sont enfin présentées à l'utilisateur par ordre de similarité décroissante.

   Le profil de l'utilisateur est représenté par N vecteurs de dimension M (M prédéfini), correspondant à N domaines d'intérêts. Chaque dimension étant associée à un terme. La valeur de TF-IDF qui présente la fréquence d'apparition d'un terme dans une collection de documents, est utilisée comme coordonnée. Ce profil est mis à jour chaque fois que l'utilisateur marque un document comme « Intéressant ». Les domaines sont comparés à l'aide de la mesure de similarité cosinus habituelle :
   $sim(V_j, V_k) = V_j.V_k/(|V_j| * |V_k|), j, k \in \{1, .., n, i\}$. Les deux vecteurs les plus proches sont alors combinés : $V_l = V_l + V_m$ (où l et m maximisent la mesure de similarité ci-

dessus). V_l est ensuite tronqué à M éléments, après les avoir triés en ordre décroissant pour pouvoir conserver ceux ayant le poids le plus élevé. WEBMATE construit le vecteur TF-IDF pour chacune de ces pages et calcule la similarité avec le profil utilisateur courant. Si la similarité est supérieure à un certain seuil, l'agent recommande la page à l'utilisateur et trie toutes les pages recommandées selon un ordre décroissant de similarité pour former le journal personnel. Dans le cas où l'utilisateur ne fournit aucune des URLs qu'il veut avoir comme source d'information, WEBMATE construit une requête utilisant les différents mots situés à la tête du profil courant et l'envoie aux moteurs de recherche.

2. Le deuxième volet porte sur le raffinement des requêtes par l'expansion de mots-clés et les bouclages de pertinence.

   L'algorithme proposé se base sur un modèle des paires de mots « trigger pairs » pour raffiner la recherche et aide à enlever les ambiguïtés. Dans ce modèle, si un mot S est significativement correlé avec un autre mot T, alors (S, T) est considéré comme un trigger pair. Cela veut dire que si S apparaît dans un document, alors il déclenche T. Autrement dit, quand le mot S apparaît à un certain point dans un texte, nous nous attendons à ce que le mot T apparaîsse quelque part après S avec une certaine confiance.
   Les paires de mots (trigger pairs) sont construites à partir de corpus importants (remarque : les pairs dépendent des corpus utilisés) en sélectionnant les mots voisins (à l'intérieur d'une distance donnée, 500 mots ici). Les mots associés à un terme sont classés dans l'ordre décroissant d'information mutuelle. L'expansion est calculée de la façon suivante : si la requête contient m mots-clés k_1 à k_m, auquels sont associés des ensembles $S_i = \{_{S_{ij}}\}$, les termes $_{S_{ij}}$ étant triés par ordre décroissant d'information mutuelle $MI(_{S_{ij}}, k_l)$, définie par $MI(_{S_{ij}}, k_l) = P(_{S_{ij}}, k_l) log(\frac{P(_{S_{ij}}, k_l)}{P(_{S_{ij}} P(k_l))})$, et le nombre de termes complémentaires souhaités est n, le principe consiste à :
   - Créer l'ensemble des combinaisons de n ensembles dans l'ensemble des m S_i associés aux termes,
   - Prendre pour chaque combinaison l'ensemble résultant de l'intersection des S_i qui lui appartiennent,
   - considérer l'union de ces intersections, en les classant par ordre décroissant d'information mutuelle. Si le nombre de termes est supérieur ou égal à n, cet ensemble est celui des termes complémentaires de la requête, sinon, on recommence avec des combinaisons de $n - 1$ ensembles.

     L'expansion avec « bouclage de pertinence » utilise une page considérée comme pertinente et examine le voisinage (5 mots précédents, 5 mots suivants) de chaque terme de la requête. Les fréquences d'apparition des termes dans ces voisinages, dans la page, sont calculées et les 5 termes les plus fréquents sont utilisés pour compléter la requête.

Dans WEBMATE, le contexte des mots-clés de recherche dans les pages web pertinentes est

utilisé pour raffiner la recherche en se basant sur l'idée que si l'utilisateur signale au système quelques pages pertinentes à sa recherche, le contexte des mots-clés (un ensemble de mots-clés précédant le mot-clé considéré) de la recherche est plus informatif que le contenu de la page.

2.4.3.2 Filtrage collaboratif

Dans cette section, nous présentons quelques systèmes qui ont utilisé le filtrage collaboratif pour assister les utilisateurs dans des tâches de recherche d'information (Berrut et Denos, 2003).
Siteseer (Rucker et Polanco, 1997) est un système de recommandation de pages web qui utilise les signets personnels et leur organisation en répertoires pour prédire et recommander des pages pertinentes. Il utilise chaque signet d'utilisateur comme une déclaration implicite d'intérêt pour le contenu, et le classement de ces signets comme une indication de cohérence sémantique ou un regroupement pertinent entre des sujets. Au cours du temps, Siteseer apprend les préférences et les catégories à travers lesquelles les utilisateurs perçoivent le monde, et en même temps, apprend pour chaque page web, quelles sont les différentes communautés ou groupes d'affinités qui s'y intéressent. Siteseer génère alors des recommandations organisées et contextualisées en les délivrant dans leur répertoire d'origine. Les signets offrent un mécanisme de collecte d'information sur les préférences. Ce mécanisme est directement géré par l'utilisateur et ne requiert pas de comportement additionnel pour la tâche d'information du système de recommandation. Siteseer consulte les signets de chaque utilisateur et mesure le degré de chevauchement (par exemple, URLs communes) de chaque répertoire avec les répertoires d'autres utilisateurs, pour donner un poids additionnel aux URLs ; le chevauchement de contenu permet de déterminer les similarités entre répertoires et de former dynamiquement des communautés virtuelles d'intérêt, particulières pour chaque utilisateur et spécifiques à chaque catégorie d'intérêt. En calculant l'adhésion relative d'une communauté à chaque répertoire, et en évitant de former un ensemble définitif de clusters, Siteseer n'impose pas de catégorisation rigide. Le système ne tire aucune sémantique ni du contenu des URLs, ni du nom du répertoire. Il utilise simplement l'URL d'une ressource comme identifiant unique et ignore complètement le titre. Les principales limites de Siteseer proviennent de son approche purement collaborative. Il est incapable de servir les premiers utilisateurs ou un utilisateur créant une nouvelle catégorie.

Phoaks (People Helping One Another Know Stuff) est un système expérimental de reconnaissance, de correspondance et de redistribution automatiques de recommandations sur les ressources web, extraites des messages des newsgroups de Usenet (Terveen *et al.*, 1997). Il effectue une recherche contextuelle de mentions d'URL dans ces messages, qui dans 23% des cas mentionnent des ressources web (sous forme d'URL), et 30% de ces mentions sont des recommandations. Phoaks compte toute mention comme recommandation si elle passe un certain

nombre de tests :

- un message ne doit pas être posté à plusieurs groupes en même temps, il devient alors trop général et ne peut être assez proche thématiquement de l'ensemble des groupes,
- l'URL ne fait pas partie de la signature de l'expéditeur sinon elle n'est pas considérée comme recommandation, c'est une forme d'auto-promotion,
- l'URL n'apparaît pas dans une partie entre « quotes » du message (inclus dans la partie message d'origine dans une réponse à un message) sinon elle ne sera pas retenue,
- si le contexte textuel de l'URL indique qu'il s'agit d'une recommandation et qu'elle n'est pas l'objet d'une publicité.

Phoaks se différencie des autres systèmes par la distinction qu'il fait entre les rôles de fournisseur et consommateur de recommandations ; il prend ainsi en considération le fait que seule une minorité d'utilisateurs prend la peine d'évaluer les ressources et de faire partager leur opinion avec les autres. Aussi, la réutilisation des conversations en ligne, existantes comme source des recommandations, ne demande aucune intervention de la part des recommandeurs. Avec d'autres règles plus complexes, Phoaks sélectionne et catégorise les ressources. Le nombre de recommandeurs distincts d'une même ressource a été retenu comme mesure de la qualité d'une recommandation. Une étude a montré que la multiconfirmation est une source de recommandation pertinente. En effet, une ressource a d'autant plus de chances de paraître dans une foire aux questions (les bases de FAQ maintenues par des experts humains) qu'elle est recommandée par différentes personnes. Son efficacité est évaluée par la mesure de la précision (pourcentage des ressources que les règles classifient dans la bonne catégorie) et du rappel (le pourcentage de ressources qui appartiennent à une catégorie et que la règle classe réellement dans cette catégorie). Comme continuation, les auteurs de Phoaks projettent d'améliorer le calcul de la crédibilité des recommandeurs et la recherche d'affinités entre ceux qui offrent et ceux qui sont en quête de recommandations dans un domaine particulier. Ils tentent aussi de combiner une recherche d'informations par mots-clés avec le filtrage collaboratif, dans une « recherche classée par communauté ». Le principe est de filtrer les résultats des requêtes à travers la base de Phoaks. Les résultats sont alors classés par groupe de newsgroups qui les mentionnent. Ceci permet de réduire l'ambiguïté des requêtes et de classer les résultats en fonction de leurs fréquences de mention.

Le système CoWing (COllaborative Web IndexiNG, (Kanawati et Malek, 2001)) est un système de gestion collaboratif de signets Web. Le but du système est de permettre aux utilisateurs de partager leurs signets de manière :

- implicite : dans le sens où les utilisateurs ne sont pas obligés de faire un effort supplémentaire pour utiliser le système,
- sécurisée : dans le sens où chaque utilisateur a la capacité de contrôler qui connaît quoi à propos de ses propres collections de signets,

– efficace : dans le sens où l'utilisateur est recommandé par des signets pertinents, calculés en appliquant un algorithme de filtrage collaboratif distribué.

Dans ce système, les recommandations sont automatiquement insérées dans le répertoire de signets le plus approprié dans la collection locale de l'utilisateur. Dans CoWing, chaque utilisateur est assisté par un agent appelé WING. Un agent WING observe les comportements de son utilisateur lorsqu'il gère sa collection de signets. Il apprend comment l'utilisateur classifie ses signets. Une approche hybride à base de réseau de neurones et du raisonnement à partir de cas est utilisée. Chaque agent apprend à mieux classifier et recommander. La partie problème d'un cas est composée par des attributs d'un signet et la partie solution présente l'identifiant du répertoire dans lequel le signet est classé par l'utilisateur. Le modèle de mémoire du classifieur utilisé (appelé PROBIS) est basé sur l'intégration d'un réseau de neurones à base de prototypes et une mémoire plate divisée en groupes, chacun étant représenté par un prototype. PROBIS contient deux niveaux de mémoire : le premier niveau contient les prototypes et le deuxième niveau contient les exemples. Le premier niveau de mémoire est composé de la couche cachée du réseau de neurones à base de prototypes. Le second niveau de mémoire est une mémoire plate simple où les exemples sont organisés en différentes zones d'exemples similaires. Les deux niveaux sont liés entre eux de manière à ce que chaque zone mémoire soit associée à un prototype. La zone mémoire contient tous les exemples appartenant à ce prototype. Une zone mémoire spéciale est réservée pour les exemples atypiques : ce sont les exemples qui n'appartiennent à aucun prototype. Pour apprendre à recommander, chaque agent $WING$ maintient localement deux structures de données : un agenda et une matrice de corrélation de répertoires. L'agenda a la structure d'un dictionnaire où les clés représentent les identifiants des agents $WING$ à contacter et les valeurs présentent les dates de contact prochaines. La corrélation entre deux répertoires $f1$ et $f2$ est donnée par le nombre de signets appartenant à $f2$, qui sont classifiés dans $f1$ selon la connaissance de l'agent, divisé par le nombre total de signets dans $f2$.

I-SPY est un méta-moteur de recherche orienté communauté (Balfe et Smyth, 2004). Son but est de permettre à un groupe de personnes d'intérêts similaires, de partager leurs résultats de recherche d'une façon implicite. Le système est construit d'une manière centralisée où l'on enregistre dans une matrice H_{ij}. Un élement h_{ij} donne le nombre de fois qu'un document d_j a été sélectionné par les utilisateurs parmi la liste de réponse à une requête q_i. Lorsqu'un utilisateur pose une nouvelle requête q le système cherche dans sa matrice les requêtes les plus similaires à q et recommande les documents les plus fréquement sélectionnés parmi les réponses fournies à ces requêtes. I-SPY utilise le contexte dans la recherche web afin d'adapter un moteur de recherche générique aux besoins d'un comité d'utilisateurs. Afin de favoriser la chance d'avoir des requêtes similaires ; le système propose d'avoir une page portail spécifique à chaque communauté d'intérêts. Chaque page portail est associée à une matrice d'historique qui lui est propre. Le système se base sur la personnalisation des résultats de recherche pour les besoins de

comités sans qu'il soit nécessaire de stocker des historiques de recherche individuels, sans profil utilisateur individuel et sans identification d'utilisateur. Cela est considéré comme un avantage significatif de sécurité et de vie privée par rapport aux autres approches traditionnelles de personnalisation. Cependant, ce système souffre du problème de redémarrage à froid où les pages web récemment indexées ont du mal à attirer l'attention des utilisateurs puisqu'elles ont par défaut un faible score de pertinence en utilisant les métriques de I-SPY et apparaissent ainsi en bas des listes de résultats.

Bibster (Bibliographic Semantic-Based Peer-to-Peer System) (Broekstra *et al.*, 2004) est un système de partage de données bibliographiques. Le but de ce système est la gestion locale de fichiers bibliographiques, la recherche et le partage de données bibliographiques dans un réseau de type égal-à-égal. Le système permet à un groupe de personnes de rechercher des références bibliographiques dans les bases de données personnelles d'autres utilisateurs. Le système se base sur l'utilisation de deux ontologies communes pour l'importation des données, la formulation de requêtes, le routage des requêtes et le traitement des résultats. Le filtrage collaboratif est utilisé pour trouver les pairs avec qui coopérer pour une expertise donnée. L'expertise d'un pair est l'ensemble de thèmes pour lesquels le pair est expert. Lorsqu'un pair reçoit une requête, il décide alors de la diffuser aux pairs dont l'expertise est similaire au sujet de la requête. Les pairs partagent une ontologie commune pour publier des descriptions sémantiques de leurs expertises dans le réseau. Cette connaissance sur les expertises des autres pairs forme la topologie sémantique qui est indépendante de la topologie du réseau. La publication des expertises est décidée et faite par les pairs en se basant sur une similarité sémantique entre les descriptions des expertises (c.f section 3.4).

2.4.3.3 Filtrage hybride

Le système Fab (Balabanovic et Shoham, 1997) est un système hybride qui combine les deux approches de filtrage : l'approche basée sur le contenu sémantique et l'approche collaborative. La notion de profil, basée sur l'analyse du contenu, est maintenue et les profils sont systématiquement comparés pour identifier les similarités entre utilisateurs. Un utilisateur reçoit un document soit parce qu'il correspond à son profil soit parce qu'il a été apprécié par un autre utilisateur ayant un profil ressemblant. Le processus de recommandation est divisé en deux phases :

- une phase de collecte de ressources pour constituer une base ou un index,
- une phase de sélection de ressources de cette base pour des utilisateurs particuliers.

Dans Fab, la phase de collecte consiste à rassembler des pages pertinentes pour un nombre réduit de sujets, et qui sont regroupées automatiquement selon les domaines d'intérêt des utilisateurs. Ces pages sont ensuite diffusées à un large nombre d'utilisateurs dans la phase de sélection. Un sujet peut intéresser plusieurs personnes et une personne peut être intéressée par plusieurs su-

jets. L'application utilise des agents ; les pages retrouvées par l'agent de collecte sont envoyées à un routeur central qui se charge de les transférer aux utilisateurs dont les profils correspondent. Les agents personnels de chaque utilisateur se chargent d'éliminer les pages déjà consultées. Une fois que l'utilisateur a envoyé une requête, reçu et consulté des recommandations, il lui est demandé de fournir une note de 0 à 7. Ces notes servent d'une part à mettre à jour le profil personnel et d'autre part à informer l'agent de collection. De plus, toute page très bien notée est automatiquement passée aux utilisateurs estimés les plus proches. La construction de profils représentatifs est une condition importante pour le succès du système. Ils permettent à la composante basée sur le contenu sémantique d'assurer des recommandations appropriées, et à la composante collaborative de fournir les utilisateurs ayant des profils proches. La population des agents de collecte s'adapte à la population d'utilisateurs, et non à un utilisateur particulier. Pour aider ce processus, les agents de collecte « impopulaires », dont les pages ne sont pas visualisées par un grand nombre d'utilisateurs ou ont peu de succès sont détruits et les meilleurs profils sont dupliqués pour les remplacer. Ainsi, la spécialisation des agents de collecte n'a pas à être faite d'avance, mais peut être déterminée dynamiquement et modifiée au cours du temps.

COSYDOR (COoperative SYstem for DOcument Retrieval) est un système d'aide à la recherche et à l'interrogation de bases documentaires (Jéribi *et al.*, 2001). Il aide à la reformulation de requêtes en se fondant sur la réutilisation d'expériences et sur les profils utilisateurs. L'expérience est définie comme étant une instance de recherche, caractérisée par un contexte de recherche et des résultats documentaires validés. Le contexte est caractérisé par l'utilisateur et la requête. Dans ce système, les connaissances sur l'utilisateur sont classées en quatres classes selon l'évolutivité dans le temps. Le profil d'un utilisateur est présenté sous une forme vectorielle $U =< u_1, u_2, u_3, u_4 >$ où u_i représente la $i^{ième}$ classe du profil utilisateur U. Les caractéristiques mises en évidence du profil sont les suivantes :

1. caractéristiques peu ou non évolutives : qui concernent les connaissances de l'utilisateur sur son propre monde. Par exemple, son origine, son sexe, etc.

2. caractéristiques ayant une évolution à long terme : comme par exemple, la catégorie ou la fonction de l'utilisateur, ses domaines d'intérêts, etc.

3. caractéristiques ayant une évolution à moyen terme : concernant les connaissances de l'utilisateur sur le système. Par exemple, son niveau de connaissance du corpus et du fonctionnement du système de recherche documentaire. Ce niveau appartient à $\{1, 2, 3\}$ où 1 correspond au plus faible niveau.

4. caractéristiques ayant une évolution à court terme : ce sont des caractéristiques liées à la session de recherche incluant des connaissances des tâches tels que le but de la recherche, les préférences et des connaissance sur le domaine tel que le niveau de spécialisation requis.

Une fonction de similarité entre utilisateurs est définie pour réutiliser les expériences de recherche des utilisateurs lorsqu'ils présentent des profils similaires. La mesure de similarité est une fonction pondérée de similarité correspondant aux quatres classes composant le profil. L'approche utilise la méthodologie du raisonnement à partir de cas où un cas représente une instance de recherche lors d'une session de recherche effectuée par un utilisateur donné.
$Cas =< U, Q, D, E >$ où :

- U : présente les caractéristiques de l'utilisateur.
- Q : présente la requête une sous une forme vectorielle.
  $Q =< a_1, a_2, a_3, ..., a_n >$ où a_i est le poids du $i^{ème}$ terme de la requête.
- D : est l'ensemble de documents validés par l'utilisateur.
- E : représente l'évaluation donnée par l'utilisateur sur la pertinence de D par rapport à Q. $E \in \{2, 1, 0, -1\}$ tel que 2 : « très satisfaisant », 1 : « satisfaisant », 0 : « moyennement satisfaisant » et -1 : « très insatisfaisant ».

La réutilisation consiste à enrichir la requête initiale de l'utilisateur par des termes de documents au format *html* contenus dans les cas trouvés.

Le fonctionnement de COSYDOR se fait comme suit : l'utilisateur une fois identifié, soumet sa requête. Celle-ci est analysée par un SGBD textuel (qui utilise des outils linguistiques, thesaurus, lexique pour l'indexation des documents et des requêtes, par exemple, Intermedia). Ensuite la requête est enrichie par COSYDOR (en fonction de la base de cas) et enfin soumise au même SGBD. Les résultats sont envoyés à l'utilisateur pour les évaluer. Les évaluations faites par l'utilisateur sont prises en compte pour la mise à jour de la base de cas et la mise en œuvre d'un enrichissement plus précis de la requête de l'utilisateur. Pour reformuler ou enrichir une requête utilisateur, le système calcule la similarité entre le cas courant contenant la requête et les différents cas de la base de cas pour trouver les cas les plus proches et qui pourront être utilisés pour l'enrichissement de la requête. Pour chaque cas trouvé, un indice de confiance (dépendant de l'évaluation des résultats) est calculé permettant de pondérer sa contribution dans l'enrichissement de la requête. À partir des cas similaires et des documents correspondants, COSYDOR extrait les termes ayant les poids les plus importants et les rajoute automatiquement dans la requête initiale de l'utilisateur. Dans ce cas, l'utilisateur peut soit soumettre la requête telle quelle, soit supprimer des termes qu'il juge inutiles, soit rajouter explicitement des termes. La difficulté de cette approche de réutilisation est la mise en évidence des contextes ou situations de recherche proches afin de les réutiliser. De plus, l'acquisition de connaissances sur l'utilisateur est explicite et se fait en demandant à l'utilisateur ses caractéristiques de recherche (but, domaine, etc.) ainsi que ses caractéristiques générales (pays, sexe, fonction, etc.).

MAIS (Multi-Agent based Internet Search) (Enembreck, 2003; Enembreck et Barthès, 2004) est un système multi-agents pour la recherche d'information sur le web. Dans MAIS, les documents personnels de l'utililsateur sont utilisés pour la construction son profil. Le système

est doté d'assistants personnels, d'un agent bibliothèque, d'agents de filtrage et d'agents de recherche. L'agent assistant personnel assure à l'utilisateur la gestion de ses pages favorites et la recherche de nouvelles pages. Les pages favorites sont stockées par l'assistant personnel dans l'espace de travail de l'utilisateur, dans des dossiers personnels sous forme de documents textuels. Ces documents sont ensuite utilisés par l'agent bibliothèque pour créer un modèle vectoriel composé de la liste des termes les plus importants, présents dans la collection de documents personnels. Ces termes sont choisis en utilisant la mesure de qualité TF-IDF (Term Frequency-Inverse Document Frequency) (Salton, 1989). TF-IDF considère qu'un terme est important s'il est très fréquent dans un petit nombre de documents. Les centres d'intérêt publics contiennent des documents concernant des domaines précis fournis par des experts. Ces centres d'intérêt sont communs à tous les utilisateurs mais peuvent être personnalisés au fur et à mesure que les utilisateurs ajoutent des documents à leur profil. Le principe de fonctionnement est le suivant : un utilisateur qui souhaite obtenir des informations déclenche une recherche en passant à son assistant personnel une liste de termes, le nombre de pages désiré et les centres d'intérêt sélectionnés qui peuvent être les siens ou des centres d'intérêt publics. L'assistant envoie ensuite une requête en mode broadcast. Le premier agent de filtrage disponible sera chargé de fournir la réponse.

La première tâche des agents de filtrage consiste à rassembler les résultats fournis par les agents de recherche. Ensuite, ils doivent ordonner et réduire la liste des pages reçues. Pour cela, ils utilisent la méthode d'apprentissage incrémental *ELA* (Entropy-based Learning Approach) permettant de représenter les données sous forme d'un graphe de concepts. Chaque agent de recherche gère un moteur de recherche particulier, et présente un agent OMAS (Open Multi-Agent System, (Barthès et Ramos, 2002)) pour permettre la création de nouveaux agents pendant l'exécution du système et la recherche en parallèle. L'agent de filtrage récupère donc une liste importante de pages qu'il doit réduire et ordonner, selon un processus de personnalisation, pour finalement présenter les résultats à l'utilisateur. Le processus de personnalisation consiste à ordonner la liste de pages fournies par les agents de recherche en fonction des préférences de l'utilisateur. Chaque page est classifiée selon sa classe et son degré de qualité de la classification. Les classes possibles correspondent aux dossiers créés par l'utilisateur pour organiser ses documents favoris (centres d'intérêt qu'il a choisis). Les résultats sont rassemblés et filtrés en fonction d'un profil utilisateur, calculé à partir des favoris organisés en centres d'intérêt et de centres d'intérêt publics. Les pages les plus similaires aux documents déjà contenus dans le profil de l'utilisateur sont retenues et placées en tête des réponses. Finalement, seules les meilleures pages sont sélectionnées après seuillage. L'utilisateur prend connaissance des pages qui l'intéressent et a la possibilité de les ajouter aux documents présents dans ses centres d'intérêt. Dans ce système, l'adaptation est réalisée par les agents de filtrage et se fait quand l'utilisateur ajoute ou supprime un document dans ses favoris et quand l'utilisateur demande une recherche avec des centres d'intérêt différents de ceux présents dans son profil.

MAIS présente de meilleurs résultats par rapport aux moteurs de recherche et devient plus précis lorsque l'utilisateur choisit les centres d'intérêt publics puisque les documents qui y sont associés sont validés par des experts. Ainsi, un utilisateur qui n'a pas beaucoup d'expériences peut profiter d'une certaine personnalisation grâce à ces centres d'intérêt publics. De plus, les utilisateurs doivent bien analyser leur documents avant de les ajouter à leur liste de favoris puisqu'ils peuvent diminuer la précision du système en introduisant du bruit.

Nguyen el al. (Nguyen *et al.*, 2006b) proposent un modèle permettant d'étendre les fonctionnalités des systèmes de filtrage hybride en les rendant capables de gérer des comités multicritères explicites. Elles proposent une méthode se basant sur l'exploitation des données « disponibles à froid » sur l'utilisateur (par exemple, son sexe, sa catégorie socioprofessionnelle, etc.) afin d'améliorer l'intégration des nouveaux utilisateurs dans le système (Nguyen *et al.*, 2006a). Cette méthode se base sur un processus de classification par règles qui permet d'associer automatiquement les meilleurs comités initiaux aux nouveaux utilisateurs. Les méthodes proposées sont testées avec les données du système de recommandation de films MovieLens [1] en considérant certains critères concernant l'utilisateur (sa profession, sa ville, son genre de film préféré et ses évaluations sur certains films). Ces méthodes peuvent être considérées comme un complément aux systèmes de filtrage permettant de diversifier les recommandations provenant de comités variés et de positionner les utilisateurs dans les espaces de comités selon l'importance accordée par chaque utilisateur pour chaque critère.

2.4.3.4 Synthèse

Nous présentons une synthèse de quelques systèmes assistants étudiés précédemment dans le tableau 2.1.

2.4.4 Les systèmes assistants pour notre classe de problèmes

Il s'agit de systèmes qui se basent sur la réutilisation des expériences passées des utilisateurs et qui utilisent la technique du RàPC pour calculer leur recommandations. Nous pouvons distinguer ces systèmes selon la structure du cas utilisé pour modéliser l'expérience, le type de la recommandation (implicite ou explicite) et selon l'approche de la réutilisation de l'expérience.

L'approche Broadway (Trousse *et al.*, 1999) est une approche d'aide à la navigation sur le Web centrée fouille de données. L'approche repose sur l'utilisation de techniques de raisonnement à partir de cas afin de recommander à un utilisateur des pages qui ont satisfait des utilisateurs ayant navigué d'une manière similaire. Elle s'appuie sur un filtrage basé sur la réutilisation de comportements de l'utilisateur et ou un groupe d'utilisateurs. Il s'agit d'une architecture de type client/serveur. Du côté serveur, Broadway est composé de différents

[1] http ://www.movielens.umn.edu

Système	Type de filtrage	Technique de filtrage	Approche	Évaluation	Type d'information
AIRA	contenu	classification	individuelle	explicite	termes pour étendre une requête
WebMate	contenu	statistique	individuelle	explicite	pages Web et termes pour raffiner une requête
Siteseer	collaboratif	catégorisation	collaborative	explicite	signets
Phoaks	collaboratif	catégorisation	collaborative	implicite	ressources Web (URLs)
CoWing	collaboratif	RàPC & réseau de neurones	collaborative	explicite	signets
I-SPY	collaboratif	RàPC	collaborative	explicite	documents
Bibster	collaboratif	statistique	collaborative	-	documents
Fab	hybride	statistique	collaborative	explicite	documents
COSYDOR	hybride	statistique	collaborative	explicite	termes pour améliorer la formulation de requête
MAIS	hybride	classification (ELA)	collaborative	implicite	pages Web

TAB. *2.1: Synthèse des systèmes assistants*

processus (le proxy HTTP, serveur d'information de page, serveur d'information d'utilisateur, serveur de recommandation, serveur d'annotation, etc.). Du côté client, sont exécutées plusieurs applets (gestionnaire, barre d'outils, etc.). Un cas dans BROADWAY reférence une expérience précise dans une navigation (Jaczynski, 1998). À la différence des systèmes RàPC classiques, l'alimentation de la base de cas dans BROADWAY est guidée par les cas cibles à résoudre. La partie problème d'un cas cible est composée de deux parties : 1) l'ensemble des n dernières pages visités et 2) l'ensemble, éventuellement vide, d'événements marquants dans l'historique de la navigation actuelle. Les pages bien évaluées par l'utilisateur constituent les événements marquants dans l'historique. L'ensemble des événements décrivant la partie problème sont reliés entre eux par des relations temporelles de type : avant, après, pendant, précède, etc.. À chaque changement de page pendant la navigation de l'utilisateur, le système de recommandation élabore un nouveau cas cible et cherche dans sa base de cas des cas sources similaires. La solution apportée par un cas source est constituée par les pages positivement évaluées et

qui sont visitées après l'épisode similaire au cas cible actuel. Si aucun cas source n'est trouvé, le système cherche à élaborer de nouveaux cas sources à partir de la base de navigations. La mesure de similarité entre les cas est une mesure complexe qui prend en compte les relations d'ordre entre les événements composant les cas. Parmi les applications de cette approche :

- Le système BroadwayV1 (Browsing advisor reusing pathways) (Jaczynski, 1998) est un assistant de navigation sur le Web permettant de suivre un utilisateur et de recommander des pages à visiter selon son comportement. Broadway est implementé sous forme de *proxy* qui s'interpose entre le groupe d'utilisateurs et le web. Le proxy, composé d'un ensemble de serveurs, a la charge de tracer les navigations des utilisateurs dans une base de navigations. Une navigation est constituée d'une séquence de pages visitées par un utilisateur. Le système repose sur l'hypothèse que les navigations sont mono-thème ; autrement dit l'utilisateur ne change pas d'objectif durant une navigation. Pour chaque page visitée le système enregistre son adresse (i.e. URL), un vecteur de mots-clés décrivant le contenu de la page, une éventuelle évaluation explicite de l'intérêt de la page et une évaluation implicite de la page qui est calculée en fonction de la taille de la page et de son temps d'affichage. Afin de permettre de collecter les évaluations des pages, le proxy Broadway insère dans chaque page visitée deux applets Java : une barre d'évaluation explicite de la page et une autre qui permet d'informer le proxy de l'affichage effectif de la page (Shahabi *et al.*, 2001). Cette dernière applet permet alors de ne pas comptabiliser le temps de transfert de la page comme temps d'affichage mais aussi de pallier au problème de cache en informant le proxy de la visite de la page même si celle-ci est dans la mémoire cache du navigateur de l'utilisateur.

  C'est un système accessible par un groupe d'utilisateurs, qui permet ainsi une collaboration indirecte en réutilisant les cas issus des navigations du groupe. Broadway observe les navigations de différents utilisateurs, récolte les évaluations et les annotations de ces utilisateurs pour établir une liste de documents pertinents. Dans ce système, un cas est une expérience précise extraite des navigations antérieures avec une évaluation des situations comportementales. Broadway utilise les navigations des utilisateurs pour en extraire des expériences utiles (cas) permettant d'associer à un comportement (succession de pages visitées) un ensemble de pages qui seront proposées à l'utilisateur.
- Le système BeCBKB (Kanawati *et al.*, 1999b) propose à l'utilisateur des recommandations pour reformuler ses requêtes (i.e. ajout et suppression de mots dans les requêtes). Le système applique l'approche Broadway pour le calcul de recommendations. Dans ce système, un cas représente une expérience précise dans une session de recherche. La partie problème est constituée d'une partie instantanée composée du contexte de la session à partir de laquelle le cas est extrait et d'une partie comportementale contenant les p dernières requêtes observées, l'ensemble des documents retenus ou exclus et les dernières configurations des requêtes qui ont mené à l'échec. La partie solution est composée des

recommandations positives et négatives. Une recommandation positive améliore la satisfaction de l'utilisateur alors qu'une recommandation négative décrit une configuration de requête ayant abouti à un échec. Le cas comprend également une autre partie évaluation. Cette partie contient un indice mesurant la confiance de la solution positive proposée par le cas. Cette confiance est augmentée à chaque fois que la solution est approuvée par l'utilisateur. Une expérimentation dans le monde réel est nécessaire pour valider le système de recommandation proposé.

Le système MUSETTE (Modelling USEs and Tasks for Tracing Experience) (Champin *et al.*, 2004) présente un environnement pour la capture de connaissances à partir d'expériences. Ce système à base de RàPC permet la modélisation de l'expérience d'utilisation d'un système informatique afin de réutiliser cette expérience pour assister l'utilisateur à effectuer ses tâches. Dans ce système, un agent observe les interactions de l'utilisateur avec le système et génère les traces primitives conformes à un modèle d'utilisation général. Ensuite, un analyseur de traces génériques extrait les épisodes significatifs à partir de la trace générique et selon les signatures de tâches expliquées. La notion de signature de tâche expliquée désigne un motif récurrent dans les traces, qui est un événement marquant relié à au moins une situation. L'expérience de l'utilisateur est récupérée à partir de la trace primitive. Une trace est une séquence d'états et de transitions présentant l'activité de l'utilisateur. Une trace peut être composée par des objets d'intérêt (OI) qui peuvent appartenir à une des trois catégories suivantes :

- les entités,
- les événements,
- les relations.

Le modèle d'utilisation d'un système particulier décrit les types d'entités, d'événements et des relations pouvant être observables pour produire des traces primitives. Des contraintes aditionnelles incluant des contraintes sur les structures internes des objets d'intérêt peuvent faire partie du modèle d'utilisation. Le modèle d'observation décrit un ensemble de moyens pour accéder aux données pertinentes dans le système aussi bien que les règles contraignant le processus de production de trace (comme par exemple, détérminer les sous-ensembles de toutes les entités observables qui doivent être écrites dans une trace à un instant donné) (Mille, 2006b). Les traces sont produites à partir de l'observation des interactions entre l'utilisateur et le système. La structure d'une trace n'est pas limitée à un flot continu d'entités et d'événements, éventuellement en relation. En effet, les entités sont utilisées pour représenter l'état du système à un moment donné (ou durant une période considérée comme un instant) dans le contexte du modèle d'utilisation. Un épisode présente n'importe quelle partie de trace qui correspond à une expérience spécifique dans l'exécution d'une tâche spécifique et qui peut être réutilisée dans une situation similaire. Les parties de traces sont reconnues comme des épisodes reliés à une tâche particulière grâce aux « signatures de tâches expliquées ». Les épisodes peuvent être annotés ou expliqués de deux façons : annotation en texte libre ou annotation en connaissances

formelles (Stuber *et al.*, 2004). Le système se base sur deux types de modélisation :

- Modélisation de l'environnement : l'environnement de partage est constitué de documents utilisés par les acteurs humains durant leurs activités.
- Modélisation des traces d'utilisation : le modèle MUSETTE se caractérise par une structure de la forme état/transition. Les objets d'intérêt (entités et événements) observables par la tâche sont décrits dans un modèle d'utilisation. Une trace d'utilisation décrit les changements d'états du système observables par l'utilisateur et provoqués par ses opérations (représentées par des événements). Lorsqu'une partie d'une trace d'utilisation vérifie une signature de tâche expliquée, elle constitue un « cas d'utilisation ». Une trace est un ensemble d'épisodes, une action est le début d'un épisode, une séquence est une suite indivisible d'états et de transitions, une alternative est une séquence permutable avec une autre séquence ou bien avec une action imbriquée (Stuber *et al.*, 2004).

Le système permet donc le partage de connaissances et d'expériences lors de la réalisation d'une tâche collective complexe. Il s'agit de mettre l'expérience collective à disposition des utilisateurs par le biais d'un assistant contextuel. Cette expérience collective est la combinaison des expériences individuelles acquises à partir des traces d'utilisation laissées par les acteurs dans le système.

Le projet PIXED (Projet d'Intégration de l'eXpérience pour l'Enseignement à Distance) (Héraud, 2002) est une plate-forme d'enseignement à distance par Internet implémentant des hypermédias adaptatifs grâce à l'expertise et la réutilisation de l'expérience. C'est un outil qui permet l'échange et la réutilisation des connaissances entre apprenants, entre apprenants et enseignants, et entre enseignants. Ce système se caractérise par l'utilisation de l'expérience concrête du système (épisodes d'apprentissage). Un cas constitue un modèle initial d'un apprenant, des traces de navigation et l'état final du modèle de l'apprenant. Le système propose aux apprenants consultant un cours en ligne, de réutiliser le parcours d'apprentissage d'autres apprenants. PIXED permet de renvoyer à l'apprenant une possibilité d'exploiter les traces d'apprentissage comme source de connaissances pour assister son orientation dans la progression d'apprentissage dans un cours. La trace est considérée comme un cas composé d'une séquence d'actions que nous pouvons interpréter selon des contextes différents. L'intégration du paradigme RàPC a pour objectif de répondre à une double problématique : aider à la navigation et la construction du domaine pour l'expertise. Il offre la possibilité d'annotation de la représentation du domaine ainsi que des documents. Cette annotation peut être du texte libre et de type choix multiple. L'apprenant a la possibilité de garder ses annotations pour lui ou de les partager avec les autres apprenants ou enseignants. Les apprenants disposent d'un outil d'édition de leur modèle du domaine. Cet outil permet d'annoter la vision que le système a d'eux et les apprenants peuvent ainsi exprimer leurs accords et désaccords sur cette vision. Concrètement, ils peuvent donner leur estimation sur la maîtrise qu'ils ont de chaque notion

et y ajouter une annotation. Cette annotation est à destination de l'enseignant responsable du cours et éventuellement de tous les autres étudiants si l'apprenant le désire. Le domaine est représenté sous forme d'un réseau notionnel construit par les enseignants qui peuvent éditer librement leur représentation du domaine. Chaque enseignant possède sa propre représentation du modèle et plusieurs enseignants ont la possibilité de partager une représentation commune. L'expérience du système est constituée des épisodes (traces) d'apprentissage, que le système enregistre au cours de son utilisation, stockés sous la forme de cas. Un cas de la base est un triplet constitué du modèle initial d'un apprenant, de sa trace de navigation et de l'état final de son modèle d'apprenant. L'utilisation de l'expérience pour l'aide à la navigation se résume comme suit : le système élabore un cas à partir du nouvel apprenant puis il cherche dans la base de cas les cas similaires. Cette similarité est fonction des distances particulières entre notions identiques, des annotations et de la notion objectif d'apprentissage. Lors de la phase d'apprentissage, PIXED conserve aussi la trace de réutilisation (le cas élaboré, les cas similaires et le cas stocké). Ce second type de trace va constituer une seconde base de cas qui a pour but de permettre au système d'affiner ses mesures de similarité. Lorsque le système doit choisir entre plusieurs documents, il intègre l'expérience dans l'évaluation de la pertinence de chaque document. Ainsi, un document qui aura beaucoup été consulté dans des cas similaires d'apprentissage positifs et peu dans des cas d'apprentissage négatifs sera favorisé. PIXED propose aussi à l'enseignant des modifications de son modèle permettant de mieux expliquer les réussites ou échecs des apprenants par rapport à leur utilisation de son modèle.

Un autre exemple d'utilisation d'expérience est celui de (Masterton, 1997). C'est un système à base d'agents pour assister l'enseignement dans un cadre collaboratif asynchrone. Les agents developpés utilisent une approche à base de cas en offrant de l'aide aux problèmes communs des étudiants. Les cas sont des exemples de problèmes expérimentés par des étudiants dans des années précédentes et de discussions sur comment ils ont été résolus. Un cas est alors composé d'une question et des réponses qui lui ont été faites. Le « Participant Virtuel » est un exemple de système utilisant l'enseignement à partir de cas. Il s'agit d'un assistant virtuel intervenant dans les logiciels de type forum de discussion. La base de cas est constituée de questions posées par les personnes participant au forum et de solutions (proposées par les mêmes intervenants). Cet environnement a été expérimenté dans un cours où les forums de discussion étaient utilisés. Le participant virtuel (PV) interagit avec la conférence en essayant d'identifier les conversations courantes. L'algorithme de recherche accumule les mots-clés à partir de ces conversations et les utilise pour retrouver les cas dans la base de cas. Lorsqu'un cas est retrouvé, un message de vue d'ensemble est affiché expliquant que le PV a identifié un thread de conversation d'année précédente pouvant être pertinent pour la discussion courante. Le message donne le contexte original du cas et une vue d'ensemble du problème qu'il adresse. Les étudiants sont ainsi capables de retrouver des informations complémentaires sur le cas par un nombre de questions

fixées pouvant être posées en les adressant à la conférence. Les étudiants et les tuteurs peuvent interagir directement, par contre le PV interagit seulement avec la conférence. Ce système tend à s'améliorer suite aux retours donnés par les étudiants et les tuteurs.

