Recherche d'images sur un réseau à l'aide d'un système multi-agents

David Picard

Université de Cergy-Pontoise

THÈSE

présentée pour obtenir le titre de DOCTEUR en Informatique et Traitement de l'Information

Recherche d'images sur un réseau à l'aide d'un système multi-agents

David Picard

E-mail : picard@ensea.fr

Soutenue le 5 décembre 2008 devant le jury composé de :

M. Jean-Pierre Cocquerez,	Rapporteur
M. Michel Crucianu,	Rapporteur
M. Matthieu Cord,	Directeur de thèse
M. Arnaud Revel,	Directeur de thèse
M. Guy Théraulaz,	Examinateur
M. Mathias Quoy,	Examinateur

Résumé

Cette thèse a pour objet la recherche d'images par le contenu dans un contexte de bases d'images distribuées. Nous nous plaçons dans le schéma classique des systèmes de recherche d'images interactifs, c'est-à-dire que le système présente des images à l'utilisateur et celui-ci les annote afin d'affiner la recherche, ce que l'on appelle "bouclage de pertinence". Nous proposons dans un premier temps, une extension de ce schéma à la recherche distribuée à l'aide d'un système multi-agents. Les agents parcourent le réseau à la recherche des images pertinentes et marquent les chemins pertinents afin de guider les autres agents vers les sites intéressants. Notre stratégie s'inspire du comportement des fourmis et de leur marquage de l'environnement à l'aide de phéromones. Dans un second temps, nous nous intéressons à la ré-utilisation des marquages d'une session de recherche à une autre. Cet apprentissage à long-terme permet aux agents de trouver plus facilement les sites contenant des images pertinentes, et offre une importante réduction du temps d'interaction requis pour l'obtention de bons résultats.

Abstract

In this thesis, we focus on content based image retrieval in distributed collections. We present a framework with online learning based on ant-like mobile agents. Mobile agents crawl the network to find images matching a given example query. The images retrieved are shown to the user who labels them, following the classical relevant feedback scheme. The labels are used both to improve the similarity measure used for the retrieval and to learn paths leading to sites containing relevant images. The relevant paths are learned in an ethologically inspired way. We also present an extension with the re-use of learned paths for later sessions. This long-term learning step leads to further improvement in the interaction time required to obtain good results.

Je voudrais tout d'abord remercier MM. Jean-Pierre Cocquerez, Michel Crucianu, Guy Théraulaz et Mathias Quoy d'avoir bien voulu participer à mon jury de thèse, et pour leurs remarques et leurs questions très intéressantes lors de la soutenance.

Je dois aussi de chaleureux remerciements à Matthieu Cord et Arnaud Revel pour m'avoir encadré. Leur disponibilité, leurs précieux conseils, leur ouverture d'esprit, mais aussi leur acharnement à me faire travailler ont été le principal moteur de cette thèse.

Je dois beaucoup à Laurent, Michel et Michel pour ne pas avoir craquer malgré tous les dérangements informatiques que j'ai pu causer (et qui j'espère ont toujours été réparables).

Un grand merci à l'ensemble du laboratoire ETIS. C'est un endroit où il fait bon travailler, et même si je n'ai pas été un grand assidu des réunions plénières, l'ambiance chaleureuse, souvent amicale, y est particulièrement agréable. En particulier, J'aimerais remercier Philippe avec qui j'ai partagé mon bureau, pour avoir accepté d'être constamment dérangé, pour toutes nos fructueuses discussions, scientifiques ou non. De même, j'ai une pensée amical pour Eduardo, avec qui j'ai aussi partagé un bureau, et qui a ensoleillé mes heures de rédaction. Un grand merci au reste de l'équipe MIDI, pour avoir toujours été compréhensif envers mes travaux, même quand ceux-ci s'éloignaient des thématiques en vogue. Il me faut aussi remercier les doctorants du labo, que ce soient les anciens (Joséphine, Dimitri, Adrian, Nicolas, Christophe, Mickael), ceux avec qui j'ai partagé ces trois années (Auguste, Sonia, Ayman, Lucile, Heykel) ceux qui sont arrivés après (Justine, David, JE, Shuji, Sofiane, Emmanuel) et tous ceux que j'oublie de nommer présentement, mais à qui je pense régulièrement. Ceux sont eux qui sont la vie de ce laboratoire, il fallait leur rendre hommage.

Je voudrais aussi remercier l'équipe des enseignants de l'ENSEA avec qui j'ai effectué mon monitorat. Ce sont des personnes formidables, avec lesquels j'ai beaucoup appris, et qui sont d'une grande motivation en ce qui concerne l'enseignement.

Bien moins scientifique ou pédagogique, j'aimerais remercier mes camarades de groupes de musique, avec lesquels j'ai usé mes samedis. Leur amitié et leur créativité m'ont aidé à apprécier le côté architecte de cette thèse et à ne pas brider mon imagination.

Évidemment, je dois une très large partie de cette thèse à ma mère, que je remercie pour m'avoir donner la volonté et la curiosité d'en vouloir plus. Connaissant mon naturel, sans son acharnement dès mon plus jeune âge, je ne serais jamais allé bien loin. Je lui suis éternellement reconnaissant d'avoir su insister quand il le fallait et comme il le fallait.

Enfin, le dernier et plus intense remerciement se devant d'aller à la personne qui a fait le plus gros du travail, je dois donc remercier celle qui a partagé ma vie et supporté mes humeurs variables (et pour tout dire souvent exécrables) durant ces trois années : Muriel, cette thèse t'est dédiée, tu le mérites plus qu'amplement.

Table des matières

Table des figures

"Are you ready for unorthodox procedures ?"
Jack Vance, *Tales of the Dying Earth*

Introduction

Mes dernières vacances d'été furent l'occasion de tester un nouvel appareil photographique numérique. N'étant pas un expert en photographie, c'est un appareil assez bas de gamme offrant quelques mégapixels de résolution ainsi qu'une mémoire d'un giga-octet. En à peine deux semaines, celui-ci fut rempli de quelques 500 photos, bien que loin d'être toutes une réussite parfaite. Ce nombre, en apparence anodin, se trouve être le centre d'un problème nouveau. En effet, cela fait quelques années que lorsque nous partons en vacances, mon amie et moi, nous prenons systématiquement quelques centaines de photos. Lorsqu'un évènement s'y prête, c'est une centaine de plus qui s'y ajoute, sans compter les milliers d'autres gentiment envoyées par des amis. C'en est à un tel point que j'ignore aujourd'hui combien de photos je peux avoir, éparpillées sur mes différents ordinateurs. Leur classement est, pour ainsi dire, inexistant, gardant bien souvent le nom évocateur de "*img_100683.jpg*" dans un dossier estampillé "*photos_20070823*". Vu à quel point les appareils photos numériques ont pris de la place dans les rayonnages des boutiques, j'imagine que cette pratique est généralisée à un grand nombre de personnes ; certaines sans doute encore plus astucieuses que moi dans leur rangement.

Téléphones, appareils photos, webcams ou caméscopes numériques, les sources d'images numériques sont présentes dans tous les foyers et produisent en quantités innombrables de nouvelles collections. L'évolution des espaces de stockage a supprimé le besoin de tri et d'élimination, laissant ces énormes volumes en pagaille. Ces volumes atteignent aujourd'hui des tailles supérieures à ce que l'humain est capable de traiter manuellement. Hors de question de chercher un ensemble de photos en parcourant aléatoirement des dossiers contenant des milliers d'images. Si cette tâche est devenue impossible pour l'être humain, on peut alors penser à utiliser l'outil informatique pour l'automatiser. Cette idée de faciliter l'accès à des données n'est pas neuve, des outils de recherche de fichiers ayant été développés à cet effet depuis la fin des années 70. Ces outils parcourent les arborescences des volumes de données afin de trouver les fichiers dont le nom respecte une certaine expression régulière. Les systèmes de fichiers (la manière dont sont organisés les volumes de données) et les algorithmes de recherche sont très bien optimisés et donnent leurs résultats en des temps records.