2.4.4.1 Synthèse

Une synthèse des approches de recommandation pour notre classe de problèmes est présentée dans le tableau 2.2.

Système	**Expérience**	**Type de recommandation**	**Approche**	**Retour**
MUSETTE	utilisation du système	implicite	collective	explicite
BROADWAY V1	navigation sur le Web	implicite	individuelle	explicite
BeCBKB	raffinement de requête	implicite	individuelle	explicite
PIXED	type d'enseignement	explicite	individuelle	explicite
Participant Virtuel	question/réponse	implicite	individuelle	explicite

Tab. *2.2: Une synthèse des approches de recommandation pour notre classe de problèmes*

2.4.5 Conclusion

Nous constatons que le terme « expérience » a été utilisé dans plusieurs contextes et de plusieurs façons suivant les domaines et les buts. En effet, il n'y a pas de définition précise de l'expérience. Cela est dû à ses spécificités : personnelle, relative, et contextuelle. Nous pensons que cette expérience est très importante, notamment dans les systèmes de recherche d'information et plus particulièrement dans les systèmes de recommandation. Nous constatons que la plupart des systèmes de recommandation demandent explicitement aux utilisateurs de donner des ressources (commentaires, avis, annotations, rangs) et de les partager avec les autres utilisateurs afin d'offrir une meilleure recommandation. Les avis et les conseils des collègues et des amis permettent souvent d'améliorer la qualité de la recherche personnelle.
Nous nous basons sur une approche hybride qui applique :

– le filtrage collaboratif puisque les recommandations fournies tiennent compte des avis et des expériences des autres utilisateurs similaires,

– le filtrage sur le contenu puisque l'approche permet de déduire les centres d'intérêt de l'utilisateur et surtout d'apprendre à partir de ses retours pour mieux lui recommander.

Pour réaliser cela, nous utilisons une technique de RàPC pour calculer les recommandations. Cette technique nous offre une méthodologie pour gérer l'expérience et pour apprendre à mieux calculer les recommandations. Une expérience dans notre système est structurée dans un cas. La partie problème présente un besoin d'information pour un utilisateur donné, décrit par un thème d'intérêt et son interprétation par l'utilisateur. La partie solution présente l'expérience de correspondance apprise qui indique où chercher dans la base bibliographique chez les autres utilisateurs. Les systèmes de recommandation distribués présentent, en plus de la problématique de calcul des recommandations, une autre problématique concernant la sélection des utilisateurs avec qui l'expérience sera partagée et auxquels les demandes de recommandation seront envoyées. C'est le deuxième volet de notre état de l'art que nous traitons dans le chapitre suivant.

Chapitre 3

La formation de comités dans les systèmes de type égal-à-égal

3.1 Introduction

L'architecture égal-à-égal (P2P) (Mishra et Ahuja, 2004) est une architecture réseau selon laquelle chaque nœud (appelé pair) possède des capacités et des responsabilités équivalentes. Ainsi chaque pair fonctionne à la fois comme serveur et comme client (Stoica *et al.*, 2001). Parmi les systèmes P2P les plus connus, nous trouvons Napster qui est apparu en juin 1999. Il permet à des utilisateurs de partager des fichiers musicaux. Une liste de fichiers disponibles sur le réseau est mise à disposition grâce à un serveur. Chacun peut ainsi faire ses recherches puis récupérer des fichiers. Seule la récupération d'objets est décentralisée. Le principe est simple : les membres enregistrent les références des morceaux de musique qu'ils ont à offrir dans une base de données centralisée, maintenue sur le serveur Napster. Pour obtenir un titre, il suffit d'effectuer une recherche dans la base, puis de télécharger le morceau directement à partir de l'ordinateur du membre qui en dispose. En mars 2000 est arrivé Gnutella [2] : le premier système P2P totalement décentralisé. En effet, il permet la recherche et la récupération d'objets sans nécessiter de serveur.

Les applications de la technologie P2P sont diverses. Par exemple nous trouvons :

1. **Partage de fichiers** : constitue l'utilisation la plus connue du P2P et regroupe toutes les applications permettant l'échange de fichiers. Nous trouvons comme exemples, Napster, Gnutella et ses différentes implémentations, Freenet (Clarke *et al.*, 2001).
2. **Communication** : nous y trouvons la messagerie instantanée (par exemple, AOL, ICQ[3], Groove, AIM, MSN Messenger [4], etc.), la téléphonie par Internet (Webphone, PhoneFree)

[2] http ://www.gnutella.com/
[3] http ://www.icq.com/
[4] http ://webmessenger.msn.com/

ainsi que la vidéo conférence tels que Cu-SeeMe (Dorcey, 1995), ICUII [5].

3. **Informatique répartie** : ce type d'application permet l'utilisation de ressources des ordinateurs non exploitées tel que la puissance du processeur ou le disque dur. Cela peut servir pour le fonctionnement de grandes applications. Par exemple, PopularPower [6] est une société de vente et de don (académique et scientifique) du temps CPU, OceanStore (Kubiatowicz *et al.*, 2000) présente un système de stockage persistent à grande échelle et le système Seti@home [7] pour les grilles de calcul.
4. **Collaboration** : regroupant des applications de collectitiels tels que Groove Networks [8]. C'est une application de travail en groupe qui facilite le travail en ligne. Le système permet de former des groupes de collaborateurs de façon décentralisée et ad hoc. et permet aux membres d'un groupe d'interagir dans des espaces de partage sécurisés pour supporter leur travail collaboratif en temps réel. Nous trouvons aussi les applications de jeux en réseau telles que Xtank [9].

Les avantages qu'une architecture d'égal-à-égal peut nous offrir sont :

- **le dynamisme** puisque l'architecture P2P peut supporter un environnement dynamique où un pair peut dynamiquement rejoindre ou quitter le système.
- **une meilleure résistance aux pannes** dans le sens où le système est insensible aux défaillances car la sortie d'un pair particulier n'affecte en rien le reste du système.
- **l'absence d'une autorité centrale** qui contrôle les pairs et qui limite leur autonomie. Chaque pair est capable d'agir sans intervention de l'extérieur et décide de ses propres actions.
- **une protection des contenus privés** puisque chaque utilisateur garde localement ses données et contrôle les schémas de partage avec les autres.

Une des problématiques importantes dans les systèmes P2P est la formation de comités. Le problème est le suivant : étant donnée une requête d'un utilisateur, quels les sont les pairs les plus pertinents pour cette requête. Certaines approches ont pour but de localiser des ressources précises, d'autres tendent à trouver des résultats satisfaisant les requêtes. Dans la première approche, nous disposons d'une description détaillée d'une ressource (par exemple, nom du fichier, date de l'article, auteur de l'article, etc.). Il s'agit d'une recherche exacte d'une ressource particulière. La recherche aboutit à un succès si la ressource est trouvée et un échec dans le cas inverse. La deuxième approche s'oriente plus vers une recherche plus « générale » d'information. En effet, cette approche prend comme entrée une description d'un besoin d'information dont nous ignorons les résultats et fournit comme sortie un ensemble de ressources correspondant au

[5] http ://www.icuii.com/
[6] http ://www.popularpower.com/index2.html
[7] http ://setiathome.free.fr/
[8] http ://office.microsoft.com/en-us/groove/
[9] http ://quozl.netrek.org/xtank/

besoin. C'est à cette approche que nous nous intéressons plutôt qu'à la première qui concerne plus une localisation de ressources qu'une recherche d'informations.

Nous commençons par définir la terminologie utilisée. Ensuite, nous introduisons notre problématique de formation de comités. Puis, nous présentons une classification des approches et quelques systèmes existants. Finalement, nous concluons.

3.2 Problème de formation de comités

Dans cette section, nous traitons en détail la problématique de formation de comités dans les systèmes P2P.

3.2.1 Terminologie

Nous précisons dans cette section les termes utilisées dans notre étude :

1. **Pair** : agent logiciel dans le système et nœud du réseau P2P.
2. **Voisin** : les voisins d'un pair sont les agents que ce pair connait dans le réseau et à qui il peut envoyer une requête.
3. **Ressource** : toute entité susceptible d'être identifiée, nommée et manipulée à travers ses représentations dans le réseau. Nous désignons par ressources les objets disponibles ou qui peuvent être localisés et atteints à travers le réseau. L'identification d'une ressource comprend donc deux aspects : le nommage et l'adressage.
4. **Ontologie de domaine** : c'est un ensemble structuré de thématiques qui décrivent un certain domaine. L'ontologie du domaine peut compléter la définition du centre d'intérêt en explicitant la sémantique de certains termes. La première définition de l'ontologie dans le domaine de l'informatique est proposée par Neches et al. (Neches *et al.*, 1991). Ils définissent l'ontologie comme : « les termes et les relations de base comportant le vocabulaire d'un domaine aussi bien que les règles pour combiner les termes et les relations afin de définir des extensions du vocabulaire ». Une des définitions de l'ontologie la plus citée en IA est celle de Gruber (Gruber, 1993) où l'ontologie présente une spécification explicite d'une conceptualisation [10]. Cela suppose l'existence d'un modèle abstrait d'un certain aspect du monde, qui se compose d'un ensemble de définitions de concepts et de relations. L'ontologie présente donc une spécification du modèle dans un langage non ambigu, qui peut être compris et manipulé aussi bien par les machines (agents logiciels) que par les personnes (agents humains). En représentation des connaissances, le terme ontologie fait référence à un modèle d'un domaine particulier (du monde réel). L'objectif premier d'une ontologie est de modéliser un ensemble de connaissances dans un domaine

[10] : « an ontology is an explicit specification of a conceptualization »

donné. Pratiquement, un tel modèle se présente comme un ensemble de définitions de concepts munis de propriétés et de relations entre ces concepts. L'utilisation effective d'une ontologie nécessite un langage de représentation des ontologies bien défini et des modules associés de raisonnement qui soient efficaces.

Dans notre application, l'ontologie permet de faciliter la mise en correspondance entre l'utilisation des thèmes des différents utilisateurs et ainsi faciliter la recherche des documents pertinents dans les bases des autres utilisateurs.

5. **Centre d'intérêt** : est présenté par des thématiques et leur description (par exemple, mots-clés) sur lesquelles l'utilisateur a des besoins en information. Ces thématiques appartiennent à l'ontologie du domaine. Le centre d'intérêt exprime le domaine d'expertise de l'utilisateur ou son périmètre d'exploration.
6. **Requête** : présente une demande d'information relative à un centre d'intérêt. Elle se caractérise essentiellement par une description de la thématique recherchée et l'identité du pair initiateur de la requête. Dans notre système, une requête désigne une demande de recommandation puisqu'elle va déclencher le processus de calcul de recommandations.
7. **Comité** : c'est un ensemble de pairs jugés pertinents pour une thématique donnée. Chaque agent peut avoir autant de comités que de centres d'intérêt. Un comité, pour un centre d'intérêt donné et pour un pair donné, peut changer au cours du temps.
8. **Routage de requête** : le but du routage consiste à former des comités. Nous voulons enlever toute sorte d'ambiguïté de ce terme pour ne pas le confondre avec le routage physique des messages dans le réseau. En effet, dans le routage physique, c'est la couche réseau qui se charge d'acheminer des paquets tout au long d'un parcours, d'une source jusqu'à une destination. Nous nous intéressons plutôt au routage sémantique : étant donnée une requête d'un utilisateur, quels sont les pairs les plus pertinents pour cette requête. Nous faisons abstraction sur la façon dont le message de la requête arrive. Le problème se résume donc à savoir comment un utilisateur peut localiser un pair qui offre ce qu'il cherche en l'absence d'une base de données ou d'un index centralisé.
9. **Historique d'interaction** : c'est l'ensemble des communications passées entre un pair et les autres pairs (les requêtes envoyées et les résultats envoyés par les pairs). L'historique d'interaction contient également le retour de l'utilisateur sur les résultats recommandés qui regroupe l'ensemble des informations collectées sur le comportement de l'utilisateur. Ce retour peut être directement fourni par lui (retour explicite) ou dérivé à son insu (retour implicite).

3.2.2 Définition de la formation de comités

La formation de comités est un module qui prend en entrée essentiellement une requête utilisateur et fournit en sortie un comité d'agents pertinents à qui la requête sera envoyée. En

plus de la requête, ce module utilise des informations complémentaires pour calculer l'ensemble d'agents. Les informations, la structure de la requête ainsi que le fonctionnement de ce module, dépendent de l'approche et de chaque système. Ces informations peuvent porter sur l'utilisateur, les autres utilisateurs, l'historique, etc..
La description générale du module de formation de comités est présentée dans la figure 3.1.

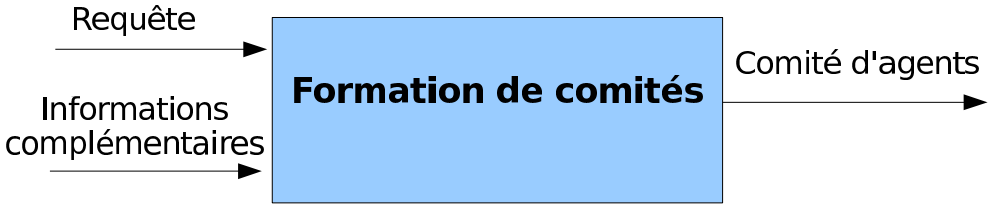

Fig. *3.1: La formation de comités*

3.3 Approches de formation de comités

Dans cette partie, nous étudions différentes approches qui ont traité le problème de la formation de comités. Tout d'abord, nous commençons par dégager des critères de classification des différentes approches. Ensuite, nous donnons les critères utilisés pour l'évaluation des approches citées. Finalement, nous présentons une classification des approches existantes et une description de certains systèmes.

3.3.1 Critères de classification des approches

Afin de pouvoir classifier et distinguer les approches, nous avons considéré les critères suivants :

1. ***Type du système P2P***
2. ***Type de la recherche***
3. ***Implication de l'utilisateur***
4. ***Utilisation d'ontologie***
5. ***Capacité d'apprentissage***
6. ***Nature du comité***

Nous détaillons chacun de ces critères et nous donnons des exemples à chaque fois.

1. ***Type du système P2P***
   En se basant sur le modèle d'architecture appliqué aux systèmes P2P, nous distinguons deux grandes classes de systèmes P2P : structurés et non-structurés (voir la figure 3.2).

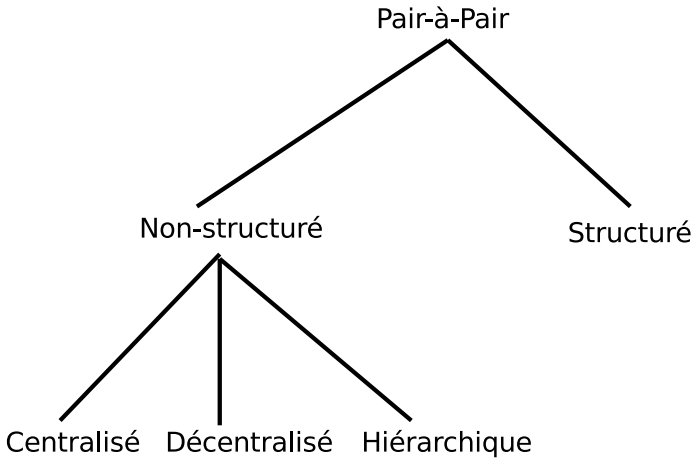

FIG. *3.2: Taxonomie des modèles des systèmes P2P*

Dans la première classe, la propagation des requêtes se fait selon une structure d'organisation prédéfinie des pairs. Dans la deuxième classe, la recherche la plus simple se base sur une propagation aléatoire des requêtes dans le graphe des pairs.

(a) ***Les systèmes structurés*** : dans ce type de système, la localisation des ressources est contrôlée par une table de hachage distribuée (DHT). C'est une technique d'indexation et de placement par mots-clés qui s'applique principalement aux systèmes de stockage distribué à grande échelle. Il s'agit d'un tableau de correspondance entres des clés et des valeurs. L'intérêt d'une table de hashage réside dans la rapidité des algorithmes de recherche. Les DHT permettent d'éclater une table de hashage sur plusieurs nœuds. Chaque machine est responsable d'un ensemble de clés. Pour chercher une ressource, il faut d'abord chercher le nœud qui supervise l'ensemble des clés associées puis l'interroger. Techniquement, un réseau P2P utilisant la DHT se base sur un algorithme de routage par contenu qui peut garantir de retrouver, le plus efficacement possible, les ressources qu'un utilisateur a déposées sur le réseau. Lorsqu'une ressource est déposée (partagée) sur un réseau, un identifiant est assigné également à la ressource sur la base d'un hachage du contenu. Une copie de la ressource sera alors gardée dans le pair ayant l'identifiant le plus proche de celui de la ressource. La technique de routage par contenu effectue une correspondance entre l'identifiant de ressource et l'identifiant de pair afin de router les requêtes vers les pairs qui ont un identifiant similaire à celui de la donnée recherchée. Dans ce type de système, l'espace de recherche est organisé à l'aide d'une structure de données adéquate. Cette structure est utilisée pour placer les ressources sur les nœuds, ce qui rend la recherche d'une ressource efficace. Il existe plusieurs systèmes tels que

CAN (Ratnasamy *et al.*, 2001), P-Grid (Aberer, 2001), Chord (Stoica *et al.*, 2001) et Pastry (Rowstron et Druschel, 2001). Ces systèmes diffèrent selon la structure de données utilisée (espace à n-dimensions pour CAN, arbre binaire de recherche pour P-Grid, anneau pour Chord et Pastry). Cependant, ces systèmes forcent le placement des ressources sur certains nœuds, ce qui limite l'autonomie des participants.

Exemple d'application : Chord (voir figure 3.3)

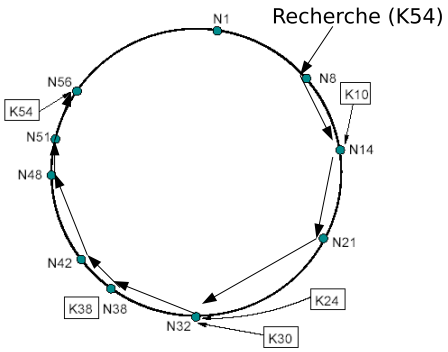

Fig. *3.3: Recherche naïve*

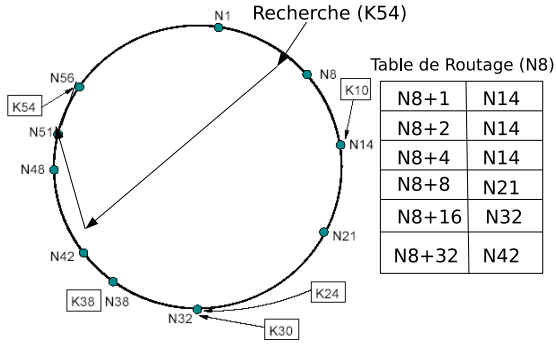

Fig. *3.4: Recherche améliorée*

Chord est une infrastructure de stockage et de routage. Les nœuds sont répartis sur un anneau logique et les ressources sont réparties sur les différents nœuds de l'anneau. Chaque nœud a la connaissance de son prédécesseur et de son successeur. Une fonction de hachage régulière génère une clé pour chaque nœud à partir de son adresse IP. Ensuite chaque nœud est placé dans l'anneau de manière à ordonner les clés par ordre croissant. Ainsi, chaque nœud est responsable de l'intervalle de clés [clé(nœud actuel), clé(nœud suivant)[.

Pour mieux comprendre le routage DHT, nous prenons un exemple qui est décrit dans la figure 3.3. Les ressources sont définies sur les différents nœuds du réseau comme suit : étant donné K, la clé de hachage d'une ressource, cette ressource doit être localisée sur le nœud N immédiatement supérieur ou égal à K. Par exemple, $K24$ et $K30$ sont localisées sur le nœud $N32$, tandis que $K38$ sur $N38$ et $K54$ sur $N56$. Supposons que nous voulons chercher une ressource avec la clé de hachage $K54$ à partir du nœud 8. Dans le routage naïf (figure 3.3), lorsqu'un nœud reçoit la requête, il la propage vers son successeur direct s'il ne possède pas la clé de hachage donnée dans la requête ($N8$ vers $N14$, puis $N21$, etc.). La complexité de cette recherche est linéaire en fonction du nombre de nœuds. En revanche, dans la figure 3.4, chaque nœud (i) a une table de routage. Cette table contient m entrées acceptant ainsi au maximum 2^m nœuds dans le réseau. La $k^{ième}$ entrée de cette table chez le nœud i contient l'identité du premier nœud, s, qui suit i par au moins 2^{k-1} dans l'anneau,

s= $successeur((i + 2^{k-1}) mod\ 2^m)$ (voir figure 3.4). Pour chaque requête reçue, le nœud cherche l'entrée avec la plus grande valeur inférieure à la clé recherchée. La requête est ensuite transmise au nœud sélectionné par cette entrée. Le processus est appliqué récursivement jusqu'à ce que la ressource soit trouvée. Le nombre moyen de messages transmis pour une requête dans ce cas est $log(N)$ avec N le nombre maximum de nœuds.

Les $DHTs$ proposent une bonne solution pour trouver rapidement une ressource bien définie. Il faut en effet connaître la clé de la ressource précisément. Une recherche de texte, ou une recherche approchée n'est pas possible.

(b) ***Les Systèmes non-structurés*** :

tous les pairs sont au même niveau sur le réseau logique et la découverte s'effectue par diffusion aux voisins. Dans ces systèmes, le principe de recherche est le suivant : un utilisateur soumet sa requête à un nœud quelconque du réseau, celui-ci résoud la requête localement et la propage récursivement à un certain nombre de nœuds (par exemple, choisis aléatoirement dans une approche naïve). La recherche s'arrête quand une condition d'arrêt est satisfaite (par exemple, quand une certaine profondeur de recherche a été atteinte). Pour cette classe de systèmes, le modèle d'architecture peut être centralisé, décentralisé ou hiérarchique (voir figure 3.2).

i. **Système centralisé**

Il s'agit d'une architecture dans laquelle un serveur se charge de mettre en relation directe tous les pairs connectés. L'intérêt de ce modèle réside dans l'indexation centralisée de tous les nœuds. En général, la mise à jour de cette base s'effectue en temps réel, dès qu'un nouveau pair se connecte ou quitte le réseau. L'avantage de l'indexation centralisée réside dans l'efficacité des recherches. L'algorithme de recherche est très simple parce que toute recherche passe par le serveur centralisé. Dès lors que des pairs sont retrouvés pour répondre à une recherche, les contacts directs seront établis entre eux. Cependant, cette architecture présente également deux principaux défauts :

- d'une part, elle ne propose qu'une seule porte d'entrée ; son serveur centralisé, ce qui peut bloquer le fonctionnement de l'ensemble du réseau en cas de défaillance de cette machine et une surcharge du traffic sur le réseau,
- d'autre part, le fait d'indexer tous les nœuds ne garantit pas le passage à l'échelle du système.

Exemple d'application : Napster (voir figure 3.5)
Dans Napster, les pairs du réseau enregistrent les fichiers dont ils disposent chez un serveur central. Ils contactent ce serveur pour obtenir les coordonnées (adresse IP et numéro de port) d'un élément possédant les fichiers recherchés.

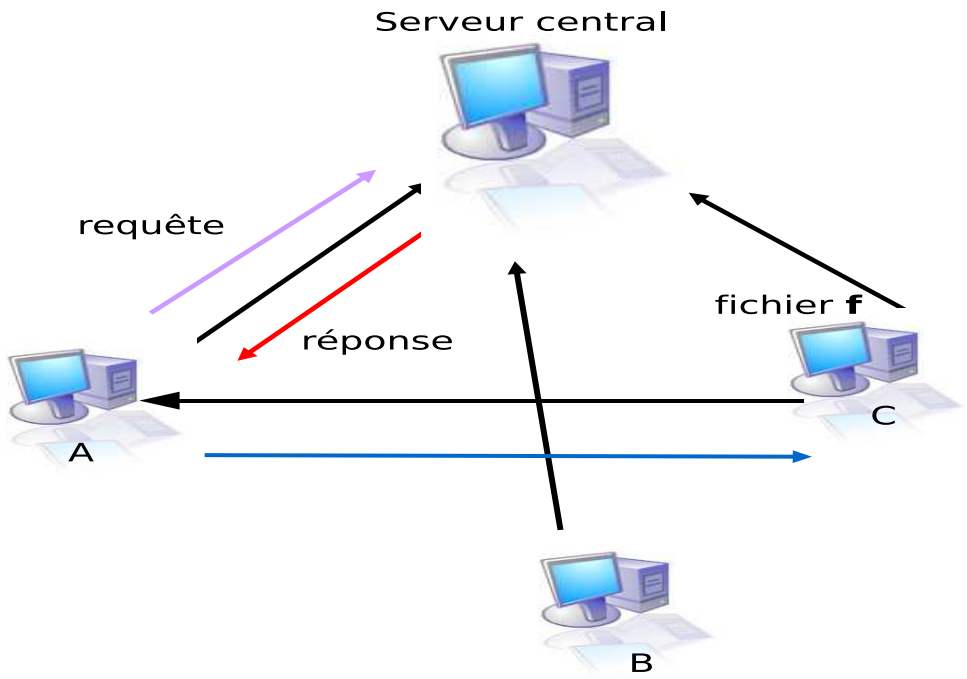

Fig. *3.5: Architecture du système Napster*

Le fonctionnement de Napster se fait comme suit :

- 1. Les nœuds se connectent auprès du serveur central qui indexe les ressources de chaque nœud.
- 2. Les nœuds envoient leurs requêtes au serveur central (par exemple, le nœud A demande un fichier f).
- 3. Les réponses sont envoyés du serveur central aux nœuds (par exemple, les nœuds qui disposent du fichier f (nœud C)).
- 4. La connexion P2P se fait enfin entre les nœuds pour récupérer les résultats demandés (par exemple, A télécharge le fichier f directement depuis le nœud C).

Dans ce modèle, le problème de routage de requête ne se pose pas.

ii. **Système décentralisé**

Aucun serveur centralisé n'existe dans cette architecture ; toutes les machines jouent un rôle identique. C'est pour cela que ce type de réseau est appelé P2P pur (voir figure 3.6). Contrairement aux systèmes centralisés, la recherche dans ces réseaux se base sur une propagation de la recherche dont le mode le plus simple est une série de diffusions. Cela a pour conséquence de ralentir les échanges de ressources entre machines. La performance d'un tel réseau purement P2P dépend donc essentiellement de l'algorithme de routage adopté pour propager

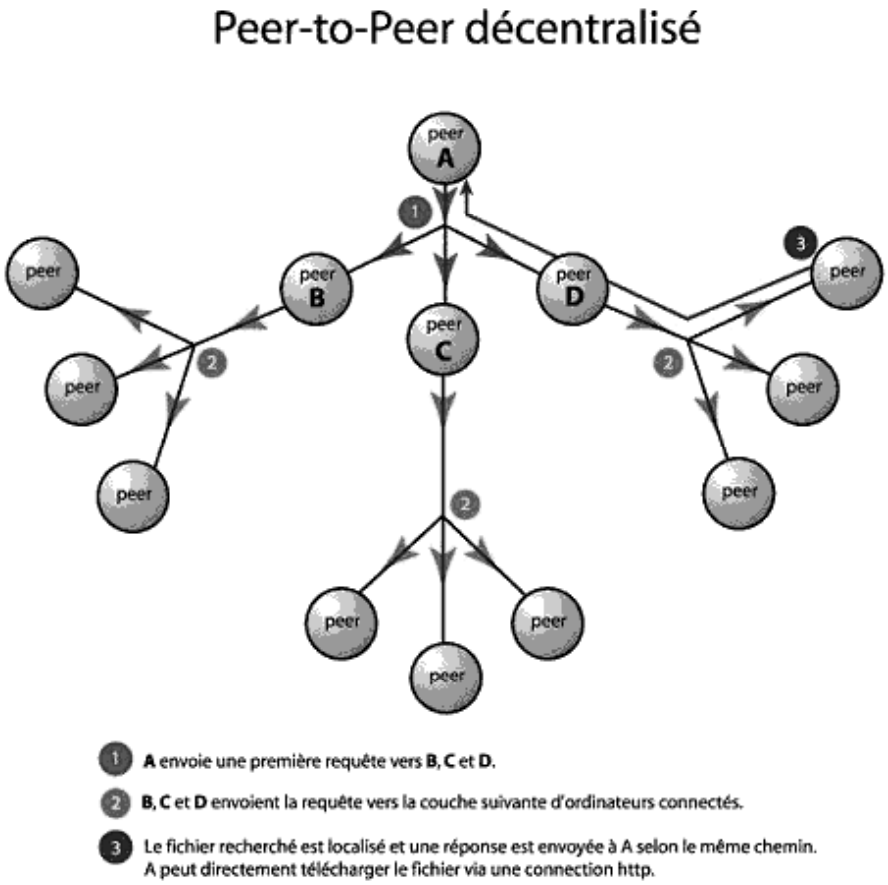

FIG. *3.6: Système P2P décentralisé.*

une requête de recherche à tous ou seulement à un certain nombre de pairs.

Exemple d'application : Gnutella (voir figure 3.7)

Nous illustrons la stratégie de localisation de ressources par un exemple : nous supposons que A cherche à obtenir le fichier f, disponible chez le nœud D. Le nœud A va donc envoyer un message de requête $Query$ au(x) premier(s) nœud(s) qu'il connaît (dans l'exemple 3.7, c'est le nœud B), qui se charge de :

A. répondre s'il dispose de la ressource (ce qui n'est pas le cas).

B. propager la requête aux nœuds qu'ils connait (dans cet exemple C).

Les autres nœuds font de même, ce qui conduit D à répondre $QueryHit$ (réponse à Query contenant des informations sur les données trouvées). Le nœud A peut alors envoyer à D le message *get* f, et le transfert direct peut alors commencer entre les deux nœuds.

iii. **Système hiérarchique (Superpair)**

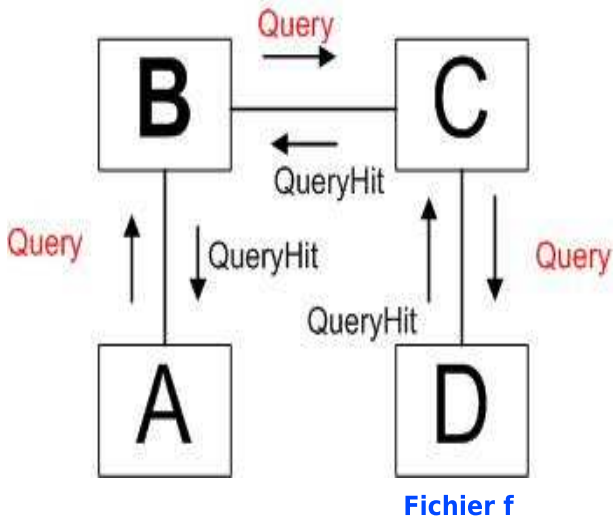

Fig. *3.7: La recherche dans Gnutella*

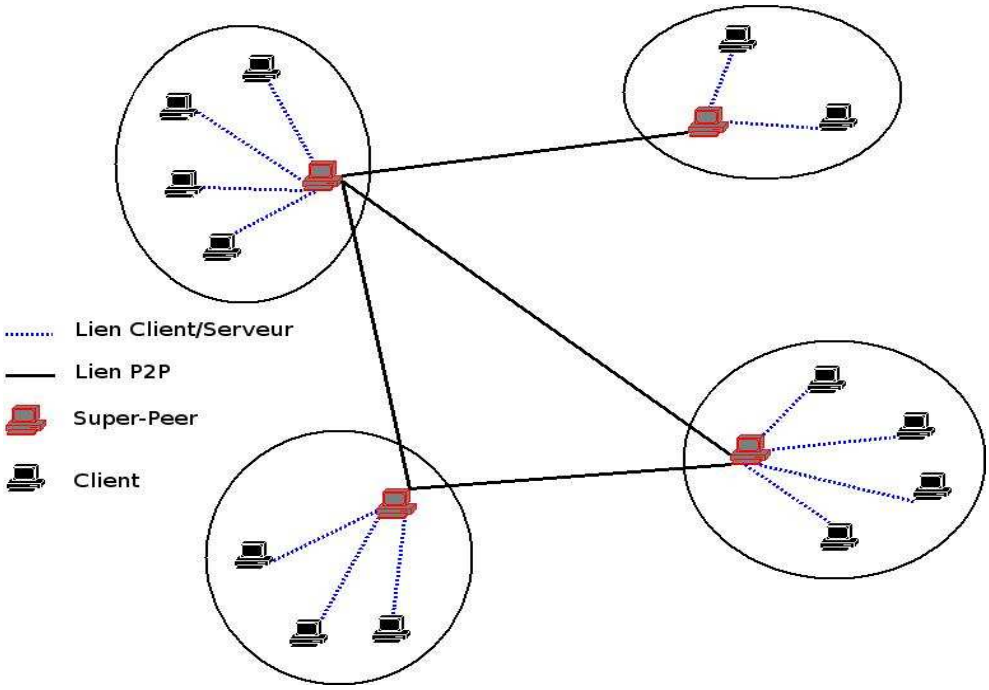

Fig. *3.8: Système P2P hiérarchique*

Le modèle superpair introduit une hiérarchie entre des pairs appelés superpairs (ou nœuds puissants) et des pairs ordinaires (ou nœuds faibles) connectés à un superpair. Les superpairs fonctionnent en mode P2P, alors qu'à l'intérieur d'un groupe, il s'agit d'un mode client-serveur classique. Le modèle hiérarchique a

pour but d'utiliser les avantages des deux types de systèmes (centralisé et décentralisé). Nous avons alors un réseau de superpairs où chacun joue le rôle d'un serveur d'indexation de ses pairs. Cette topologie permet de diminuer le nombre de connnexions sur chaque serveur d'indexation, et ainsi d'éviter les limitations de bande passante. Un mécanisme issu des réseaux décentralisés est utilisé pour tenir à jour, dans chaque superpair, un index des ressources à partir des informations provenant des pairs du groupe ainsi que des autres superpairs. Un superpair agit comme un répertoire centralisé pour le compte d'un ensemble fini de pairs. Le routage dans les réseaux de superpairs est plus efficace que dans les réseaux purement P2P puisque le routage se limite aux superpairs dans ces réseaux. Cette solution permet de résoudre le problème d'extensibilité de l'approche purement distribuée tout en gardant l'efficacité de la solution centralisée. Cependant, de tels systèmes supposent l'existence de pairs pouvant jouer le rôle de superpairs. De tels pairs existent mais les techniques mises en œuvre afin de les désigner sont complexes. En effet, élire un pair comme superpair nécessite de disposer de nombreux paramètres sur ses capacités propres mais aussi et surtout sur les capacités, notamment en terme de bande passante, du réseau qui le connecte. Des règles pour améliorer la conception d'un superpair sont définies dans (Yang et Garcia-Molina, 2003).

Des exemples d'application sont : Kazaa [11] et Gnutella2 [12].
L'approche non-structurée, bien que moins efficace sur le plan du routage des requêtes, offre l'avantage de respecter l'autonomie des pairs et de pouvoir supporter des langages de requêtes plus expressifs. C'est pour cela que nous nous intéressons aux systèmes non-structurés puisque nous privilégions l'autonomie des pairs.

2. ***Type de la recherche***
Les méthodes de recherche dans les systèmes P2P non-structurés peuvent être classées en deux catégories : aveugle ou informée (Tsoumakos et Roussopoulos, 2003b).

   (a) ***La recherche aveugle*** : dans ce type de recherche, les nœuds ne disposent pas d'information concernant la localisation des ressources dans le réseau. La recherche aveugle se fait par un parcours partiel de l'espace de recherche. En effet, quand une requête est soumise à un nœud, elle est résolue sur celui-ci et propagée récursivement sur un ensemble de nœuds voisins. Le processus de propagation s'arrête lorsqu'un certain seuil est atteint (généralement un nombre fixé de rebonds). Ce seuil n'est pas forcément facile à déterminer si nous n'avons pas d'information sur la topologie du réseau et/ou si nous souhaitons la complétude de la recherche. Ces systèmes

[11] http ://www.kazaa.com/us/index.htm
[12] http ://www.gnutella2.com/

fonctionnent assez bien lorsque les ressources partagées sont répliquées sur de nombreux nœuds et donc sont facilement trouvées. En revanche, plusieurs études ont montré que les performances de ces systèmes sont très faibles et que le nombre de messages échangés est très important (Tsoumakos et Roussopoulos, 2003b). Parmi les approches se basant sur la recherche aveugle, nous citons :

– Gnutella : l'algorithme original de Gnutella utilise la propagation de recherche en largeur d'abord (BFS : Breadth-first search) pour la découverte d'objets et contacte tous les nœuds accessibles dans les limites de la valeur de la durée de vie de la requête (TTL : Time-To-Live). Bien que cette approche soit simple et tende à découvrir un nombre maximum d'objets, l'approche ne passe pas à l'échelle, produisant une grande charge à un nombre important de pairs (Tsoumakos et Roussopoulos, 2003b).
– Recherche en largeur modifiée (BFS modifié) (Kalogeraki *et al.*, 2002) : c'est une variation du schéma de propagation où les pairs choisissent uniquement un sous-ensemble de leurs voisins pour leur envoyer la requête. Cet algorithme réduit la moyenne des messages échangés par rapport à la méthode précédente mais contacte toujours un grand nombre de pairs.
– Marche aléatoire (Random Walks (Lv *et al.*, 2002)) : dans cet algorithme, le pair initiateur envoie la requête à un nombre k de voisins choisis au hasard. Chacun de ces messages suit son propre chemin, passant par des nœuds intermédiaires qui le font suivre à un voisin choisi au hasard. Ainsi la requête effectue une marche aléatoire dans le réseau des nœuds. Chaque requête a une durée de vie exprimée en nombre maximum de nœuds à visiter. Une requête termine avec un succès ou un échec. L'échec peut être déterminé par deux méthodes différentes : méthode à base de la durée de vie et méthode de vérification où les requêtes contactent périodiquement la source de la requête pour voir si les conditions de terminaison sont satisfaites. L'avantage majeur de l'algorithme est la réduction significative des messages qu'il réalise. Dans le pire des cas, l'algorithme produit $k \times TTL$ messages. Cependant, l'inconvénient essentiel est sa grande variation de performance. Le taux de succès varie énormément selon la topologie du réseau et les choix aléatoires effectués. De plus, rien n'est appris à partir des succès et des échecs précédents des requêtes.

D'autres protocoles de recherche agissant dans les réseaux P2P hybrides sont apparus. Par exemple, Guess et Gnutella2 qui sont des améliorations du protocole de Gnutella :

– GUESS (Gnutella UDP Extension for Scalable Searches) (Daswani et Fisk, 2002) : ce protocole se base sur la notion de superpairs. Une recherche est guidée par le contact itératif de différents superpairs qui interrogent tous leurs nœuds feuilles

jusqu'à ce qu'un nombre d'objets soit trouvé. L'ordre de choix des superpairs est non spécifié. La solution proposée par GUESS est de laisser à l'initiateur de la recherche le soin d'interroger successivement un ensemble de superpairs jusqu'à ce qu'il obtienne un nombre de résultats satisfaisants.

– Gnutella2 (Stokes, 2002) : se base sur le principe suivant : quand un superpair reçoit une requête d'une feuille, il la propage à ses feuilles pertinentes et à ses superpairs voisins. Ces superpairs traitent la requête localement et la propagent à leurs feuilles pertinentes. Les superpairs voisins échangent régulièrement leurs tables locales pour éliminer le trafic non nécessaire entre eux. Ces deux derniers algorithmes réduisent le flux en utilisant une structure hybride du réseau.

Tous ces protocoles de base et leurs améliorations tendent à réduire la charge du réseau en diminuant le nombre de messages échangés. Ce flux peut être réduit avec l'utilisation d'une structure hybride de type superpair. Cependant, cette catégorie d'algorithmes de recherche reste très pauvre dans le sens où aucune sémantique n'est prise en compte lors du routage des requêtes.

(b) ***La recherche informée*** : la recherche se base sur l'utilisation d'informations concernant la localisation des ressources qui peut être centralisée (c'est-à-dire qu'il existe un répertoire connu par tous les nœuds à l'instar de Napster) ou décentralisée (chaque nœud détient une partie de l'information, (Kalogeraki *et al.*, 2002), (Crespo et Garcia-Molina, 2002a)). Le principe consiste à ajouter de la connaissance au contenu des nœuds pour permettre une propagation de requêtes non plus à l'aveugle, mais guidée par cette connaissance. Parmi les algorithmes de recherche existants, nous citons les suivants :

– Recherche en largeur intelligente (Intelligent-BFS (Kalogeraki *et al.*, 2002)) : c'est une version informée de l'algorithme BFS modifié. Dans ce cas, les nœuds stockent les couples (requête, IdVoisin) des requêtes récemment traitées par ou à travers leurs voisins. Un pair identifie toutes les requêtes similaires à la requête courante selon une métrique de similarité. Ensuite, il choisit de faire suivre la requête à un ensemble de voisins ayant retourné le plus de résultats pour ces requêtes. Si le succès se produit, la requête prend le chemin inverse pour informer le pair initiateur et les index locaux sont mis à jour. Cette méthode est plutôt orientée découverte d'objets que réduction de messages.

– Recherche probabiliste adaptative (APS : Adaptive Probabilistic Search) (Tsoumakos et Roussopoulos, 2003a) : chaque nœud garde un index local qui consiste en une entrée pour chaque objet qu'il a demandé et par voisin. La valeur de cette entrée reflète la probabilité de choisir ce voisin au prochain saut pour une requête future de l'objet spécifique. La recherche est basée sur le déploiement de k marcheurs indépendants et l'acheminement probabilistique. Chaque nœud in-

termédiaire envoie la requête à un de ses voisins avec une probabilité donnée par son index local. Les valeurs des index sont mises à jour en utilisant les retours des marcheurs. Si un marcheur réussit (respectivement échoue), les probabilités relatives des nœuds sur le chemin du marcheur sont augmentées (respectivement diminuées). Chaque nœud sur les marcheurs déployés met à jour son index selon les résultats de recherche, ainsi les pairs partagent, raffinent et ajustent leurs connaissances de recherche avec le temps.

– Indices locaux (Local indices (Yang et Garcia-Molina, 2002)) : selon cette approche, chaque nœud indexe les fichiers stockés chez tous les nœuds dans un rayon r fixé et peut répondre aux requêtes au nom d'eux tous. La recherche est effectuée selon l'algorithme BFS mais seulement les nœuds accessibles à partir du pair initiateur, à certaines profondeurs, traitent la requête. Cette approche rassemble les deux schémas de recherche des réseaux hybrides. D'une part, la précision de la méthode est élevée puique chaque nœud contacté indexe plusieurs pairs. D'autre part, la production de messages est comparable au schéma d'inondation bien que le temps de traitement soit beaucoup plus réduit puisque la requête n'est pas traitée par tous les nœuds.
– Protocole de localisation de ressource distribuée (DRLP : Distributed Resource Location Protocol) (Menascé et Kanchanapalli, 2002) : selon cette approche, les nœuds ne disposant pas d'information sur la localisation d'un fichier, envoient la requête à un sous-ensemble de leurs voisins. Si un objet est trouvé, la requête prend le chemin inverse vers le pair initiateur en stockant la localisation du document chez ces nœuds. Pour les requêtes ultérieures, les nœuds avec l'information de localisation indexée contactent directement le nœud spécifié. Si ce nœud n'obtient pas le document, alors il initie une nouvelle recherche. Cet algorithme dépense initialement beaucoup de messages pour trouver la localisation des objets. Pour des requêtes ultérieures, il peut utiliser seulement un seul message pour découvrir le document. Une petite production de messages est réalisée seulement avec une large charge de travail qui permettra au coût initial d'être amorti sur plusieurs recherches. Dans les réseaux qui évoluent rapidement, cette approche échoue et beaucoup de nœuds doivent faire une recherche aveugle. Ce schéma est très dépendant des paramètres de l'application et/ou du réseau.

Le choix d'un pair dans un routage aveugle se fait complètement au hasard tandis qu'un routage informé ne fait le choix que des pairs pouvant bien répondre à la requête. Ce dernier routage exploite plus d'information et de connaissances sur les pairs afin d'assurer un routage « intelligent ». Il permet également de réduire le nombre de messages envoyés pour une requête donnée. C'est à ce type de routage « sémantique » que nous nous intéressons.

3. ***Implication de l'utilisateur*** : peut concerner trois niveaux :
   (a) Expression des besoins en terme de sujet de la requête. Est-ce que la détermination des sujets de la requête est explicite ou implicite ? Autrement dit, l'utilisateur exprime-t-il directement et explicitement ses besoins en lançant une requête par rapport à une thématique donnée, ou la requête est-elle lancée automatiquement suite à une détection implicite du centre d'intérêt de l'utilisateur ?
   (b) Routage sémantique de la requête dans le réseau : l'utilisateur joue-t-il un rôle ou non dans le routage de sa requête à travers le réseau ? Autrement dit, l'utilisateur intervient-il pour aider au bon acheminement de sa requête comme par exemple le choix de son comité ?
   (c) Retour : l'approche considère-t-elle le retour (feedback) de l'utilisateur une fois les résultats de recherche sont présentés ou non ? Et s'il est pris en compte, comment se fait-il ? Autrement dit, l'utilisateur évalue-t-il explicitement le résultat qui lui a été proposé (par exemple, en donnant une note) ou cela peut-il être déduit implicitement à travers l'observation de ses réactions sur le résultat (par exemple, un document est jugé très intéressant si l'utilisateur l'ajoute dans sa base locale) ?
4. ***Utilisation d'ontologie*** : l'approche repose-t-elle ou non sur l'emploi d'une ou plusieurs ontologies ? Quelles sont les caractéristiques des ontologies utilisées ?
   Parmi ces caractéristiques, nous trouvons :
   (a) Le niveau d'ontologie : indépendamment de leurs formalismes, différents niveaux d'ontologies sont distingués selon le domaine modélisé et éventuellement les tâches pour lesquelles elles sont conçues (Chandrasekaran *et al.*, 1999), (Burgun et Bodenreider, 2001), nous trouvons :
      – *Les ontologies d'application* ont un domaine de validité restreint et correspondent à l'exécution d'une tâche.
      – *Les ontologies de domaine* ont un faisceau plus large, une bonne précision et ne sont pas propres à une tâche particulière.
      – *Les ontologies générales* ne sont pas propres à un domaine. Leur précision est moyenne.
      – *Les ontologies supérieures* ou de haut niveau représentent des concepts généraux comme l'espace, le temps ou la matière.
   (b) L'ontologie peut être commune ou individuelle : dans le premier cas, tous les pairs partagent une même ontologie ce qui facilite leur collaboration puisqu'ils utilisent le « même langage ». Dans le second cas, chaque pair dispose de sa propre ontologie et donc d'autres outils de médiation seront nécessaires pour faciliter la mise en correspondance sémantique entre les différentes ontologies.

(c) L'ontologie peut être fixe ou dynamique : est ce que l'ontologie utilisée est figée ou au contraire peut-elle évoluer au cours du temps ? Dans le deuxième cas, comment le système peut-il s'adapter suite aux changements de l'ontologie ?

(d) L'ontologie peut être simple ou composée : l'ontologie utilisée est-elle dans sa forme la plus simple (hiérarchie) ou plus riche en comportant diverses relations sémantiques entre ses concepts ?

5. ***Capacité d'apprentissage*** : l'approche inclut-elle de l'apprentissage automatique ? Nous désignons par apprentissage, l'acquisition de nouvelles connaissances suite aux expériences passées d'interaction entre les pairs. Le but est d'améliorer le routage de la requête au cours du temps et d'apprendre à mieux former son comité. Quels sont les moyens utilisés pour acquérir de nouvelles connaissances : retour de l'utilisateur, historique d'interaction ?

6. ***Nature du comité*** : qui désigne la spécificité du comité de pairs considéré dans le système P2P. Est-ce qu'il s'agit de comité structuré, de taille limitée et dont les membres se connaissent (par exemple, une équipe de recherche) où s'agit-il de comité de taille illimitée dont les membres changent souvent ?

3.3.2 Critères d'évaluation des approches

Deux types de critères sont considérés : critères qualitatifs et critères quantitatifs :

1. **Critères qualitatifs** : ces critères évaluent la qualité de l'approche en considérant sa capacité à évoluer avec les changements et son applicabilité.
   - ***Prise en compte de l'aspect dynamique du système*** : l'approche proposée tient-elle compte ou non des problèmes liés à la dynamique des réseaux et aux évolutions du système ? Cette dynamique peut concerner plusieurs aspects, à savoir :
     (a) Les connexions/déconnexions des pairs.
     (b) Les centres d'intérêt d'un utilisateur : les centres d'intérêt de l'utilisateur sont-ils les mêmes ou ses intérêts évoluent-ils au cours du temps ? Comment tenir compte de ces changements pour mettre à jour son ou ses comités ?

     Quelles sont les stratégies utilisées pour faire face à tous ces changements ?
   - **Échelle** : l'approche considérée se place-t-elle dans un contexte de grande échelle (de l'ordre de dizaines de milliers de nœuds) ou est-elle plutôt limitée ?

2. **Critères quantitatifs** : nous considérons le rappel, la précision, le nombre de messages envoyés et la taille relative du comité de pairs contactés, qui sont définis comme suit :
   - ***Le rappel*** : qui présente le taux de bons pairs (pairs proposant un bon résultat) contactés par rapport à tous les bons pairs qui existent dans le système.

$$Rappel \quad = \quad \frac{(Bons\_pairs\_contactés)}{(Tous\_les\_bons\_pairs)} \tag{3.1}$$

– ***La précision*** : qui présente le taux de bons pairs contactés par rapport à tous les pairs contactés.

$$Précision \quad = \quad \frac{(Bons\_pairs\_contactés)}{(Tous\_les\_pairs\_contactés)} \tag{3.2}$$

– ***Le nombre de messages*** : qui sont échangés dans le réseau pour une requête d'utilisateur donnée.

– ***La taille relative du comités*** : présente le rapport du nombre de pairs contactés sur le nombre de tous les pairs existants dans le système.

$$Taille_{relative}\_Comité \quad = \quad \frac{(Nb\_pairs\_contactés)}{(Tous\_les\_pairs\_existants)} \tag{3.3}$$

3.4 Description de quelques systèmes existants

Nous décrivons dans ce qui suit quelques exemples de systèmes P2P non-structurés pour le routage sémantique de requêtes.

1. **SON (Semantic Overlay Networks) (Crespo et Garcia-Molina, 2002b)**

   Il s'agit d'une application de partage de musique. La problématique consiste à connecter des pairs en se basant sur leur contenu pour que les requêtes soient routées aux SONs appropriés afin d'accélérer la recherche et de réduire la charge de recherche des pairs ayant des contenus non reliés. La solution proposée repose sur un modèle d'architecture superpair où les nœuds proches sémantiquement sont regroupés en clusters (un cluster par concept, par exemple, Rock, Jazz, etc.). Le système utilise une hiérachie commune de thèmes dans le domaine de la musique où les différents nœuds présentent des styles musicaux (par exemple, Rock, Dance, Pop, etc.) ou des décennies (par exemple, les années 90, les années 70, etc.). Un pair peut faire partie d'un ou de plusieurs SONs tels que les pairs E et F de la figure 3.9. Par exemple, tous les nœuds possédant des chansons rock sont regroupés en établissant des connexions entre eux. L'exemple présenté dans la figure 3.9 montre 8 nœuds classés par genre de musique. Nous avons quatre SONs, par exemple, le SON Rock est formé par les nœuds A, B et C.

   Le principe consiste à déterminer, pour une requête donnée, le cluster qui puisse contenir des documents correspondants. Un pair décide quel(s) SON(s) joindre en se basant sur la classification de ses documents. Une requête peut être aussi simple que l'identificateur d'un document, des mots-clés ou une requête SQL plus complexe.

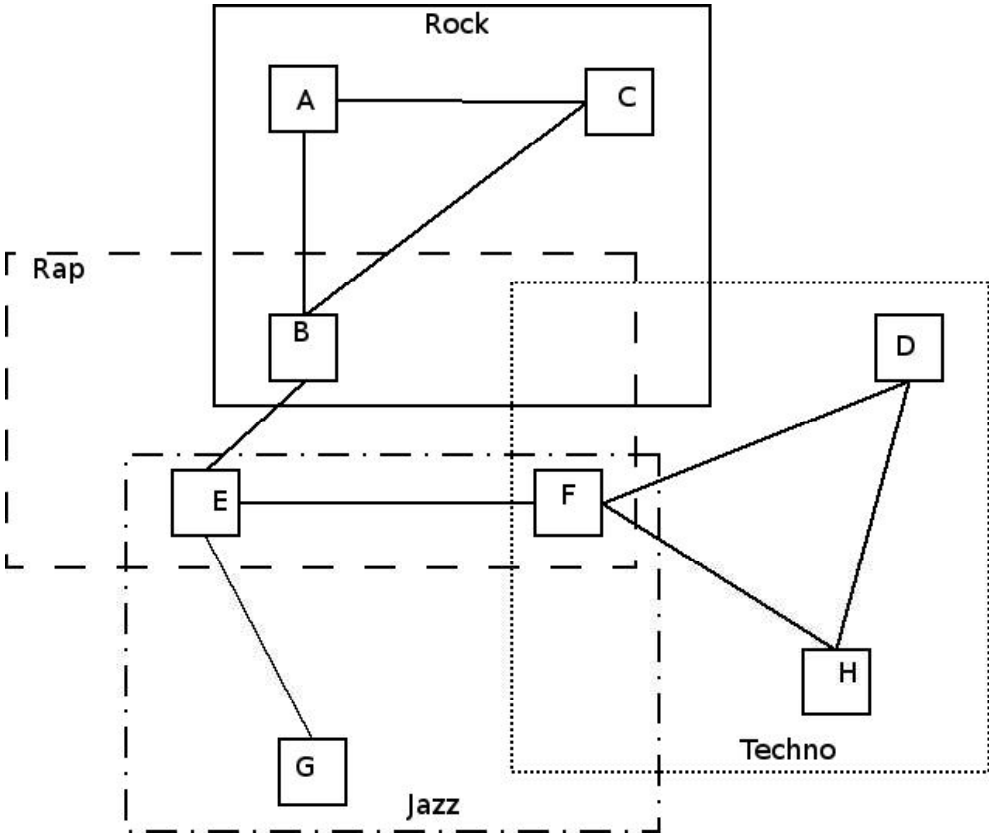

Fig. *3.9: Exemples de SONs*

Par exemple, il joint (SON_c) si un nombre significatif de documents sont classifiés sous le concept c. Une fois la requête classifiée, elle est envoyée au(x) SON(s) associés au(x) concept(s) de base de la classification de la requête. Finalement, la requête est progressivement envoyée dans la hiérarchie jusqu'à ce que suffisament de résultats soient trouvés. Si la classification de la requête donne plus d'un concept, alors une recherche séquentielle se fait tout d'abord dans tous les concepts trouvés avant d'aller explorer la hiérarchie. L'expérimentation a été réalisée avec 1.800 pairs Napster et cette stratégie a amélioré les performances de requête en terme de nombre moyen de messages pour une requête. Cependant, cette approche présente des problèmes concernant surtout le choix d'une hiérarchie de classification, la classification de requêtes et de documents (le nombre de requêtes est important mais elles sont courtes alors que la classification des documents est rare mais longue) et le choix de la granularité dans la hiérarchie des clusters.

Les requêtes sont obtenues à partir des traces de requêtes passées et sont classifiées manuellement par les auteurs eux-mêmes (Yang et Garcia-Molina, 2001), sous un ou plusieurs sous-styles de musique, un seul style ou sous le thème « musique » qui présente

Chemin	Nombre de documents	BDD	Langages	Réseaux	Théorie
B	100	20	30	0	10
C	1000	0	50	300	0
D	200	100	150	0	100

TAB. *3.1: Exemple d'indices de routage pour le nœud A*

la racine de la hiérarchie. Les documents sont classifiés en explorant la base de données du guide de « All Music » du site allmusic.com. C'est une base de données maintenue par des volontaires qui classifient manuellement les chansons et les artistes. Dans cette base de données, les chansons et les artistes sont classifiés en utilisant une hiérarchie de concepts : styles et sous-styles. Dans chaque nœud, il y a un ensemble de fichiers de format « répertoire/auteur-titreChanson.mp3 ». Le classifieur de documents extrait l'auteur et le titre de la chanson à partir du fichier. Ensuite, le classifieur explore la base de données avec cet auteur et ce titre de la chanson et obtient une liste de chansons correspondantes. Finalement, le classifieur sélectionne les chansons les mieux classées et trouve son style et ses sous-styles. S'il n'y a pas de correspondance dans la base de données, le classifieur attribue « non connu » au style et sous-style du fichier.

2. **Routing indices (Crespo et Garcia-Molina, 2002a)**

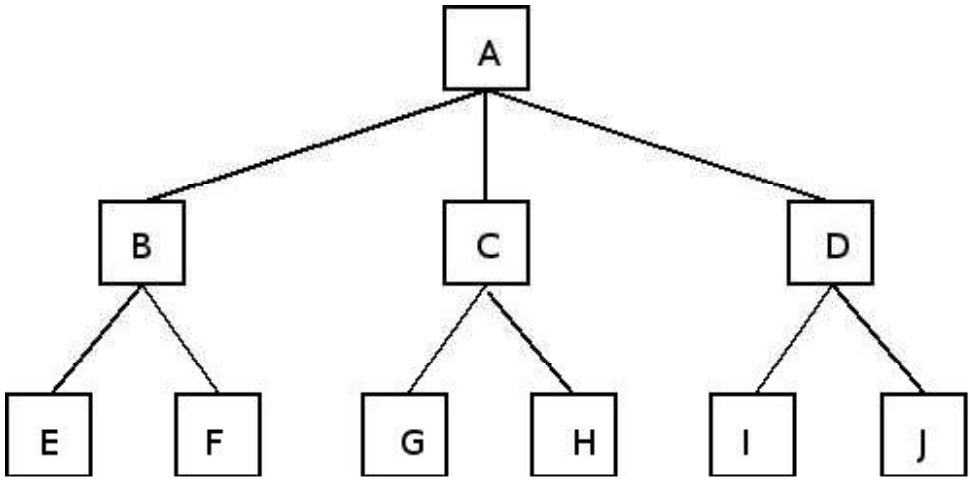

FIG. *3.10: Exemple de Routing Indices*

L'approche se base sur un modèle d'architecture décentralisée. Les pairs partagent un ensemble de thèmes pour classifier leurs documents. Chaque document peut avoir zéro ou plusieurs thèmes. Le principe repose sur la notion d'indices de routage. Un indice

de routage (voir figure 3.10) est une structure de données (et des algorithmes associés) attachée à chaque nœud qui, pour une requête donnée, retourne un ensemble de voisins triés selon leur qualité pour la requête. Cette qualité peut varier mais reflète en général le nombre de documents à proximité (en considérant le chemin entre les nœuds) des nœuds. Cet indice de routage contient un résumé de l'index local (matrice contenant différents thèmes et le nombre de documents associés) et la même chose pour les autres nœuds mais cette fois-ci en considérant le nombre de documents qui peuvent être accessibles à travers les nœuds visibles considérés. Comme le montre le tableau 3.1, le nœud B peut accéder à 100 documents, dont 20 traitent les BDD, 30 les *Langages* et 10 la *Théorie*. Chaque nœud a sa propre base de documents pouvant être accédée à travers son indice local. Ils proposent trois mécanismes de recherche pouvant être classifiés en trois catégories : mécanismes sans indice, mécanismes avec des nœuds d'indice spécialisé (recherche centralisée) et mécanismes avec des indices sur chaque nœud (recherche distribuée). Les utilisateurs soumettent leurs requêtes (des conjonctions de thèmes) avec une condition d'arrêt (par exemple, le nombre de résultats). L'objectif du mécanisme de recherche est de répondre aux requêtes en trouvant un ensemble de documents de taille fixée dans la condition d'arrêt qui correspondent à la requête. L'algorithme de recherche consiste à résoudre une requête Q localement. Tant qu'il n'y a pas assez de résultats, il faut évaluer la proximité des successeurs en se basant sur le composant d'indices de routage, prendre le successeur S le plus proche et non encore exploré puis faire la recherche (Q, S). Cette approche s'inscrit dans un contexte large échelle (60.000 nœuds pour la simulation sur une base de documents). Elle réduit le nombre de messages par rapport à Gnutella mais rencontre certains problèmes. En effet, l'exploration est restreinte aux nœuds ayant la plus grande probabilité de succès. Il n'y a pas d'information sur le nombre de sauts nécessaires et il n'y a pas de garantie d'avoir tous les résultats. De plus, les coûts de la mise à jour des indices de routage sont élevés. L'approche est plutôt orientée recherche des k meilleurs résultats.

3. **Padoue** [13] (Doucet et Lumineau, 2003) : est un système pour le partage d'informations hétérogènes dans des environnements distribués en exploitant les expériences des communautés scientifiques. L'application se base sur une architecture de type superpair et s'inscrit dans le cadre d'un contexte de grande échelle. Son principe consiste à regrouper un ensemble d'utilisateurs sur un même nœud en fonction de leurs centres d'intérêt. Le but de l'application est de faciliter la découverte de nœuds contenant des ressources pertinentes et de réduire le nombre de messages utilisés durant la propagation de la requête aussi bien que le temps de réponse. Dans le système Padoue, c'est l'utilisateur qui soumet sa requête et choisit son comité (appelé communauté). L'approche adoptée

[13] http ://www-poleia.lip6.fr/padoue/

est inspirée des travaux sur le filtrage collaboratif : la différence est qu'en général, le filtrage se fait sur les utilisateurs alors que l'approche proposée effectue plutôt un filtrage sur les comités. Les nœuds sont libellés pour décrire le domaine des données qu'ils stockent. Chaque comité est défini par des mots-clés. Les thèmes (chaque thème est un mot-clé représentant un domaine scientifique bien précis) sont établis par des experts humains et présentent une ressource globale partagée par tous les pairs et connus par tous les utilisateurs du réseau. L'approche utilise une hiérarchie commune pour classifier les nœuds. Le routage de requête se base sur des liens sémantiques intra et inter-comités. Les liens intra-comités sont des liens basés sur la pertinence qui matérialisent la connaissance communautaire en pointant sur les nœuds jugés les plus pertinents par les membres d'un comité. Les liens inter-comités relient deux comités de centres d'intérêt proches définis sur deux nœuds voisins. Un comité regroupe un ensemble d'utilisateurs en fonction de leurs centres d'intérêt associés à un même nœud (par exemple, utilisateurs appartenant à une même organisation, un même département, etc.). Il est défini par un ensemble restreint de thèmes caractérisant un centre d'intérêt (par exemple, océanographie, hydrologie, etc.). Le processus de propagation d'une requête Q, initiée par un membre du comité C, définie sur le nœud N, se fait donc en deux temps. Dans un premier temps, la requête Q est transmise aux voisins pointés par les liens basés sur la pertinence sur lesquels Q sera exécutée de manière à extraire les méta-données. De plus, la requête Q est transmise aux nœuds voisins contenant des comités pointés par les liens inter-comités pour être dans un second temps transmise aux nœuds pointés par les liens basés sur la pertinence définie par les comités distants. L'utilisateur commence par choisir son comité à travers une liste de comités disponibles (exemples de comités : hydrogeologists, climatologists, ecologists). S'il n'y a pas de comité pertinent, l'utilisateur peut créer un nouveau comité. Ensuite, l'utilisateur envoie sa requête dans le réseau P2P et le système lui retourne des méta-données potentiellement pertinentes pour la requête. Les méta-données concernent des ressources, ces méta-données peuvent être le titre, la date, l'auteur, la localisation, etc.. Une requête peut être un ensemble de mots-clés. Cette approche suppose que les ressources stockées sur un même nœud sont reliées aux mêmes thèmes. Ainsi, un nœud est jugé pertinent si des ressources pertinentes (méta-données) sont trouvées dans ce nœud. Finalement, l'utilisateur indique son évaluation des méta-données proposées (1 si elles sont pertinentes, -1 sinon) et sélectionne les ressources pertinentes à charger. Le routage de la requête se fait comme suit : une fois que les nœuds associés à une aggrégation de retours positifs sont déterminés, la requête sera envoyée aux nœuds susceptibles d'avoir des ressources pertinentes. De plus, la requête est envoyée aux comités trouvés à l'aide des liens inter-comités stockés sur les nœuds trouvés. La requête ne sera pas exécutée sur le nœud du comité trouvé mais plutôt sur les nœuds qu'il considère pertinents. Ainsi la requête est propagée directement aux nœuds pertinents afin de trouver les ressources

pertinentes. L'évolution du comité est prise en compte en se basant sur les retours de l'utilisateur qui présente une connaissance dynamique. L'expérience de comité (liste de nœuds évalués et leur retour aggrégé) sont utilisés pour calculer les corrélations entre les comités. Cette approche se base sur l'introduction de connaissances dans le processus de propagation des requêtes. Le type de connaissance considéré est l'appartenance d'un utilisateur à un comité et sur les liens qui peuvent être établis entre les comités reliés. Cette approche sollicite l'utilisateur tout au long du processus de routage de la requête. En effet, c'est l'utilisateur qui soumet sa requête explicitement, choisit son comité et effectue des retours explicites sur les méta-données proposées. Cela permet de mieux guider le routage des requêtes mais peut aussi réduire l'utilité de cette approche puisque l'implication de l'utilisateur est importante.

4. **Bibster** (Broekstra *et al.*, 2004) : est une application visant à partager des données bibliographiques. Le modèle d'architecture utilisé est P2P décentralisé. Les données sont obtenues à partir des fichiers BibTex ou des serveurs bibliographiques tels que DBLP [14] ou Citeseer[15]. Le système se base sur l'utilisation de deux ontologies communes pour l'importation des données, la formulation de requêtes, le routage des requêtes et le traitement des résultats : $SWRC$ et ACM. L'ontologie $SWRC$ (Semantic Web Research Community) est utilisée pour décrire les aspects génériques des méta-données bibliographiques (titre, auteur, etc.). L'ontologie ACM (Association for Computing Machinery) est utilisée pour décrire les catégories spécifiques du domaine (thèmes).
   Les requêtes sont formulées soit en terme d'attributs (par exemple, auteur, type de document, etc.) selon l'ontologie SWRC, soit en terme de thèmes spécifiques (de la hiérarchie ACM). Le routage de requêtes se base sur le principe de la publication des expertises des pairs dans le réseau. L'expertise d'un pair est l'ensemble de thèmes ACM pour lesquels le pair est expert. Ces modèles d'expertise des pairs décrivent pour quels thèmes de l'ontologie un pair peut répondre aux requêtes. Le routage se fait ainsi selon ce classement sémantique. Les pairs commencent par envoyer des publications de leur expertise à tous les pairs qu'ils connaissent. Lorsqu'un pair reçoit une requête, il décide alors de la diffuser aux pairs dont l'expertise est similaire au sujet de la requête. Les pairs partagent une ontologie commune pour publier des descriptions sémantiques de leurs expertises dans le réseau. Cette connaissance sur les expertises des autres pairs forme la topologie sémantique qui est indépendante de la topologie du réseau. La publication des expertises est décidée et faite par les pairs en se basant sur une similarité sémantique entre les descriptions des expertises. Les pairs décident d'une façon autonome, à qui transmettre les publications et lesquelles accepter. Cette stratégie donne de meilleurs résultats par rapport à la diffusion de la requête à tous les pairs disponibles ou à un ensemble de pairs

[14] http ://dblp.uni-trier.de/
[15] http ://citeseer.ist.psu.edu/

pris au hasard mais n'exploite pas les expériences passées pour apprendre et faire évoluer la topologie sémantique produite.

5. **Semantic Publish/Subscribe System** (Chirita *et al.*, 2006)
   Chirita et al. utilisent le système de publication/souscription P2P-DIET [16] et l'infrastructure superpair Edutella (Nejdl *et al.*, 2003) pour présenter un système de publication/souscription sémantique s'appuyant sur des méta-données et un langage de requête basé sur le modèle de description de ressources RDF [17]. C'est une alternative aux systèmes à base de requêtes où la même information est demandée de nouveau et où les clients souhaitent avoir des réponses mises à jour pendant une période de temps. Le modèle d'architecture utilisé est de type superpair. L'approche se base essentiellement sur trois compétences :
   - ***la publication*** qui présente une annonce du contenu que les pairs veulent publier,
   - ***la souscription*** des pairs chez les superpairs se fait en définissant les types de documents qu'ils veulent obtenir,
   - ***la notification*** des pairs dès que de nouvelles ressources deviennent disponibles.

   Les superpairs sont responsables des notifications, souscriptions et publications.
   Dans ce système, une requête est une conjonction de variables avec des contraintes selon le langage de requête RDF. Par exemple, la requête X telle que :

   `X : t(X, <rdf :type>, <dc :article>)` $\wedge$ `t(X, <dc :creator>, Y)` $\wedge$
   `t(X, <dc :date>, D)` $\wedge$ `(Y` $\sqsupseteq$ « `Nejdl` » $\vee$ `Y` $\sqsupseteq$ « `Koubarakis` ») $\wedge$ `D=2004)`

   exprime la souscription suivante : « je suis intéressé par les articles écrits par Nejdl ou Koubarakis en 2004 ».
   Une publication est un couple (T, I) où T est une description de la ressource en RDF et I est l'identificateur du pair (C3 dans l'exemple suivant). Par exemple, la publication suivante correspond à la souscription décrite précédemment :

   ```
   ({ t(<is1 :esws04.pdf>, <rdf :type>, <dc :article>),
   t(<is1 :esws04.pdf>, <dc :creator>, « Koubarakis »),
      t(<is1 :esws04.pdf>, <dc :date>, 2004) }, C3)
   ```

   Chirita et al. proposent un mécanisme pour le cas où quelques pairs ne sont pas en ligne quand de nouvelles ressources, qui les intéressent, deviennent disponibles. Ce mécanisme se base sur la notification hors ligne et la prise de rendez-vous chez le superpair. Le principe est le suivant : quand un pair se déconnecte, son point d'accès garde l'information

[16] http ://www.intelligence.tuc.gr/p2pdiet/
[17] Resource Description Framework

d'identification du pair et ses souscriptions pour une période de temps spécifiée. Ce qui ne reflète pas que le pair a quitté le réseau. Ainsi les notifications peuvent toujours arriver au point d'accès qui enregistre ces notifications et les délivre au pair, une fois qu'il se reconnecte. Dans un autre cas, le propriétaire d'une ressource est déconnecté alors qu'un autre agent la demande, supposons que c'est l'agent A. L'agent A va s'adresser à l'agent B directement à partir du point d'accès de B qui est inclus dans la ressource recherchée. Dans ce cas, l'agent A peut demander un rendez-vous avec la ressource r du point d'accès de B. Quand B se reconnecte, son point d'accès l'informe qu'il doit télécharger la ressource r pour l'agent A dans son point d'accès. Si l'agent A est connecté, alors son point d'accès lui envoie la ressource r. Dans le cas inverse, la ressource est stockée dans le répertoire de rendez-vous du point d'accès de l'agent A qui le notifie dès qu'il se reconnecte. Le problème d'authentification des pairs a été également traité en utilisant des algorithmes de cryptographie afin de fournir des identificateurs uniques des pairs dans le réseau. Cela permettra d'éviter les pairs malveillants.

Cette approche diffère des autres approches puisqu'elle tend à chercher des documents précis correspondant aux souscriptions pour des requêtes de durée longue ou pour une durée de temps donnée. Ce mécanisme de recherche est complémentaire et peut être combiné avec les systèmes de recommandation classiques afin de recommander les utilisateurs de façon continue à chaque fois que de nouvelles ressources apparaissent. Nous travaillons dans un cadre plus général où nous disposons uniquement d'une description des besoins de l'utilisateur à qui nous essayons de recommander les ressources les plus pertinentes et non d'une recherche d'une ressource particulière selon des détails précis (tels que le titre d'un fichier, l'auteur et la date d'un article, etc.).

6. **Semantic query routing** (Tempich et Staab, 2006)
   L'algorithme REMINDIN (Routing Enabled by Memorizing INformation about Distributed INformation) est dédié au routage de requêtes. Le principe se base sur l'exploitation de la métaphore sociale pour résoudre cette tâche dans le sens où chaque pair retient des méta-informations sur les connaissances des autres. Chaque pair décide d'interroger un ou quelques pairs en se basant sur son estimation de leur couverture et de leur fiabilité d'information sur des thèmes particuliers. La méthode permet aux pairs d'observer quelles requêtes ont été traitées avec succès par les autres pairs. Ensuite, ces observations sont mémorisées et les informations correspondantes sont utilisées pour sélectionner les pairs auxquels la requête sera envoyée. L'application utilise la plate-forme SWAP et un modèle de méta-données de SWAP qui consiste en deux classes RDFS : La classe Swabbi pour les objets utilisés afin d'obtenir des méta-informations sur les ressources et les états ; La deuxième classe est Peer qui contient des informations sur le pair (nom, id, etc.).
   Chaque éditeur correspond à un pair. Pour chaque thème, un ou plusieurs éditeurs sont

responsables de définir les thèmes et les sous-thèmes reliés.

L'expérimentation a été réalisée avec 1.024 pairs et 1.657 requêtes. Chaque pair est initialement connecté à cinq pairs au hasard. Ensuite, chaque pair utilise REMINDIN pour sélectionner au maximum 2 pairs à qui envoyer la requête. Chaque pair ayant reçu la requête essaie de la résoudre. Dans chaque message de requête, nous trouvons le chemin parcouru par la requête et si un pair est déjà apparu dans le chemin, alors il ne sera pas sélectionné. Chaque requête est réenvoyée jusqu'à atteindre le nombre maximum de rebonds (égal à 6). Ainsi, le répertoire du nœud local d'un pair contient la hiérarchie de thèmes et les liens aux pages web exterieures des thèmes dont il est responsable. Par conséquence, un éditeur demande des liens pour les pages web extérieures appartenant à un certain thème. Les requêtes sont générées en instanciant le modèle suivant : (*, <rdf :type>, topic) tels que les thèmes sont arbitrairement choisis à partir d'un ensemble de thèmes ayant au moins une instance. L'expérimentation ne tient pas compte du fait qu'un pair joigne et quitte le réseau. L'ensemble de requêtes considéré (1.657) est divisé en deux sous-ensembles de même taille pour les deux phases de l'expérimentation de REMINDIN :

- Une première ***phase d'apprentissage*** où le réseau est confronté à un ensemble de requêtes (826).
- Une deuxième phase explicite appelée ***phase de test*** qui permet d'observer comment les pairs du réseau réajustent le deuxième ensemble de requêtes.

Les résultats obtenus montrent que le système apprend et devient plus performant avec le temps. La combinaison de l'algorithme REMINDIN (permettant l'apprentissage) avec l'approche à base de publication (BIBSTER qui n'est pas affecté par le changement de requêtes) est intéressante. L'introduction d'un peu de hasard permet aussi d'améliorer l'efficacité (rappel) de REMINDIN. Les critères d'évaluation considérés sont le rappel (document) permettant d'évaluer l'efficacité de REMINDIN et le nombre de messages qui présente le coût de la recherche par requête, utilisé afin de justifier indirectement le passage à l'échelle de l'approche. La précision des pairs et le nombre de pairs contactés pour une requête donnée ne sont pas mentionnés. C'est pourtant un critère très important qui permet d'évaluer les performance de l'approche et d'améliorer le routage des requêtes uniquement aux pairs intéressants.

7. **Piazza** (Halevy *et al.*, 2003)

   C'est un système permettant l'échange de données relationnelles, XML et RDF. L'application de PIAZZA s'intéresse au domaines de recherche en bases de données pour l'échange de données scientifiques et académiques. Le modèle d'architecture est décentralisé. Le prototype est formé uniquement par 15 nœuds où chacun concerne un aspect du domaine de recherche en base de données. Les groupes de base de données peuvent représenter des

universités, des laboratoires de recherche, des sites Web pour des conférences de bases de données importantes tels que SIGMOD et VLDB. Le principe de routage se base sur les correspondances sémantiques ou les médiations de schémas entre les nœuds. Cette mise en correspondance peut être une inclusion ou une égalité. En présence de différents schémas de requête et de différentes représentations du domaine, les pairs intéressés par l'échange de données (données relationnelles et XML) définissent des correspondances sémantiques entre eux, deux à deux ou entre petits groupes de pairs. Chaque pair exprime ses requêtes sur son propre schéma. Les requêtes sont décrites selon le schéma logique utilisé par le nœud. Par exemple, la requête XQuery suivante est posée à partir du nœud source S1 et réclame tous les « étudiants d'Ullman » :

```
<result> {
for $faculty in /S1/people/faculty,
  $name in $faculty/name/text(),
$advisee in $faculty/advisee/text()
     where $name = « Ullman »
          return
  <student> {$advisee} </student>
               }
          </result>
```

Les deux schémas des pairs $S1$ et $S2$ sont définis dans la figure 3.11.

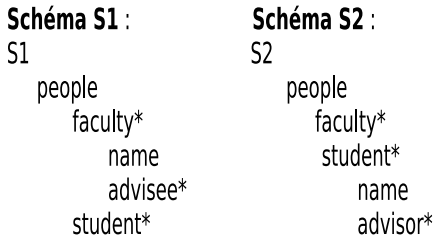

Fig. *3.11: Schémas des deux pairs $S1$ et $S2$*

La mise en correspondance entre le modèle de la requête et le modèle du schéma des correspondances est décrite dans la figure 3.12.

Le schéma de mise en correspondance correspondant au graphe présenté dans la figure 3.12 est décrit dans la figure 3.13.

L'élément « result » dans la requête spécifie simplement l'élément racine du document résultant. L'algorithme proposé représente graphiquement les requêtes et les corespon-

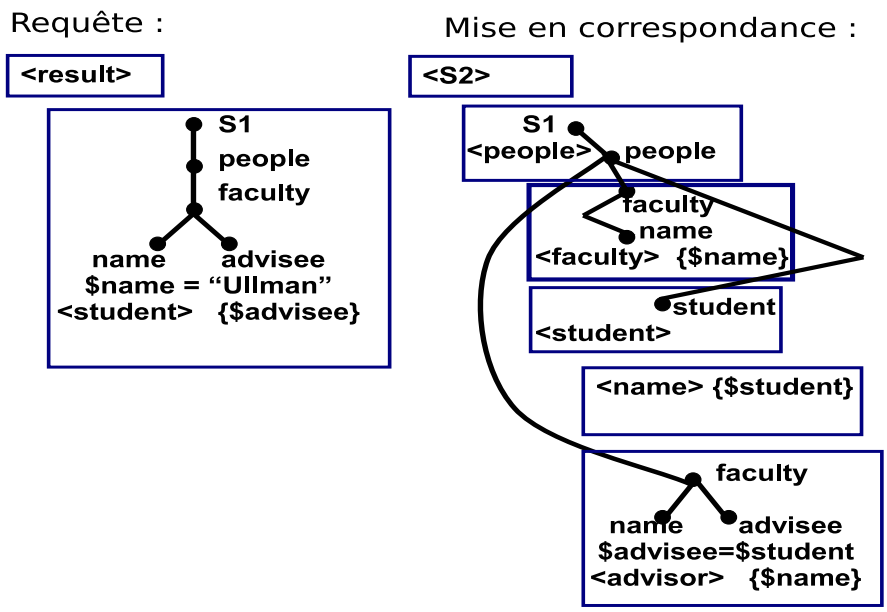

FIG. *3.12: Arbres des modèles de la requête et des correspondances*

```
<S2>
 for $people in  /S1/people
 return
 <people>
     for $name in $people/faculty/name/text()
     return
     <faculty>{$name}</faculty>
      for $student in  $people/student/text()
      return
      <student>
          <name> {$student} </name>
          for $faculty in $people/faculty,
                      $name=$faculty/name/text(),
                   $advisee in $faculty/advisee/text()
                      where $advisee=$student
                      return
              <advisor> {$name}</advisor}
    </student>
  </people>
</S2>
```

FIG. *3.13: Mise en correspondance de $S1$ et $S2$*

dances sous forme d'arbre de requête (respectivement d'arbre de modèle). La réécriture de la requête résultante pour l'exemple précédent chez le nœud S2 est :

```
<result> {
for $student in /S2/people/student,
$advisor in $student/advisor/text(),
   $aname in $student/name/text()
    where $advisor = « Ullman »
              return
    <student> {$name} </student>
              }
</result>
```

Les requêtes sont, dans ce cas, évaluées globalement sur un réseau de pairs sémantiquement liés par les correspondances. Piazza combine et généralise les formalismes LAV (Local-As-View) et GAV (Global-As-View) proposés dans la médiation de schémas dans les systèmes d'intégration de données et les étend aux documents XML. Le langage d'expression des correspondances pour les données relationnelles est PPL (Peer Programming Language) tandis que celui utilisé pour les documents XML est basé sur XQuery. La réécriture des requêtes est basée sur un modèle de correspondance entre les expressions XQuery et les correspondances sémantiques, et elle est faite de manière centralisée. L'application du prototype est préliminaire mais l'architecture fournit une médiation pratique de données structurées hétérogènes et permet l'ajout de nouvelles sources plus facile que dans les environnements deux-tiers traditionnels. L'approche utilisée dans Piazza présente des insuffisances liées à la difficulté de décrire les correspondances, de les construire mais aussi à la maintenance de ces dernières. De plus, la reformulation des requêtes est faite par un nœud central.