Cependant, ces méthodes de recherche exacte sur les noms de fichiers ne sont pas adaptées pour la recherche d'images. En effet, la grande majorité des images numériques ont été nommées automatiquement par les appareils les ayant créées. Les noms des fichiers sont ainsi sans aucun rapport avec la sémantique que l'on associe au contenu des images. Il en est de même pour les méta-données contenues dans les formats d'images

(résolution, facteur de compression, etc.). De fait, il faut réinventer des outils facilitant la recherche dans des grandes bases d'images numériques. Ces outils doivent travailler sur le contenu des images, permettant ainsi de s'approcher de la sémantique qu'on leur associe généralement. Cette problématique touche de nombreux domaines en posant des questions complexes, encore ouvertes aujourd'hui, telles que "Comment mesurer si un chat est présent dans cette image ?" ou encore "Quelle grandeur numérique permet de dire qu'une image est plus similaire à celle-ci qu'à une autre ?". Il s'agit donc de formuler de manière numérique un sens (un concept sémantique) que nous avons l'habitude de manipuler avec des mots (c'est-à-dire de manière symbolique). Cette problématique fait partie du cœur du travail de cette thèse et nous reviendrons plus en détail sur les techniques efficaces de recherche d'image par le contenu au prochain chapitre.

Si les volumes de documents multimédia se sont envolés ces dernières années, que dire alors de l'essor des réseaux informatiques. Depuis son ouverture au public dans les années 90 jusqu'à l'arrivée massive du *"Web 2.0"* tout récemment, Internet est passé d'une vitrine pour quelques professionnels à un média de communication généralisé. Alors que l'on considérait jusqu'à maintenant Internet comme une source de données, il est aussi récemment devenu un moyen de publication et de partage. Entretenir un blog et l'alimenter par des images ou des vidéos (en 2008, 183 millions de personnes possèdent un blog[1]), partager ses exploits photographiques sur des sites de galeries tels que Flickr[2], illustrer ses propos sur un forum par quelques dessins, ou bien étayer d'une illustration un article sur un média collectif comme Agoravox[3] sont des pratiques courantes. Tout ceci nous amène à considérer le web comme un ensemble gigantesque de bases d'images. Effectuer des recherches dans cet ensemble ajoute à la complexité de la recherche d'images précédemment évoquée, celle de la recherche dans des bases distribuées. La problématique de la recherche d'informations dans des bases distribuées n'est pas non plus récente, et on peut découper le problème en deux grands ensembles :

- La sélection des bases pertinentes
- La fusion des résultats de recherche dans chacune des bases sélectionnées

Le problème de cette approche dès lors qu'elle est appliquée au web, c'est d'une part le gigantisme de l'ensemble des bases considéré, et d'autre part l'absence d'outils fiables de description et de sélection des sites pertinents. Il n'existe pas de véritable organisation d'Internet, ni de cartographie sémantique détaillant les catégories de données se trouvant sur chaque site. Autrement dit, sélectionner les sites ou les bases qui contiennent des images pertinentes vis-à-vis de la recherche effectuée n'est pas une tâche aisée.

En considérant la nouvelle extension d'Internet, les réseaux pair-à-pair, on s'aperçoit que la nature distribuée et désorganisée précédemment décrite y est encore plus présente. Dans ce type de réseaux, les liens reliant un site à un autre sont extrêmement volatiles (ils peuvent changer rapidement). Ainsi, il semble difficile voire impossible de tenir à jour une cartographie précise du réseau identifiant les pairs de manière fiable. Une cartographie du

[1] http ://www.caslon.com.au/weblogprofile1.htm
[2] http ://www.flicr.com
[3] http ://www.agoravox.com

contenu du réseau est tout bonnement impossible à faire de manière fiable. La recherche d'information, et plus particulièrement d'images, consiste à déterminer les pairs du réseau possédant des documents relatifs à la requête effectuée. Il s'agit alors de propager une requête vers ces pairs puis de fusionner les résultats obtenus. Dans les deux cas (web ou pair-à-pair), la problématique qui semble émerger est celle de la localisation de l'information (et donc des images) pertinente.

Cette double problématique de la recherche d'images par le contenu et la recherche d'information dans des bases distribuées soulève au fond les deux grandes questions qui font l'objet des travaux de cette thèse. La première est de *savoir où se trouvent sur un réseau les images qui correspondent à la catégorie recherchée par l'utilisateur*. Il s'agit alors de mettre en place des méthodes permettant de trouver les collections (que ce soit les sites ou les pairs) qui contiennent des images pertinentes et de focaliser les recherches vers elles. La seconde est *d'exprimer le concept sémantique auquel l'utilisateur du système pense quand il effectue sa recherche à l'aide des données numériques que l'on peut extraire des images*. Dans les travaux présentés ici, nous nous sommes efforcés de garder une approche système et de mettre en interaction ces deux aspects. En effet, il semble réaliste de considérer que les sites sont spécialisés et contiennent des documents ayant un rapport entre eux. Autrement dit, la distribution des données est liée à la sémantique. Nous faisons de cette hypothèse le point de départ de nos travaux et tentons d'utiliser l'information de localisation afin d'améliorer la recherche de sémantique, et réciproquement, d'utiliser l'information sémantique obtenue sur les documents afin d'en faciliter la localisation.

Pour ce qui est de la première question, le succès des différents moteurs de recherche (*Google* en tête) ces dernières années montre bien l'importance des travaux dans ce domaine. Que ce soit sur le web, dans les systèmes pair-à-pair ou même à l'échelle d'un intranet, l'explosion des volumes de données a fait naître toute une problématique de la recherche de la bonne source d'informations.

En ce qui concerne la deuxième question, le sujet est un des champs les plus actifs du traitement d'images depuis une quinzaine d'années. Les premiers systèmes, tels que QBIC [Niblack et al., 1993], ont utilisé des résumés des couleurs de l'image afin de déterminer la pertinence des images au regard d'une certaine requête. La mesure se fait généralement sur la comparaison des statistiques des couleurs sur les images. La réalisation de systèmes entièrement automatiques, opérant à partir d'une requête sémantique, s'est avérée trop complexe, peu performante ou très limitée, et des systèmes semi-automatiques, fonctionnant en interaction avec l'utilisateur se sont rapidement développés. Le cycle d'interaction consistant en la présentation de résultats et leur annotation par l'utilisateur afin de raffiner la recherche, communément appelé *bouclage de pertinence*, a permis des gains considérables dans la qualité des résultats retrouvés. Dans ce contexte, on dispose à chaque cycle d'un lot d'images et de leur annotation et de techniques d'apprentissage artificiel pour améliorer les résultats.

Dans les travaux présentés ici, nous nous sommes placé dans le schéma classique

d'apprentissage interactif (en ligne) pour la recherche par le contenu dans un petit nombre de bases d'images. Partant de ce contexte d'utilisation du système, nous avons cherché à exploiter l'information donnée par l'utilisateur grâce aux annotations afin d'améliorer la sélection des sites contenant des images intéressantes ainsi que la mesure de pertinence basée sur le contenu des images. Nous avons élaboré des stratégies de sélection des images à présenter à l'utilisateur pour qu'il les annote de manière à réduire le nombre d'itérations du bouclage de pertinence.

Nous nous sommes concentrés sur des méthodes d'apprentissage utilisant des systèmes multi-agents. Nous avons utilisé des programmes autonomes ayant la faculté de se déplacer sur le réseau, appelés *agents mobiles*, afin d'effectuer la recherche d'images. Dans le système ainsi obtenu, les agents mobiles parcourent le réseau à la recherche d'images pertinentes relatives à la requête de l'utilisateur. Dès qu'ils ont trouvé des images pertinentes, ils reviennent sur l'ordinateur de l'utilisateur afin de lui présenter les résultats. Le but est de faire apprendre aux agents d'une part les chemins menant aux collections contenant les images pertinentes, et d'autre part la mesure numérique permettant de discriminer les images pertinentes à partir de leur contenu. L'apprentissage de la mesure de pertinence est basé sur les techniques d'apprentissage automatique utilisées en recherche d'images. L'apprentissage des chemins pertinents est issue de l'émergence d'une optimisation à travers les interactions entre les agents. Les comportements des agents sont inspirés de ceux des insectes sociaux, en particulier des fourmis, et consistent à collectivement marquer l'environnement (le réseau) afin de faire apparaître les chemins pertinents. Nous avons essayé de maintenir un couplage entre ces deux apprentissages, de sorte que l'apprentissage du réseau ait une influence sur l'apprentissage des mesures de similarité et réciproquement.

Nos contributions se décomposent en deux grandes parties. La première fait l'objet de recherches sur l'élaboration d'une stratégie de recherche interactive dans un contexte distribué, et montre comment les méthodes d'apprentissage actif peuvent être étendues à des collections distribuées à l'aide d'un système multi-agents. Nous avons effectué de nombreuses expériences sur des bases de grandes tailles (75 000 images pour la base *TrecVid'05*) et sur de nombreux paramètres du système. Ces contributions ont été publiées dans [Picard et al., 2006a, Picard et al., 2006b, Picard et al., 2008a].