8. **SenPeer** (Faye *et al.*, 2006)

   C'est un système de gestion de données distribuées permettant le partage décentralisé de données entre plusieurs experts de différents domaines de compétences. Ces compétences publient des données environnementales relatives à la mise en valeur de la vallée du fleuve Sénégal. SenPeer est un système de type superpair où les pairs sont organisés en domaines sémantiques (thèmes). Ces pairs peuvent publier des bases de données relationnelles, des objets ou des documents XML. Chaque pair dispose de sa propre interface d'accès et de son propre langage d'interrogation.

   Dans ce système, chaque pair correspond à un expert. Les données de la vallée du fleuve Sénégal (thèmes) sont organisés selon une taxonomie. Les thèmes ne sont pas forcément disjoints et chacun correspond à un domaine particulier (par exemple, hydrologie, météorologie). Les superpairs ont un pouvoir de calcul additionnel et une plus grande bande passante par rapport aux pairs, et effectuent des tâches administratives. Ils sont chargés de la gestion d'un ensemble de pairs. Dans chaque domaine se trouve un superpair

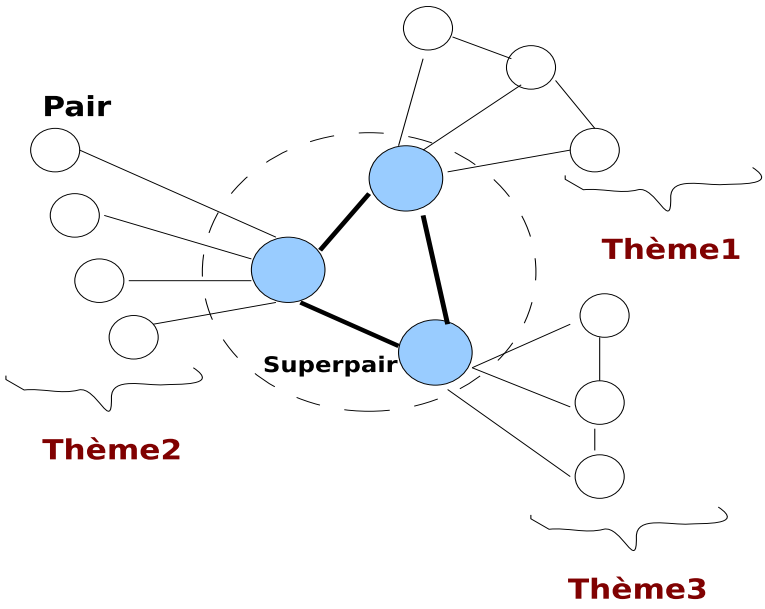

FIG. *3.14: Architecture superpair de SenPeer organisée par thèmes*

auquel est attaché un réseau sémantique expliquant les données du domaine. Ce réseau sémantique est purement conceptuel, les données nécessaires pour répondre aux requêtes sont stockées dans les pairs du domaine. Les pairs d'un même domaine sont directement rattachés au superpair correspondant, mais peuvent décrire leurs données avec des vocabulaires différents. Il devient donc nécessaire d'établir une médiation sémantique entre les pairs d'un domaine et leur superpair, mais aussi entre les superpairs puisque les domaines ne sont pas disjoints. Le principe de la méthode de routage se base sur la médiation sémantique et sur les matrices de correspondance. Dans un but de médiation flexible, chaque pair exporte ses données dans un formalisme pivot commun ayant la structure de graphe (graphe sémantique). Ce graphe est enrichi sémantiquement par des mots-clés destinés à guider la découverte de correspondances entre les éléments des schémas. Ces correspondances sont découvertes grâce à un ensemble de mesures de similarité floues. Ce qui permet l'établissement d'un réseau sémantique permettant un routage intelligent des requêtes et leur réécriture sur d'autres schémas. Le routage de requêtes se base sur les matrices de correspondances gérées par les superpairs. Une matrice de correspondance établit les liens sémantiques de type superpair/superpair et de type superpair/pair. Une médiation sémantique est utilisée afin de faciliter le partage de données entre pairs. Cette médiation passe par l'établissement de correspondances sémantiques entre les schémas des données qui vont être utilisées pour la réécriture des requêtes. Étant donnée une requête formulée avec un langage donné sur le schéma d'un pair, la réécriture de requête consiste

à reformuler cette dernière sur le schéma d'autres pairs avec le langage d'interrogation de ces pairs.

Par exemple, supposons que le pair « SAED » appartenant au domaine « Agriculture » exprime la requête suivante : quels sont le nom, le taux de salinité et l'humidité des sols sur lesquels le riz est cultivé ?

Cette requête est exprimée dans son propre schéma avec son propre vocabulaire en R1 :

(R1)
<nom_sol, taux_sel, humidité>
<nomc = « riz »> <sol, culture>

Lorsque cette requête est envoyée au superpair du domaine Agriculture, elle est réécrite au sein du domaine. D'après les matrices de correspondance SP/P existante (Agriculture/SAED et Agriculture/ISRA), la requête est réécrite avec le vocabulaire du pair $ISRA$ et devient $R2$ telle que :

(R2)
<nomS, taux_sel, hygrometrie>
<terre, culture>
<nomc = « riz »>

La requête $R2$ est reformulée ensuite avec le langage d'interrogation du pair $ISRA$ et les résultats sont envoyés au superpair Agriculture. D'autre part, la matrice de correspondance SP/SP existante (Agriculture/Pédologie) permet de réécrire la requête $R1$, avec le vocabulaire du domaine Pédologie, en une requête $R3$ telle que :

(R3)
<nomTerre, taux_sel, infiltration>
<sol>

Celle-ci est évaluée dans le domaine et les résultats sont ensuite envoyés au superpair Agriculture qui peut enfin composer les résultats et les envoyer au pair SAED ayant initié la requête. Il y a deux niveaux de médiation : la médiation entre les pairs d'un domaine puisque les schémas de ces derniers ont été conçus par des experts différents et la médiation entre superpairs puisque les domaines ne sont pas disjoints. Pour les correspondances sémantiques intra-domaine, un pair sollicitant le partage de données envoie une méta-requête décrivant son domaine par un ensemble de mots-clés au réseau de superpairs afin de trouver le superpair responsable de son domaine. Quand un pair se connecte à un superpair, les informations le concernant (par exemple, id, bande passante, etc.) sont enregistrées dans une table localisée au niveau du superpair et qui contient des informations sur les pairs du domaine. Ensuite, le pair est indexé par le superpair en établissant des correspondances sémantiques (déduites grâce à des mesures de similarité)

entre le modèle interne du pair et le modèle du superpair, exprimé lui aussi sous forme de réseau sémantique. Les correspondances trouvées sont ajoutées dans une matrice de correspondances stockée au niveau du superpair. Quand un pair quitte le réseau, toutes les références le concernant dans la matrice de correspondances sont supprimées. À l'arrivée d'un nouveau superpair, ce dernier envoie aussi une méta-requête décrivant ses centres d'intérêt au réseau de superpairs dans le but de trouver les superpairs dont les domaines recoupent le sien. De la même façon les correspondances entre superpairs sont stockées dans une matrice de correspondances superpair/superpair (SP/SP). Chaque superpair abrite un tel type de matrice. À la connection d'un superpair, ce dernier envoie à ses voisins son modèle interne qui présente son expertise. Ces derniers peuvent alors calculer l'affinité entre leur domaine et celui du superpair initial. En fonction de leurs affinités, ils peuvent accepter les correspondances trouvées en les stockant dans leurs matrices de correspondances et propager la requête à leurs voisins. En cas d'acceptation du partage, ils envoient les correspondances trouvées au superpair initial qui les ajoute à son tour dans sa matrice de correspondances. Le réseau sémantique s'établit donc à partir de la matrice de correspondances et sera utile pour le routage des requêtes hors du domaine. La grande difficulté de cette méthode est les correspondances sémantiques entre les différents éléments des schémas des pairs qui se basent sur des mesures de similarité floues. De plus, il faut trouver les bons algorithmes pour la décomposition, la réécriture et le routage des requêtes.

Nous décrivons également d'autres systèmes posant une problématique qui nous semble être similaire mais dans des contextes différents tels que :

1. **Les middle-agents** (Decker *et al.*, 1997) qui ont pour but d'aider à la localisation des fournisseurs de services. Il s'agit d'un système multi-agents (SMA) où les agents peuvent être des pages jaunes qui traitent les publications, des tableaux noirs qui collectent les requêtes et des courtiers qui traitent les deux. Le principe de la méthode repose sur la mise en correspondance basée sur deux types particuliers d'informations : les compétences et les préférences des agents. Dans un SMA, une préférence est une méta-connaissance sur les types d'information ayant une utilité pour un demandeur. Une compétence est une méta-connaissance sur les types de requêtes qui peuvent être servies par un fournisseur. Similairement à l'utilisation d'une ontologie pour sélectionner les pairs, la mise en correspondance est basée sur les compétences où les spécifications des requêtes sont mises en correspondance avec un ensemble de compétences d'agents ou de services. Ce type de mise en correspondance est utilisé dans les services Web tels que (Li et Horrocks, 2003).
2. **Comités dynamiques** : dans un cadre de tâches de classification où la solution est déterminée en sélectionnant à partir d'un ensemble de solutions énumérées, Ontañón et Plaza (Ontañón et Plaza, 2003) proposent une autre stratégie permettant de sélection-

ner les agents qui vont joindre un comité pour résoudre un problème. L'idée consiste à formuler la problématique d'adhésion d'agents au comité comme une tâche d'apprentissage de l'agent. Un comité est une collection d'agents qui coopèrent pour la résolution d'un problème en donnant un vote pour une solution et où la solution finale est celle ayant le nombre maximum de votes. La motivation des agents pour la coopération sous forme de comité réside dans la possibilité d'améliorer leurs performances dans la résolution de problèmes. Cette stratégie se base sur l'apprentissage sur quand et qui ? En effet, l'agent apprend à évaluer la probabilité qu'un comité courant pourra donner une solution correcte. Si cette probabilité n'est pas assez élevée, l'agent devra inviter un nouvel agent pour joindre le comité et doit déterminer lequel. L'agent apprend à former un comité dynamiquement et à prendre des décisions tels que l'ajout ou non d'un nouvel agent, quand résoudre individuellement un problème. Le processus de cette structure se déclenche quand l'agent convoquant reçoit un problème formant ainsi un comité de taille un. Il décide donc si le problème peut être résolu en isolation ou s'il est mieux de convoquer un comité. C'est une approche d'apprentissage proactive, dans laquelle un agent exécute son activité dans le système multi-agents pour apprendre cette procédure de décision appelée *politique de comité dynamique*. L'agent fait de l'apprentissage dans l'espace de situations de votes. Spécifiquement, un agent utilisant une approche d'apprentissage proactive induit un arbre de décisions qui classifie la situation de vote courante comme positive (la solution semble être correcte) ou négative, et dans ce cas, le comité doit être élargi. Les agents considérés dans le SMA sont capables de résoudre les problèmes individuellement. Le travail considéré se restreint aux tâches analytiques (classification), il n'est donc pas évident de décomposer un problème en sous-problèmes. Ainsi, quand un agent veut coopérer avec un autre pour résoudre un problème, il ne peut pas décomposer le problème pour que chacun résolve une partie. La seule façon de le faire consiste à envoyer le problème complet à l'agent A_j. Quand ce dernier envoie la solution à l'agent A_i, ce dernier compare la solution reçue à sa solution. Si les deux solutions se confirment, alors l'agent A_i peut augmenter le degré de confidence de la solution trouvée. Cependant, si les deux solutions sont en désaccord, il se peut être intéressant d'envoyer le problème à un autre agent afin d'avoir un troisième avis. Le scénario d'interaction (quand l'agent A_i demande à un autre agent A_j de l'aider à résoudre un problème P) se déroule comme suit : l'agent A_i envoie une description du problème à l'agent A_j, l'agent A_j essaie de résoudre le problème en utilisant sa base de cas, renvoie un message $Sorry$ s'il ne peut pas résoudre le problème P, ou un enregistrement de contravention de solution SER ayant la forme $\langle\{(S_k, E_k^j)\}, P, A_j\rangle$ où la collection de solutions approuvées (S_k, E_k^j) signifie que l'agent A_j a trouvé des cas dans sa base de cas approuvant la solution S_k. Ensuite, les agents utilisent un mécanisme de vote afin d'aggréger les informations contenues dans les divers SER provenant des autres agents. Ce mécanisme de vote se base sur le principe

que les agents votent pour les classes de solutions selon le nombre de cas trouvés qui approuvent ces classes. Une fonction de normalisation est définie afin d'éviter le nombre illimité de votes par un agent et donc permettre à chaque agent d'avoir un seule vote. La classe de solution choisie sera celle avec le nombre maximum de votes. La formation de comité est dynamique, c'est-à-dire à chaque fois qu'un agent a besoin de résoudre un nouveau problème, cet agent doit décider lequel des sous-comités est le meilleur pour résoudre ce problème. Afin de réaliser cela, Plaza et al. proposent un protocole incrémental pour la construction d'un comité en se basant sur des techniques d'apprentissage. Chaque agent peut apprendre sa propre politique de comité dynamique en utilisant une stratégie d'apprentissage proactive. Chaque situation de vote est représentée par un vecteur attribut/valeur comportant la classe solution candidate, les votes pour la classe de solution candidate, la somme des votes du reste des classes et le ratio des votes pour la solution candidate.

Les agents apprennent la politique de comité dynamique en utilisant un algorithme d'apprentissage basé sur les arbres de décision avec des techniques de discrétisation pour les attributs numériques. Il s'agit d'un contexte différent dans le cadre de tâche de résolution de problème où la collaboration se fait uniquement en cas de nécessité pour résoudre un problème global alors que nous privilégions la collaboration pour faire des recommandations.

3.5 Conclusion

Les difficultés dans les approches qui se basent sur une architecture de type superpair concernent le choix de la hiérarchie de classification, le niveau de granularité et la classification des pairs. Dans ce type d'approche, les pairs sont supposés être classifiés et regroupés en clusters. Cela peut assurer un meilleur routage de la requête et diminuer le nombre de messages échangés, mais limite l'autonomie et l'évolution des pairs dans le réseau. En effet, chaque pair doit appartenir à un ou plusieurs clusters. Un autre problème qui découle de certains systèmes ayant ce type d'architecture est la classification des requêtes afin de trouver le bon cluster qui peut contenir des documents leur correspondant. L'architecture de type décentralisé assure complètement l'autonomie des pairs sans imposition de contrainte. Chaque pair n'appartient à aucun groupe prédéfini et évolue librement dans le réseau. Les approches qui se basent sur ce type d'architecture P2P pur, utilisent en général l'ontologie pour la classification des documents (contenu des pairs) alors que pour celles basées sur le modèle superpair l'utilisent dans un premier temps pour classifier les pairs en clusters. Ce qui réduit l'aspect autonome et évolutif des pairs.

Afin de garder l'autonomie totale des pairs, nous avons opté pour une architecture décentralisée. Or, dans ce type d'architecture, trouver les pairs pertinents, pour une requête donnée, devient

plus difficile. Pour cela, il faut prévoir un mécanisme de routage « intelligent » de requête dans le réseau. Nous pensons que les expériences d'interaction passées entre les pairs peuvent fournir une bonne source de connaissances sur les pairs et sur leur contenu. Ces expériences peuvent être réutilisées afin d'assurer un meilleur routage de requête entre les pairs pertinents et afin de réduire la charge du réseau en terme de nombre de messages échangés. C'est pour cela que nous nous basons, dans notre approche, sur l'exploitation de l'historique des interactions et des mécanismes d'apprentissage à partir du passé. Ces historiques sont utilisés pour faire les mises à jour des connaissances sur les pairs connus et ainsi apprendre à mieux former son comité par rapport à une thématique donnée. Cela est réalisé en utilisant la méthodologie du raisonnement à partir de cas qui permet l'apprentissage incrémental et la réutilisation des expériences passées.

Dans toutes les approches, la recherche est déclenchée de manière explicite par l'utilisateur en introduisant ses critères de recherche ou en lançant directement une requête. Une approche demandant moins d'effort à l'utilisateur sera plus pratique. C'est pour cela que nous avons choisi de se baser sur l'observation de l'utilisateur afin de déduire ses centres d'intérêt et ses évaluations des résultats qui lui sont proposés.

Notre approche est développée dans le cadre d'une application pilote **Cobras** (Karoui *et al.*, 2006). C'est un système pour la gestion et la recommandation de références bibliograhiques. Son but est le partage de références bibliographiques à travers la recommandation implicite entre des groupes organisés d'utilisateurs. Nous avons adopté une architecture décentralisée afin d'assurer l'autonomie des pairs. Notre approche se base sur l'exploitation des traces des interactions entre les utilisateurs et l'apprentissage continu pour améliorer les recommandations fournies. Tous les pairs utilisent une simple ontologie de domaine pour classifier leurs références mais l'utilisent différemment. L'utilisateur n'intervient à aucun moment ; ni pour exprimer explicitement ses centres d'intérêt, ni pour aider le routage de sa requête. Tout se fait d'une manière implicite en se basant sur l'observation de ses comportements par un agent assistant qui lui est associé. Ce système sera détaillé dans le chapitre suivant du manuscrit.

Système	Ontologie	Historique	Apprentissage	Recherche/ Recommandation	Architecture	Echelle	Dynamisme	Critères quant.
Bibster	2 ontologies communes	non	non	Recherche explicite	décentralisée	moyen	connexion /déconnexion	rappel, précision & nombre de messages
SON	1 ontologie commune	non	non	Recherche explicite	super-pair	petit	-	nombre de messages
Routing indices	1 ontologie commune	non	non	Recherche explicite	décentralisée	petit	-	nombre de messages
SenPeer	1 ontologie commune	-	-	Recherche explicite	super-pair	moyen	-	-
Piazza	1 hiérarchie de thèmes	-	-	Recherche explicite	décentralisé	petit	-	-
Padoue	1 ontologie commune	oui	oui	Recherche explicite	super-pair	large	-	-
REMINDIN	1 hiérarchie de thèmes & modèle de métadonnées	oui	oui	Recherche explicite	décentralisée	moyen	-	nombre de messages & rappel
Semantic publish/ subsribe system	1 ontologie commune	non	non	Recherche explicite	super-pair	petit	connexion /déconnexion	-

TAB. *3.2: Une synthèse des approches de formation de comité de notre classe de problèmes*

Chapitre 4

COBRAS : Système coopératif pour la recommandation de références bibliographiques

Dans ce chapitre, nous décrivons une application pilote que nous avons développée afin d'illustrer notre approche de recommandation de documents. L'application, nommée COBRAS, est détaillée dans la section 5.2.2. Nous expliquons le principe de calcul des recommandations dans la section 4.2. L'approche de formation de comités proposée est détaillée dans la section 4.3. Finalement, nous terminons par une conclusion dans la section 4.4.

4.1 Description Générale

L'objectif du système consiste à aider les utilisateurs dans la gestion de leur base bibliographique et à échanger des données bibliographiques entre des groupes d'utilisateurs partageant des centres d'intérêt. L'architecture de COBRAS est de type égal-à-égal où chaque utilisateur est assisté par un agent logiciel personnel, comme l'illustre la figure 4.1. Chaque agent personnel offre des services à son utilisateur tels que l'édition de références, la correction de références et surtout la recommandation de références pertinentes d'une manière implicite et intelligente.

Chaque utilisateur gère localement une base bibliographique (BB). Chaque agent assistant observe le comportement et les actions de son utilisateur sur la base locale. L'agent détecte alors les thèmes d'intérêt courants de son utilisateur et déclenche le processus de recommandation afin de trouver les références pertinentes à lui proposer. Le système de recommandation est composé de deux modules : un module de formation de comités et un module de calcul des recommandations. Le premier vise à trouver un comité d'agents pertinents par rapport à un centre d'intérêt de l'utilisateur, à qui la demande de recommandation sera envoyée. Le deuxième module a pour fonction de proposer des références locales en réponse à une requête

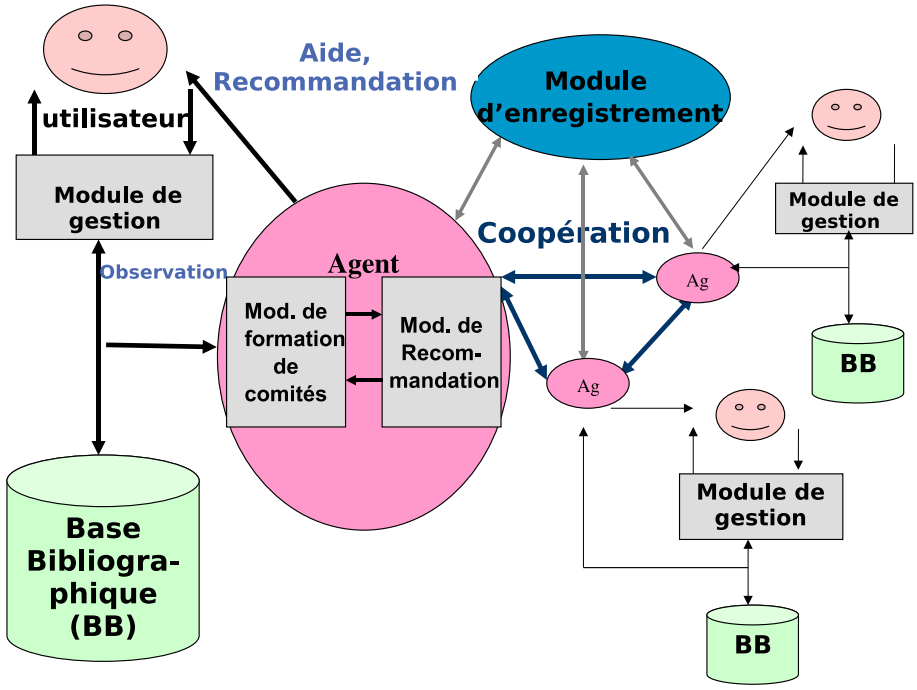

FIG. *4.1: Architecture globale du système COBRAS*

de recommandation envoyée par un autre agent.

Afin d'exploiter ses propres expériences passées et celles des autres et de permettre un apprentisage automatique, chaque agent implémente ces deux modules en utilisant la technique du raisonnement à partir de cas (RàPC). Le système offre un agent d'enregistrement qui détient des informations sur les différentes connexions/déconnexions des agents et d'autres informations système.

Afin de faciliter les opérations d'exportation/transformation, les références bibliographiques sont stockées sous un format XML. Chaque référence r est décrite par un enregistrement contenant les informations suivantes.

- ***Données bibliographiques*** (notées $r.biblio$) : ce sont des données classiques composant une référence bibliographique telles que le type (Article, InProceedings, Report, etc.), la liste des auteurs, le titre, etc.
- ***Mots-clés*** (notés $r.keywords$) : c'est une liste de mots-clés donnée par l'utilisateur décrivant la référence.
- ***Thèmes*** (notés $r.topics$) : c'est une liste de thèmes auxquels la référence est liée. La même hiérarchie de thèmes est utilisée par tous les utilisateurs. Il est raisonnable de supposer qu'une même équipe de recherche utilise les mêmes arbres de thèmes pour indexer les références bibliographiques. Cependant, nous soulignons que la même hiérarchie peut être utilisée différemment par différents utilisateurs.

– ***Annotation*** (notée *r.annotation*) : c'est un champ de texte libre permettant à l'utilisateur d'ajouter des détails sur une référence.

4.1.1 Fonctionnement

Le processus de calcul des recommandations est décrit par le diagramme de séquence dans la figure 4.2 et se déroule comme suit :

– Chaque agent observe les actions (ajout, suppression, recherche, édition) de son utilisateur sur la base de références bibliographiques locale. Ces actions vont permettre de calculer l'importance d'un thème pour l'utilisateur selon un algorithme détaillé en section 4.1.2.
– Pour chaque thème d'intérêt courant détecté, l'agent, appelé *agent initiateur*, envoie une requête de recommandation à un *comité* de pairs. Une requête de recommandation est composée d'un thème et de son interprétation par l'agent initiateur. Un comité est un ensemble d'agents susceptibles d'avoir de bonnes références associées au thème courant. Ce comité est calculé par l'agent lui-même en explorant les traces de ses interactions avec les autres agents et en suivant les recommandations des autres agents (voir section 4.3).
– Chaque agent contacté (appelé *agent de recommandation*) cherche des références satisfaisant la requête reçue. Il envoie ses références trouvées ainsi qu'un ensemble d'agents qu'il juge : a) similaires à l'agent initiateur en terme de requêtes posées, b) intéressants ou c) non intéressants par rapport au sujet de la requête. La recherche des références est basée sur une autre approche RàPC afin de réutiliser les solutions pour la résolution de problèmes futurs similaires et afin d'accélérer la recherche dans la base bibliographique (cf. section 4.1.3).
– L'agent initiateur traite les résultats reçus des différents agents de recommandation. Le résultat contient non seulement des listes de références, mais également des agents recommandés. L'agent initiateur met à jour ses connaissances sur les agents qu'il a contactés et sur les agents qui lui ont été recommandés, classe les références obtenues suivant la valeur de leur similarité avec la requête et propose les k meilleures références à son utilisateur. k peut être un paramètre fixé par l'utilisateur. L'utilisateur peut évaluer les références recommandées pendant une durée bien déterminée δ_t (par exemple, une semaine, un mois), qu'il peut fixer lui-même ou qui peut être déterminée par son agent assistant. Une fois la période écoulée, l'agent initiateur évalue les agents qui ont proposé les références selon leur apport et envoie ses évaluations aux agents concernés.
– Chaque agent de recommandation apprend à partir du retour reçu de la part de l'agent initiateur et met à jour sa base de cas de références et son évaluation par l'agent initiateur par rapport au sujet de la requête, utilisées pour mieux calculer les bonnes recommandations.

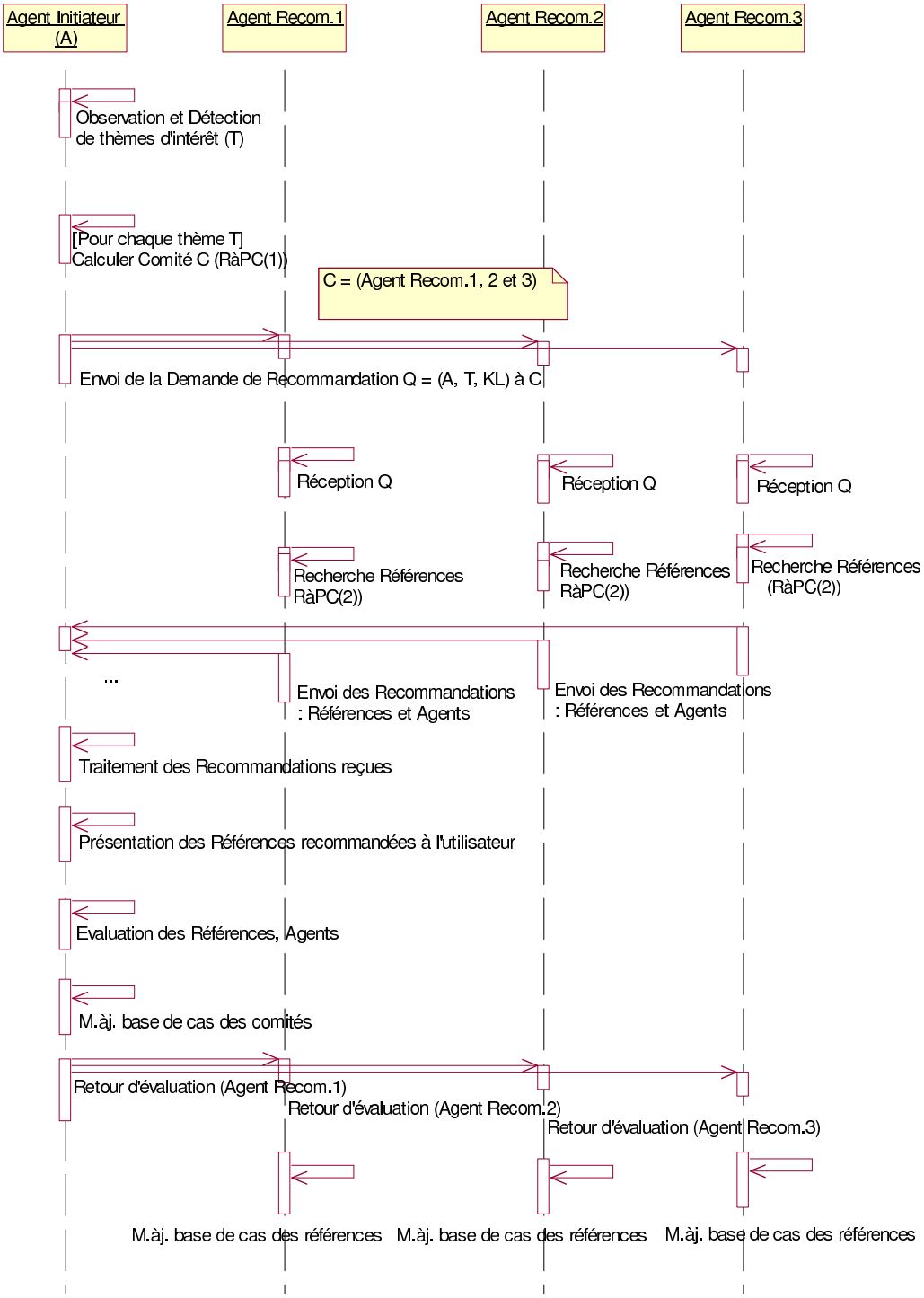

FIG. *4.2: Diagramme de séquence de la coopération des agents pairs dans COBRAS*

4.1.2 Calcul des thèmes d'intérêt courants

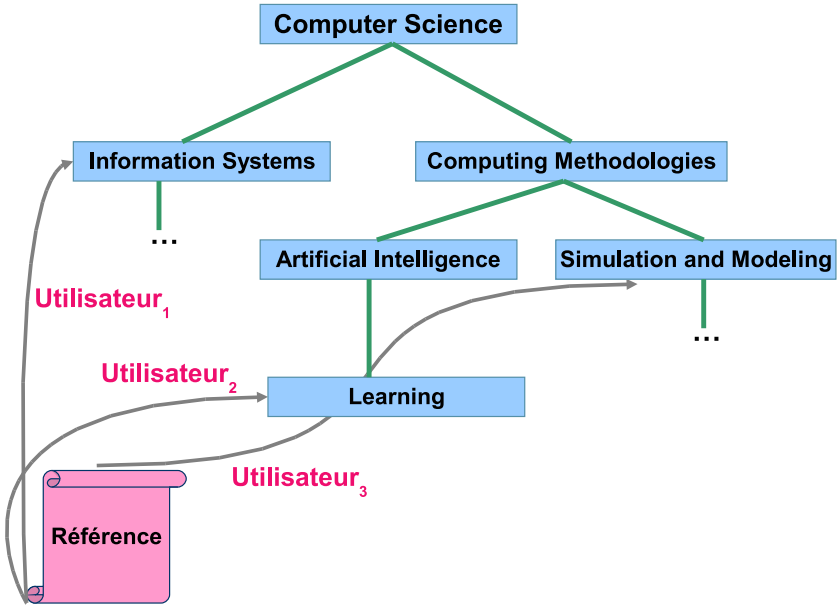

FIG. *4.3: Exemple d'utilisations de la hiérarchie de thèmes ACM*

Tous les utilisateurs partagent une même hiérarchie de thèmes mais l'utilisent différemment comme le montre la figure 4.3. Par exemple, une même référence R peut être classée sous le thème « Information Systems » par un $utilisateur_1$, sous le thème « Learning » selon un $utilisateur_2$ et sous le thème « Simulation and Modeling » selon un troisième utilisateur. Tous les thèmes ne présentent pas le même intérêt pour chaque utilisateur. De plus, pour chaque utilisateur, l'ensemble de thèmes intéressants change au cours du temps. Les recommandations fournies ne doivent pas être intrusives : les références à suggérer doivent être pertinentes et doivent se présenter à l'utilisateur au bon moment. En effet, si les utilisateurs sont réellement intéressés par le domaine de recommandation, ils seront plus enclins à évaluer ces recommandations. Dans le but de calculer la liste des thèmes courants d'un utilisateur, nous proposons un algorithme simple qui « mesure la température » de chaque thème à chaque période de temps δ_t. L'analogie employée consiste à augmenter la « température » d'un thème s'il est impliqué dans une action de l'utilisateur. Les thèmes ayant une température au-dessus d'un certain seuil σ seront considérés comme des *thèmes d'intérêt courants*. L'algorithme de détermination des thèmes d'intérêt courants est décrit dans la figure 4.4 et se fait comme suit :

- Initialement, tous les thèmes ont une température égale à zéro. Chaque action exécutée par l'utilisateur invoquant un thème t va modifier alors la température de t d'une quantité spécifique nommée θ.
- Les actions typiques qui modifient la température sont : l'ajout, la suppression d'une réfé-

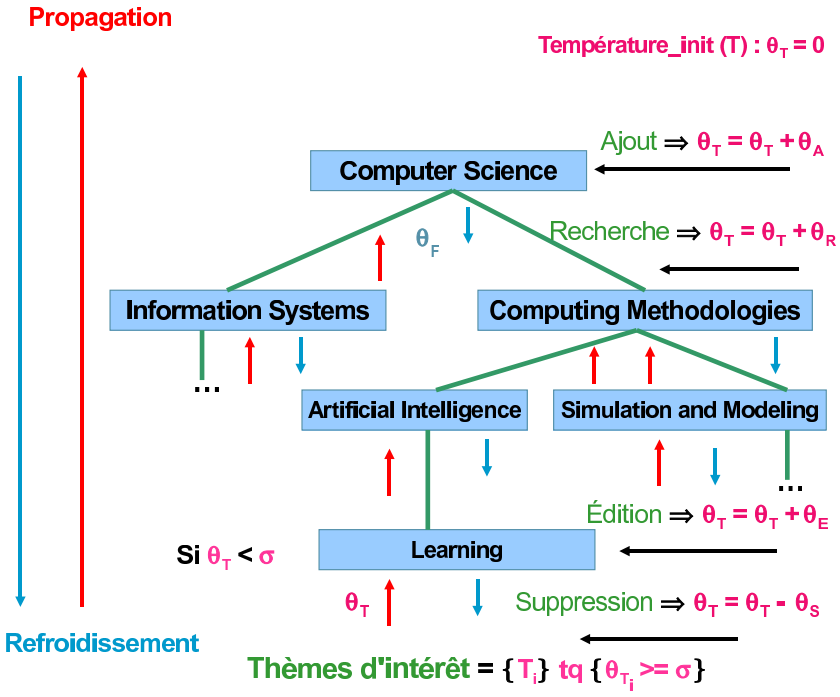

FIG. *4.4: Calcul des thèmes d'intérêt courants*

rence, l'édition ou la recherche d'une référence associée à un thème. Les différentes actions modifient différemment les valeurs de la température courante du thème respectivement θ_A, θ_S, θ_E et θ_R comme le montre la figure 4.4.

- Une *fonction de refroidissement* est aussi appliquée périodiquement afin de diminuer la température des thèmes non utilisés par l'utilisateur d'une quantité spécifique, nommée θ_F.
- Afin de déterminer les thèmes les plus spécifiques à l'activité de l'utilisateur, une *fonction de propagation de température* est appliquée. À partir des feuilles de l'arbre des thèmes, chaque thème propage sa température courante au thème parent. Les thèmes ayant une température au-dessus d'un certain seuil fixé σ cessent de propager leur température et seront ajoutés à la liste des *thèmes d'intérêt courants*.
- Les n thèmes ayant la température la plus élevée qui ont été ajoutés à la liste des thèmes d'intérêt courants, après le parcours de l'arbre d'une façon ascendante, seront retournés. L'heuristique consiste à retourner les thèmes les plus spécifiques qui révèlent un certain niveau des intérêts de l'utilisateur.

4.1.3 Calcul des recommandations de références

L'objectif du service de la recommandation est de trouver des correspondances entre l'utilisation de la hiérarchie de thèmes par un utilisateur et les différentes utilisations chez les

autres utilisateurs. Le module de calcul des recommandations de références prend comme entrée une requête de recommandation et fournit comme résultats une liste de références et une liste d'agents comme le montre la figure 4.5. Un agent interroge ses pairs en leur adressant une demande de recommandation (requête) décrivant un centre d'intérêt de son utilisateur. Une requête utilisateur présente un thème d'intérêt et son interprétation par l'utilisateur (par exemple, des mots-clés).

FIG. *4.5: Module de calcul des recommandations*

Quand un agent de recommandation reçoit une requête, il commence la recherche dans sa base locale pour des références qui correspondent au sujet de la requête. L'adéquation référence/requête est évaluée par une simple fonction de similarité qui mesure la similarité entre une requête Q et une référence r (voir équation 4.5 en section 4.2.4). Utilisant ces fonctions de similarité, l'agent local essaye de retourner les m références les plus pertinentes qui correspondent à la requête reçue. À partir du thème cible T, l'agent cherche les références reliées qui sont similaires au-dessus d'un certain seuil σ_r. Puis il examine les références reliées à des thèmes plus spécifiques, puis il cherche dans des thèmes plus généraux. Le processus de recherche se termine quand il n'y a plus de thèmes à visiter. Chaque référence retournée est associée à un score représentant sa valeur de similarité avec la requête initiale. L'agent initiateur fusionne et filtre les références reçues : il élimine d'abord les références dupliquées et celles déjà connues par l'utilisateur (celles déjà stockées dans la base de données locale) même si elles sont associées à d'autres thèmes, ensuite, il les présente à son utilisateur.

COBRAS permet donc d'offrir à l'utilisateur des recommandations implicites provenant d'autres utilisateurs partageant des centres d'intérêt. Pour permettre cela, le système utilise deux modules de RàPC. Le premier permet de calculer, pour une requête donnée, un comité d'agents avec lesquels il va interagir. Le second permet de chercher les références à proposer à l'utilisateur chez les différents agents de recommandation formant le comité trouvé par le premier composant.

4.2 La recommandation de références

Suite à l'envoi d'une demande de recommandation, l'agent initiateur interagit avec les agents de recommandation qu'il a contactés comme le décrit le diagramme de la figure 4.6. Une requête

Q est un triplet $Q = \langle A, T, KL \rangle$ où :

- A est l'identificateur de l'agent initiateur qui envoie la requête.
- T est un thème d'intérêt (de la hiérarchie partagée).
- KL est la liste de mots-clés calculée à partir de l'ensemble des listes de mots-clés décrivant les références liées directement ou indirectement au thème T. Une référence est indirectement reliée à un thème T si elle est reliée (directement ou indirectement) à un thème T' plus spécifique que T. Les thèmes spécifiques considérés sont ceux qui ont participé à la sélection de T comme thème d'intérêt en propageant leur température (c.f. section 4.1.2).

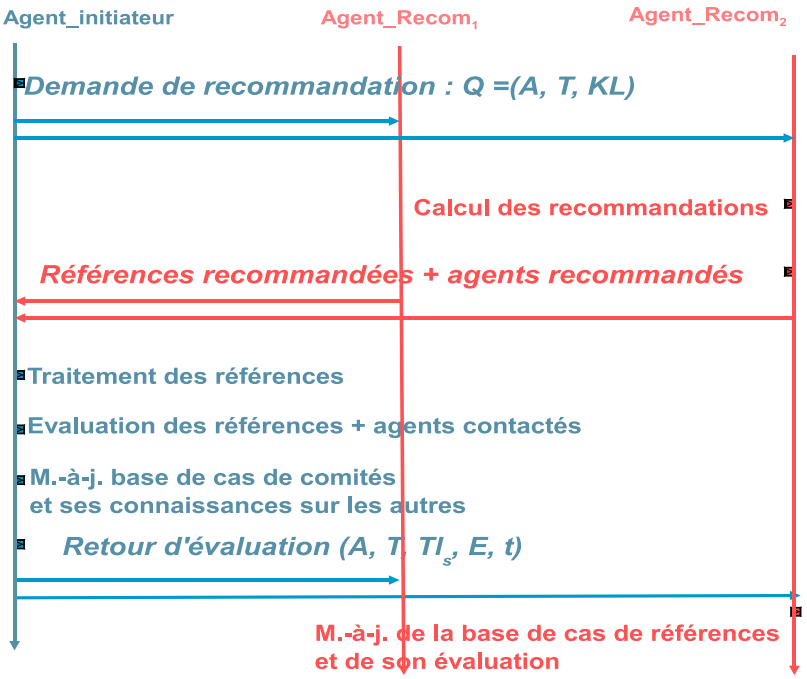

FIG. *4.6: Interactions entre l'agent initiateur et les agents de recommandation*

Chaque agent de recommandation ($agent_j$) traite la requête reçue et répond à l'agent initiateur par une liste de références ainsi qu'une liste d'agents recommandés, pondérés par leur score d'évaluation (c.f. section 4.3.3). L'agent initiateur traite ensuite les recommandations reçues. Il commence par traiter les références et les proposer à son utilisateur. L'étape suivante consiste à mettre à jour ses connaissances sur les agents. Nous distinguons entre deux types d'agents : les agents contactés et les agents recommandés par les agents contactés. La mise à jour des évaluations des agents recommandés se fait immédiatement en intégrant les scores des agents dans la partie correspondante de la base de connaissances. Quant à la mise à jour des évaluations des agents contactés, elle se fera en mode différé après l'écoulement d'une période de temps δ_t, selon l'évaluation de l'utilisateur. Nous supposons que l'utilisateur sélectionne les références

qui lui paraissent pertinentes. Ces références seront ajoutées dans sa base bibliographique locale.

Une bonne référence est donc une référence qui a été sélectionnée par l'utilisateur. Nous supposons également que toutes les références associées au thème T dans la base locale de l'agent sont jugées bonnes puisqu'il les a gardées.
Finalement l'agent initiateur va évaluer chaque agent et va lui envoyer son évaluation (E) dans le message de retour (A, T, TL_s, E, t) par rapport à la requête initiale où :

- A est l'identificateur de l'agent initiateur,
- T est le thème d'intérêt courant,
- TL_s présente la liste des thèmes des bonnes références proposées par l'agent de recommandation,
- E est l'évaluation des résultats par l'agent initiateur,
- t est le temps d'élaboration du cas.

L'évaluation présente la valeur de la précision locale de l'agent par rapport au sujet de la requête T qui est faite par l'agent initiateur (i). L'agent de recommandation va donc mettre à jour ses connaissances et son évaluation par l'agent concerné.

4.2.1 Représentation de cas

Les cas sont élaborés à partir des données brutes. Ces cas seront utilisés pour trouver les correspondances d'utilisation de la hiérarchie de thèmes chez l'agent initiateur et les autres agents. Ainsi, les cas indiquent où chercher dans les bases locales des autres agents. Cela permet d'accélérer la recherche au lieu de repartir à chaque fois des données brutes. Dans le cas où il n'y a pas de correspondance trouvée, le système va permettre d'élaborer de nouveaux cas à partir des données brutes en cherchant directement dans la base bibliographique. Le RàPC permet donc un apprentissage incrémental de ces expériences de mise en correspondance entre les agents pour leur utilisation différente de la même hiérarchie.

Un cas fournit pour un thème d'intérêt et son interprétation par un utilisateur donné, un ensemble thèmes correspondant et leur interprétation chez un autre utilisateur.
Structure d'un cas référence :
Un cas source a la structure suivante « (A, T, KL), (TL_s, KL_s, E, t) » où :

- **La partie problème (A, T, KL)** : présente un centre d'intérêt (T) de l'utilisateur et son interprétation par des mots-clés (KL), déduits directement de la requête de recommandation.
- **La partie solution (TL_s, KL_s, E, t)** : présente la correspondance du thème d'intérêt et de son interprétation (la partie problème) chez un autre utilisateur. Cette correspondance est définie par un ensemble de thèmes (TL_s) et un ensemble de mots-clés (KL_s), présen-

tant des indices pertinents pour la recherche des références dans la base bibliographique de l'agent de recommandation. Une indication sur la pertinence du cas par rapport à la requête est aussi prévue. La solution comprend donc :

1. TL_s : un ensemble de thèmes triés selon leur spécifité (du plus spécifique au plus général) et/ou leur pertinence (par exemple, selon le nombre de bonnes références associées), calculés lors de l'évaluation des références par l'agent initiateur (c.f. section 4.1.1).
2. KL_s : un ensemble de mots-clés général et bien représentatif des différents thèmes des références trouvées. Il est composé par l'union des intersections des mots-clés des références associées aux thèmes TL_s.
3. E : l'évaluation de l'application du cas faite par l'agent initiateur une fois que les résultats lui sont proposés. Elle est exprimée en terme de précision locale des références présentant le taux de bonnes références proposées par l'agent par rapport à toutes les références qu'il propose à l'agent initiateur (voir équation 5.4 du chapitre 5).
4. t présente la date d'élaboration du cas, qui servira à la mise à jour des cas dans le futur.

Ainsi, un cas présente une source d'information que ce soit sur les données (contenu de la base : thèmes et mots-clés) ou sur les autres agents (pour quel agent ce cas a été crée, correspondances d'utilisation de la hiérarchie, centre d'intérêt des autres utilisateurs, agents similaires, etc.). Cette partie solution présente un ensemble d'indices importants, utilisés par l'agent pour faciliter la recherche des références dans la base bibliographique locale.

4.2.2 Cycle RàPC

Le cycle RàPC pour le calcul des recommandations chez un agent, pour une requête donnée, est décrit dans la figure 4.7. Chaque agent dispose de deux mémoires : une base de cas contenant un ensemble de cas sources et une base bibliographique contenant les références bibliographiques de l'agent. Les cas sources sont rémémorés à partir de la base de cas et les références sont extraites à partir de la base bibliographique.

Nous détaillons maintenant les différentes phases d'un cycle RàPC pour la recommandation de références bibliographiques (voir figure 4.7). Nous commençons par la phase de recherche puisque le cas cible est élaboré directement à partir de la requête. C'est un triplet $Q = (A, T, KL)$ dont les éléments sont déjà définis dans la section 4.2.1.

1. ***La phase de recherche***

   La recherche se base sur une mesure de similarité, qui compare le thème et les mots-

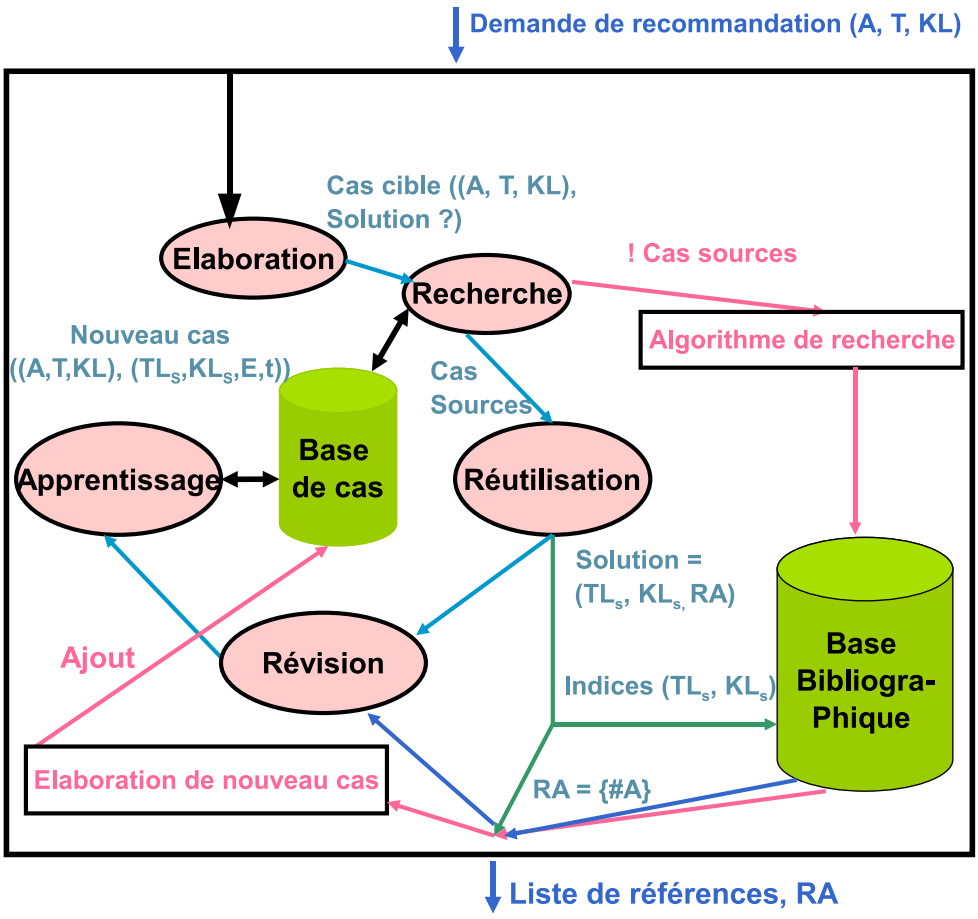

FIG. *4.7: Cycle RàPC pour le calcul de recommandations*

clés du cas source (CS) à ceux du cas cible (CC). La recherche de la solution du cas cible est effectuée en remémorant les cas sources dont la similarité au cas cible dépasse un certain seuil σ_c. La mesure de similarité (voir équation 4.1 dans la section 4.2.3) est une aggrégation pondérée de deux similarités de base : une similarité de thèmes et une similarité de mots-clés ($Sim_{Keywords}$).

Deux cas de figures se présentent :

(a) la recherche aboutit à des cas sources : dans ce cas, nous passons à la phase suivante du cycle,

(b) la recherche n'aboutit pas à des cas sources : dans ce cas, la recherche se fait dans la base biliographique (section 4.2.4) selon un algorithme de recherche spécifique (algorithme 5) et va aboutir à l'élaboration de nouveaux cas sources potentiels.

2. ***La phase de réutilisation***

Le but de cette phase consiste à proposer une solution à partir d'un ensemble de solutions des cas sources trouvés au cours de l'étape précédente. Rappelons qu'un cas source a la structure suivante : $CS = ((A, T, KL), (TL_s, KL_s, E, t))$. Au cours de cette phase, le système forme la solution qui présente un ensemble d'indices (thèmes et mots-clés) correspondant à la description du problème cible. Ces indices sont utilisés pour chercher des références dans la base bibliographique locale comme le décrit l'algorithme 4, de la section 4.2.4. Plusieurs alternatives peuvent ainsi être appliquées afin de former la solution. Nous avons choisi de prendre l'union des solutions trouvées afin d'élargir l'espace de recherche éventuellement intéressant. En plus de ces indices retournés par cette phase, le système calcule une liste d'agents à recommander (nommée (RAL)). À chaque agent de RAL est attribué un score (défini dans la section 4.3.3) qui présente son évaluation par l'agent qui le recommande. Les agents recommandés peuvent être de trois types :

(a) Agents similaires (AS) : en terme de thème d'intérêt. Ces agents ont posé des requêtes similaires dans le passé mais n'ont pas été évalué par l'agent de recommandation. Ces agents peuvent donc être des agents intéressants et leur score d'évaluation est nulle.

(b) Agents intéressants (AI) : jugés intéressants par l'agent de recommandation par rapport au thème du cas cible. Ces agents ont leur score d'évaluation strictement positif.

(c) Agents non intéressants (ANI) : jugés non intéressants par l'agent de recommandation par rapport au thème du cas cible. Ces agents sont donc à éviter et ont leur score d'évaluation strictement négatif.

La liste des agents similaires (AS) est déduite à partir des cas sources trouvés. Quant aux deux autres types agents, ils sont déterminés par l'agent de recommandation en consultant sa matrice d'évaluation définie dans la section 4.3.2.

$RAL \leftarrow (\{AS\} \cup \{AI\} \cup \{ANI\})$. Si un agent apparaît à la fois dans AS et dans AI (respectivement dans ANI) alors il sera considéré comme un agent intéressant (respectivement non intéressant) puisqu'il a été déjà évalué par l'agent qui le recommande.

Algorithme 1 Algorithme de choix de la solution.

// Initialisation de la solution au cas cible
$Solution \leftarrow \emptyset$
Pour chaque $cas_i \in Liste_{cas}$ **Faire**
// Construction de la solution
$Solution.TL_s \leftarrow Solution.TL_s \cup \{cas_i.TL_s\}$
$Solution.KL_s \leftarrow Solution.KL_s \cup \{cas_i.KL_s\}$
$AS \leftarrow AS \cup \{cas_i.A\}$
Fin Pour

Une fois calculée, l'agent de recommandation envoie sa réponse à la requête à l'agent initiateur. La réponse inclut une liste de références bibliographiques ainsi qu'une liste d'agents recommandés. Il attend ensuite l'évaluation de sa solution par l'agent initiateur afin de réviser sa solution cible. Quand l'agent reçoit une évaluation, il passe alors à la phase suivante pour réviser sa solution.

3. ***La phase de révision***
   Cette phase s'effectue suite à l'évaluation des références par l'utilisateur. Elle se fait en mode différé. L'agent initiateur va évaluer les agents en calculant leur précision (équation 5.4 du chapitre 5) par rapport aux références qu'ils ont proposées et va l'envoyer à l'agent concerné. Suite à la réception du message de l'évaluation (A, T, TL_s, E, t), l'agent de recommandation va élaborer son cas et passera à la phase suivante. Suite à la réception de son évaluation par l'agent initiateur, l'agent va mettre à jour sa matrice d'évaluation (détaillée dans la section 4.3.2).
   L'élaboration d'un nouveau cas source se fait comme suit :
   – la partie problème est formée du problème cible (A, T, KL),
   – la partie solution est composée de la liste de thèmes (TL_s) des références trouvées, la liste de mots-clés (KL_s) décrivant ces références, l'identificateur de l'agent initiateur (A) et son évaluation (E). t correspond à la date d'élaboration du cas.

4. ***La phase d'apprentissage***
   Nous distinguons entre deux cas :
   – Si la phase de recherche n'a pas abouti à des cas sources, alors le nouveau cas source potentiel élaboré sera ajouté à la base de cas de l'agent si son évaluation est satisfaisante (E).
   – Si la phase de recherche a abouti à des cas sources et si le cas a été bien évalué par

l'agent, alors
- si le cas existe déjà pour le même agent, il ne sera pas inséré dans la base de cas, mais le cas sera mis à jour en remplaçant les valeurs de E et de t par les nouvelles valeurs.
- sinon, le cas sera ajouté à la base de cas.

Chaque agent pourra apprendre de cette expérience en ajoutant, dans sa base de cas, les nouveaux cas bien évalués.

Nous précisons que la réponse à une demande de recommandation (l'ensemble des références trouvées) est le résultat de l'application de la solution retournée par le système RàPC (l'ensemble d'indices) sur la base bibliographique. Autrement dit, l'application du système RàPC ne retourne pas un ensemble de références mais un ensemble d'indices (les correspondances) qui vont être utilisés pour guider la recherche dans la base de références bibliographiques. Donc, une nouvelle application de ces mêmes indices (la même solution RàPC) sur la base peut fournir une solution différente en terme de références si le contenu de la base bibliographique a changé.

4.2.3 Mesures de similarités

La mesure de similarité de cas est une aggrégation pondérée de deux similarités de base : une similarité de thèmes (Sim_{Topics}) et une similarité de mots-clés ($Sim_{Keywords}$).

$$\begin{aligned} Sim_{Cas}(CS, CC) = \alpha\ & Sim_{Topics}(CS.T, CC.T) + \\ & \beta\ Sim_{Keywords}(CS.KL, CC.KL) \end{aligned} \tag{4.1}$$

où α et β sont les poids respectifs de la similarité de thèmes et de la similarité de mots-clés, tels que $\alpha + \beta = 1$.

1. **Similarité de mots-clés** : c'est une fonction simple qui mesure le nombre de mots en commun entre deux ensembles (A, B). Elle est donnée par la formule suivante :

$$Sim_{Keywords}(A, B) \quad = \quad \frac{|A \cap B|}{|A \cup B|} \tag{4.2}$$

2. **Similarité de thèmes** : est une aggrégation pondérée de deux similarités. En effet, nous proposons une nouvelle mesure qui dépend de deux types de relations (voir la figure 4.8) : relation de spécialisation que nous appelons $Sim_{Proximité}$ et relation de liaison que nous appelons $Sim_{EnRelation}$:

$$Sim_{Topics}(T1, T2) = \lambda\ Sim_{Proximité}(T1, T2) + \sigma\ Sim_{EnRelation}(T1, T2) \tag{4.3}$$

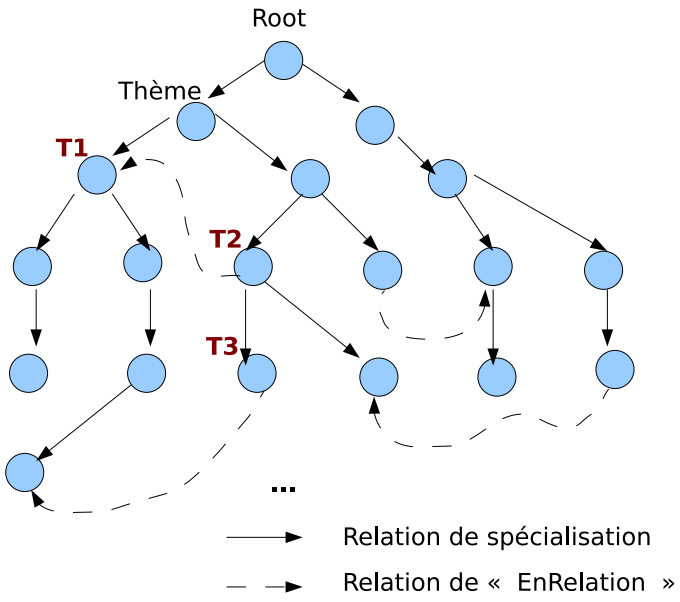

FIG. *4.8: Les deux types de relations dans la hiérarchie de thèmes*

λ et σ étant les poids respectifs de la relation de proximité et de la relation « EnRelation » qui existent entre les deux thèmes $T1$ et $T2$, où $\lambda + \sigma = 1$.

(a) ***La relation de spécialisation*** : dans un premier temps, nous considérons uniquement la relation de spécialisation qui présente les liens de navigation (flèches en trait continu dans la figure 4.8) dans l'arbre et nous calculons la proximité de deux thèmes en utilisant la structure hiérarchique. L'heuristique appliquée est la suivante : la similarité entre deux thèmes dépend de la longueur du chemin qui lie les deux thèmes ainsi que la profondeur des thèmes dans la hiérarchie (Jaczynski, 1998; Jaczynski et Trousse, 1998). Donc, une correspondance avec des nœuds spécifiques plus proches feuilles conduit à une valeur de similarité plus importante que les nœuds correspondant à des niveaux plus hauts dans l'arbre.

$$Sim_{Proximité}(T1, T2) = 1 - \frac{path(T1, MSCA(T1, T2)) + path(T2, MSCA(T1, T2))}{path(T1, root) + path(T2, root)} \quad (4.4)$$

où :
- $path(a, b)$ est la longueur du chemin entre les nœuds a et b.
- $root$ est le thème racine de l'arbre.
- $MSCA(a, b)$ retourne l'ancêtre commun le plus spécifique des nœuds a et b.

(b) ***La relation « EnRelation »*** : dans un deuxième temps, nous considérons la relation « EnRelation » qui existe entre des thèmes (flèches en pointillés dans la

figure 4.8) et nous calculons la valeur de similarité de ces deux thèmes. La similarité selon la relation « EnRelation » prend une valeur maximale si les deux thèmes sont directement liés et diminue au fur et à mesure que les thèmes sont indirectement liés. Deux thèmes sont indirectement liés si l'un des descendants ou ascendants du premier thème est lié à un des descendants ou ascendants du deuxième thème. Plus la relation est directe, plus la valeur de similarité correspondante est importante.

Le calcul de la valeur de cette distance se fait selon l'algorithme 2. La relation « EnRelation » n'est pas symétrique. Ce qui veut dire que si nous cherchons des références associées à un thème $T1$, nous pouvons chercher dans le thème $T2$ et pas forcément l'inverse.

Prenons un exemple simplifié : d'après la figure 4.8 et avec $\lambda = \sigma = \frac{1}{2}$:
$Sim_{Topics}(T1, T2) = \frac{1}{2} \times \frac{2}{5} + \frac{1}{2} \times 0 = \frac{1}{5}$
$Sim_{Topics}(T2, T1) = \frac{1}{2} \times \frac{2}{5} + \frac{1}{2} \times 1 = \frac{7}{10}$

4.2.4 Recherche des références dans la base bibliographique

Cette étape se déroule après l'exécution des deux premières phases du cycle RàPC (la recherche et la réutilisation). À ce niveau, deux cas se présentent : soit le système de RàPC retourne des cas sources, soit le système n'a pas trouvé de cas source correspondant à la requête.

1. ***cas sources retenus***
   Dans ce cas, la solution calculée lors de la phase de réutilisation présente un ensemble d'indices qui vont être utilisés par l'agent de recommandation afin de chercher des références dans sa propre base bibliographique. La liste de thèmes de la partie solution et l'ensemble de mots-clés sont utilisées pour accélérer et guider la recherche suivant l'algorithme 4. Son principe consiste à chercher dans la liste de thèmes trouvés, les références qui vérifient un certain seuil de similarité de mots-clés avec la liste de mots-clés de la solution retournée par la phase de réutilisation du cycle RàPC.
2. ***pas de cas sources retenus***
   Dans ce cas, l'adéquation référence(r)/requête(Q) est évaluée par une simple fonction de similarité $Sim_{Ref}(Q, r)$. Nous rappelons qu'une référence r se compose des éléments suivants : a) biblio , b) keywords, c) topics et d) annotation.

$$Sim_{Ref}(Q, r) = \alpha \max(Sim_{Topics}(Q.T, t)) + \beta Sim_{RefKw}(Q.KL, r.keywords) \quad (4.5)$$

   où $t \in r.topics$, puisqu'une référence peut avoir plusieurs thèmes.

   C'est une aggrégation pondérée des deux similarités : Sim_{Topics} (déjà définie dans l'équation 4.3) et Sim_{RefKw} définie dans l'équation 4.6. Elle mesure la similarité de mots-clés entre une référence r et un ensemble de mots-clés de la requête (Q). Les deux ensembles

Algorithme 2 Algorithme du $Sim_{EnRelation}$.

Entrée: $T1, T2 \in$ thèmes, $ListeSim_{EnRelation} \leftarrow \emptyset$;

Sortie: valeur maximale du lien qui lie T1 à T2

ThemesAVisiter $\leftarrow$ {T1} ; $changerDirection \leftarrow false$;
Pour tout $theme1 \in ThemesAVisiter$ **Faire**
Si Related(theme1, T2) **Alors**
$poidsLien \leftarrow CalculPoids(T1, T2, theme1, T2)$
$ListeSim_{EnRelation} \leftarrow ListeSim_{EnRelation} \cup \{poidsLien\}$
Fin Si
$DescendantsT2 \leftarrow \{fils(T2)\}$
Pour tout $theme2 \in DescendantsT2$ **Faire**
Si Related(theme1, theme2) **Alors**
$poidsLien \leftarrow CalculPoids(T1, T2, theme1, theme2)$
$ListeSim_{EnRelation} \leftarrow ListeSim_{EnRelation} \cup \{poidsLien\}$
Fin Si
$DescendantsT2 \leftarrow DescendantsT2 \cup fils(theme2)$
$DescendantsT2 \leftarrow DescendantsT2 \setminus \{theme2\}$
Fin Pour
$AscendantT2 \leftarrow parent(T2)$
Tant que $AscendantT2 \neq null$ **Faire**
Si Related(theme1, AscendantT2) **Alors**
$poidsLien \leftarrow CalculPoids(T1, T2, theme1, AscendantT2)$
$ListeSim_{EnRelation} \leftarrow ListeSim_{EnRelation} \cup \{poidsLien\}$
Fin Si
$AscendantT2 \leftarrow parent(AscendantT2)$
Fin Tant que
Si $(\neg changerDirection)$ **Alors**
$DescendantsT1 \leftarrow \{fils(theme1)\}$
Si $DescendantsT1 \neq null$ **Alors**
$ThemesAVisiter \leftarrow ThemesAVisiter \cup DescendantsT1$
Sinon
$changerDirection \leftarrow true$
Fin Si
Fin Si
Si $(changerDirection)$ **Alors**
$AscendantT1 \leftarrow parent(theme1)$
Si $(AscendantT1 \neq null)$ **Alors**
$ThemesAVisiter \leftarrow ThemesAVisiter \cup AscendantT1$
Fin Si
Fin Si
$ThemesAVisiter \leftarrow ThemesAVisiter \setminus \{theme1\}$
Fin Pour
return $max(ListeSim_{EnRelation})$

Algorithme 3 Algorithme de la fonction CalculPoids.

Entrée: $T1, T2, theme1, theme2 \in$ thèmes

diminution : constante pour diminuer le poids du lien

$profondeur$: profondeur de l'arbre de thèmes selon la relation de spécialisation

// $2 \times profondeur$ est la longueur maximale entre deux nœuds de l'arbre, utilisée pour la normalisation,

|theme1ToT1| : longueur du chemin entre les thèmes $theme1$ et $T1$

|theme2ToT2| : longueur du chemin entre les thèmes $theme2$ et $T2$

Sortie: valeur du lien qui lie theme1 à theme2

return (1 - [(|theme1ToT1|+|theme2ToT2|)/$(2 \times profondeur)$])

Algorithme 4 Algorithme de recherche avec indices.

Entrée: *Solution* retournée par la phase de réutilisation du cycle RàPC

Sortie: liste de références à recommander (Résultat)

// Initialisation de la liste Résultat

Résultat $\leftarrow$ ()

Pour $thème_i \in Solution.TL$ **Faire**

// LR_i : *la liste des références associées au* $thème_i$

$LR_i \leftarrow Références(thème_i)$

Pour chaque $ref_j \in LR_i$ **Faire**

Si $(Sim_{RefKw}(ref_j.keywords, Solution.KL) \geq \sigma_k)$ **Alors**

$Résultat \leftarrow Résultat \cup \{ref_j\}$

Fin Si

Fin Pour

Fin Pour

n'étant pas de même ordre, nous divisions uniquement par la taille des mots-clés de la référence.

$$Sim_{RefKw}(a, B) = \frac{|a \cap B|}{|a|} \tag{4.6}$$

où a et B sont deux ensembles de mots-clés tel que $|a| << |B|$ (dans notre expérimentation, le rapport a sur B est variable selon le thème mais en moyenne, il est de l'ordre de $0,01$). Dans ce cas, la recherche dans la base des références bibliographiques se fait comme le décrit l'algorithme 5. Le principe de l'algorithme consiste à filtrer les documents indexés par thèmes et mots-clés.

L'agent cherche les références associées qui sont similaires au-dessus d'un certain seuil σ_r. Le processus de recherche se termine quand il n' y a plus de thème à visiter, c'est-à-dire

Algorithme 5 Algorithme de recherche sans indices.

Entrée: T : thème du cas cible

Sortie: liste de références à recommander (Résultat)

// Initialisation de la liste Résultat et des variables
$Résultat \leftarrow ()$, $continue = true$, $change_{direction} = false$;
// Commencer la recherche par le thème cible T
$ListeThèmes \leftarrow \{T\}$
// Utilisation de la structure arborescente de la hiérarchie des thèmes.
Tant que (continue) **Faire**
  Pour tout $thème \in ListeThèmes$ **Faire**
    Si $((Sim_{Topics}(Q.T, thème) \geqslant \sigma_t)$ Et $(\neg change_{direction}))$ **Alors**
      // $ListeRefs_i$ reçoit la liste de références associées au $thème$
      $ListeRefs_i \leftarrow Références(thème)$
      Pour tout $(ref_j \in LR_i)$ **Faire**
        Si $(Sim_{Ref}(ref_j, Q) \geq \sigma_r)$ **Alors**
          $Résultat \leftarrow Résultat \cup \{ref_j\}$
        Fin Si
      Fin Pour
      // Recherche dans les sous-thèmes de $< thème >$
      $ListeThèmes.append(fils(thème))$
    Fin Si
    $ListeThèmes \leftarrow ListeThèmes \setminus \{thème\}$
  Fin Pour
  $ListeThème \leftarrow \{parent(T)\}$
  Si $((\neg change_{direction})$ Et $((Sim_{Topics}(Q.T, thème) < \sigma_t)))$ **Alors**
    // Changer de direction de parcours des thèmes de l'arbre en explorant les sur-thèmes
    $change_{direction} = true$
    Pour tout $thème \in ListeThèmes$ **Faire**
      Si $(\ (Sim_{Topics}(Q.T, thème) \geqslant \sigma_t))$ **Alors**
        $ListeRefs_i \leftarrow Références(thème)$
        Pour tout $(ref_j \in LR_i)$ **Faire**
          Si $(Sim_{Ref}(ref_j, Q) \geq \sigma_r)$ **Alors**
            $Résultat \leftarrow Résultat \cup \{ref_j\}$
          Fin Si
        Fin Pour
      Fin Si
      $ListeThèmes.append(parent(thème))$
      $ListeThèmes \leftarrow ListeThèmes \setminus \{thème\}$
    Fin Pour
    $continue = false$
  Fin Si
Fin Tant que

la similarité entre les thèmes est inférieure à un certain seuil. L'élaboration d'un nouveau cas source potentiel se fait comme le décrit l'algorithme 6. Nous signalons que la valeur de l'évaluation n'est pas obtenue en même temps que les autres attributs. En fait, elle sera connue une fois la réponse envoyée à l'agent initiateur et une fois que l'utilisateur aura terminé son évaluation des références proposées.

Algorithme 6 Algorithme d'élaboration d'un nouveau cas source.

Partie Problème $\leftarrow (T, KL, A)$
// T et KL constituent respectivement le thème et la liste
// de mots-clés du cas cible.
Partie Solution $\leftarrow (TL_s, KL_s, E, t)$
// TL_s et KL_s présentent respectivement la liste de thèmes et la liste
// de mots-clés des références trouvées (contenues dans la liste Résultat),
// construites dans l'étape de recherche.
$A \leftarrow l'identificateur\ de\ l'agent\ initiateur$
$E \leftarrow l'évaluation\ du\ Résultat\ par\ l'agent\ A$
$TL_s \leftarrow Thèmes(Ref(Résultat)))$
// la liste de thèmes des références dans la liste Résultat calculée par l'algorithme 4 ou 5
$KL_s \leftarrow \cup\ (\cap\ MotsCles(Ref(T_i)), tq\ T_i \in TL_s))$
// les mots-clés des références associées à chaque thème dans TL_s
// (et non de chaque référence résultat)

4.2.5 Conclusion

Nous avons présenté le processus de calcul des recommandations où deux parcours de recherche sont possibles suivant le résultat de la phase de recherche du cycle RàPC. Chaque agent détient deux types de mémoires : la première est une base de cas permettant la réutilisation des expériences qui présentent les différentes correspondances d'utilisation de la hiérarchie de thèmes entre les différents utilisateurs. La deuxième contient des données brutes et permet l'élaboration de nouveaux cas potentiels et ainsi capitaliser les différentes connaissances apprises.

Ce module RàPC travaille en collaboration avec un deuxième module qui se charge de calculer le comité d'agents auquel la requête sera adressée.

4.3 La formation de comités

Étant donnée une requête de recommandation, la problématique de formation de comités consiste à trouver un sous-ensemble de pairs disponibles pouvant fournir les résultats les

plus pertinents. L'objectif est double : d'une part, améliorer les performances du système en réduisant la charge du réseau et celles des agents. D'autre part, améliorer la qualité des recommandations fournies en évitant de fausser les résultats par des informations bruitées. Notre approche consiste à exploiter les historiques des interactions des agents afin qu'ils apprennent à sélectionner les agents pertinents de leur comité par rapport à un thème donné.

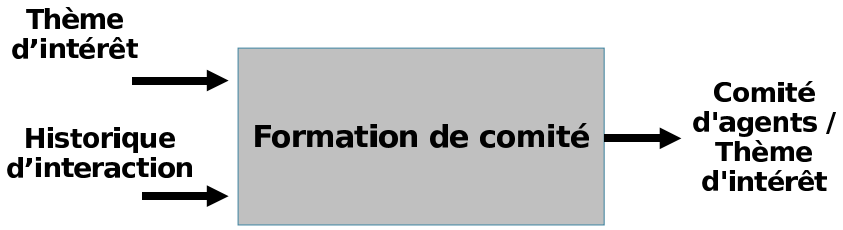

FIG. *4.9: Module de la formation de comités*

4.3.1 Description

Le but de ce module consiste cette fois-ci à faire partager les expériences des utilisateurs au niveau de leurs collaborateurs et par rapport à un centre d'intérêt donné. Ces expériences sont stockées dans des cas afin d'être réutilisées et partagées. En effet, les cas contiennent une liste d'agents qui ont été jugés pertinents par rapport à un thème d'intérêt. Ces cas sont élaborés à partir des traces des interactions des agents. Le RàPC va permettre de réutiliser ces cas et de contacter des agents potentiellement pertinents au lieu de contacter à chaque fois un ensemble d'agents pris au hasard. Chaque agent garde dans sa base de cas de comités les agents qu'il a bien évalués par rapport à un thème donné. L'agent garde également une liste contenant des agents qui lui étaient recommandés par les autres agents et qu'il n'a pas encore évalués par rapport au thème en question et aussi une liste d'agents qu'il a jugés non intéressants (voir section 4.3.2). Suite à une détection de thème d'intérêt courant de son utilisateur, l'agent va essayer de calculer un comité d'agents auxquels il va envoyer sa requête. L'agent va donc construire la liste d'agents candidats formée par les agents retournés par la phase de recherche du système RàPC pour la formation de comités et aussi par les agents qui lui ont été recommandés par rapport à ce thème. À partir de cette liste, l'agent va choisir un sous-ensemble d'agents en se basant sur un critère qui est leur score total. Ce score total regroupe l'avis de l'agent lui même et aussi les avis des autres par rapport à un agent donné et pour un thème donné. Les agents les mieux classés formeront son comité qui sera contacté pour traiter la requête de recommandation. Initialement l'agent initiateur ne dispose pas encore de comités, il va contacter le module d'enregistrement afin de connaître les agents connectés. Il sélectionne ensuite un sous-ensemble de ces agents pris au hasard pour leur envoyer la requête.

4.3.2 Matrice d'évaluation d'un agent

C'est une matrice carrée de taille n, relative à un thème d'intérêt d'un $agent_i$ (notée $MatEval(t)$). Elle contient des évaluations des agents les uns des autres selon leur précision locale qui présente le taux de bonnes références proposées par l'agent par rapport à toutes les références qu'il propose (voir équation 5.4 du chapitre 5). La taille n correspond au nombre d'agents connus directement ou indirectement par l'agent local.
$e_{xy}(t)$: est la valeur de l'évaluation de l'agent x par l'agent y par rapport au thème t. e_{ii} présente l'évaluation de l'$agent_i$ par lui-même et est égale à 1.

$$MatEval(t)_i = \begin{pmatrix} e_{11} & e_{12} & \cdots & e_{1i} & \cdots & e_{1n} \\ e_{21} & e_{22} & \cdots & e_{2i} & \cdots & e_{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ e_{i1} & e_{i2} & \cdots & e_{ii} & \cdots & e_{in} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ e_{n1} & e_{n2} & \cdots & e_{ni} & \cdots & e_{nn} \end{pmatrix}_t \tag{4.7}$$

La matrice n'est pas symétrique ($e_{xy}(t) \neq e_{yx}(t)$, $x \neq y$) puisque les évaluations sont individuelles.
L'ensemble des valeurs de ces évaluations est $\{-1\} \cup \{0\} \cup]0, 1]$ tel que :

1. $e_{xy}(t) =$ -1 signifie que l'agent x a été contacté par l'agent y et qu'il l'a jugé non intéressant par rapport au thème t.
2. $e_{xy}(t) = 0$ signifie que l'agent x n'a pas été encore contacté par l'agent y et est donc non encore évalué par rapport au thème t.
3. $e_{xy}(t) > 0$ signifie que l'agent x a été contacté par l'agent y et a été « bien » évalué. Cette évaluation prend la valeur de la précision locale de l'agent x en terme de références.

La matrice d'évaluation d'un agent est une source riche d'informations portant sur :

- Sa propre évaluation des agents qu'il connaît. C'est une évaluation directe faite par l'agent lui-même sur les autres agents qu'il connaît. Elle reflète son propre avis sur les agents par rapport à un thème donné. Ces informations sont obtenues lors des interactions entre les agents pour résoudre les requêtes reçues.
- Les évaluations des agents. Cette information présente les avis des autres agents sur les agents qu'ils ont recommandés par rapport à un thème t.

La matrice d'évaluation est mise à jour lors de la phase de révision du cycle RàPC pour la formation de comités pour les agents contactés. En ce qui concerne les agents recommandés, leurs scores sont directement mis à jour dans la matrice d'évaluation.

4.3.3 Score d'un agent

Chaque agent a un score total, noté ST. Le score total d'un agent A_j chez un agent i est calculé suivant la formule suivante :

$$ST_i(A_j, T) = \sum_{t \in Thèmes\_similaires(T)} w_t \sum_{k \in Agents(t)} e_{ki}(t) \times e_{jk}(t) \tag{4.8}$$

avec :

- $Thèmes\_similaires(T)$: représente l'ensemble de thèmes similaires au thème T selon la similarité de thèmes et qui vont être considérés dans le calcul du score total de l'agent A_j. Ces thèmes sont déterminés lors de la phase de recherche du cycle RàPC : ils représentent les thèmes des cas sources trouvés et vérifiant un certain seuil de similarité avec le thème de la requête.

- w_t : c'est le poids du thème t, il est égal à la valeur de la similarité (sim_{Topic}) entre les thèmes T et t.

- Agents(t) : représente la liste d'agents connus par l'agent initiateur par rapport au thème t. Elle inclut les agents qui ont été évalués par l'agent i et les agents qui lui ont été recommandés par les autres par rapport au thème t.

- $e_{xy}(t)$: est la valeur de l'évaluation de l'agent x par l'agent y par rapport au thème t.

4.3.4 Représentation de cas

Pour chaque thème d'intérêt, un agent doit contacter un nombre d'agents pour trouver des documents pertinents pour son utilisateur. Au lieu de contacter à chaque fois un ensemble d'agents choisis aléatoirement, nous utilisons le RàPC qui permet de mémoriser des expériences des interactions passées dans sa mémoire sous forme de cas. Ainsi, l'agent cherche dans sa base de cas des comités pertinents par rapport au thème d'intérêt. Si la recherche n'aboutit pas, l'agent contacte un nombre fixé d'agents disponibles et de nouveaux cas potentiels seront élaborés et éventuellement mémorisés dans la base de cas de l'agent. Un cas présente une expérience de collaboration entre agents par rapport à un thème d'intérêt. Un cas comité a la structure suivante : *cas = (T, C, E, t)* où :

1. Problème = T représente un thème d'intérêt.
2. Solution = (C, E, t) décrit le comité formé et son évaluation tels que :
   - C : présente un comité d'agents intéressants par rapport au thème T,
   - E : présente l'évaluation du comité retenu. Elle désigne la précision en terme d'agents, obtenue avec le comité C. Cette précision présente le taux de bons agents du comité parmi tous les agents du comité C.

– t : présente la date d'élaboration du cas.

4.3.5 Cycle RàPC

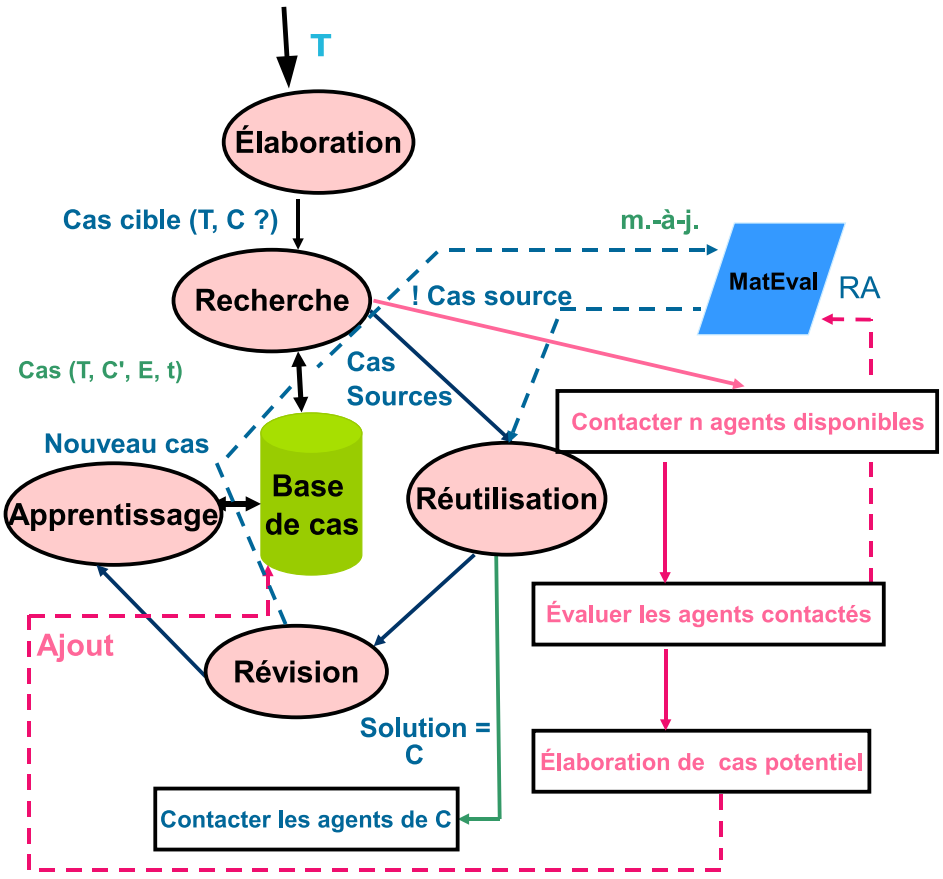

FIG. *4.10: Cycle RàPC pour la formation de comités*

Nous décrivons dans ce qui suit les différentes phases du système RàPC pour la formation de comités chez l'agent initiateur (voir figure 4.10). Nous commençons par la phase de recherche puisque la phase d'élaboration est directe. Notre problème cible est le thème d'intérêt T de la requête de recommandation.