La seconde partie de nos contributions se consacre à la réutilisation des informations obtenues lors des précédentes sessions d'utilisation du système afin d'améliorer les futures recherches. Nous avons ainsi développé une stratégie interactive à long terme en étendant les capacités de notre système multi-agents. Des expériences ont été conduites afin de bien déterminer le comportement du système proposé, les gains qu'il apporte, mais aussi ses limitations. Ces travaux ont fait l'objet de publication dans [Picard et al., 2008c, Picard et al., 2008b].

Le présent manuscrit se déroule comme suit : une première partie en trois chapitres regroupe l'état de l'art et la description de notre système. Un premier chapitre décrit les outils de recherche d'images par le contenu afin de bien garder à l'esprit le point de départ de nos travaux, tandis que le deuxième fait le point sur les systèmes multi-agents

d'inspiration éthologique afin de présenter les méthodes que nous avons utilisées pour répondre à la problématique. Enfin, un troisième chapitre présente nos motivations quant à l'utilisation des différents outils énoncés précédemment pour la recherche d'images dans des bases distribuées et se poursuit par une description approfondie du système proposé. Puis, une seconde partie en deux chapitres est consacrée à l'adaptation des techniques interactives de recherche d'images au contexte distribué. Le quatrième chapitre développe les algorithmes d'apprentissage actif proposés pour notre système, tandis que le chapitre cinquième détaille les expériences et les résultats obtenus pour ces algorithmes. Finalement, une troisième partie, aussi en deux chapitres, expose l'extension de nos travaux à un contexte multi-utilisateurs dans lequel on dispose de plusieurs sessions de recherche. Le chapitre sixième explicite les algorithmes d'apprentissage à long-terme proposés. Il s'agit de fusionner les informations obtenues par les différents utilisateurs aux cours de leurs sessions de recherche afin d'en bénéficier pour les recherches ultérieures. Le septième chapitre présente les expériences et les résultats relatifs à ces algorithmes. Enfin, un chapitre de conclusion générale de nos travaux, ainsi que quelques perspectives en continuation de ceux-ci, termine ce mémoire.

Première partie

État de l'art et motivations

Chapitre 1

Recherche d'images

La première question soulevée par la problématique de cette thèse est celle de la recherche d'images basée sur le contenu (*CBIR* pour *Content Based Image Retrieval*). L'idée générale de la recherche par le contenu est d'extraire de l'image (considérée comme un tableau à deux dimensions de pixels) un résumé exprimant de manière numérique la sémantique que l'on associe à l'image. Ces résumés sont comparés afin de déterminer une similarité entre les images. Cette similarité est utilisée soit à des fins d'indexation (rangement, classement) ou bien de recherche (trouver des résultats en fonction d'une requête donnée).

Afin d'appuyer nos travaux effectués sur cet axe, nous détaillons dans ce chapitre l'état de l'art en matière de recherche d'images par le contenu. Nous commençons par présenter en détail les mécanismes mis en œuvre dans la recherche d'images par le contenu, puis nous analysons quelques systèmes afin de voir l'intégration de ces mécanismes au sein d'une architecture unie. Nous étendons ensuite cette description à la recherche dans des bases distribuées et montrons les limitations des approches classiques lors de leur application à la recherche d'images par le contenu. Enfin, nous clôturons ce chapitre par la présentation des méthodes fonctionnant à long-terme, c'est-à-dire des méthodes qui permettent de réutiliser l'information fournie par les utilisateurs d'une session sur l'autre.

1.1 Mécanismes de la recherche d'images

Dans ce que l'on entend par recherche d'images par le contenu, on peut distinguer trois paradigmes différents en fonctions des objectifs de l'utilisateur, tels que définis par Smeulders [Smeulders et al., 2000] :

- la recherche associative
- la recherche de cibles
- la recherche de catégories

La recherche associative consiste à aider l'utilisateur à explorer la base d'images, alors qu'il n'a pas de but véritablement défini. Le procédé généralement utilisé consiste

à présenter un ensemble d'images à l'utilisateur et, via une interaction, à affiner avec lui l'objet de sa recherche. Le second paradigme consiste à retrouver une image particulière dans une base, comme par exemple un tableau bien précis dans une collection de musée. Il s'agit typiquement des problématiques de détection de copies. L'image recherchée peut avoir subi des transformations géométriques ou colorimétriques et les méthodes tentant de résoudre ce problème se basent généralement sur une certaine robustesse des algorithmes à ces transformations. Le dernier paradigme est celui qui nous intéresse dans ce manuscrit. L'utilisateur a en tête une catégorie, comme *"voiture"* ou *"éléphant"* (figure 1.1), et le système doit retrouver un maximum d'images correspondant à cette catégorie.

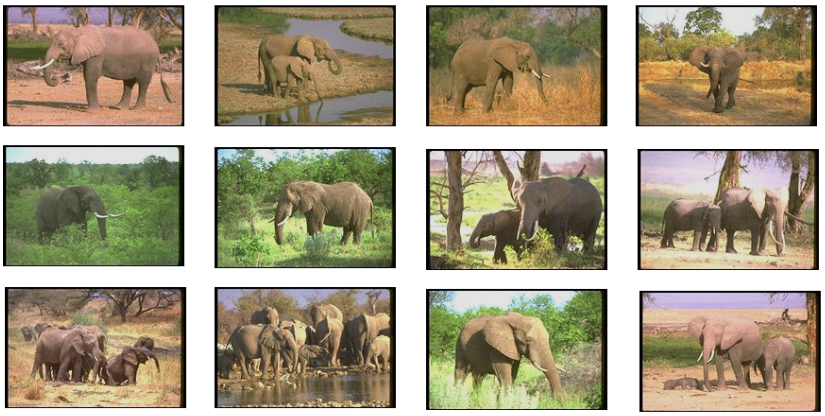


Fig. 1-1. Exemple d'une catégorie contenant des éléphants. On notera la grande variabilité à l'intérieur de la catégorie : un ou plusieurs éléphants, type de fond, angle de la photo, luminosité...

Cependant, les images numériques sont stockées sur les ordinateurs sous la forme de fichiers aux formats souvent bruts. Ceux-ci ne contiennent généralement que les données nécessaires à la construction d'une visualisation de l'image. Dans le cas du format *jpeg*, par exemple, ces données sont une transformation en fréquence des pixels de l'image. Ces formats présentent l'avantage d'être liés à des méthodes de compressions efficaces permettant de stocker un très grand nombre d'images. Ils ont le très gros inconvénient de ne contenir que peu, si ce n'est aucune, information sur l'interprétation sémantique du contenu (à la différence de certains formats multimédia comme le *mp3* qui embarquent des *"tags"* décrivant le style, l'interprète, etc.). Ainsi, les seules informations directement disponibles dans une image numérique afin d'en déterminer le type sont le nom du fichier et les données stockées. Dès lors, l'idée de la recherche d'images par le contenu est d'extraire de l'information des pixels de l'image et de se baser sur cette information pour effectuer la recherche.

1.1.1 Architecture des systèmes

L'idée d'extraire une sémantique du contenu de l'image afin d'indexer de grandes bases d'images est née avec le projet *QBIC* [Niblack et al., 1993]. Le procédé général est

le suivant : dans un premier temps, des caractéristiques visuelles sont extraites directement des pixels de l'image, telles par exemple les couleurs, les formes ou bien les textures.

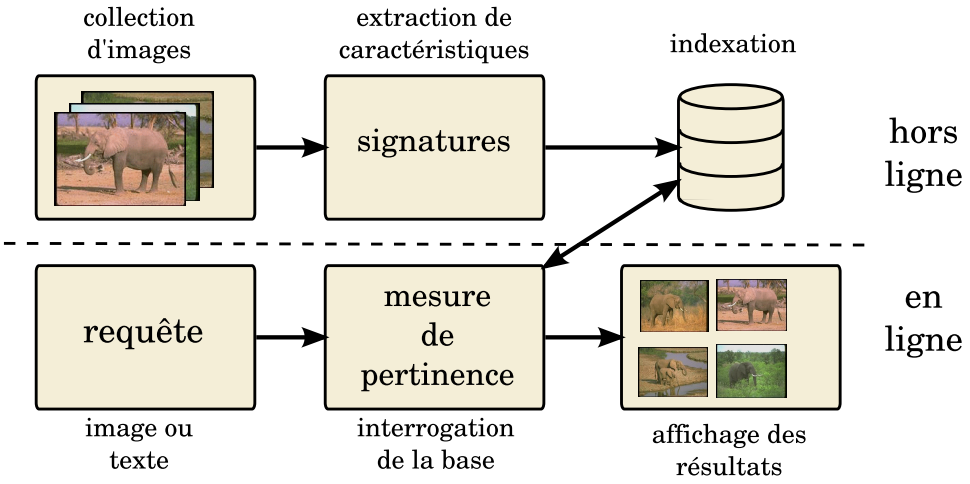


Fig. 1-2. Déroulement de la recherche d'images par le contenu. La première étape, hors-ligne, consiste à extraire des caractéristiques visuelles d'une collection d'images et à les indexer. La seconde étape se déroule en ligne et considère une requête formulée par l'utilisateur, dont elle compare la signature à celles indexées au moyen d'une mesure de pertinence afin d'afficher les résultats pertinents à l'utilisateur.