1. ***La phase de recherche***
   La recherche de comités se base sur la mesure de similarité de thèmes (voir équation 4.3), elle compare le thème du cas cible (CC) à celui des cas sources (CS) dans la base de cas de comités de l'agent. Si cette similarité est au-dessus d'un certain seuil σ_t, alors le cas sera remémoré. Le résultat de cette phase est donc un ensemble de cas sources comportant un ensemble de comités d'agents (voir l'algorithme 7). Si la recherche n'aboutit pas à

des cas sources, alors le système va contacter un ensemble d'agents disponibles choisis aléatoirement et un ensemble d'agents qui lui ont été recommandés par rapport au thème T en consultant sa matrice d'évaluation correspondante. Cela permettra d'élaborer de nouveaux cas sources potentiels qui pourront être réutilisés pour des problèmes similaires.

Algorithme 7 Algorithme de remémoration de cas.

Entrée: T, BCC, CC ;
Sortie: un comité d'agents
  $com \leftarrow \emptyset$, $Thèmes\_similaires(T) \leftarrow \{T\}$;
  Pour chaque $CS_i \in BCC$ **Faire**
    // BCC : *Base de cas des comités*
    // Recherche dans la base de cas
    Si $Sim_{Topics}(CS_i.T, CC.T) \geq \sigma_t$ **Alors**
      $com \leftarrow com \cup \{CS_i.C\}$
      // Liste des thèmes à considérer dans le calcul des scores
      $Thèmes\_similaires(T) \leftarrow Thèmes\_similaires(T) \cup \{CS_i.T\}$
    Fin Si
  Fin Pour
  retourne *com*

2. ***La phase de réutilisation***
   Le but de cette phase consiste à calculer un sous-ensemble d'agents à partir de l'ensemble de cas sources retournés par la phase précédente et auxquels la requête de recommandation sera diffusée. Les étapes consistent à :
   – Déterminer les matrices d'évaluation à considérer à partir de la phase précédente. Nous considérons les matrices correspondant aux thèmes formant la liste des thèmes similaires trouvés ($Thèmes\_similaires(T)$).
   – Déterminer les agents candidats pour lesquels nous calculons le score total : en effet, au lieu de considérer toutes les colonnes des matrices trouvées (et donc tous les agents), nous considérons uniquement les colonnes correspondant aux agents des comités trouvés (*com* dans l'algorithme 7). Les agents d'une matrice qui auront leur score calculé sont les agents qui ont été recommandés par les agents du comité trouvé. Nous calculons ensuite les scores des agents trouvés par rapport au thème T. Ce même traitement va être effectué sur chaque matrice correspondant à un thème de la liste $Thèmes\_similaires(T)$. Finalement, nous calculons les scores totaux de tous les agents trouvés. La sélection d'un sous-ensemble d'agents peut se faire selon le nombre d'agents voulu et selon la valeur de leur score total en choisisant les n meilleurs par exemple. La demande de recommandation est envoyée aux agents choisis.

3. ***La phase de révision***

La solution retournée par l'étape précédente est évaluée par l'agent initiateur en observant la réaction de son utilisateur face aux références proposées par les agents contactés. Si l'utilisateur semble être intéressé par les recommandations fournies par quelques agents (par exemple, l'utilisateur ajoute quelques références dans sa base bibliographique), alors les agents qui ont participé à la solution seront bien évalués (notés). Il faut noter que cette évaluation peut se faire en plusieurs fois et pendant une période de temps pour donner plus de liberté à l'utilisateur.

Cette phase consiste à :

– choisir le comité d'agents à retenir parmi tous les agents contactés,
– élaborer un nouveau cas source.
– mettre à jour les matrices d'évaluation des agents.

L'évaluation des agents se fait par rapport à leur propre apport (références proposées) et par rapport aux apports des autres.

– Par rapport à leur apport : nous considérons la valeur de précision locale de chaque agent.
– Par rapport aux apports des autres : nous considérons la précision de tout le comité et nous comparons les apports des agents par rapport à ceux des autres.

La formation du comité retenu se fait comme suit : la première étape consiste à grouper tous les agents ayant une valeur de précision locale non nulle (ce sont les agents qui ont proposé au moins une bonne référence). Ensuite les agents sont classés selon leur valeur de précision. Finalement, le comité retenu sera formé par les n meilleurs agents classés. L'idée consiste à ne garder que les agents intéressants, qui sont bien évalués par rapport aux autres.

Une fois le comité formé, un nouveau cas (T, C, E, t) lui correspondant sera élaboré tel que :

– T : le thème de la requête envoyée.
– C : le comité des agents retenus.
– E : l'évaluation du comité qui présente sa valeur de précision. La précision du comité C est la moyenne des précisions des agents (équation 5.4 de la section 5.3.2) qui le forment.
– t : la date d'élaboration du nouveau cas source, qui sera très utile surtout pour la phase de maintenance à laquelle nous ne nous sommes pas intéressés pour le moment.

La mise à jour d'une matrice d'évaluation d'un agent i et correspondant à un thème T donné, chez l'agent initiateur j se fait comme suit :

– Si l'agent est déjà évalué alors sa valeur d'évaluation sera mise à jour comme suit : $e_{ij}^{t+1}(T) = (e_{ij}^{t}(T) + Nouvelle(e_{ij}(T)))/2$.
– Si l'agent n'est pas encore évalué alors sa valeur d'évaluation dans la matrice aura la

nouvelle valeur : $e_{ij}^{t+1}(T) = Nouvelle(e_{ij}(T))$.

4. ***La phase d'apprentissage***

   Lors ce cette phase, nous décidons si le nouveau cas source sera ajouté ou pas dans la base de cas. Deux cas sont possibles :

   – Si la phase de recherche n'a pas abouti à des cas sources, alors le nouveau cas source élaboré sera ajouté à la base de cas de l'agent si son évaluation est satisfaisante (E).
   – Si la phase de recherche a abouti à des cas sources et si le cas a été bien évalué par l'agent, alors :
     – si le cas existe déjà pour le même thème, il ne sera pas inséré dans la base de cas, mais le cas sera mis à jour en remplaçant les valeurs de E et de t par les nouvelles valeurs.
     – sinon, le cas sera ajouté à la base de cas.

   Une phase de maintenance serait nécessaire afin de mettre à jour la base de cas. Dans cette maintenance, les attributs E et t respectivement des pertinences des cas et leur date d'élaboration seront d'une grande utilité.

Remarque

Même si l'agent de recommandation ne trouve pas dans sa base des références à recommander à l'agent initiateur, il peut toujours être utile en envoyant non une recommandation de références mais une recommandation d'agents qu'il juge similaires et intéressants à l'agent initiateur. Donc, suite à une requête de recommandation de références, l'agent initiateur reçoit de chaque agent membre de son comité, deux types de recommandations : recommandation de références et recommandation de comité d'agents.

4.4 Conclusion

Nous avons vu dans ce chapitre tous les détails de fonctionnement du système COBRAS et la complémentarité des deux modules dont le but final est de fournir les meilleures recommandations à l'utilisateur. Ces recommandations sont calculées d'une manière implicite par les agents assistants personnels. Afin de valider nos approches proposées, nous avons implémenté le système COBRAS et nous avons effectué des expérimentations. C'est le sujet du chapitre suivant du manuscrit.

Chapitre 5

Mise en œuvre et validation

Dans ce chapitre, nous détaillons la mise en œuvre du système COBRAS dans la section 5.1. Ensuite, nous présentons l'expérimentation réalisée dans la section 5.2. Finalement, l'évaluation des approches proposées dans le système est détaillée dans la section 5.3.

5.1 Mise en œuvre

Nous décrivons dans cette partie tout le processus de mise en œuvre que nous avons effectué pour la validation des approches proposées dans notre système COBRAS. Nous avons choisi JADE comme plate-forme pour implémenter les agents de notre application. Nous commençons par présenter la structure d'un agent, ensuite nous présentons l'ensemble de données des agents : partagées et individuelles.

5.1.1 Structure d'un agent

Chaque pair est un agent de la plate-forme Jade (Java Agent DEvelopment Framework) [18] (Bellifemine *et al.*, 1999). Jade est une plate-forme multi-agents développée en Java et compatible FIPA (fondation d'agents physiques intelligents, formé en 1996 afin de construire des standards pour les agents hétérogènes, en interaction et les SMA, [19] (O'Brien et Nicol, 1998) qui facilite le développement de systèmes multi-agents).

La plate-forme multi-agents inclut trois agents fondamentaux qui sont automatiquement créés et activés quand la plate-forme est activée :

1. Le Système de Gestion d'Agents (Agent Management System - AMS) est l'agent qui exerce la supervision de l'accès et l'usage de la plate-forme ; il maintient une liste de tous les agents qui résident sur la plate-forme et contrôle l'accès ainsi que l'utilisation du canal de communication des agents (ACC).

[18] http ://jade.tilab.com/
[19] http ://www.fipa.org/

2. Le Canal de Communication entre Agents (Agent Communication Channel - ACC) est l'agent qui gère les communications entre agents. C'est la méthode de communication implicite qui offre un service fiable pour le routage des messages.
3. Le Facilitateur d'Annuaire (Directory Facilitator - DF) est l'agent qui fournit un service de pages jaunes à la plate-forme multi-agents. Il enregistre les descriptions des agents ainsi que les services qu'ils offrent. Les agents peuvent enregistrer leurs services auprès d'un DF ou demander à DF de découvrir les services offerts par d'autres agents (pages jaunes). Un service est caractérisé par son nom et son type.

JADE nous offre les avantages suivants :
- L'enregistrement automatique d'agents dans le Système de Gestion d'Agents (AMS).
- Un certain nombre de DF (Facilitateurs d'Annuaire) compatibles FIPA.
- Une communication transparente par message de langage de communication d'agents (ACL), spécifié par le standard FIPA.
- Un service d'attribution de noms compatible FIPA ; quand la plate-forme est lancée, un agent obtient un identificateur unique (Globally Unique Identifier - GUID) contenant des informations telles que le nom de l'agent, le nom du contenaire, le numéro de port, etc..

JADE conçoit un agent comme étant un processus autonome et indépendant ayant une identité et ayant besoin de la communication (collaboration ou compétition) avec d'autres agents afin de réaliser ses tâches. Chaque agent a son propre cycle de vie, possède un ou plusieurs comportements (Behaviours) qui définissent ses actions et communique et interagit avec les autres agents. Cette communication est implémentée à travers le passage de messages asynchrones et en utilisant le langage de communication d'agent (FIPA ACL [20]).

De point de vue technique, un agent JADE est simplement une classe Java qui étend la classe de base « Agent ». Cela permet à l'agent d'hériter d'un comportement fondamental caché (qui traite toutes les tâches liées à la plate-forme, telles que l'enregistrement, la configuration, la gestion à distance, etc.), et un ensemble de méthodes qui peuvent être appelées pour implémenter les tâches spécifiques à l'agent, par exemple envoi des messages, utilisation des protocoles d'interaction standard, enregistrement sur plusieurs domaines, etc.

Nous avons utilisé la plate-forme Jade puisqu'elle nous facilite certaines fonctionnalités nécessaires pour notre système. En effet, Jade nous simplifie le problème de communication entre les agents et nous offre un service d'annuaire. Cela nous a permis de se concentrer sur le comportement interne d'un agent. Chaque agent de notre système est un agent Jade qui dipose d'une interface graphique afin de permettre à l'utilisateur de gérer sa base bibliographique et pour lui offrir des recommandations. JADE utilise l'abstraction « Behaviour » pour modéliser les tâches qu'un agent peut exécuter et les agents instancient leurs comportements selon leurs besoins et leurs capacités. Nous avons donc étendu la classe Agent et implémenté les tâches

[20] http ://www.fipa.org/repository/aclspecs.html

spécifiques de l'agent en instanciant la classe « Behaviour » et ses différentes sous-classes. En effet, chaque agent implémente deux cycles de RàPC sous forme de comportements principaux : le premier relatif à la formation de comités et le second relatif au calcul des références.

Chaque agent manipule un ensemble de données afin de réaliser ses tâches :

- ***Données partagées*** : ce sont des données communes à tous les agents notamment la hiérarchie de thèmes. Deux types de relations peuvent lier deux thèmes : relation de composition qui exploite la structure hiérarchique de l'arbre et une relation sémantique qui peut lier des thèmes appartenant aux mêmes branches ou à des branches différentes de l'arbre de thèmes.
- ***Données individuelles*** : ce sont des données propres à chaque agent, nous trouvons :
  1. La base de références bibliographiques.
  2. Les bases de cas relatives aux deux modules de RàPC.
  3. La base des évaluations des agents où sont stockées les matrices d'évaluation.

5.1.2 Ensemble de données individuelles d'un agent

Les données principales d'un agent présentent ses différentes bases qui sont toutes implémentées sous le format XML pour des raisons de clarté et de réutilisabilité. Nous présentons dans ce qui suit la structure de chaque base ainsi que son rôle :

- ***Base de références*** : présente la base de toutes les références bibliographiques d'un agent. Les références bibliographiques sont représentées par un schéma XML et peuvent être de plusieurs types comme le montre la figure 5.1. Chaque type de référence a ses propres attributs et doit être conforme à un schéma XML bien défini. En plus de cette base, nous avons prévu une base d'annotations. Les annotations peuvent porter sur des auteurs, des références ou même des conférences (voir figure 5.2). Un exemple de la structure d'une annotation relative à un auteur est donné dans la figure 5.3.
- ***Base de cas des références*** : c'est la base de cas utilisée par le système RàPC pour la recommandation de références. La base est conforme à un schéma XML où chaque élément, présente un cas, ayant la structure suivante :

  ```
  <case Id="">
  <problem Topic="" KWS="" Agent=""/>
  <solution TList="" KWL="" E="" t=""/>
  </case>
  ```

  Un exemple de cas dans la base de cas des références est présenté dans la figure 5.4. Dans les exemples donnés, l'attribut t n'apparaît pas puisque nous l'avons pas considéré dans cette expérimentation.

```
<xs:element name="references">
<xs:complexType>
<xs:choice minOccurs ="1" maxOccurs ="unbounded">
<xs:element ref= "unpublished" minOccurs ='0' maxOccurs ='unbounded'/>
<xs:element ref ="electronicdoc" minOccurs ='0' maxOccurs ='unbounded'/>
<xs:element ref ="misc" minOccurs ='0' maxOccurs ='unbounded'/>
<xs:element ref ="manual" minOccurs ='0' maxOccurs ='unbounded'/>
<xs:element ref ="technicalreport" minOccurs ='0' maxOccurs ='unbounded'/>
<xs:element ref ="brochure" minOccurs ='0' maxOccurs ='unbounded'/>
<xs:element ref ="memory" minOccurs ='0' maxOccurs ='unbounded'/>
<xs:element ref ="phdthesis" minOccurs ='0' maxOccurs ='unbounded'/>
<xs:element ref ="hdrthesis" minOccurs ='0' maxOccurs ='unbounded'/>
<xs:element ref ="article" minOccurs ='0' maxOccurs ='unbounded'/>
<xs:element ref ="book" minOccurs ='0' maxOccurs ='unbounded'/>
<xs:element ref ="inbook" minOccurs ='0' maxOccurs ='unbounded'/>
<xs:element ref ="incollection" minOccurs ='0' maxOccurs ='unbounded'/>
<xs:element ref ="inproceedings" minOccurs ='0' maxOccurs ='unbounded'/>
</xs:choice>
</xs:complexType>
</xs:element>
```

FIG. *5.1: Types de références bibliographiques*

```
<!--annotations base-->
<xs:element name ="annotations">
<xs:complexType>
<xs:choice>
<xs:element ref="AnotAut" minOccurs='0' maxOccurs='unbounded'/>
<xs:element ref="AnotRef" minOccurs='0' maxOccurs='unbounded'/>
<xs:element ref="AnotProc" minOccurs='0' maxOccurs='unbounded'/>
</xs:choice>
<xs:attribute name="id" type="xs:ID" use='required'/>
</xs:complexType>
</xs:element>
```

FIG. *5.2: Types d'annotations*

- ***Base de comités*** : c'est la base de comités d'agents qui sont sélectionnés par l'agent. Les comités sont organisés par thèmes. Chaque élément présente un cas contenant un thème, un comité d'agents intéressants par rapport à ce thème ainsi qu'une évaluation du comité. L'attribut t correspond à la date d'élaboration du cas.

  Un cas comité a donc la structure suivante :

  ```
  <case Id="">
  <problem Topic=""/>
  <solution Committee="" E="" t=""/>
  </case>
  ```

  Un exemple de cas dans la base de cas de comités est donné dans la figure 5.5.
- ***Base des matrices d'évaluation*** : cette base contient des matrices d'évaluation.

```
<xs:element name="AnotAut" type="TypeAnotAut"/>
<xs:complexType name="TypeAnotAut">
<xs:choice>
<xs:element name="Activity" type ="xs:string" minOccurs="0"
maxOccurs="unbounded" />
<xs:element name="Mail" type ="xs:string" minOccurs="0" maxOccurs="1" />
<xs:element name="Website" type ="xs:string" minOccurs="0" maxOccurs="1"
/>
<xs:element name="ResearchAreas" type ="xs:string" minOccurs="0"
maxOccurs="unbounded" />
<xs:element name="ResearchTeam" type ="xs:string" minOccurs="0"
maxOccurs="1"/>
<xs:element name="Others" type ="xs:string" minOccurs="0" maxOccurs="1"
/>
<xs:element ref="linkaut" minOccurs='0' maxOccurs='1' />
</xs:choice>
</xs:complexType>
```

FIG. *5.3: Structure d'une annotation d'auteur*

```
<CaseBase>
<case Id="7">
<problem Topic="/computing methodologies/artificial intelligence/learning"

KWS="learning, rules, learning classifiers, models, optimization"

Agent="Agentai00@aubergine.lipn.univ-paris13.fr:1099/JADE"/>

<solution TList="/computing methodologies/artificial intelligence/learning,
/computing methodologies/artificial intelligence/deduction and theorem
proving/logic programming, /computing methodologies/artificial
intelligence/natural language processing/machine translation, /computing
methodologies/symbolic and algebraic manipulation/algorithms"

KWL="constructive induction, learning, reinforcement learning, induction,
unsupervised data, prolog definitons, planning, automatic generation,
bayesian network, automating model acquisition, nonstationary environments,
propagation method"  E="0,735"  t= ""/>
</case>
...
</CaseBase>
```

FIG. *5.4: Extrait de la base de cas de références*

Chaque matrice correspond à un thème d'intérêt courant et a la structure suivante :

```
<Evaluations>
<MatEvalList>
<TopMat Topic=""  id="" />
<TopMat Topic=""  id="" />
...
```

```
<CommitteeBase>
<case Id="1">
<problem
Topic="/computer applications/physical sciences and
engineering/engineering"/>
<solution
Committee="Agentcc96@aubergine.lipn.univ-paris13.fr:1099/JADE,
                Agentre04@aubergine.lipn.univ-paris13.fr:1099/JADE,
                Agentre99@aubergine.lipn.univ-paris13.fr:1099/JADE,
                Agenteh03@aubergine.lipn.univ-paris13.fr:1099/JADE,
                Agentai01@aubergine.lipn.univ-paris13.fr:1099/JADE "

E=" 0,9"  t=""/>
</case>
...
</CommitteeBase>
```

FIG. *5.5: Extrait de la base de cas de comités*

```
</MatEvalList>
<Matrices>
<Matrix id="">
<Cell rowIndex="" columnIndex=""> </Cell>
<Cell rowIndex="" columnIndex=""> </Cell>
<Cell rowIndex="" columnIndex=""> </Cell>
...
</Matrix>
</Matrices>
</Evaluations>
```

Comme le montre l'extrait juste au-dessus, la base de matrices d'évaluation d'un agent est un document XML de racine l'élément « Evaluations ». Chaque matrice est relative à un thème : l'élément « TopMat ». Les différents éléments « Cell » d'une matrice correspondent aux évaluations e_{ij}. La matrice est implémentée de telle façon qu'une nouvelle ligne sera créée dès qu'une nouvelle entrée se présente (e_{ij}) pour optimiser l'espace mémoire et la recherche. Un exemple de matrice d'évaluation correspondant à l'agent « Agentai03 » et relatif au thème « Learning » est donné dans la figure 5.6. Dans cet exemple, l'évaluation de l'agent « Agentai03 » par l'agent « Agentos74 » est égale à -1, alors que celle attribuée par l'agent « Agenteh05 » est de $0,59$.

```
<?xml version="1.0" encoding="UTF-8"?>
<Evaluations>
<MatEvalList>
<TopMat Topic="/computing methodologies/artificial intelligence/learning" id="0" />
</MatEvalList>
<Matrices>
<Matrix id="0">
<Cell rowIndex="Agentai03@aubergine.lipn.univ-paris13.fr:1099/JADE"
 columnIndex="Agentos74@aubergine.lipn.univ-paris13.fr:1099/JADE">-1.0</Cell>
<Cell rowIndex="Agentai03@aubergine.lipn.univ-paris13.fr:1099/JADE"
 columnIndex="Agentih02@aubergine.lipn.univ-paris13.fr:1099/JADE">-1.0</Cell>
<Cell rowIndex="Agentai03@aubergine.lipn.univ-paris13.fr:1099/JADE"
 columnIndex="Agenteh05@aubergine.lipn.univ-paris13.fr:1099/JADE">0.59</Cell>
<Cell rowIndex="Agentai03@aubergine.lipn.univ-paris13.fr:1099/JADE"
columnIndex="Agentcg00@aubergine.lipn.univ-paris13.fr:1099/JADE">0.92</Cell>
<Cell rowIndex="Agentai03@aubergine.lipn.univ-paris13.fr:1099/JADE"
columnIndex="Agenteh00@aubergine.lipn.univ-paris13.fr:1099/JADE">-1.0</Cell>
<Cell rowIndex="Agentcg00@aubergine.lipn.univ-paris13.fr:1099/JADE"
columnIndex="Agentai03@aubergine.lipn.univ-paris13.fr:1099/JADE">0.0</Cell>
<Cell rowIndex="Agenteh05@aubergine.lipn.univ-paris13.fr:1099/JADE"
columnIndex="Agentai03@aubergine.lipn.univ-paris13.fr:1099/JADE">0.0</Cell>
</Matrix>
...
</Matrices>
</Evaluations>
```

FIG. *5.6: Exemple de matrice d'évaluation*

5.2 Expérimentation

Nous détaillons dans cette section toutes les étapes éffectuées afin de réaliser nos expérimentations. Nous commençons par décrire le scénarios d'utilisation du système dans la section 5.2.1. Ensuite, nous passons dans la section 5.2.2 à la description des données utilisées et aux différents traitements effectués afin de réaliser l'expérimentation. Enfin, nous décrivons l'ensemble de données considéré dans l'expérimentation dans la section 5.2.3.

5.2.1 Scénario d'utilisation du système

Dans COBRAS, chaque agent est spécialisé dans une édition de conférence : il dispose de toutes les références associées à cette édition de conférence (par exemple ai05, re83, asc00, etc.). Dans notre simulation, nous supposons que le thème d'intérêt courant d'un agent est le thème qui dispose du plus grand nombre de références dans la base de l'agent en question. Chaque agent du système formule une requête (demande de recommandation) associée à son thème d'intérêt courant et contacte certains agents pour obtenir des recommandations. Initialement, comme toutes les bases de cas de notre système sont vides, chaque agent envoie sa requête à un sous-ensemble d'agents choisis aléatoirement (x% du nombre d'agents disponibles). Ensuite,

chaque agent utilise son module RàPC afin de trouver son comité. Si le système retourne des cas sources, alors le comité sera formé par les agents trouvés et les agents recommandés pour ce thème. Das le cas inverse, le comité sera formé par des agents choisis aléatoirement et des agents recommandés. Notre but consiste à évaluer les performances globales du système en fonction du nombre de lancements d'une même requête pour chaque agent (cycles).
La requête de chaque agent sera donc envoyée plusieurs fois, le système va utiliser ses deux composants RàPC pour traiter cette requête. Nous mesurons à chaque fois les valeurs du rappel et de la précision du système pour ses deux composants (pour la recommandation de références et pour la formation de comité d'agents) et nous évaluons le système en se basant sur ces critères. Les valeurs du rappel et de la précision du système présentent les moyennes des valeurs des rappels (respectivement des précisions) locaux de tous les agents du système. Pour pouvoir calculer ces valeurs, nous devons dans un premier temps et pour chaque agent, envoyer la requête à tous les agents du système afin de déterminer la liste des agents intéressants et la liste de toutes les bonnes références pour chaque agent. Cela veut dire que dans ce premier test, tout le monde contacte tout le monde, c'est-à-dire chaque agent envoie sa requête aux 99 autres agents. Ensuite nous calculons, à chaque lancement de requête, les valeurs des critères choisis, de chaque agent, puis celles du système afin de l'évaluer. Les critères sont définis dans la section 5.3.

5.2.2 Description des données

Afin de réaliser notre expérimentation, nous nous sommes basés sur les données de la base *DBLP* (Digital Bibliography and Library Project) [21] qui correspond à notre type d'application puisqu'elle traite des données présentant des références bibliographiques et qui sont au format XML. Cette base se compose essentiellement d'un ensemble de conférences (proceedings au nombre de 7.919) et des actes de conférences (inproceedings au nombre de 481.000). Comme le montre la figure 5.7, nous distinguons trois niveaux de données :

1. ***Les conférences*** : par exemple, la conférence d'intelligence artificielle qui est notée « ai ».
2. ***Les éditions de conférence*** : chaque conférence a plusieurs éditions au cours des années. Par exemple, « ai2002 » et « ai1997 » sont des éditions de la conférence « ai ».
3. ***Les références*** : qui présentent l'ensemble des actes des différentes éditions de conférences.

Nous avons organisé les références en plusieurs bases rangées par conférence puis par édition de conférence. Ensuite, afin de simuler le rôle de l'utilisateur, nous avons attribué des thèmes et des mots-clés à ces références pour mieux les décrire et donc les retrouver.

[21] http ://dblp.uni-trier.de/

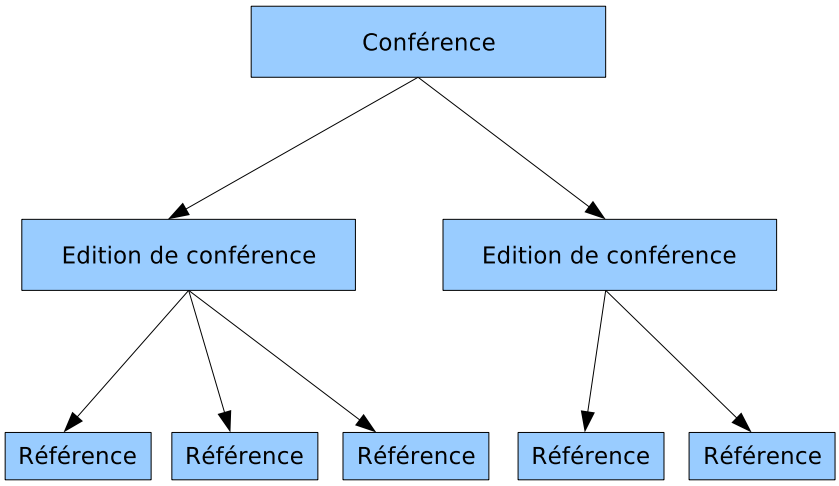

FIG. *5.7: Hiérarchie des données*

5.2.2.1 Extraction et organisation des données

Nous avons extrait nos données de la base DBLP puis nous les avons organisées en formant deux types de bases XML : la base des éditions de conférences et la base des actes de conférences (ou références bibliographiques). Par exemple, la base nommée « aiBase » contient toutes les éditions de conférences relatives à la conférence « ai ». La base de références « aiBase01.xml » contient tous les actes de conférences qui correspondent à la conférence « ai » pour l'édition 2001. Un exemple de référence bibliographique est décrit dans la figure 5.8.

```
<inproceedings mdate="2002-12-11" key="conf/dl/PaynterWCB00">
<author>Gordon W. Paynter</author>
<author>Ian H. Witten</author>
<author>Sally Jo Cunningham</author>
<author>George Buchanan</author>
<title>Scalable browsing for large collections: a case study.</title>
<pages>215-223</pages>
<year>2000</year>
<booktitle>ACM DL</booktitle>
<ee>http://doi.acm.org/10.1145/336597.336666</ee>
<url>db/conf/dl/dl2000.html#PaynterWCB00</url>
</inproceedings>
```

FIG. *5.8: Exemple de référence bibliographique*

5.2.2.2 Organisation de la hiérarchie ACM

Nous avons utilisé la hiérarchie ACM (Association for Computing Machinery) [22] pour classifier par thèmes les références bibliographiques. Cette hiérarchie présente un ensemble de 1.474 thèmes sous un format XML. Chaque thème représente un nœud dans l'arbre qui dispose d'un identifiant unique (#id) et d'un nom (#label). Chaque thème peut être composé de plusieurs sous-thèmes. Cette relation est exprimée par l'élément « isComposedBy » dans la hiérarchie ACM. En sus de cette relation de composition, des thèmes peuvent être liés entre eux sémantiquement. Ce type de relation est exprimé par l'élément « isRelatedTo ». Un extrait de la hiérarchie de thèmes ACM d'origine est illustré dans la figure 5.9.

```
<?xml version="1.0" encoding="UTF-8"?>
<node id="acmccs98" label="ACMCCS98">
<isComposedBy>
<node id="A." label="General Literature">
<isComposedBy>
<node id="A.0" label="GENERAL">
<isComposedBy>
<node label="Biographies/autobiographies"/>
<node label="Conference proceedings"/>
<node label="General literary works (e.g., fiction, plays)"/>
</isComposedBy>
</node>
<node id="A.1" label="INTRODUCTORY AND SURVEY"/>
<node id="A.2" label="REFERENCE (e.g., dictionaries, encyclopedias,
glossaries)"/>
<node id="A.m" label="MISCELLANEOUS"/>
</isComposedBy>
</node>
<node id="B." label="Hardware">
<isComposedBy>
<node id="B.0" label="GENERAL"/>
<node id="B.1" label="CONTROL STRUCTURES AND MICROPROGRAMMING">
<isRelatedTo>
<node id="D.3.2"/>
</isRelatedTo>
<isComposedBy>
<node id="B.1.0" label="General"/>
<node id="B.1.1" label="Control Design Styles">
<isComposedBy>
<node label="Hardwired control">
...
</node>
```

FIG. *5.9: Extrait de la hiérarchie ACM*

Dans la hiérarcgie ACM, nous avons constaté qu'un même label peut exister plusieurs fois même si les chemins qui y mènent ne sont pas les mêmes. Afin d'accélérer la recherche et pour ne pas calculer le chemin des thèmes à chaque fois, nous avons construit, à partir de cette

[22] http ://www.acm.org/

hiérarchie, une autre base de thèmes où chaque élément présente à la fois le label du thème ainsi que son chemin dans la hiérarchie. Ce chemin permet ainsi l'identification unique d'un thème dans l'arbre. Il est construit à partir des relations de composition : la relation « isComposedBy » de la hiérarchie ACM. Un extrait de cette base modifiée est décrit dans la figure 5.10.

```
<HierarchicalTopics>
<label path="/General Literature">General Literature</label>
<label path="/Hardware">Hardware</label>
<label path="/Computer Systems Organization">Computer Systems
Organization</label>
<label path="/Software">Software</label>
<label path="/Data">Data</label>
<label path="/Theory of Computation">Theory of Computation</label>
<label path="/Mathematics of Computing">Mathematics of Computing
</label>
<label path="/Information Systems">Information Systems</label>
<label path="/Computing Methodologies">Computing
Methodologies</label>
<label path="/Computer Applications">Computer Applications</label>
<label path="/Computing Milieux">Computing Milieux</label>
<label path="/Hardware/CONTROL STRUCTURES AND
MICROPROGRAMMING">
CONTROL STRUCTURES AND MICROPROGRAMMING</label>
<label path="/Hardware/ARITHMETIC AND LOGIC STRUCTURES">
ARITHMETIC AND LOGIC STRUCTURES</label>
<label path="/Hardware/MEMORY STRUCTURES">MEMORY STRUCTURES
</label>
<label path="/Hardware/CONTROL STRUCTURES AND MICROPROGRAMMING/
Control Design Styles">Control Design Styles</label>
<label path="/Hardware/CONTROL STRUCTURES AND MICROPROGRAMMING/
Control Structure Performance Analysis and Design Aids">
Control Structure Performance Analysis and Design Aids</label>
<label path="/Hardware/CONTROL STRUCTURES AND MICROPROGRAMMING/
Control Structure Reliability, Testing, and Fault-Tolerance">
Control Structure Reliability, Testing, and Fault-Tolerance</label>
...
</HierarchicalTopics>
```

FIG. *5.10: Extrait de la hiérarchie ACM modifiée*

Dans l'exemple suivant, le thème a comme label *Microprogram Design Aids* et comme chemin */Hardware/Control Structures and Microprogramming/Microprogram Design Aids.*

```
<label path="/Hardware/Control Structures and Micro-programming/
Microprogram Design Aids"> Microprogram Design Aids
</label>
```

Quant aux relations sémantiques qui lient ces thèmes, nous les avons groupées dans une base nommée « RelatedTopics ». Un extrait de cette base est décrit dans la figure 5.11.

```
RelatedTopics>
<label path="/Hardware/CONTROL STRUCTURES AND
MICROPROGRAMMING">
CONTROL STRUCTURES AND MICROPROGRAMMING
<isRelatedTo>
<label path="/Software/PROGRAMMING LANGUAGES/
Language Classifications">Language Classifications</label>
</isRelatedTo>
</label>
<label path="/Hardware/CONTROL STRUCTURES AND MICROPROGRAMMING/
Control Structure Reliability, Testing, and Fault-Tolerance">
Control Structure Reliability, Testing, and Fault-Tolerance
<isRelatedTo>
<label path="/Hardware/PERFORMANCE AND RELIABILITY">
PERFORMANCE AND RELIABILITY</label>
</isRelatedTo>
</label>
<label path="/Hardware/CONTROL STRUCTURES AND MICROPROGRAMMING/
Microprogram Design Aids">Microprogram Design Aids
<isRelatedTo>
<label path="/Software/SOFTWARE ENGINEERING/Design Tools and
Techniques">Design Tools and Techniques</label>
<label path="/Software/SOFTWARE ENGINEERING/Software Program
Verification">Software Program Verification</label>
<label path="/Software/PROGRAMMING LANGUAGES/Language
Classifications">Language Classifications</label>
<label path="/Hardware/INPUT/OUTPUT AND DATA COMMUNICATIONS/Data
Communications Devices/Processors"> Data Communications Devices/
Processors</label>
</isRelatedTo>
</label>
...
</RelatedTopics>
```

FIG. *5.11: Extrait de la base des thèmes en relation*

5.2.2.3 Classification des données par thèmes de la hiérarchie ACM

Pour pouvoir réaliser le scénario, nous avons besoin des références qui soient classifiées par thème et décrites par mots-clés. Cette tâche est effectuée par l'utilisateur. Pour simuler cela, nous avons proposé un algorithme de classification automatique des références par thèmes et un autre plus simple pour l'attribution de mots-clés aux références.
Nous avons commencé tout d'abord par la classification des conférences par thèmes selon la hiérarchie ACM.

La classification des conférences est faite selon un algorithme simple qui consiste à faire une mise en correspondance entre le titre d'une conférence et la hiérarchie de thèmes ACM. Nous avons réussi à classifier 5.513 conférences, soit $69,61\%$ de l'ensemble. Comme nous allons considérer, dans notre expérimentation, qu'un sous-ensemble de données de la base DBLP, nous avons choisi, à partir de cet ensemble classifié, les 100 premiers types de conférences, soit

629 éditions de conférences et 32.624 références. La deuxième étape consiste à classifier les références de cet ensemble sélectionné par thèmes, toujours selon la hiérarchie ACM puis à les valider (algorithme 8). Le principe consiste tout d'abord à trouver la liste de thèmes candidats en faisant une correspondance entre le titre de la référence et les thèmes de la hiérarchie. Ensuite, nous passons à la validation des thèmes candidats trouvés. Une première étape consiste à filtrer les thèmes candidats selon leur label. En effet, si un label est inclus dans un autre, nous le supprimons afin de ne garder que les thèmes les plus spécifiques. La deuxième étape consiste à valider les thèmes trouvés en utilisant cette fois le titre de l'édition de conférence auquelle appartient la référence (voir algorithme 8).

Les résultats obtenus sont décrits dans le tableau 5.1.

Nombre total d'éditions de conférences	7.919
Nombre d'éditions de conférences classifiées	5.513 (69, 61%)
Nombre de conférences sélectionnées	100
Nombre d'éditions de conférences associées	629
Nombre de références associées	32.624
Nombre de références classifiées	19.456 (60%)

TAB. *5.1: Résultats de la classification*

5.2.2.4 Association de mots-clés aux références

Une fois, l'ensemble de références est classifié par thèmes, l'étape suivante consiste à attribuer des mots-clés aux références pour mieux les décrire. Pour réaliser cela, nous avons adopté un algorithme très simple qui permet d'extraire des groupes nominaux à partir du titre d'une référence. Les mots-clés associés à une référence sont les groupes nominaux ainsi trouvés. Cet algorithme nous garantit au moins une description objective des références.

Un exemple de référence classifiée par thèmes et décrite par mots-clés est donné dans la figure 5.12.

5.2.3 Notre ensemble de données

Nous avons choisi de prendre un petit ensemble de données pour réaliser nos expérimentations et valider les approches proposées. Notre ensemble de données est une base de données XML contenant 2.162 références bibliographiques regroupant 100 éditions de conférences qui appartiennent à 21 conférences différentes. Ce sous-ensemble est choisi aléatoirement mais tout en gardant le nombre d'éditions de conférences réel pour chaque conférence afin de refléter le

Algorithme 8 Algorithme de classification des références par thèmes ACM et leur validation.

Entrée: ACMlabels ← {labels des thèmes dans la hiérarchie ACM}
ListeRéférences : liste de références à classifier
Sortie: liste de références classifiées par thèmes

```
Pour tout (référence ∈ ListeRéférences) Faire
  // ThèmesValidés : liste de thèmes validés de la référence, initialisée à vide
  ThèmesValidés← ∅
  // Attribution de thèmes
  Titre ← {mots du titre de la référence}
  TitreProceedings ← {mots du titre de l'édition de la référence}
  // Mise en correspondance des titres et de la hiérarchie ACM
  Labels ← Titre ∩ ACMlabels
  //Validation de thèmes
  Pour tout l ∈ Labels Faire
    //Filtrage initial selon les labels
    Si (∃ k ∈ Labels tel que l ⊆ k) Alors
      Labels ← Labels \ {l}
    Fin Si
  Fin Pour
  Pour tout l ∈ Labels Faire
    // Détermination de l'ensemble des chemins menant au label
    // qui forme un ensemble de thèmes candidats
    l.chemins ← TrouverChemins(racine, l)
  Fin Pour
  Pour tout l ∈ Labels Faire
    Pour tout c ∈ l.chemins Faire
      c.elements ← {labels \ {l}}
      // c.elements est l'ensemble de labels composant le chemin c
      Si (c.elements ∩ TitreProceedings ≠ ∅) Alors
        ThèmesValidés ← ThèmesValidés ∪ {(l, c)} ;
        Validation ← true
      Fin Si
    Fin Pour
    Si (Validation == false) Alors
      // Filtrage selon la longueur du chemin
      ThèmesValidés ← ThèmesValidés ∪ {(l, c ∈ l.chemins tel que longueur(c) =
      min(longueur(l.chemins)))}
    Fin Si
Fin Pour
  // Affectation des thèmes validés à la référence
  référence.thèmes ← ThèmesValidés
Fin Pour
```

```
<inproceedings mdate="2002-12-11" key="conf/dl/PaynterWCB00">
<author>Gordon W. Paynter</author>
<author>Ian H. Witten</author>
<author>Sally Jo Cunningham</author>
<author>George Buchanan</author>
<title>Scalable browsing for large collections: a case study.</title>
<pages>215-223</pages>
<year>2000</year>
<booktitle>ACM DL</booktitle>
<ee>http://doi.acm.org/10.1145/336597.336666</ee>
<url>db/conf/dl/dl2000.html#PaynterWCB00</url>
<keywords>scalable browsing, large collections, case study</keywords>
<topics>/information systems/information storage and retrieval/digital
libraries/collection</topics>
</inproceedings>
```

FIG. *5.12: Exemple de référence bibliographique complétée*

plus possible le cas réel (voir tableau 5.2).