Le challenge est d'extraire des caractéristiques qui soient pertinentes au regard de ce que l'on attribue comme sémantique à l'image. Dans un deuxième temps, il s'agit de construire des signatures à partir des caractéristiques extraites afin de pouvoir mesurer des similarités entre signatures et donc entre images. En effet, les caractéristiques obtenues ne sont pas nécessairement des objets mathématiques facilement manipulables. Il faut alors générer sur la base de ces caractéristiques, des signatures que l'on peut comparer afin de déterminer les images proches de ce que l'utilisateur recherche.

Lors d'une session de recherche, l'utilisateur exprime sa requête au système (le plus souvent à partir d'une image exemple, ou bien en utilisant un mot-clé). À l'aide d'une mesure de similarité entre les signatures, la base (ou *collection*) d'image est triée et des résultats sont proposés à l'utilisateur (Fig. 1-2).

1.1.2 Caractéristiques visuelles

Dans sa thèse, Nuno Vasconcelos [Vasconcelos, 2000] appelle *"caractéristiques"* le résultat d'une transformation de l'espace des pixels vers un espace d'observation. Les couleurs, les formes ou les textures présentes dans une images sont des caractéristiques visuelles courantes. Ces observations vont servir à retrouver la cible, la catégorie d'images, ou a produire les associations entre images, en fonction du paradigme choisi par le système de recherche.

Il y a principalement deux approches pour les caractéristiques qui peuvent être extraites. La première est la construction de descripteurs globaux à toute l'image. Dans ce

cas, il s'agit de fournir des observations sur la totalité de l'image. L'avantage des descripteurs globaux sont la simplicité des algorithmes mis en œuvre, et le nombre réduit d'observations que l'on obtient (ce qui peut se répercuter avantageusement sur les calculs qui seront effectués par la suite). Cependant, l'inconvénient majeur des descripteurs globaux est la perte de l'information de localisation des éléments de l'image, comme par exemple le fait qu'un bateau est toujours au dessus de l'eau ou bien qu'une moto est normalement toujours en dessous du motard. Pour un objet relativement petit dans l'image, le descripteur contiendra principalement l'information sur l'arrière-plan qui peut être non pertinente.

La seconde approche est locale et consiste à calculer des attributs sur des portions restreintes de l'image. L'avantage des descripteurs locaux est de conserver une information localisée dans l'image, évitant ainsi que certains détails (par exemple un petit objet tel que des lunettes) ne soient noyés par le reste de l'image. L'inconvénient majeur est que la quantité d'observations produite est très grande, ce qui implique un gros volume de données à traiter.

Le choix des caractéristiques extraites est souvent guidé par une volonté d'invariance (ou de robustesse) par rapport à des transformations de l'image (géométriques, radiométriques, etc).

1.1.2.1 Couleur

La couleur est très utilisée dans les systèmes d'indexation d'images [Smeulders et al., 2000, Veltkamp, 2002]. Les images numériques sont stockées dans l'espace couleur *rgb*, qui, malgré ses avantages pour la projection des images sur les différents appareils de visualisation (moniteurs, écrans, dalles LCD, projecteurs), ne correspond pas aux caractéristiques de la vision humaine. Il existe bien d'autres espaces de représentation de la couleur, tels que YC_rC_b, HSV ou encore *Lab*. L'espace YC_rC_b comporte une composante de luminance (intensité Y), et deux composantes de chrominance (C_r et C_b). Cet espace présente des avantages en matière de codage de l'image, car l'œil est plus sensible à l'intensité qu'à la couleur, ce qui permet d'accorder plus de bits à la composante de luminance qu'aux autres composantes (c'est le cas notamment en vidéo - ce format étant un standard développé par l'industrie télévisuelle sous le nom *ITU-R BT.601*).

L'espace HSV comporte une composante d'intensité (V - *Value*), une composante de couleur pure (H - *Hue*, teinte) et une composante de saturation (quantité de blanc S - *Saturation*). Cet espace est assez proche de la perception humaine de la couleur, ce qui en fait un bon candidat pour l'extraction de caractéristiques [Smith and Chang, 1996]. L'espace *Lab* est très proche de l'espace HSV à ceci près que les couleurs sont distribuées uniformément dans un cube de $\mathbb{R}^3$ [Carrilero, 1999]. Il est alors très simple de mesurer des distances entre les couleurs.

Chacune des composantes est codée sur huit bits, ce qui donne un total de 2^{24} couleurs possibles. Le nombre de caractéristiques extraites par un décompte des occurrences de chacune des couleurs est donc très grand.

Concernant la robustesse aux transformations de l'image, on peut aussi définir des espaces colorimétriques qui présentent de bonnes propriétés concernant l'invariance à une illumination non homogène, colorée ou aux propriétés de réflexions des objets de la scène (comme les éléments métalliques, par exemple) [Geusebroek et al., 2001]. Ces modèles sont cependant complexes à mettre en œuvre, et dépendent des images à traiter (objets en gros plan, paysages, ...).

1.1.2.2 Texture

Il n'y a pas de définition unique de la texture. Elle représente les zones d'une image n'ayant pas une couleur unie et que pourtant l'on considère comme "homogène" ou "cohérente" à l'œil. Les primitives formant cette zone de l'image peuvent prendre un aspect aléatoire ou bien répétitif. Par exemple, un mur de briques possède une texture clairement identifiable à l'œil, tout comme un morceau de marbre, alors que celui-ci ne possède pas de motifs répétables.

Il existe de nombreuses méthodes de caractérisation de la texture, comme les matrices de cooccurrences [Aksoy and Haralick, 1998], la décomposition paramétrique Wold [Liu and Picard, 1996] ou les filtres de Gabor [Gabor, 1946]. Les filtres de Gabor, outre le fait d'être relativement proches de ce qui se passe dans la vision humaine [Petkov and Kruizinga, 1997], peuvent être vus comme *"des détecteurs de coins et de lignes paramétrables en orientation et en échelle"* [Manjunath and Ma, 1996]. Un paramètre d'orientation permet de fixer la direction principale de la texture et un paramètre de fréquence son échelle. En prenant par exemple quatre orientations et trois échelles, on obtient alors douze filtres et autant d'images réponses. En codant chaque réponse d'un pixel sur huit bits (ce qui correspond au codage de la valeur d'intensité du pixel), on obtient $2^{8\times12}$ valeurs possibles, soit un nombre de caractéristiques immense.

Tout comme la couleur, la texture est très utilisée dans les systèmes d'indexation d'images [Ma and Manjunath, 1995, Pala and Santini, 1999, Randen and Husoy, 1997, Wolf et al., 2000].

1.1.2.3 Descripteurs locaux

Il existe deux grandes catégories de descripteurs locaux : les régions et les points d'intérêt. Les régions sont produites à partir d'un partitionnement de l'image (segmentation). Il existent énormément de méthodes de segmentation, cette problématique étant très étudiée en recherche d'images [Cocquerez and Philipp, 1995, Guigues et al., 2006]. Cependant, il est difficile d'évaluer la qualité d'une segmentation, car on ne dispose pas d'une vérité terrain sur les images naturelles [Philipp-Foliguet and Guigues, 2006]. On affecte alors des descripteurs à chacune des régions, qui peuvent être des distributions de couleurs, de textures ou bien des attributs de forme. L'avantage indéniable de cette approche locale est sa pertinence lorsque l'on recherche un objet dans l'image. Pour peu que la segmentation ait été bien faite, les attributs de la ou les régions comprenant l'objet ne contiennent majoritairement que de l'information relative à cet objet.