5.3 Évaluation

Nous consacrons cette section à l'évaluation du système. Nous commençons par définir les concepts utilisés dans l'expérimentation et les critères d'évaluation considérés. Ensuite, nous présentons et nous interprétons les résultats obtenus concernant les deux modules : le module de formation de comités et le module de recommandation de références. Enfin, nous terminons par une synthèse.

5.3.1 Définition des concepts

Nous donnons dans cette partie les définitions concrètes des concepts utilisés dans l'expérimentation.

- ***Thème d'intérêt courant*** : c'est un thème d'intérêt qui est déterminé par l'agent selon l'algorithme décrit dans la section 4.1.2. Dans notre expérimentation, nous considérons un thème d'intérêt courant d'un agent, le thème qui dispose du plus grand nombre de références associées dans la base de références bibliographiques de l'agent.
- ***Bonne référence*** : c'est une référence jugée pertinente par l'utilisateur. Dans cette expérimentation, nous considérons qu'une bonne référence est une référence qui correspond à la requête et qui vérifie donc un certain niveau de similarité. Nous avons utilisé la fonction de similarité entre une référence r et une requête Q, que nous avons définie dans le chapitre 4, section 4.2.4, équation 4.5. Elle dépend de la similarité de thèmes et de la

Nom de la conférence	**Conférence**	**Nombre d'éditions**	**Nombre de références classifiées**
Artificial Intelligence	ai	9	237
Algebraic Methods	am	2	19
Artificial Intelligence and Soft Computing	asc	1	48
Compiler Construction	cc	15	187
Computers and Games	cg	4	37
Computational Logic	cl	1	54
Digital Libraries	dl	7	165
Data Base Management Systems	ds	8	218
Evolvable Hardware	eh	7	156
Electronic Publishing	ep	1	32
Intrusion Detection and Network Monitoring	id	1	5
Information Hiding	ih	7	92
Integrated Network Management	im	4	219
Logic Programming	lp	5	52
Machine Intelligence	mi	2	38
Optimal Algorithms	oa	1	12
Neural Networks	nn	3	30
Performance Engineering	pe	1	21
Pacific Conference on Computer Graphics and Applications	pg	8	226
Operating Systems	os	2	18
Requirements Engineering	re	11	294

TAB. *5.2: Détails sur l'ensemble de données de l'expérimentation*

similarité de mots-clés. Nous avons fixé le seuil de similarité à la valeur $0,5$.

– ***Agent intéressant*** : c'est un agent qui recommande de bonnes références. Les agents intéressants ne sont pas de même importance ; certains agents intéressants peuvent être plus importants que d'autres suivant leur précision ou leur rappel.

5.3.2 Critères d'évaluation du système

Afin d'évaluer notre système, nous considérons plusieurs critères qui décrivent la qualité de la solution et les performances du système.

1. **La qualité de la solution** : présentée par le rappel et la précision et concerne les deux modules RàPC : module de recommandation de références et module de la formation de comités d'agents.

   (a) **Pour le module de recommandation de références**

   – ***Le rappel*** : le rappel de recommandation de références du système est la moyenne des rappels locaux des agents du système. Le rappel local de recommandation de références d'un $agent_i$ représente le taux de bonnes références recommandées par $l'agent_i$ parmi toutes les bonnes références qui existent chez tous les agents du système (les n agents).

$$RappelRef_{Sys} = Moyenne(RappelRef_{Local_i}), i = 1..n \tag{5.1}$$

$$RappelRef_{Local_i} = \frac{|(Bonnes\_références\_recommandées)_i|}{|(Bonnes\_références)|} \tag{5.2}$$

   – ***La précision*** : la précision de recommandation de références du système est la moyenne des précisions locales des agents du système. La précision locale de recommandation de références d'un $agent_i$ représente le taux de bonnes références recommandées par l'$agent_i$ parmi toutes les références qu'il a recommandées.

$$PrécisionRef_{Sys} = Moyenne(PrécisionRef_{Local_i}), i = 1..n \tag{5.3}$$

$$PrécisionRef_{Locale_i} = \frac{|(Bonnes\_références\_recommandées)_i|}{|(Toutes\_les\_références\_recommandées)_i|} \tag{5.4}$$

   (b) **Pour le module de la formation de comités**

– ***Le rappel*** : le rappel de recommandation d'agents du système est la moyenne des rappels agents locaux des agents du système. Le rappel local de recommandation d'agents d'un $agent_i$ représente le taux d'agents intéressants contactés par l'$agent_i$ parmi tous les agents intéressants qui existent dans le système.

$$RappelAg_{Sys} = Moyenne(RappelAg_{Local_i}), i = 1..n \tag{5.5}$$

$$RappelAg_{Local_i} = \frac{|(Agents\_contactés\_intéressants)_i|}{|(Agents\_intéressants)|} \tag{5.6}$$

– ***La précision*** : la précision de recommandation d'agents du système est la moyenne des précisions locales des agents du système. La précision locale de recommandation d'agents d'un $agent_i$ représente le taux d'agents intéressants contactés par l'$agent_i$ parmi tous les agents qu'il a contactés.

$$PrécisionAg_{Sys} = Moyenne(PrécisionAg_{Locale_i}), i = 1..n \tag{5.7}$$

$$PrécisionAg_{Locale_i} = \frac{|(Agents\_contactés\_intéressants)_i|}{|(Tous\_les\_agents\_contactés)_i|} \tag{5.8}$$

Remarque
Nous rappelons que l'ensemble de toutes les bonnes références et l'ensemble de tous les agents intéressants pour un $agent_i$ sont calculés lors d'un premier test qui consiste à envoyer la demande de recommandation de l'$agent_i$ à tous les agents du système.

2. **Les performances du système** : sont obtenus par la taille relative du comité et le nombre de messages.
   – ***Taille relative du comité*** : est le nombre d'agents contactés par rapport à tous les agents disponibles.

$$Taille_{relative}\_Comité = \frac{|Agents\_contactés|}{|Tous\_les\_agents\_disponibles|} \tag{5.9}$$

   – ***Nombre de messages*** : est le nombre moyen de messages échangés pour une requête utilisateur. Ce nombre est égal à deux fois la taille moyenne du comité. La taille moyenne du comité présente la moyenne des tailles des comités des agents du système. La taille du comité d'un $agent_i$ est égale au nombre d'agents contactés par l'$agent_i$, pour une requête donnée.

5.3.3 Résultats et interprétations

Les résultats du premier test (c.-à-d. tout le monde contacte tout le monde) montrent que pour un agent donné, les agents intéressants peuvent appartenir à une même conférence, à des conférences totalement différentes ou à une combinaison des deux. Cela montre que des conférences et des éditions de conférences peuvent être jugées similaires, non pas à travers leur propres thèmes mais à travers leurs références associées. Ainsi, des éditions de conférences appartenant à des conférences différentes peuvent être plus similaires entre elles que des éditions de conférences appartenant à une même conférence. Nous notons également que le nombre de bonnes références et d'agents intéressants pour chaque agent varient énormément d'un agent à un autre. Par exemple, nous avons trouvé que le nombre d'agents intéressants varie entre 0 et 81 agents et le nombre de bonnes références varie entre 0 et 1.992.

5.3.3.1 Description

1. ***Rappel et Précision***
   - Alternative 1 : dans ce cas, nous avons lancé l'expérimentation (comme le décrit le scénario de la section 5.2.1) avec x (le nombre d'agents aléatoires) égal à 10. Initialement, chaque agent envoie donc sa requête à 10 agents au hasard. Les résultats obtenus sont décrits dans la figure 5.13 pour le module de recommandation de références et dans la figure 5.14 pour le module de la formation de comités. Nous remarquons que la valeur de la précision est importante pour les deux modules : elle est supérieure à $0,7$ pour le module de calcul des références et supérieure à $0,9$ pour le module de formation de comités au bout du $5^{ème}$ cycle. Cependant la valeur du rappel est assez faible pour les deux modules. Cela veut dire que le système est précis et propose donc à l'utilisateur des bonnes références, évitant ainsi de le submerger par des références inutiles et de réduire l'effort supplémentaire de choix des références parmi l'ensemble proposé. Les rappels et les précisions augmentent en général en fonction du nombre de cycles. Ainsi, le système apprend petit à petit à former ses comités et à trouver les références pertinentes.
   - Alternative 2 : comme le nombre d'agents intéressants varie énormément d'un agent à un autre, nous avons fixé dans ce cas la valeur de x à la valeur de la médiane, c'est-à-dire 28. Les nouveaux résultats obtenus sont décrits dans la figure 5.15 pour le module de recommandation de références et dans la figure 5.16 pour le module de la formation de comités. Nous remarquons pour les deux modules que la valeur du rappel s'est améliorée et a plus que doublé (figure 5.17). Quant à la précision, elle est toujours très importante.
2. ***Taille du comité et nombre de messages***
   Nous considérons dans cette partie la taille relative du comité et le nombre de messages

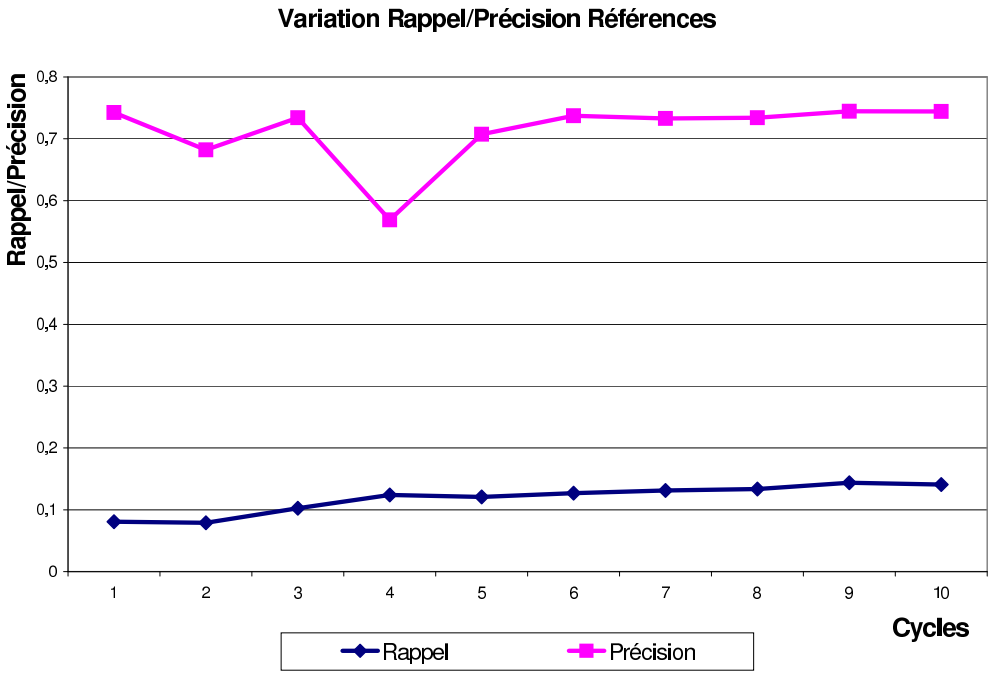

FIG. *5.13: Rappels et précisions du module de recommandation de références*

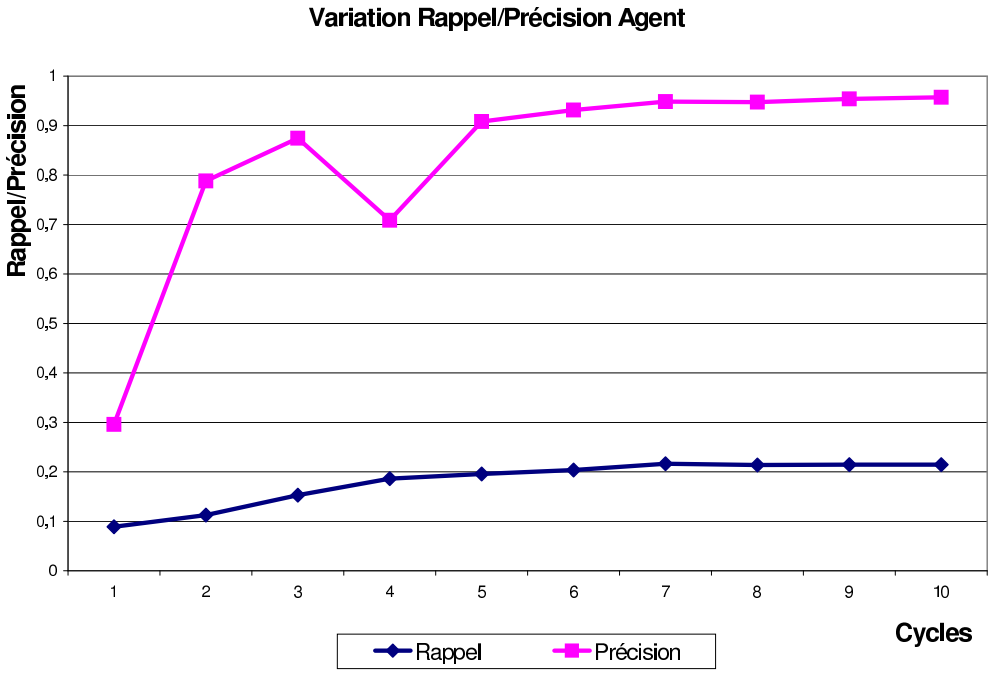

FIG. *5.14: Rappels et précisions du module de la formation de comités*

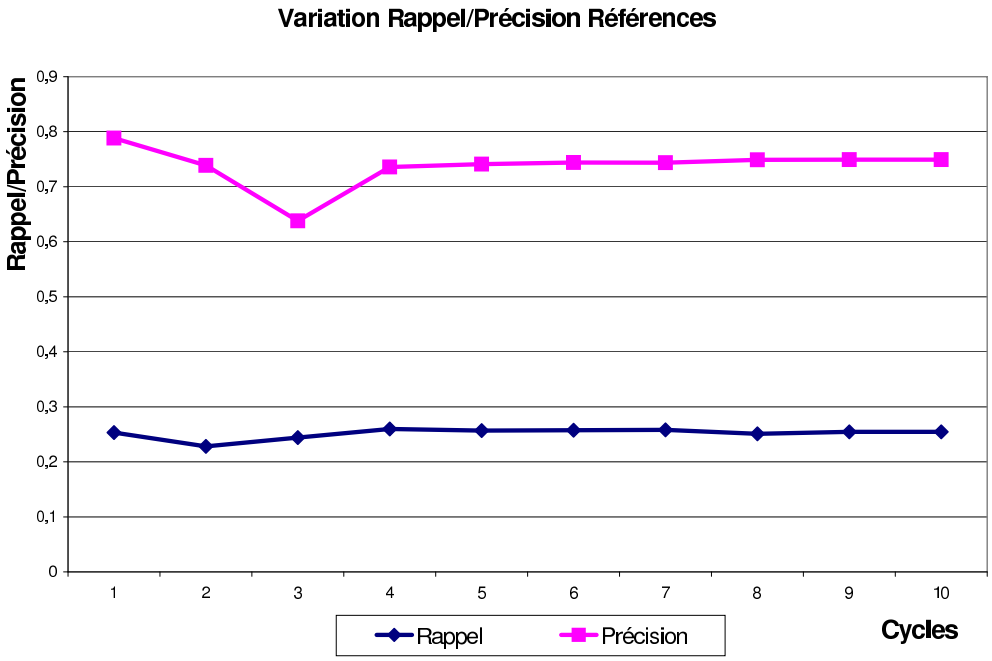

FIG. *5.15: Rappels et précisions du module de recommandation de références*

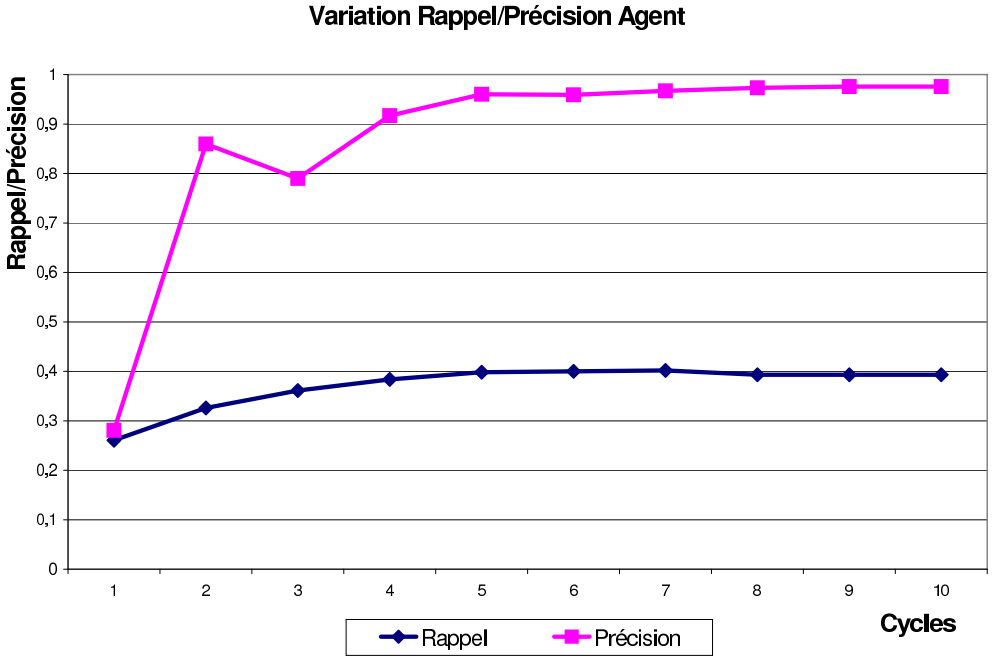

FIG. *5.16: Rappels et précisions du module de la formation de comités*

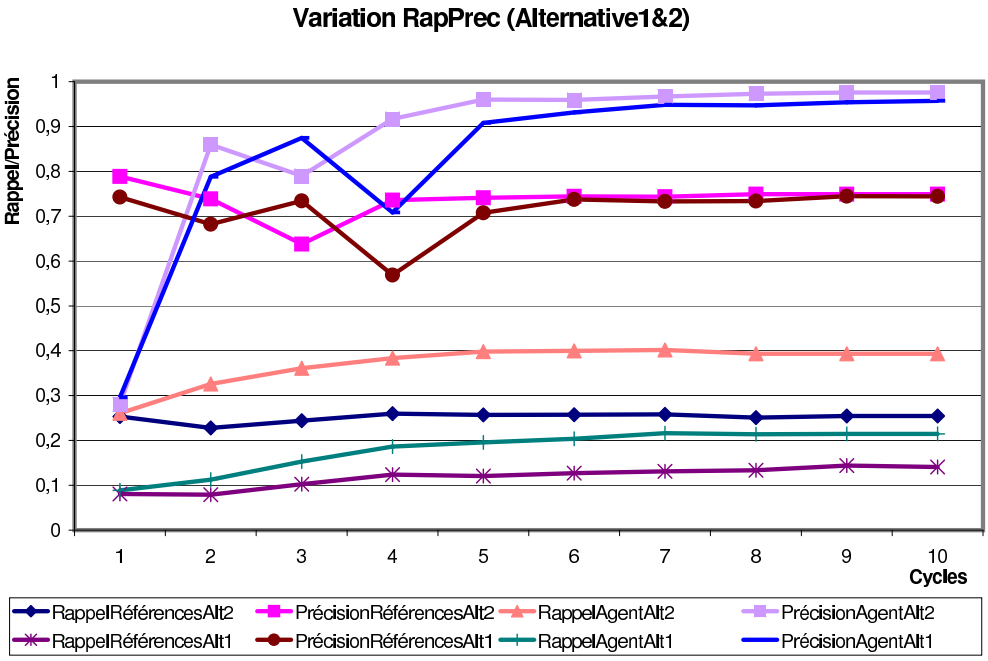

FIG. *5.17: Variation du rappel et de la précision des deux modules*

pour les deux alternatives (1 et 2). La figure 5.18 montre l'évolution de la taille du comité au fur et à mesure du lancement de la requête pour chaque agent. Cette taille est importante au début et se stabilise au fur et à mesure puisque l'agent apprend petit à petit à former son comité en ne recontactant que les agents intéressants. La taille relative du comité obtenue avec l'alternative 2 (environ 10%) est beaucoup plus importante que celle obtenue avec l'alternative 1 (environ 5%) mais le comité de l'alternative 2 est plus intéressant que celui de l'alternative 1 puisqu'il contient plus d'agents intéressants (voir précision et rappel de la recommandation d'agents). Le nombre de messages échangés pour les deux alternatives est décrit dans la figure 5.19. Nous remarquons que le nombre de messages est globalement faible pour les deux alternatives. Il est initialement important mais se stabilise rapidement avec le nombre de cycles. Le nombre de messages est plus important avec l'alternative 2 puisqu'elle sollicite plus d'agents mais qui sont intéressants.

5.3.3.2 Analyse des résultats

Puisque le groupe d'agents considéré n'est pas très homogène en terme de nombre d'agents intéressants et de nombre de bonnes références pour chaque agent, nous avons analysé les effets de ces différences sur les performances moyennes du système. Pour cela, nous avons divisé l'ensemble des agents en cinq groupes selon la « popularité » de leur thème d'intérêt en se basant sur leur nombre d'agents intéressants. Les cinq groupes ainsi formés sont décrits dans

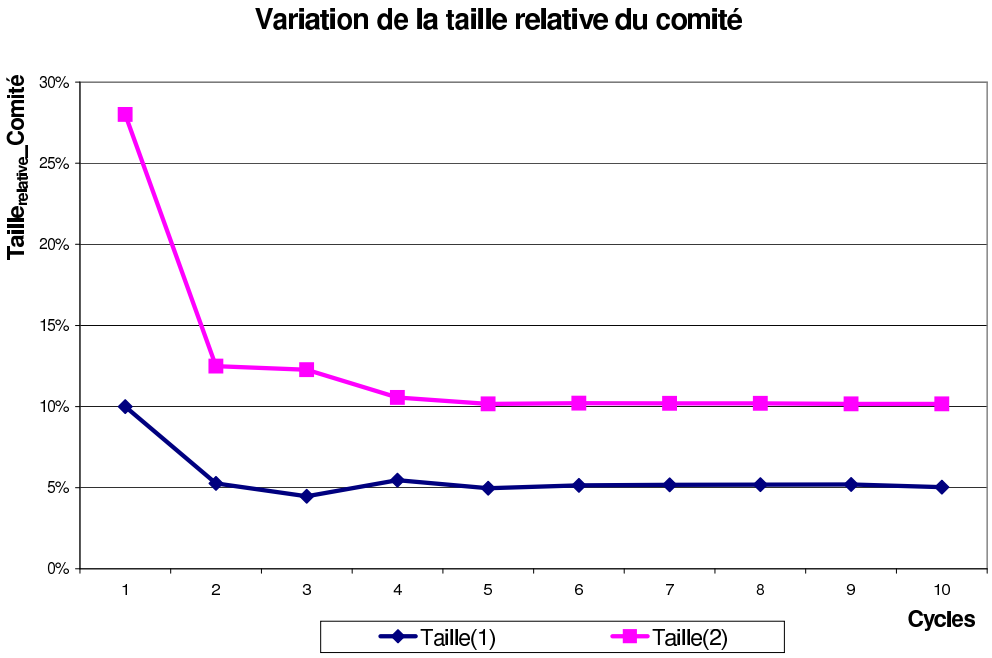

FIG. *5.18: Variation de la taille relative du comité*

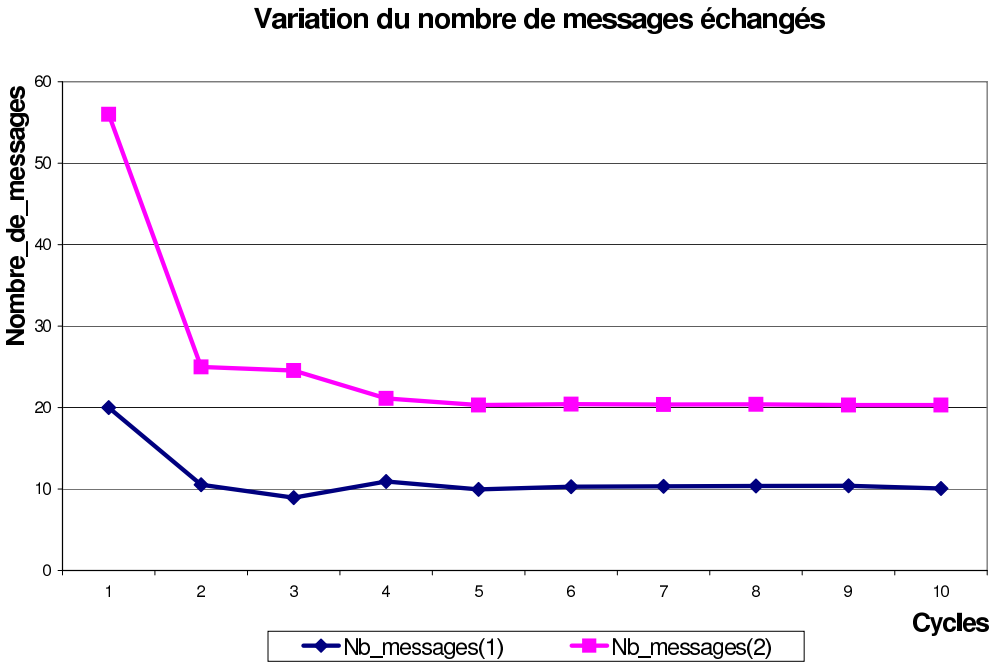

FIG. *5.19: Variation du nombre de messages échangés*

Groupe d'agents	Nb. d'agents intéressants	Taille du groupe
Très populaire	$\geqslant 70$	3
Populaire	$\geqslant 40$ et < 70	23
Peu populaire	$\geqslant 10$ et < 40	49
Rare	$\geqslant 5$ et < 10	12
Très rare	< 5	13

TAB. *5.3: Description des cinq groupes d'agents*

le tableau 5.3.

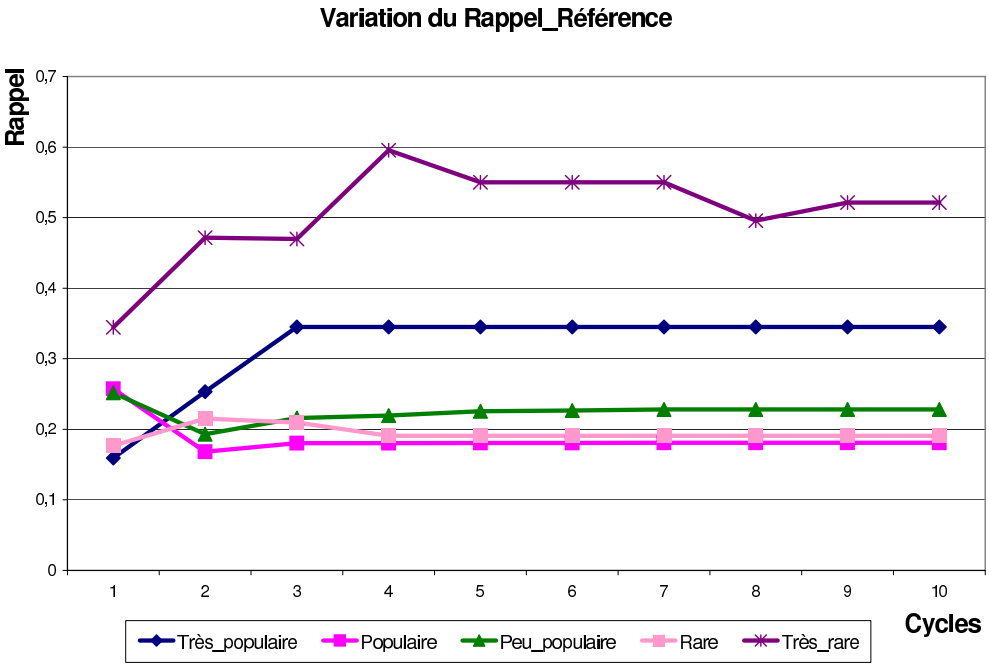

FIG. *5.20: Variation du rappel références pour les cinq groupes*

Les résultats de cette analyse sont décrits dans les figures 5.20, 5.21, 5.22 et 5.23. Nous remarquons que les résultats obtenus par les différents groupes ne sont pas les mêmes. En effet, les valeurs de la précision et surtout du rappel varient considérablement selon le groupe d'agents. En ce qui concerne le rappel et pour les deux modules, les valeurs maximales sont obtenues avec le groupe « Très rare » qui atteint la valeur de $0,6$ et les valeurs minimales sont obtenues avec le groupe « Populaire ». En ce qui concerne la précision, les variations entre les groupes sont assez importantes dans les premiers cycles mais se rapprochent au fur et à mesure des cycles. Les valeurs de la précision pour les deux modules et pour les cinq groupes sont toujours importantes et se stabilisent rapidement. Les valeurs maximales sont obtenues

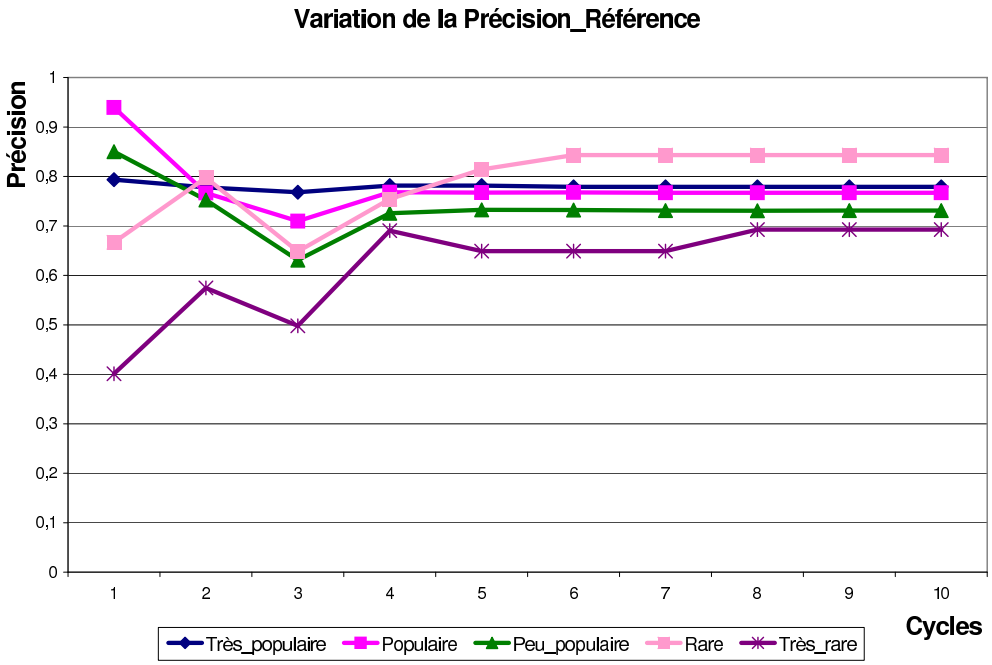

FIG. *5.21: Variation de la précision références pour les cinq groupes*

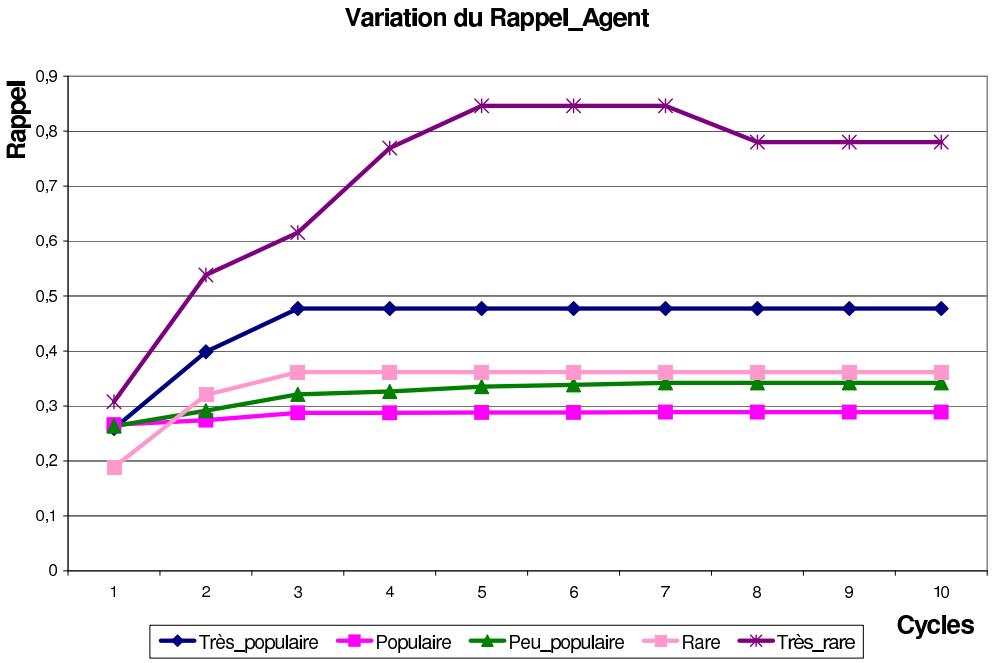

FIG. *5.22: Variation du rappel agent pour les cinq groupes*

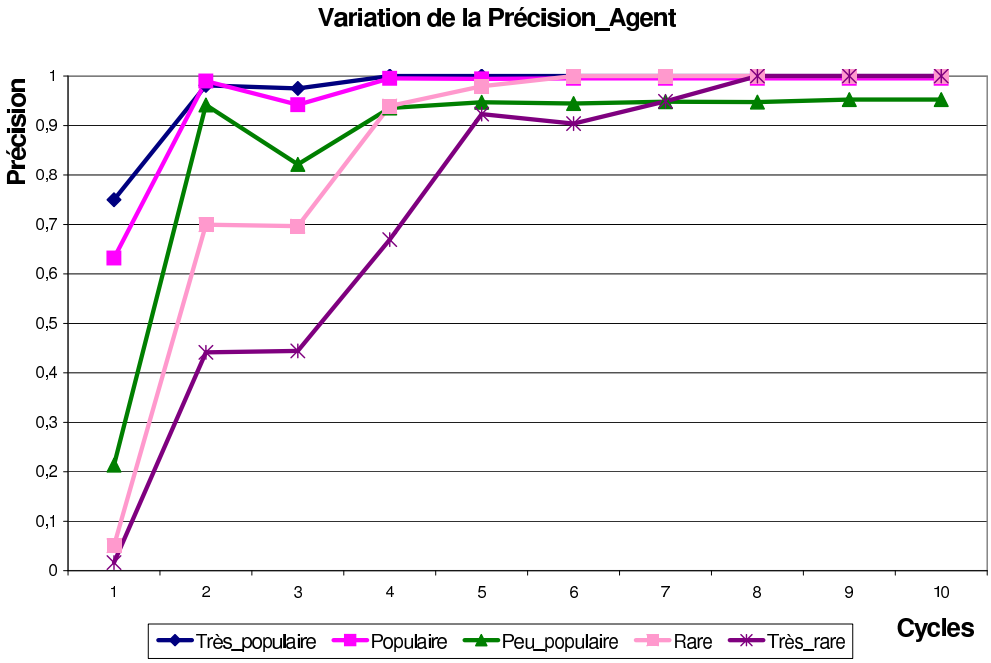

FIG. *5.23: Variation de la précision agent pour les cinq groupes*

avec le groupe « Rare » pour le module de recommandation de références et le groupe « Très populaire » pour le module de formation de comités. Les valeurs minimales sont obtenues avec le groupe « Très rare ». Les valeurs de la précision sont assez similaires pour les cinq groupes alors que les valeurs du rappel sont très variables.

5.3.4 Synthèse

Globalement, nous remarquons que les résultats obtenus par les approches proposées dans le système COBRAS sont satisfaisants. La caractéristique principale de ces approches est leur précision importante, évitant ainsi un effort supplémentaire de la part de l'utilisateur, nécessaire pour sélectionner les bonnes références à partir d'un ensemble important de recommandations bruitées. De plus, les valeurs du rappel sont assez satisfaisantes.

Plus spécifiquement, en étudiant l'homogénéité de l'ensemble d'agents considérés dans l'expérimentation, nous avons distingué cinq groupes selon la popularité de leur thème d'intérêt chez les autres agents. Nous remarquons qu'en général, les résultats obtenus sont satisfaisants pour les différents groupes mais la différence de leur apport dans les performances moyennes du système est importante. Globalement, les groupes « Très populaire » et « Très rare » donnent les meilleurs résultats. Cependant, il n'y a pas de meilleur groupe qui donne les meilleurs résultats pour tous les critères : cela depend du ou des critères à optimiser (voir tableau 5.4). Par exemple, le groupe « Très rare » optimise les valeurs du rappel de recommandation de

Groupe Critère	**Rap. Réf.**	**Préc. Réf.**	**Rap. Ag.**	**Préc. Ag.**	**Apprentissage**
Très populaire	Importante	Importante	Importante	Maximale	Très rapide
Populaire	Minimale	Importante	Minimale	Importante	Très rapide
Peu populaire	Moyenne	Importante	Moyenne	Moyenne	Rapide
Rare	Faible	Maximale	Moyenne	Moyenne	Lent
Très rare	Maximale	Minimale	Maximale	Minimale	Très lent

TAB. *5.4: Comparaison des résultats des différents groupes d'agents*

références tandis que le groupe « Rare » optimise les valeurs de la précision de recommandation de références. Le groupe « Très populaire » améliore ces deux critères. Nous remarquons également que l'apprentissage est plus rapide quand le groupe est plus ou moins populaire que les groupes rares puisque la probabilité de les retrouver est plus importante.

Chapitre 6

Conclusion

Nous avons implémenté une application pilote pour la gestion et la recommandation de références bibliographiques. Le but de cette application consiste à étudier les moyens de réutilisation et de partage automatique des expériences des différents utilisateurs du système. Pour cela, nous avons exploré la méthodologie du raisonnement à partir de cas et l'architecture de type égal-à-égal afin d'améliorer l'accès à l'information et proposer des recommandations provenant des différents utilisateurs du système. Cette coopération entre les utilisateurs du système se fait via leur agent personnel logiciel.

Chaque agent implémente deux modules RàPC complémentaires. Le premier sert à calculer pour une requête donnée, un ensemble d'agents pertinents à interroger. Le second module sert à calculer des recommandations de références bibliographiques par l'ensemble d'agents trouvés. Les deux modules collaborent afin d'améliorer leur propre fonctionnement et le résultat final : *diffuser la demande de recommandation aux* ***bons agents*** *pour avoir les* ***bonnes références*** *à proposer à l'utilisateur.*

6.1 Bilan

Les contributions apportées dans notre travail de thèse consistent essentiellement à proposer des approches collaboratives de partage d'expériences entre groupes organisés d'utilisateurs. Les utilisateurs partagent une même hiérarchie de thèmes mais l'utilisent différemment. Nous avons proposé une approche qui détermine, pour un thème d'intérêt chez un utilisateur l'ensemble de thèmes correspondant chez d'autres utilisateurs. Cette correspondance permet de faciliter la recherche des documents et d'offrir des recommandations aux utilisateurs. Le partage se fait d'une manière implicite. Notre approche consiste à associer un agent assistant à chaque utilisateur. L'agent assistant observe le comportement de l'utilisateur associé afin de déduire ces intérêts. Les agents coopèrent entre eux pour échanger les expériences acquises dans l'objectif d'améliorer les recommandations faites à leur utilisateur.

L'utilisation d'un système RàPC de type égal-à-égal pose de nouveaux problèmes spécifiques tels que :

- comment choisir les agents intéressants pour le thème d'intérêt actuel ?
- comment effectuer la coopération entre les agents : échange de données brutes, de cas, ou de solutions de cas ?