Les points d'intérêt sont des points de l'image extraits par un détecteur auquel on affecte un descripteur caractérisant le point. Les point extraits sont généralement les points ayant une forte information de contour, comme les parties anguleuses [Harris and Stephens, 1988]. Un autre extracteur typique est un filtre en différence de gaussiennes qui approxime le laplacien de l'image (détecteur *SIFT*). L'avantage de ces extracteurs est d'être relativement stables et de donner des points à des positions relatives proches pour des objets similaires. Les caractéristiques associées aux points peuvent être calculées sur un voisinage. Le descripteur SIFT [Lowe, 2003], par exemple, contient un histogramme des gradients dans un voisinage du point. Ce descripteur a l'énorme avantage d'être robuste à de nombreuses transformations de l'image, comme des translations ou des rotations, ou encore des variations dans l'acquisition de l'image (changement de point de vue, par exemple). Les points d'intérêt sont en général très discriminants et fonctionnent très bien dans le cadre de la recherche de cible (chercher exactement le même objet), mais, de fait, n'ont pas assez de pouvoir de généralisation pour faire de la recherche de catégories.

1.1.3 Représentation des données

À partir des caractéristiques extraites, il faut produire une signature. Selon le type d'objets mathématiques obtenus (vecteurs, graphes, ...), il reste alors à définir une fonction de similarité mesurant la proximité des images.

1.1.3.1 Signatures des images

Le moyen le plus simple pour produire une telle signature est de faire l'histogramme des primitives, c'est-à-dire de compter les occurrences de chacune des primitives dans l'image ou la portion de l'image [Stricker and Orengo, 1995, Brunelli and Mich, 2001]. Cependant, les caractéristiques extraites sont, comme on l'a vu précédemment, composées d'un très grand nombre de primitives. La signature obtenue est alors de très grande dimension.

Les approches les plus courantes pour la conception de signatures sont celles définissant un dictionnaire visuel. On quantifie l'espace des primitives, puis on calcule l'histogramme à partir des représentants. Le dictionnaire visuel est constitué de l'ensemble des représentants. Cette quantification peut être fixe ou bien adaptée à la base afin d'obtenir un dictionnaire plus représentatif des données [Fournier et al., 2001a]. L'histogramme est alors rangé dans un vecteur de $\mathbb{R}^N$, où N est le nombre d'éléments du dictionnaire et où chaque dimension est le nombre d'occurrences correspondant à chacune des entrées du dictionnaire visuel. La figure 1.1.3.1 illustre l'effet de la quantification des couleurs à une palette réduite sur une image.

Dans le cas où plusieurs types différents de caractéristiques ont été extraits (couleurs et textures, par exemple), on peut tout simplement concaténer les vecteurs résultant de chacune des caractéristiques. Ce procédé s'appelle la fusion précoce [Snoek et al., 2005].

Fig. 1-3. Exemple d'image dont chaque pixel à été remplacé par l'entrée du dictionnaire couleur correspondante.

Les approches basées sur la construction d'un dictionnaire visuel sont les plus utilisées en recherche d'images par le contenu, y compris pour les systèmes basés sur des caractéristiques locales [Jurie and Triggs, 2005, Philbin et al., 2007].

Les signatures peuvent être des objets plus complexes que des vecteurs, comme par exemple un ensemble de vecteurs (on parle alors de *sac* de vecteurs) ou bien des graphes. Dans le cas de caractéristiques locales comme des points d'intérêt, on peut avoir un sac regroupant les signatures de tous les points d'intérêt de l'image [Gosselin et al., 2007a]. Dans ces conditions, la fonction de similarité à définir est moins intuitive que dans le cas vectoriel.

1.1.3.2 Similarité

À partir des signatures $\mathbf{x}_i$ et $\mathbf{x}_j$ de deux images i et j on souhaite définir une fonction s mesurant la similarité entre ces deux signatures.

La notion de similarité est un domaine vaste. On peut néanmoins considérer la mesure de distance entre les signatures comme une approche intuitive. Il existe de très nombreuses distances, dépendant de la nature des signatures à comparer. Pour les vecteurs, on peut citer notamment les distance L_1, L_2 ou L_∞. Pour des histogrammes, on peut soit les considérer comme des vecteurs et utiliser des distances vectorielles, soit utiliser des distance spécifiquement conçues, comme la *earth mover distance* [Rubner et al., 1998]. Pour les distributions, on peut utiliser des distances statistiques, comme la distance du χ^2.

Dans le cas où $\mathbf{x}_i$ et $\mathbf{x}_j$ sont des vecteurs de $\mathbb{R}^N$, une fonction de similarité très simple est un produit scalaire $\langle ., . \rangle$ dans $\mathbb{R}^N$:

$$s(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle \tag{1.1}$$

Pour pouvoir utiliser un produit scalaire dans le cas où les signatures ne sont pas des vecteurs, il faut d'abord projeter les signatures dans un espace hilbertien à l'aide d'une fonction d'induction ϕ, puis opérer un produit scalaire dans cet espace :

$$s(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle \tag{1.2}$$

1.1.3.3 Noyaux

Cette transformation ϕ peut aussi permettre de "réarranger" les données de manière à ce que la similarité soit plus satisfaisante par rapport à la sémantique associée aux images.

On peut faire la combinaison des deux étapes (ϕ puis $\langle ., . \rangle$) en une seule fonction appelée *noyau* [Smola and Scholkopf, 2002] :

$$k(\mathbf{x}_i, \mathbf{x}_j) = s(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle \tag{1.3}$$

La théorie des noyaux est un formalisme puissant qui permet de définir des similarités entre des éléments d'une manière très élégante, dont l'intérêt est triple. En premier lieu, il masque la fonction d'induction ϕ, ce qui évite d'avoir à expliciter l'induction ϕ vers un espace hilbertien. Ensuite il permet de travailler des objet éventuellement non vectoriels en entrée. Enfin, il permet de modifier la représentation des données (les signatures) de façon non linéaire (c'est à dire que la fonction ϕ peut être non linéaire).

La conception de noyaux est un problème complexe faisant l'objet de nombreuses études. Il existe des travaux tentant de créer des noyaux ayant des propriétés intéressantes, comme par exemple l'invariance en échelle.

On peut citer quelques exemples de noyaux sur des vecteurs :
- le noyau linéaire : $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$
- le noyau polynomial : $k_d(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j)^d$
- le noyau gaussien (avec une distance L_2) : $k(\mathbf{x}_i, \mathbf{x}_j) = \exp^{-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma^2}}$

Récemment, des noyaux ont été définis pour différents types de signatures, comme les sacs de points d'intérêts ou de régions [Gosselin et al., 2007b, Gosselin et al., 2007a] ou bien les graphes [Suard, 2006].

1.1.4 Recherche

Une fois les caractéristiques visuelles extraites, les signatures et la mesure de similarité associée construites durant la phase hors-ligne, la phase de recherche, qui se fait en-ligne, peut commencer. Dans notre cas de recherche de catégories, l'expression la plus simple du schéma de la recherche consiste en une requête posée par l'utilisateur au système, suivi de la réponse du système.

1.1.4.1 Requête

L'expression de la requête qui marque le commencement de la recherche est un point très important des systèmes de recherche d'images par le contenu. Il existe plusieurs manières d'exprimer cette requête.

Textuelle Le texte est la manière la plus naturelle d'exprimer une requête. En effet, lorsque l'on recherche une catégorie d'images, un mot-clé ou un ensemble de mots-clés est le moyen le plus facile de décrire la sémantique associée à la catégorie. Cependant, si les images ne sont pas accompagnées d'annotations textuelles, il est difficile de traduire la requête dans l'espace des signatures qui ont servi à indexer la base, de manière à calculer une similarité entre la requête de l'utilisateur et le reste de la base. [Wang and Ma, 2005] proposent par exemple d'utiliser un réseau de neurones afin d'apprendre l'association entre les caractéristiques des images et un ensemble de mots-clés prédéfinis. Une autre approche consiste à utiliser comme requête l'image (ou les images) renvoyée(s) par un moteur de recherche web pour le mot-clé donné.

Image De nombreux systèmes de recherche d'images par le contenu préfèrent se débarrasser de cette étape de traduction et commencer directement la recherche à l'aide d'une image exemple (*query by example* en anglais). Dans ce cas, l'utilisateur présente une image sur laquelle le système calcule une signature avec la même méthode que celle utilisée pour l'indexation de la base. C'est cette signature qui fait office de requête.

Lot d'images Le point de départ de la recherche peut être un lot d'images représentant le concept recherché. Dans ce cas, la requête peut être construite de manière géométrique (par exemple en considérant le barycentre des images de la requête), ou bien de manière statistique (estimateur de densité, *One Class SVM* [Chen et al., 2001]). Dans le cas où on dispose d'un lot d'images annotées, certaines appartenant à la catégorie recherchée, certaines ne lui appartenant pas, on peut alors utiliser des techniques issues de l'apprentissage artificiel afin de construire une estimateur de la pertinence des images de la base au regard de la catégorie recherchée.

1.1.4.2 Tri et affichage des résultats

Une fois la requête traduite de manière à pouvoir estimer la similarité de chacune des images par rapport à la requête, un tri est effectué des images les plus pertinentes aux moins pertinentes. Dans le cas où la requête s'exprime comme un élément de l'espace des signatures, la mesure de similarité est directement utilisée pour former le tri, et ce sont les images les plus similaires qui sont présentées en premier. Le résultat de ce tri est affiché à l'utilisateur.

1.1.4.3 Bouclage de pertinence

Dans certains systèmes, l'utilisateur a la possibilité d'annoter les résultats qui lui sont présentés, c'est ce que l'on appelle le bouclage de pertinence (*relevance feedback*) [Rui et al., 1998, Rui and Huang, 2000, Huang and Zhou, 2001]. Dans ce cas, on se retrouve avec une requête contenant un lot d'images et les annotations qui leur sont associées. Le système peut alors mettre à jour la fonction mesurant la pertinence des images de la base et réordonner la base avec cette nouvelle mesure. L'avantage du bouclage de

pertinence est de pouvoir affiner la pertinence de chaque image par rapport à ce que l'utilisateur recherche réellement. Le fait de disposer d'un lot d'images et de leurs annotations permet d'utiliser des algorithmes issus des méthodes d'apprentissage artificiel. Cependant, à la différence de l'apprentissage classique, la base d'apprentissage est construite en ligne à partir d'exemples choisis depuis la base d'évaluation.

1.1.4.4 Apprentissage en ligne

Afin de construire la mesure de pertinence exprimant une requête utilisant un lot d'images ou bien dans le cadre du bouclage de pertinence, on peut utiliser des méthodes d'apprentissage. En classification, disposant d'un ensemble d'images $\{\mathbf{x}\}$, et de leur classe $\{y\}$ (pertinente, non-pertinente), on désire construire une fonction f prédisant la classe y d'une nouvelle image $\mathbf{x}$:

$$f : \mathbf{x} \longrightarrow y \tag{1.4}$$

Il existe une grande quantité d'algorithmes d'apprentissage pour ce contexte, on peut citer notamment les réseaux bayesiens, les réseaux de neurones, le boosting, les machines à vaste marge (*support vector machines - SVM*) [Cortes and Vapnik, 1995].

Une approche consiste à modéliser la pertinence par une probabilité de chaque image d'être pertinente. L'apprentissage consiste alors à estimer la densité de probabilité sur l'ensemble de la base. La majorité des système reposant sur ce principe utilise un mélange de gaussiennes [Vasconcelos, 2000, Najjar et al., 2003].

Dans le cas des réseaux de neurones, les neurones d'entrées sont les valeurs de chacune des composantes du vecteur de signatures (dans le cas où les signatures sont des vecteurs). Le réseau peut disposer de plusieurs couches cachées et possède un seul neurone de sortie dont l'activité représente le degré d'appartenance à la classe.

Le boosting [Tieu and Viola, 2000] fonctionne à partir d'un ensemble de classifieurs mineurs (par exemple un seuillage sur une dimension des vecteurs de signatures), dont il détermine le meilleur du point de vue de l'erreur de classification sur l'ensemble d'apprentissage. Les exemples mal classés sont pondérés de manière à être plus importants dans le calcul de l'erreur, et un second classifieur mineur est choisi comme étant celui qui corrige au mieux les erreurs du premier. Plusieurs classifieurs mineurs sont ainsi itérativement sélectionnés et le classifieur final est construit comme étant la combinaison linéaire habilement choisie de tous les classifieurs mineurs.

L'utilisation des *SVM* a récemment permis de grandement améliorer les performances des systèmes de recherche d'images [Chapelle et al., 1999, Chen et al., 2001, Zhang et al., 2001]. Ce classifieur construit un hyperplan séparant les données de manière à maximiser la distance entre les exemples d'apprentissage et la frontière. Cette frontière est écrite comme une combinaison linéaire des exemples d'apprentissage :

$$f(\mathbf{x}) = \sum_i \alpha_i y_i k(\mathbf{x}, \mathbf{x}_i) \tag{1.5}$$

Avec $\mathbf{x}$ une image de la base et $(\mathbf{x}_i, y_i)$ les images et leurs annotations formant la requête. Il existe un bon nombre d'algorithmes d'optimisation permettant d'obtenir les α_i maximisant la marge entre les deux classes. Alors que les *SVM* sont utilisés dans le cadre de la classification (où seul le signe de f est utilisé), c'est la distance à la marge qui sert de pertinence. Dans ce cas, f est normalisée de manière à donner des valeurs comprises dans $[-1; 1]$, -1 étant le moins pertinent et 1 le plus pertinent.

Il existe aussi des méthodes cherchant directement à optimiser le tri des documents. On sort alors du cadre de la classification. Ces méthodes ont été appliquées aux documents structurés (comme le *XML*) [Usunier, 2006]. On peut également chercher à maximiser la qualité du tri des documents, plutôt que de minimiser l'erreur de classification [Cord et al., 2008].

1.1.4.5 Stratégie active

Lorsque la catégorie recherchée dans un système de recherche d'images n'est pas connue à l'avance, il est alors impossible d'utiliser un ensemble d'apprentissage *a priori* pour construire le classifieur. Dans ce cas, les systèmes de recherche d'images fonctionnent de manière interactive avec l'utilisateur [Rui et al., 1998, Santini et al., 2001]. Un certain nombre d'images lui sont présentées et celui-ci les annote (pertinentes, non-pertinentes). Ainsi, l'ensemble d'apprentissage est construit itérativement lors du bouclage de pertinence.

On sort alors du cadre classique de l'apprentissage supervisé. On ne dispose pas d'une base d'apprentissage et d'une base de test pour l'entraînement du classifieur, puis d'une base sur laquelle les données vont être évaluées, mais d'une base unique dans laquelle certains des éléments vont servir à l'entraînement du classifieur. Dans ces conditions, et en prenant en compte les attentes de l'utilisateur, la problématique de l'apprentissage est plutôt de trouver les meilleurs éléments de la base du point de vue de l'entraînement du classifieur en un minimum de labels [Lewis and Catlett, 1994, Cohn, 1996]. Afin de répondre à cette problématique, une stratégie de sélection des exemples à faire annoter par l'utilisateur, stratégie dite *active*, est mise en place [He et al., 2004, Park, 2000, Lindenbaum et al., 2004, Tong and Chang, 2001].

La stratégie la plus courante est basée sur l'incertitude. Il s'agit de faire annoter par l'utilisateur les images dont le classifieur est le plus incertain afin de lever cette ambiguïté. Dans le cas de classifieur mesurant la probabilité d'une image d'appartenir à la classe recherchée, on prendra ainsi les images avec une probabilité la plus proche de 0.5. Dans le cas du SVM, il s'agit de prendre les images non annotées les plus proches de la marge. Cette stratégie montre de meilleurs résultats que la sélection des exemples les plus pertinents à chaque itération du bouclage de pertinence [Ferecatu, 2005].

1.2 Aperçu des systèmes existants

Dans cette partie nous détaillons quelques systèmes de recherche d'images par le contenu. Nous les examinons par ordre chronologique afin de voir l'évolution des techniques employées, jusqu'à l'apparition des méthodes d'apprentissage qui nous intéressent particulièrement pour nos travaux. Nous commençons par le projet *QBIC* [Niblack et al., 1993] vers 1995, puis nous détaillons *SIMPLIcity* [Wang et al., 2001] développé en 2000, puis nous terminons par le système *RETIN* développé par le laboratoire ETIS [Fournier, 2002, Gosselin, 2005].

1.2.1 QBIC

Le système *QBIC* (*Query by Image Content*) [Niblack et al., 1993] est l'un des tous premiers systèmes de recherche d'images par le contenu. Il a été développé par *IBM* au début des années 90 et sert encore aujourd'hui à l'indexation des images du musée de l'Hermitage à Saint-Petersbourg.