Pour résoudre ce problème, nous avons proposé une autre approche qui permet le partage automatique des expériences de collaboration entre les utilisateurs d'un groupe dans un environnement distribué. La coopération entre les agents se fait par échange de solutions de cas. L'expérience présente un ensemble d'utilisateurs intéressants pour un thème d'intérêt. Ces expériences sont construites à partir des traces des interactions des utilisateurs et sont mémorisés dans des bases de cas sous forme de cas. La méthodologie du raisonnement à partir de cas est employée dans cette approche puisque son principe, consistant à résoudre de nouveaux problèmes en adaptant des solutions approuvées de problèmes similaires résolus dans le passé, s'accorde bien avec les objectifs des systèmes d'aide à la recherche d'information qui visent à proposer à l'utilisateur des recommandations qui correspondent à ses centres d'intérêt. La caractéristique d'apprentissage incrémental de nouveaux cas dans les systèmes RàPC permet d'apprendre continuellement les expériences en recherche d'informations des utilisateurs et les correspondances dans l'utilisation de la hiérarchie commune. Ce qui va favoriser la collaboration des utilisateurs au sein de leur groupe.

Plus spécifiquement, les apports tout au long de la thèse, peuvent se résumer comme suit :

- Nous avons proposé une approche pour le calcul des recommandations de références bibliographiques. Cette approche se base sur la méthodologie du raisonnement à partir de cas afin de permettre la réutilisation des expériences passées pour des problèmes futurs similaires.
- Nous avons proposé une approche pour la formation de comités. Cette approche se base aussi sur la méthodologie du raisonnement à partir de cas et sur une architecture de type égal-à-égal afin de permettre le partage des expériences entre les utilisateurs du système et la détermination des collaborateurs intéressants pour une requête donnée.
- Nous avons proposé un système d'application P2P coopératif (COBRAS) pour la gestion et la recommandation de références bibliographiques. COBRAS est un système de recommandation complet traitant à la fois les problèmes de formation de comités et de calcul des recommandations. Il permet un partage automatique et un apprentissage incrémental des expériences des utilisateurs. Tester le système sur un vrai réseau en utilisant une technologie de réseau égal-à-égal telle que java/JXTA serait intéressant. Ce test peut se faire rapidement puisque nous utilisons déjà le langage Java et les données sont au format XML.
- Afin de minimiser l'implication de l'utilisateur dans l'utilisation du système, nous avons doté les agents des capacités de déduction des besoins en informations de leur utilisateur

(par exemple, ses centres d'intérêt). Cette déduction se fait d'une manière implicite par l'agent assistant en observant les comportements et les actions de son utilisateur sur la base de documents.

6.2 Perspectives

Il reste de nombreuses extensions et directions à explorer, nous pensons tout particulièrement aux points suivants :

- Dans notre travail, nous avons considéré les références bibliographiques comme étant des entités individuelles. Il serait intéressant de considérer les références comme un ensemble et d'étudier les différents types de relations qui lient ses éléments. Nous construisons à partir de cet ensemble un réseau social liant les différentes références. Plusieurs types de liens sémantiques peuvent être considérés dans la construction du réseau en exploitant les informations sur les références : même conférence, même auteur, même année, etc.. La valeur du poids du lien qui lie deux références est une aggrégation pondérée des similarités élémentaires correspondant aux attributs pertinents d'une référence tels que le nom de la conférence ou du journal, le ou la liste des auteurs, la date, etc. Cela peut aussi être utile dans la sélection des références à la fois pertinentes et diversifiées pour une requête utilisateur.
- Dans notre travail, nous avons considéré que tous les utilisateurs partagent une même hiérarchie pour classifier leur références bibliographiques. Chaque utilisateur peut avoir sa propre hiérarchie (ou ontologie) pour classer ses documents. Dans ce cas, la mise en correspondance entre les différents thèmes ou plus généralement les concepts sera plus difficile et il faudra trouver des moyens pour calculer les correspondances d'ontologie, c'est le problème d'alignement d'ontologie (Ehrig et Sure, 2004). Par exemple, l'utilisation des mots-clés des documents et l'exploitation des réseaux sociaux des documents des différents utilisateurs peuvent être d'une grande utilité pour la découverte des correspondances entre les thèmes des différentes ontologies.
- Une perspective à long terme consiste à proposer un système RàPC plus général pour la recommandation d'expériences dans le domaine de la recherche d'information. Dans ce cas, chaque utilisateur est assisté par un groupe d'agents logiciels personnels. Chaque agent est spécialisé dans un type d'activité de recherche d'information. À ce niveau de nouvelles problématiques intéressantes apparaissent telles que :
  - la modélisation de l'expérience : qu'est-ce qu'une expérience de recherche d'information ? Comment est-elle représentée ?
  - la coopération entre les agents : cette coopération peut se faire entre :
    1. des agents homologues, c'est-à-dire des agents spécialisés dans une même tâche ou activité de recherche d'information mais qui sont associés à d'autres utilisateurs.

2. des agents hétérogènes de point de vue tâche ou activité de recherche d'information. Par exemple, des agents d'aide à la navigation sur le Web, des agents d'aide à la gestion de signets, des agents d'aide au tri des résultats des moteurs de recherche, etc. Dans ce cas, la coopération se fait entre le groupe d'agents personnels et sert à renforcer le modèle et les préférences de l'utilisateur appris par les agents assistants.

Liste de publications

Revues

H. Karoui, R. Kanawati and L. Petrucci. An intelligent peer-to-peer multi-agent system for collaborative management of bibliographic databases. In SGAI's expert update magazine, 22-31, 8(3), 2006.

Conférences internationales et workshops

H. Karoui, R. Kanawati and L. Petrucci. COBRAS : Cooperative CBR System for Bibliographical Reference Recommendation In the 8^{th} European Conference on Case-Based Reasoning (ECCBR'06), LNCS 4106, Ölüdeniz/Fethiye, Turkey, September 2006.

H. Karoui, R. Kanawati and L.Petrucci. Cooperative CBR System for Peer Agent Committee Formation. In the 5^{th} Workshop on Agents and Peer-to-Peer Computing (AP2PC'06), Hakodate, Japan, May 2006.

H. Karoui, R. Kanawati and L. Petrucci. An intelligent peer-to-peer multi-agent system for collaborative management of bibliographic databases. In the 9^{th} UK Workshop on Case-Based Reasoning, Queens' College, Cambridge, UK, December 2004.

Conférences nationales et ateliers

H. Karoui. Système de recommandation de références bibliographiques et de formation de comités. $3^{èmes}$ Rencontres Inter-Associations (RIA's 2007), Recherche et extraction d'information. Laboratoire IRIT, Université Paul Sabatier Toulouse III, Mars 2007.

H. Karoui, R. Kanawati and L. Petrucci. Une approche RàPC pour la formation de comité d'agents dans un système de recommandation égal à égal. $14^{ème}$ Atelier de Raisonnement

à Partir de Cas, Besançon, Mars 2006.

H. Karoui. Agent RàPC pour la gestion coopérative de bases bibliographiques personnelles. Plate-forme AFIA' 2005. $13^{ème}$ Atelier de Raisonnement à Partir de Cas, Maison du Séminaire, Nice, Mai 2005.

Bibliographie

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

[Aberer, 2001] K. Aberer. P-Grid : A self-organizing access structure for P2P information systems. In 6^{th} *International Conference on Cooperative Information Systems (CoopIS'01)*, Trento, Italy, September 2001.

[Allan, 1997] J. Allan. Building hypertext using information retrieval. *Information Processing and Management*, 33(2) :145–159, 1997.

[Amato et Straccia, 1999] G. Amato et U. Straccia. User profile modeling and applications to digital librairies. In *the* 3^{rd} *European Conference on Research and Advanced Technology for Digital Librairies*, pages 184–197, London, UK, 1999. Springer-Verlag.

[Balabanovic et Shoham, 1997] M. Balabanovic et Y. Shoham. Fab : content-based, collaborative recommendation. *the Communications of the ACM*, 40(3) :66–72, March 1997.

[Balfe et Smyth, 2004] E. Balfe et B. Smyth. Case-based collaborative web search. In *the* 7^{th} *European Conference on Advances in Case-based Reasoning (ECCBR'04)*, volume 3155 de *LNCS*, pages 489–503, Madrid, Spain, 2004. Springer.

[Barthès et Ramos, 2002] J.-P. Barthès et M. Ramos. Agents assistants personnels dans les systèmes multi-agents mixtes - réalisation sur la plate-forme OMAS. *Technique et Science Informatiques*, 21(4) :473–498, 2002.

[Belkin et Croft, 1992] N.R. Belkin et W.B. Croft. Information filtering and information retrieval : two sides of the same coin ? *Communications of the ACM*, 35(12), december 1992.

[Belkin, 1996] N.J. Belkin. Intelligent information retrieval : Whose intelligence ? In *ISI'96 : Proceedings of the* 5^{th} *International Symposium for Information Science*, pages 25–31, Konstanz : Universtaetsverlag Konstanz, 1996.

[Bellifemine *et al.*, 1999] F. Bellifemine, A. Poggi, et G. Rimassa. JADE - A FIPA-compliant agent framework. In *the Practical Applications of Intelligent Agents*, pages 98–108, London, Avril 1999.

[Benabdeslem et Bennani, 2007] K. Benabdeslem et Y. Bennani. Approche connexionniste pour l'analyse des données issues d'usages d'internet : Classification et visualisation. *Numéro spécial : Fouille du Web de la revue des nouvelles technologies de l'information (RNTI)*, pages 23–36, 2007.

[Bergmann et Wilke, 1998] R. Bergmann et W. Wilke. Towards a new formal model of transformational adaptation in case-based reasoning. In *the* 13^{th} *European Conference on Artificial Intelligence, (ECAI'98)*, pages 53–57, Brighton, UK, August 1998.

[Bergmann, 2002] R. Bergmann. *Experience Management, Foundations, Development Methodology, and Internet-Based Applications*, volume 2432 de *LNAI*. Springer, 2002.

[Berrut et Denos, 2003] C. Berrut et N. Denos. *Assistance intelligente à la recherche d'informations (Traité des sciences et techniques de l'information*, chapitre Filtrage Collaboratif., pages 242–269. Hermes-Lavoisier, Juin 2003.

[Bottraud *et al.*, 2003a] J.C. Bottraud, G. Bisson, et M.F. Bruandet. An adaptive information research personal assistant. In *the Workshop AI2IA (Artificial Intelligence, Information Access and Mobile Computing), IJCAI'03*, Acapulco, Mexico, August 2003.

[Bottraud *et al.*, 2003b] J.C. Bottraud, G. Bisson, et M.F. Bruandet. Apprentissage de profils pour un agent de recherche d'information. In *Actes de la Conférence Apprentissage (CAp'03)*, pages 31–46, Laval, France, Juillet 2003. PUG.

[Bottraud *et al.*, 2004] J.C. Bottraud, G. Bisson, et M.F. Bruandet. Expansion de requêtes par apprentissage automatique dans un assistant pour la recherche d'information. In *Actes de la Conférence en Recherche d'Information et Applications (CORIA'04)*, Toulouse, France, Mars 2004.

[Bouzeghoub et Kostadinov, 2005] M. Bouzeghoub et D. Kostadinov. Personnalisation de l'information : aperçu de l'état de l'art et définition d'un modèle flexible de profils. In *COnférence en Recherche d'Infomations et Applications (CORIA'05)*, pages 556–567, Grenoble, France, Mars 2005.

[Breese *et al.*, 1998] J. Breese, D. Heckerman, et K. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *the* 14^{th} *uncertainty in Artificial Intelligence*, éditeur Morgan Kaufman, pages 43–52, May 1998.

[Brin et Page, 1998] S. Brin et L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1-7) :107–117, 1998.

[Broekstra *et al.*, 2004] J. Broekstra, M. Ehrig, P. Haase, F. Harmelen, M. Menken, P. Mika, B. Schnizler, et R. Siebes. Bibster -a semantics-based bibliographic peer-to-peer system. In *Proceedings of SemPGRID'04,* 2^{nd} *Workshop on Semantics in Peer-to-Peer and Grid Computing*, pages 3–22, New York, USA, may 2004.

[Budzik et Hamond, 2000] J. Budzik et K.J. Hamond. User interactions with every applications as context for just-in-time information access. In *the* 5^{th} *International Conference on Intelligent User Interfaces*, pages 44–51, New Orleans, Louisiana, United States, January 9-12 2000. ACM Press.

[Burgun et Bodenreider, 2001] A. Burgun et O. Bodenreider. Mapping the UMLS semantic network into general ontologies. In *the American Medical Informatics Association (AMIA) Annual Symposium*, pages 81–85, Washington, November 2001.

[Challam, 2004] V.K.R. Challam. Contextual information retrieval using ontology-based user profiles. Master's thesis, Information and Communication Technology Center, University of Kansas, 2004.

[Champin *et al.*, 2004] P.A. Champin, Y. Prié, et A. Mille. MUSETTE : a framewok for knowledge capture from experience. In $12^{ème}$ *Atelier du Raisonnement à Partir de Cas (RàPC'04)*, pages 85–97, Villetaneuse, France, Mars 2004.

[Chandrasekaran *et al.*, 1999] B. Chandrasekaran, J. R. Josephson, et V.R. Benjamins. What are ontologies and why do we need them ? *the IEEE Intelligent Systems*, 14(1) :20–26, 1999.

[Chau *et al.*, 2001] M. Chau, D. Zeng, et H. Chen. Personalised spiders for web search and analysis. In *ACM/IEEE Joint Conference on Digital Libraries*, pages 79–87, Roanoke, VA, USA, June 24-28 2001.

[Chen et Sycara, 1998] L. Chen et K. Sycara. WebMate : A personal agent for browsing and searching. In *the* 2^{nd} *International Conference on Autonomous Agents (Agents'98)*, éditeurs Katia P. Sycara et Michael Wooldridge, pages 132–139, New York, May 1998. ACM Press.

[Chidlovski, 2000] Boris Chidlovski. Collaborative re-ranking of search results. In *the AAAI workshop on artificial intelligence for Web search*, pages 18–22, 2000.

[Chirita *et al.*, 2006] P-A. Chirita, S. Idreos, M. Koubarakis, et W. Nejdl. Designing semantic publish/subscribe networks using super-peers. In *Semantic Web and Peer-to-Peer : Decentralized Management and Exchange of Knowledge and Information*, éditeurs S. Staab et H. Stuckenschmidt, pages 159–179, Berlin, Germany, July 2006. Springer Berlin Heidelberg.

[Clarke *et al.*, 2001] I. Clarke, O. Sandberg, B. Wiley, et T.W. Hong. Freenet : A distributed anonymous information storage and retrieval system. *LNCS : Designing Privacy Enhancing Technologies*, 2009 :46–66, 2001.

[Cordier *et al.*, 2007] A. Cordier, B. Fuchs, J. Lieber, et A. Mille. Failure Analysis for Domain Knowledge Acquisition in a Knowledge-Intensive CBR System. In *the* 7^{th} *International Conference on Case-Based Reasoning, (ICCBR'07)*, éditeurs Rosina Weber et Michael Richter, volume 4626 de *LNCS*, pages 463–477, Northern Ireland, UK, August 2007. Springer.

[Crespo et Garcia-Molina, 2002a] A. Crespo et H. Garcia-Molina. Routing indices for peer-to-peer systems. In *the* 22^{th} *International Conference on Distributed Computing Systems (ICDCS'2002)*, pages 23–34, Vienna, Austria, July 2002. IEEE.

[Crespo et Garcia-Molina, 2002b] A. Crespo et H. Garcia-Molina. Semantic overlay networks for P2P systems. Rapport technique, Computer Science Dpt., Stanford University, 2002.

[Daswani et Fisk, 2002] S. Daswani et A. Fisk. Gnutella UDP extension for scalable searches (guess). LimeWire LLC, August 2002.

[Decker *et al.*, 1997] K. Decker, K. Sycara, et M. Williamson. Middle-agents for the internet. In *the International Joint Conference on Artificial Intelligence (IJCAI'97)*, volume 1, pages 578–583, Nagoya, Japan, January 1997.

[Dorcey, 1995] T. Dorcey. CU-SeeMe desktop videoconferencing software. *Connexions*, 9(3), March 1995.

[Doucet et Lumineau, 2003] A. Doucet et N. Lumineau. A collaborative approach for query propagation in peer-to-peer systems. In *the* 1^{st} *International Workshop on Semantic Web and Databases (SWDB'03) co-located with VLDB 2003*, pages 251–257, Berlin, Germany, September 2003.

[Dreilinger et Howe, 1997] D. Dreilinger et A. Howe. Experiences with selecting search engines using metasearch. *the ACM Transactions on Information Systems*, 15(3) :195–222, 1997.

[Dumais *et al.*, 2003] S. Dumais, E. Cadiz Cuttrel, J.J. Jancke, G. Sarin, et D.C. Robbins. Stuff I've seen : A system for a personal information retrieval and re-use. In *the* 26^{th} *ACM SIGIR International Conference on Research and Development*, pages 72–79. ACM Press, 2003.

[Ehrig et Sure, 2004] M. Ehrig et Y. Sure. Ontology Mapping- An Integrated Approach. In *the Semantic Web : Research and Applications,* 1^{st} *European Semantic Web Symposium, (ESWS'04)*, éditeurs Christoph Bussler, John Davies, Dieter Fensel, et Rudi Studer, volume 3053 de *LNCS*, pages 76–91, Heraklion, Crete, Greece, May 2004. Springer.

[Enembreck et Barthès, 2004] F. Enembreck et J.-P. Barthès. *Fouille de textes et organisation de documents*, volume 8, chapitre MAIS-Un système multi-agents pour la recherche d'information sur le web, pages 83–106. Lavoisier, 2004.

[Enembreck, 2003] F. Enembreck. *Contribution à la conception d'agents assistants personnels adaptatifs.* PhD thesis, Université de Technologie de Compiègne, Décembre 2003.

[Faye *et al.*, 2006] D. Faye, G. Nachouki, et P. Valduriez. Un système pair-à-pair de médiation de données. In *Revue Africaine de la Recherche en Informatique et Mathématique Appliquées (ARIMA)*, volume 4, pages 24–52, 2006.

[Fuchs *et al.*, 1999] B. Fuchs, J. Lieber, A. Mille, et A. Napoli. Towards a unified theory of adaptation in case-based reasoning. In *Case Based Reasoning Research and Development, The* 3^{rd} *International Conference on Case-Based Reasoning (ICCBR'99)*, volume 1650 de *LNAI*, pages 104–117, Seeon Monastery, Germany, August 25-27 1999. Springer-Verlag.

[Fuchs *et al.*, 2000] B. Fuchs, J. Lieber, A. Mille, et A. Napoli. An algorithm for adaptation in

case-based reasoning. In *the* 14^{th} *European Conference on Artificial Intelligence (ECAI'00)*, volume 14, pages 45–49, Amsterdam, The Netherlands, August 20-25 2000. IOS Press.

[Fuchs et Mille, 2000] B. Fuchs et A. Mille. Une modélisation au niveau connaissance du raisonnement à partir de cas. In *Actes des Journées Ingénierie des Connaissances (IC'00)*, éditeur N. Aussenac-Gilles, pages 3–11, Toulouse, France, Mai 2000.

[Gauch *et al.*, 2003] S. Gauch, J. Chaffe, et A. Pretschner. Ontology-based personalized search and browsing. *Web Intelligence and Agent System*, 1(3-4) :219–234, 2003.

[Glover *et al.*, 1999] E.J. Glover, S. Lawrence, W.P. Birmingham, et C.L. Giles. Architecture of a metasearch engine that supports user information needs. In *the* 8^{th} *International Conference on Information and Knowledge Management (CIKM'99)*, pages 210–216, Kansas City, MO, November 1999. ACM Press.

[Gowan, 2003] J.P. Mc Gowan. A multiple model approach to personnalized information access. Master's thesis, Faculty of Science, University College Dublin, February 2003.

[Gruber, 1993] T.R. Gruber. Towards principles for the design of ontologies used for knowledge sharing. In *the Formal Ontology in Conceptual Analysis and Knowledge Representation*, éditeurs N. Guarino et R. Poli, Deventer, The Netherlands, March 1993. Kluwer Academic Publishers.

[Halevy *et al.*, 2003] A. Halevy, Z. Ives, P. Mork, et I. Tatarinov. Piazza : data management infrastructure for semantic web applications. In *the* 12^{th} *International Conference on World Wide Web*, pages 556–567, Budapest, 2003.

[Hanney et Keane, 1996] K. Hanney et M. T. Keane. Learning adaptation rules from a case-base. In *the* 3^{rd} *European Workshop on Advances in Case-Based Reasoning (EWCBR'96)*, volume 1168 de *LNCS*, pages 179–192, London, UK, November 1996. Springer-Verlag.

[Heckerman, 1999] D. Heckerman. A tutorial on learning with bayesian networks. In *the Learning in Graphical Models*, éditeur M. Jordan. MIT Press, Cambridge, MA, 1999.

[Héraud, 2002] J. M. Héraud. *Pixed : Une approche collaborative de l'expérience et l'expertise pour guider l'adaptation des hypermédias.* PhD thesis, Université Lyon I, Décembre 2002.

[Hibou, 2006] M. Hibou. Réseaux bayésiens pour la modélisation de l'apprenant en eiah : modèles multiples versus modèle unique. In $1^{ères}$ *Rencontres Jeunes Chercheurs en EIAH, (RJC-EIAH'06)*, pages 40–47, Paris, France, Mai 2006.

[Ingwersen, 1996] P. Ingwersen. Cognitive perspectives of information retrieval interaction : elements of a cognitive IR theory. *the Journal of Documentation*, 52(1) :3–50, 1996.

[Jaczynski et Trousse, 1998] M. Jaczynski et B. Trousse. WWW assisted browsing by reusing past navigations of a group of users. In *the* 4^{th} *European Workshop on Case-Based Reasoning (EWCBR'98)*, volume 1488 de *LNCS*, pages 160–171, Dublin, Ireland, 1998.

[Jaczynski, 1998] M. Jaczynski. *Modèle et plate-forme à objets pour l'indexation par situations comportementales : application à l'assistance à la navigation sur le Web.* PhD thesis, Université de Nice-Antipolis, 1998.

[Jenning, 1993] N.R. Jenning. Commitments and conventions : The foundation of coordination in multi-agent systems. *The Knowledge Engineering Review*, 8(3) :223–250, 1993.

[Jéribi *et al.*, 2001] L. Jéribi, B. Rumpler, et J.M. Pinon. Système d'aide à la recherche et à l'interrogation de bases documentaires, basé sur la réutilisation d'expériences. In *Congrès Informatique des Organisations et Systèmes d'Information et de Décision (INFORSID'2001)*, pages 443–463, Martigny, Suisse, 2001.

[Kalogeraki *et al.*, 2002] V. Kalogeraki, D. Gunopulos, et D. Zeinalipour-Yazti. A local search mechanism for peer-to-peer networks. In *the* 11^{th} *International Conference on Information and knowledge management, (CIKM'02)*, pages 300–307, New York, NY, USA, 2002. ACM Press.

[Kanawati *et al.*, 1999a] R. Kanawati, M. Jaczynski, B. Trousse, et J.-M. Anderloi. Applying the broadway recommendation computation approach for implementing a query refinement service in the CBKB meta-search engine. In *Actes de RàPC'99*, pages 17–26, Palaiseau, France, 1999.

[Kanawati *et al.*, 1999b] R. Kanawati, M. Jaczynski, B. Trousse, et J.M. Andreoli. Applying the broadway recommendation computation approach for implementing a query refinement service in the CBKB meta search engine. In *Actes de Raisonnement à Partir de Cas (RàPC'99)*, pages 17–26, Plaiseau, 1999.

[Kanawati et Malek, 2000] R. Kanawati et M. Malek. Informing the design of shared bookmark systems. In *the proceedings of RIAO'2000 : Content-based Multimedia Information Access*, pages 170–180, Collège de France, Paris, France, April 2000.

[Kanawati et Malek, 2001] R. Kanawati et M. Malek. Cowing : A collaborative bookmark management system. In *the International workshop on Cooperative Information Agents (CIA'01)*, volume 2182 de *LNAI*, pages 34–39, 2001.

[Kanawati et Malek, 2002] R. Kanawati et M. Malek. A multi-agent system for collaborative bookmarking. In *the* 1^{st} *International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'02)*, pages 1137–1138. ACM Press, 2002.

[Karoui *et al.*, 2006] H. Karoui, R. Kanawati, et L. Petrucci. COBRAS : Cooperative CBR system for bibliographical reference recommendation. In *the* 8^{th} *European Conference on Case-Based Reasonning (ECCBR'06)*, éditeurs Thomas Roth-Berghofer, Mehmet H. Göker, et H. Altay Güvenir, volume 4106 de *LNCS*, pages 76–90, Ölüdeniz/Fethiye, Turkey, September 2006. Springer.

[Kolodner, 1993] J. Kolodner. *Case-Based Reasoning.* Morgan Kaufmann, Calif. US., 1993.

[Kolodner, 1996] J. Kolodner. Making the implicit explicit : Clarifying the principles of case-based reasoning. In *Case-Based Reasoning : Experiences, Lessons and Future Directions*, pages 349–370. AAAI press, Mit Press, 1996.

[Kubiatowicz *et al.*, 2000] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, et B. Zhao. Oceanstore : An architecture for gobal-scale persistent storage. *ACM SIGPLAN Notices*, 35(11) :190–201, 2000.

[Li et Horrocks, 2003] L. Li et I. Horrocks. A software framework for matchmaking based on semantic web technology. In *the* 12^{th} *World Wide Web Conference (WWW'03)*, pages 331–339, Budapest, Hungary, May 2003. ACM.

[Lieber *et al.*, 2001] J. Lieber, P. Bey, F. Boisson, B. Bresson, P. Falzon, A. Lesur, A. Napoli, M. Rios, et C. Sauvagnac. Acquisition et modélisation de connaissances d'adaptation, une étude pour le traitement du cancer du sein. In *Actes des Journées Ingénierie des Connaissances (IC'01)*, pages 409–426, Grenoble, France, Juin 2001.

[Lieber *et al.*, 2004] J. Lieber, M. d'Aquin, S. Brachais, et A. Napoli. Une étude comparative de quelques travaux sur l'acquisition des connaissances d'adaptation pour le raisonnement à partir de cas. In $12^{ème}$ *Atelier de Raisonnement à Partir de Cas (RàPC'04)*, pages 53–60, Villetaneuse, France, Mars 2004.

[Lieber, 2002] J. Lieber. Recopier c'est déjà adapter : six types d'adaptation par copie. In *the* $10^{ème}$ *Séminaire Français de Raisonnement à Partir de Cas (RàPC'02)*, pages 11–21, Paris, France, Juin 2002.

[Lin, 2005] J. Lin. Evaluation of resource for question answering evaluation. In *Actes of the* 28^{th} *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR'05*, pages 392–399, Salvador, Bahia, Brazil, 15-19 August 2005.

[Liu et Croft, 2004] X. Liu et W.B. Croft. Cluster-based retrieval using language models. In *the* 27^{th} *Annual International ACM SIGIR Conference (SIGIR'04)*, pages 186–193, South Yorkshire, UK, July 2004. ACM.

[Lv *et al.*, 2002] C. Lv, P. Cao, E. Cohen, K. Li, et S. Shenker. Search and replication in unstructured peer-to-peer networks. In *the* 16^{th} *International Conference on Supercomputing (ICS'02)*, pages 84–95, New York, USA, 2002. ACM Press.

[Main *et al.*, 2000] J. Main, T.S. Dillon, et S.C.K. Shiu. *Soft Computing in Case-Based Reasoning.* Springer-Verlag, October 2000.

[Malek et Kanawati, 2001] M. Malek et R. Kanawati. COBRA : A cbr-based approach for predicting users actions in a web site. In *Case-Based Reasoning Research and Development, the* 4^{th} *International Conference on Case-Based Reasoning, (ICCBR'01)*, volume 2080 de *LNCS*, pages 336–346, Vancouver, Canada, July 30 - August 2 2001. Springer.

[Malek et Kanawati, 2004] M. Malek et R. Kanawati. A data driven CBR approach : A contineous maintenance strategy. In *the International Conference on Advances in Intelligent Systems : Theory and Applications*, pages 281–290, Luxemburg, November 2004.

[Martin *et al.*, 1999] F.J. Martin, E. Plaza, et J.L. Arcos. Knowledge and experience reuse through communication among competent (peer) agents. *International Journal of Software Engineering and Knowledge Engineering*, 9(3) :319–341, 1999.

[Masterton, 1997] S. Masterton. The virtual participant : Lessons to be learned from a case-based tutor's assistant. In *Computer Support for Collaborative Learning (CSCL'97)*, Toronto, Canada, December 1997.

[McSherry, 2002] D. McSherry. A generalized approach to similarity-based retrieval in recommender systems. *Artificial Intelligence Review*, 18 :309–341, 2002.

[Menascé et Kanchanapalli, 2002] D.A. Menascé et L. Kanchanapalli. Probabilistic scalable P2P resource location services. *the ACM SIGMETRICS Performance Evaluation Review*, 30(2) :48–58, September 2002.

[Mille et Prié, 2006] A. Mille et Y. Prié. Une théorie de la trace informatique pour faciliter l'adaptation dans la confrontation logique d'utilisation/logique de conception. In $13^{ème}$ *Journées de Rochebrune - Traces, Enigmes, Problèmes : Emergence et construction du sens-Rencontres interdisciplinaires sur les systèmes complexes naturels et artificiels*, Rochebrune, France, Janvier 2006.

[Mille, 2006a] A. Mille. From case-based reasoning to traces-based reasoning. *Annual Reviews in Control*, 30(2) :223–232, october 2006.

[Mille, 2006b] A. Mille. Traces based reasoning (TBR) definition, illustration and echoes with story telling. Rapport Technique RR-LIRIS-2006-002, LIRIS UMR 5205 CNRS/INSA de Lyon/Université Claude Bernard Lyon 1/Université Lumière Lyon 2/Ecole Centrale de Lyon, january 2006.

[Minsky, 1975] M. Minsky. A framework for representing knowledge. In *the psychology of Computer Vision*, éditeur P.H. (Ed.) Winston, pages 211–279, New York, McGraw-Hill, 1975.

[Mishra et Ahuja, 2004] J. Mishra et S. Ahuja. A survey on the state of the art in peer to peer computing. In *the Conference on Communication Internet and Information Technology (IASTED'04)*, pages 1–22, St. Thomas, US Virgin Islands, 2004. Springer.

[Nanas *et al.*, 2003] N. Nanas, V. Uren, A. de Roeck, et J. Domingue. Building and applying a concept hierarchy representation of a user profile. In *th* 26^{th} *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 198–204, New York, USA, 2003. ACM Press.

[Neches *et al.*, 1991] R. Neches, R. Fikes, T. Finin, T. Gruber, R. Patil, T. Senator, et W.R. Swartout. Enabling technology for knowledge sharing. *Artificial Intelligence Magazine*, 12(3) :36–56, 1991.

[Nejdl *et al.*, 2003] W. Nejdl, M. Siberski, C. Schmitz, M. Schlosser, I. Brunkhorst, et A. Loser. Super-peer based routing and clustering strategies for rdf-based peer-to-peer networks. In *the* 12^{th} *International World Wide Web Conference (WWW'03)*, pages 536–543, New York, USA, 2003. ACM Press.

[Neumann et Morgenstern, 1947] J. Von Neumann et O. Morgenstern. *Theory of Games and Economic Behavior*. Princeton : Princeton University Press, Princeton, 1947. 2^{nd} Edition.

[Nguyen *et al.*, 2006a] A-T. Nguyen, N. Denos, et C. Berrut. Exploitation des données « disponibles à froid » pour améliorer le démarrage à froid dans les systèmes de filtrage d'information. In $33^{ème}$ *Congrès INFORSID*, Hammamet, Tunisie, Juin 2006.

[Nguyen *et al.*, 2006b] A-T. Nguyen, N. Denos, et C. Berrut. Modèle d'espaces de communautés orienté vers la diversité de recommandations pour les systèmes de filtrage. *Revue I3 (Information - Interaction - Intelligence)*, 6(2) :125–150, 2006.

[O'Brien et Nicol, 1998] P. O'Brien et R. Nicol. FIPA - towards a standard for software agents. *the BT Technology Journal*, 16(3) :51–59, 1998.

[Ontañón et Plaza, 2003] S. Ontañón et E. Plaza. Learning to form dynamic committees. In *the* 2^{nd} *International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'03)*, pages 504–511, Melbourne, Australia, July 2003. ACM Press.

[Ratnasamy *et al.*, 2001] S. Ratnasamy, P. Francis, M. Handley, R. Karp, et S. Shenker. CAN : a scalable content-addressable network. In *the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM'01)*, pages 161–172, New York, USA, August 2001. ACM Press.

[Resnick et Varian, 1997] P. Resnick et H.R. Varian. Recommender systems. *the Communications of the ACM*, 40(3) :56–58, March 1997.

[Riesbeck et Schank, 1989] C. K. Riesbeck et R. C. Schank. *Inside Case-Based Reasoning*. Lawrence Erlbaum Associates, Inc., Mahwah, NJ, USA, 1989.

[Rowstron et Druschel, 2001] A. Rowstron et P. Druschel. Pastry : Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *the IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, Heidelberg, Germany, November 2001.

[Rucker et Polanco, 1997] J. Rucker et M.J. Polanco. Siteseer : Personalized navigation for the web. *Communications of the ACM*, 40(3) :73–76, Mars 1997.

[Salton, 1971] G. Salton. *The SMART Retrieval System-Experiments in Automatic document Processing*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1971.

[Salton, 1989] G. Salton. *Automatic Text Processing : The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley, 1989.

[Saracevic, 1997] T. Saracevic. The stratified model of information retrieval interaction : Extension and applications. *Proceedings of the American Society for Information Science*, 34(2) :313–327, 1997.

[Schank, 1983] R. Schank. *Dynamic Memory ; a Theory of Reminding and Learning in Computers and People*. Cambridge University Press, New York, USA, 1983.

[Secchi, 1999] P. Secchi. Proceedings of alerts and lessons learned : An effective way to prevent failures and problems. Rapport technique, WPP-167, Noordwijk, The Netherlands : ESTEC, 1999.

[Shahabi *et al.*, 2001] C. Shahabi, F. Banaei-Kashani, et J. Faruque. A reliable, efficient, and scalable system for web usage data acquisition. In *the Proceedings of WebKDD Workshop in conjunction with the ACM-SIGKDD*, August 2001.

[Sire, 1999] S. Sire. Enrichir les échanges de données avec les collecticiels. In $11^{ème}$ *Conférence Francophone Interaction Homme-Machine (IHM'99)*, pages 91–105, Montpellier, France, Novembre 22-26 1999.

[Smith *et al.*, 1998] R. Smith, R. Hixon, et B. Horan. Supporting flexible roles in a shared space. In *the ACM Conference on Computer Supported Cooperative Work (CSCW'98)*, pages 197–206, New York, USA, 1998. ACM Press.

[Smyth et Keane, 1993] B. Smyth et M.T. Keane. Retrieving adaptable cases : The role of adaptation knowledge in case retrieval. In *the European Workshop on Case-Based Reasonin (EWCBR'93)*, pages 209–220, Kaiserslautern, Germany, November 1993.

[Smyth et Keane, 1995] B. Smyth et M.T. Keane. Experiments on adaptation-guided retrieval in a case-based design system. In *Case-Based Reasoning : Research and Development (ICCBR'95)*, éditeurs M. Veloso et A. Aamodt, volume 1010 de *LNAI*, pages 313–324, Sesimbra, Portugal, October 23-26 1995. Springer-Verlag.

[Smyth et McClave, 2001] B. Smyth et P. McClave. Similarity versus diversity. In *the* 4^{th} *International Conference on Case-Based Reasoning ICCBR'01*, volume 2080 de *LNCS*, pages 347–361, London, UK, 2001. Springer-Verlag.

[Sonnenwald, 1999] D.H. Sonnenwald. Perspectives of human information behaviour : Contexts, situations, social networks and information horizons. *Exploring the Contexts of Information Behaviour*, pages 176–190, 1999.

[Stewart *et al.*, 1998] B. M. Stewart, Lades H. Martin, et Sejnowski T.J. Independent component representations for face recognition. In *Symposium on Electonic Imaging : Science and Technology ; Human Vision and Electronic Imaging III (SPIE)*, éditeur B. Pappas T. Rogowitz, volume 3299, pages 528–539, San Jose, CA, January 1998. SPIE Press.

[Stoica *et al.*, 2001] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, et H. Balakrishman. Chord : a scalable peer-to-peer lookup service for internet applications. In *the Conference on*

Applications, technologies, architectures, and protocols for computer communications ACM (SIGCOMM'01), pages 149–160, New York, NY, USA, August 2001. ACM Press.

[Stokes, 2002] M. Stokes. Gnutella2 : Specifications part one. Rapport technique, 2002.

[Stuber *et al.*, 2004] A. Stuber, S. Hassas, et A. Mille. Combiner le paradigme multi-agents et le raisonnement à partir d'expérience pour assister la réalisation collective de tâches. In $12^{ème}$ *Atelier de Raisonnement à Partir de Cas RàPC 2004*, pages 99–104, Villetaneuse, France, 2004.

[Sycara, 1988] K. Sycara. Using case-based reasoning for plan adaptation and repair. In *the Case-Based Reasoning workshop, DARPA*, pages 425–434, Clearwater Beach, Florida, 1988. Morgan Kaufmann.

[Tautz *et al.*, 2000] C. Tautz, M. Althoff, et M. Nick. A case-based reasoning approach for managing qualitative experience. Rapport technique, Fraunhofer Publica, Germany, March 2000.

[T.Berners-Lee, 1999] T.Berners-Lee. *Weaving the Web.* Harper San Francisco, New York, USA, 1999.

[Tempich et Staab, 2006] C. Tempich et S. Staab. *Semantic Web and Peer-to-Peer : Decentralized Management and Exchange of Knowledge and Information*, chapitre Semantic Query Routing in Unstructured Networks Using Social Metaphors, pages 107–123. Springer-Verlag, Berlin, Germany, July 2006.

[Terveen *et al.*, 1997] L. Terveen, W. Hill, B. Amento, D. McDonald, et J. Creter. Phoaks : a system for sharing recommendations. *the Communications of the ACM*, 40(3) :59–62, March 1997.

[Trousse *et al.*, 1999] B. Trousse, M. Jaczynski, et R. Kanawati. Using user behavior similarity for recommendation computation : the broadway approach. In *the* 8^{th} *International Conference on Human-Computer Interaction (HCI'99)*, volume 2, pages 85–89, Munich, Germany, August 1999. Lawrence Erlbaum.

[Tsoumakos et Roussopoulos, 2003a] D. Tsoumakos et N. Roussopoulos. Adaptive probabilistic search (APS) for peer-to-peer networks. In *the* 3^{rd} *International Conference on Peer-to-Peer Computing (P2P'03)*, page 102, Washington, DC, USA, 2003. IEEE Computer Society.

[Tsoumakos et Roussopoulos, 2003b] D. Tsoumakos et N. Roussopoulos. A comparison of peer-to-peer search methods. In *the* 6^{th} *International Workshop on the Web and Databases (WebDB'03)*, pages 61–66, San Diego, USA, 2003.

[Weber *et al.*, 2001] R. Weber, D. W. Aha, et I. Becerra-Fernandez. Intelligent lessons learned systems. *Expert Syst. Appl.*, 20(1) :17–34, February 2001.

[Wen *et al.*, 2004] J.-R. Wen, N. Lao, et W.-Y. Ma. Probabilistic model for contextual retrieval.

In *the* 27^{th} *Annual International ACM SIGIR Conference on Research and development in Information Retrieval (SIGIR'04)*, pages 57–63, New York, USA, 2004. ACM Press.

[Yan *et al.*, 1996] T.W. Yan, M. Jacobsen, H. Garcia-Molina, et U. Dayal. From user access patterns to dynamic hypertext linking. In *the* 5^{th} *International World Wide Web Conference on Computer Networks and ISDN Systems*, pages 1007–1014, Amsterdam, The Netherlands, 1996. Elsevier Science Publishers B. V.

[Yang et Garcia-Molina, 2001] B. Yang et H. Garcia-Molina. Comparing hybrid peer-to-peer systems. In *the* 27^{th} *International Conference on Very Large Data Bases (VLDB'01)*, pages 561–570, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[Yang et Garcia-Molina, 2002] B. Yang et H. Garcia-Molina. Improving search in peer-to-peer networks. In *the* 22^{nd} *International Conference on Distributed Computing Systems (ICDCS'02)*, Washington, DC, USA, July 2002. IEEE Computer Society.

[Yang et Garcia-Molina, 2003] B. Yang et H. Garcia-Molina. Designing a super-peer network. In *the* 19^{th} *International Conference on Data Engineering (ICDE'03)*, éditeurs U. Dayal, K. Ramamritham, et T. M. Vijayaraman, pages 49–60, Bangalore, India, March 2003. IEEE Computer Society.

[Zemirli *et al.*, 2005] W.N. Zemirli, L. Tamine, et M. Boughanem. Accès personnalisé à l'information : vers la définition d'un profil utilisateur multidimensionnel. In *the International Symposium On Programming Systems (ISPS'05)*, pages 20–28, Alger, Algérie, Mai 2005. USTHB.