Caractéristiques

QBIC utilise des caractéristiques de couleurs prises sous la forme d'un histogramme de 256 couleurs dans l'espace *RGB*. Les textures utilisées se basent sur les notions de *granularité*, *contraste* et *orientation* définies par [Tamura et al., 1978]. *QBIC* utilise aussi des descripteurs de formes qui consistent en l'axe principal et l'excentricité calculés sur la matrice de covariance des pixels de contour. Les contours sont extraits de manière semi-automatique : l'utilisateur dessine une forme grossière qui est affinée à l'aide d'algorithmes de contours actifs.

Signature et similarité

Les caractéristiques extraites sont mises sous forme d'histogrammes. Deux mesures sont appliquées afin de déterminer la pertinence des images de la base. Dans une première passe, une distance euclidienne pondérée entre les vecteurs moyens de couleurs est appliquée afin de sélectionner un nombre réduit d'images. Dans la deuxième passe, une distance L_2 pondérée est appliquée.

Recherche

QBIC permet d'opérer des requêtes soit en présentant une image, soit en dessinant des formes de couleurs. Les résultats sont affichés par ordre décroissant de pertinence et n'importe quelle image affichée en résultat peut être réutilisée comme point de départ d'une recherche afin d'obtenir de meilleurs résultats.

Ce système reste assez rudimentaire au niveau de la mesure de similarité entre images, et ne permet pas de faire des requêtes complexes avec de l'apprentissage interactif. Cependant, il a réussi à fédérer une communauté dans le domaine de la recherche multimédia par le contenu, qui trouve aujourd'hui son expression dans la campagne d'évaluation *TrecVid*[1].

1.2.2 SIMPLIcity

SIMPLIcity est un système développé par James Wang [Wang et al., 2001] qui reprend toutes les avancées faites en recherche d'images par le contenu à la fin des années 90. Le fait qu'il soit relativement populaire lui assure sa place dans cet état de l'art.

Fig. 1-4. Capture d'écran de la version web de *SIMPLIcity* pour une requête de la catégorie éléphant. Les résultats semblent proches au niveau des teintes de couleurs, cependant, peu contiennent réellement des éléphants.

Caractéristiques visuelles

SIMPLIcity effectue une segmentation de l'image en régions. Pour cela, l'image est découpée en blocs de quatre pixels, et un vecteur de caractéristiques est associé à chacun de ces blocs. Ce vecteur comporte trois composantes couleurs et trois composantes textures (coefficients d'une transformée en ondelettes). Un algorithme de *k-means* est appliqué sur ces blocs afin d'obtenir les régions. La signature finale de l'image est un sac de régions. Chaque région a pour signature le vecteur moyen de dimension six de l'ensemble des pixels de la région.

[1] http ://www-nlpir.nist.gov/projects/trecvid/

Les images de la base sont ensuite classées en plusieurs catégories sémantiques : *texturées*, *non-texturées*, *dessinées*, *naturelles*. Un classifieur spécifique est construit pour chacune de ces classes.

Similarité

Pour mesurer la similarité entre deux images, les régions contenues dans les sacs sont appariées deux-à-deux sur la base d'une pertinence d'association suffisamment forte entre les deux. Ainsi, une région de la première image peut être associée à plusieurs autres régions de la seconde image et inversement. Chacune de ces associations est pondérée par la pertinence de l'association calculée entre les régions. La similarité globale entre deux images est la somme des similarités entre régions pondérée par la pertinence des associations. Cette similarité dépend de la catégories sémantique à laquelle appartiennent les images.

Recherche

Pour démarrer la recherche, l'utilisateur présente une image exemple de la catégorie recherchée. Si celle-ci appartient déjà à la base, la similarité avec les images de la base (en utilisant les mesures associées à la catégorie sémantique détectée) est calculée et les résultats sont renvoyés par ordre décroissant de similarité. Si l'image n'appartient pas à la base, elle est segmentée afin d'extraire un sac de régions caractéristiques. Puis elle est classée dans l'une des quatre catégories sémantiques et le système continue comme si l'image appartenait à la base.

SIMPLIcity ne dispose pas de système de bouclage de pertinence ni de techniques d'apprentissage sophistiquées et est pourtant diablement efficace, comme en témoigne la démonstration disponible sur le site du projet (voir figure 1.2.2). Plus récemment, ces travaux ont été étendus par la prise en compte de méthodes d'apprentissage sur des sacs (*Multiple Instance Learning*), appliquées ici à des ensembles de régions [Chen et al., 2004].

1.2.3 RETIN

RETIN est le système de recherche d'images par le contenu développé au laboratoire ETIS. Il a fait l'objet de nombreuses recherches tant au niveau du choix des descripteurs et des mesures de similarité qu'au niveau des algorithmes d'apprentissage. Les travaux de cette thèse concernant la partie image se basent en partie sur RETIN, c'est pourquoi il convient de le détailler ici.

Caractéristiques visuelles

Les caractéristiques visuelles utilisées sont des couleurs et des textures. Les couleurs sont le résultat d'une quantification de l'espace colorimétrique *Lab* sur un nombre de

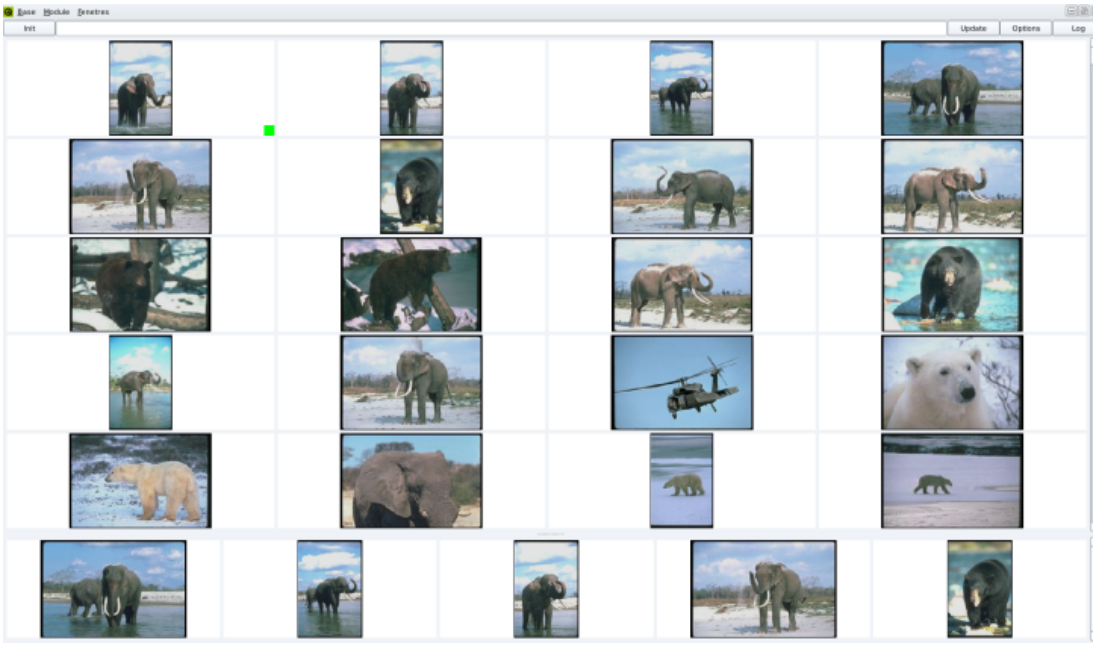


Fig. 1-5. Capture d'écran de *RETIN* pour une requête de la catégorie éléphant. Les résultats semblent plutôt satisfaisants, même si certaines images n'appartenant pas à la catégorie sont bien classées. Les résultats sont triés et affichés dans la fenêtre principale. La dernière ligne correspond à une fenêtre de visualisation particulière pour l'apprentissage en ligne (apprentissage actif).

couleurs compris entre 25 et 50. Cette quantification est effectuée sur un échantillonage des pixels de la totalité de la base par un algorithme de nuée dynamique (*k-means*). Le vecteur caractéristique couleur est la distribution des couleurs de l'image sur les bins obtenus. Les textures sont calculées sur la sortie de 12 filtres de Gabor (3 échelles et 4 orientations). Cet espace est lui aussi quantifié sur un nombre de bins compris entre 25 et 50 par le même algorithme que pour les couleurs. Le vecteur caractéristique de texture est la distribution des sorties de filtres pour chaque pixel de l'image sur les bins obtenus.

Puis chaque dimension est normalisée de manière à avoir la même variance que les autres. Ainsi, il n'y a pas de dimension contenant plus d'informations que les autres (au sens d'une analyse en composantes principales). Les vecteurs de couleurs et de textures sont concaténés l'un à la suite de l'autre pour fournir un unique vecteur de caractéristiques selon ce que l'on appelle la fusion précoce.

Classifieur

Plusieurs classifieurs sont disponibles dans RETIN (Parzen, SVM, réseaux bayesiens, etc). Le fonctionnement par défaut est d'utiliser un estimateur de densité de type *one-class SVM* tant que l'utilisateur n'a pas annoté au moins une image de chaque classe (pertinente et non-pertinente), puis d'utiliser un *SVM*. Plusieurs noyaux sont disponibles, les plus utilisés étant le noyau gaussien avec une distance χ^2 :

$$k_{gauss_{\chi^2}}(\mathbf{x}_i, \mathbf{x}_j) = e^{\frac{-1}{2\sigma^2}\sum_k \frac{(\mathbf{x}_i^k - \mathbf{x}_j^k)^2}{(x_i^k + \mathbf{x}_j^k)}} \tag{1.6}$$

Et le noyau gaussien avec une "distance χ^1" :

$$k_{gauss_{\chi^1}}(\mathbf{x}_i, \mathbf{x}_j) = e^{\frac{-1}{2\sigma^2}\sum_k \frac{|\mathbf{x}_i^k - \mathbf{x}_j^k|}{(x_i^k + \mathbf{x}_j^k)}} \tag{1.7}$$

L'avantage du noyau gaussien χ^1 est d'être invariant en échelle (le coefficient d'échelle se factorisant au numérateur et au dénominateur pour se simplifier).

Apprentissage actif

Un exemple de recherche est présenté sur la figure 1.5. RETIN dispose d'un stratégie active de construction de l'ensemble d'entraînement [Gosselin and Cord, 2006b]. Dans la fenêtre d'interface utilisateur, un cadre est utilisé pour afficher les images de la base par ordre décroissant de similarité, et un autre cadre est utilisé pour afficher les images dont l'annotation est conseillée pour accélérer la recherche (voir figure 1.5). Ces images sont choisies comme étant les plus proches de la frontière estimée entre les deux classes. La sortie du SVM étant normalisée entre -1 et 1, ce sont les images dont la sortie est la plus proche de 0 qui sont sélectionnées.

De plus, RETIN dispose de deux mécanismes afin d'améliorer cette sélection. Partant de l'observation que les images non annotées sont très nombreuses à avoir une pertinence proche de zéro, le premier mécanisme consiste à ajouter de la diversité dans l'ensemble sélectionné de manière active. Puisqu'annoter deux images très proches n'apporte pas beaucoup plus d'information qu'une seule image, l'idée est de sélectionner pour l'annotation les images les plus proches de la marge maximisant l'angle entre elles (*angle diversity* [Gosselin and Cord, 2005a]).

Le second mécanisme consiste à rééquilibrer l'ensemble d'entraînement. Puisque la catégorie recherchée est globalement plus petite que l'ensemble des images non pertinente, l'ensemble d'apprentissage est souvent majoritairement constitué d'image annotées négativement. Afin d'équilibrer correctement cet ensemble, RETIN corrige la marge du SVM lors de la sélection des exemples à annoter. Dans le cas où l'ensemble d'entraînement contient beaucoup d'images annotées négativement, la marge est rapprochée du cœur de classe, afin de sélectionner plus d'exemples positifs, et inversement dans le cas où l'ensemble d'entraînement contient beaucoup d'exemples annotés positivement (*boundary correction*).

L'actif dans l'apprentissage en ligne est largement utilisé dans d'autre systèmes aux caractéristiques variées [Tong and Chang, 2001, Ferecatu et al., 2005].

1.3 Recherche d'images distribuée

À la différence de la recherche dans une unique base d'images, la recherche dans des bases distribuées considère que les données sont réparties dans plusieurs collections. Une hypothèse couramment faite est que le contenu de ces collections est relativement homogène. Les travaux dans ce domaine se basent énormément sur la recherche de textes. Dans le champ de la recherche d'information distribuée, le problème est classiquement décomposé en trois parties [Callan, 2000] :

- L'extraction de descripteurs pour chacune des collections.
- La sélection des collections dans lesquelles la recherche d'information est effectuée.
- La fusion des résultats des différentes collections en un tri unique.

Il existe très peu de travaux pour appliquer la recherche d'images par le contenu à cette approche. Tout d'abord, les systèmes de recherche d'images ne sont efficaces que dès lors qu'ils mettent en œuvre une recherche interactive à l'aide d'un apprentissage en ligne. Or l'interaction n'est pas prise en compte dans ce contexte. Puis, les descripteurs qui sont considérés pour décrire les collections sont hautement sémantiques afin de permettre la sélection en une seule passe des bases dans lesquelles effectuer la recherche. Dans le cadre de l'apprentissage en ligne avec stratégie active, l'objet de la recherche n'est pas connu à l'avance, et donc il devient impossible de sélectionner ces collections sans interaction avec l'utilisateur. Enfin, la fusion des données est grandement facilitée dans le cas où les scores de pertinence donnés sur chacune des collections sont compatibles. Cependant, dans les algorithmes de classification tels que les SVM, le score est une fonction des éléments de la collection. C'est d'autant plus vrai lorsque l'ensemble d'apprentissage a été choisi par une stratégie active. Dès lors, le classifieur est optimisé pour la base sur laquelle il a été entraîné et rend son exploitation hasardeuse sur les autres bases.

1.3.1 Approche centralisée

L'approche naïve consisterait à récupérer toutes les données disponibles dans les diverses collections et à les regrouper en une seule base afin de se passer du caractère distribué des données (c'est l'approche des moteurs de recherche sur le web comme *Google*, qui constituent d'immenses réserves de pages mises en cache). Dès lors, les méthodes traditionnelles de recherche peuvent s'appliquer (extraction de descripteurs par quantification adaptative, apprentissage en ligne, etc.) Bien que cette technique simplifie énormément la mise en œuvre d'algorithmes, elle présente différents problèmes.

Information distribuée

Tout d'abord, en considérant l'hypothèse de collections spécialisées, le fait de centraliser les données dans une unique base peut faire perdre de l'information. En effet, la recherche n'aurait pu être effectuée que dans les bases contenant les données recherchées. Cela permet d'accélérer la construction de l'ensemble d'entraînement en supprimant de

gros volumes de données. Ces volumes tombant dans les zones d'incertitudes du classifieur nécessiteraient alors quelques annotations du point de vue de la stratégie active, alors qu'aucun élément pertinent ne s'y trouve.

Bien sûr, le sous-ensemble des collections contenant les données pertinentes n'est pas connu *a priori*. Il faut donc trouver des méthodes pour exploiter la distribution des données dans les collections, sans pour autant disposer de cette information.

Ressources calculatoires

D'autre part, la concentration de toutes les données dans une seule et unique collection induit des problèmes d'ordre calculatoire. En effet, le résultat d'une recherche est un tri de l'ensemble des données. Ce tri se faisant sur l'ensemble des données unifiées demande alors beaucoup plus de ressources localement que si chaque collection avait été triée séparément. Dans le cas de base d'images, les éléments de la collection sont relativement volumineux et unifier les collections pose alors des problèmes de stockage en mémoire RAM. De part la nature des calculs qui sont effectués, où chaque élément de la collection est examiné, un stockage qui ne serait pas intégralement en RAM rendrait la réactivité du système si lente qu'il en deviendrait inutilisable.

Persistance des données

Enfin, les données étant dupliquées depuis leur source vers le serveur central, des problèmes de mise à jour se posent. En effet, puisque le serveur central n'est pas maître de l'évolution des éléments de chacune des collections, il faut que celui-ci se synchronise régulièrement avec ceux-ci. Outre le coût en ressources évident, cela a un impact sur les méthodes employées. Dans le cas de descripteurs adaptés à la base (par une quantification adaptative, par exemple), il faut recalculer le dictionnaire employé et mettre à jour les distributions. De plus, toute tentative d'apprentissage à long terme telle que décrite en 1.4 n'est pas envisageable, car le contenu de la collection peut varier.

1.3.2 Fusion de moteurs de recherche

Dans le cas où l'on dispose d'un ensemble de collections d'images et les moteurs de recherche qui leurs sont associés, deux problèmes se posent. Dans un premier temps, il s'agit d'exprimer la requête pour chacun des moteurs disponibles, puis dans un second temps, il faut fusionner les résultats des différents moteurs en un tri unique qui sera présenté à l'utilisateur [Si and Callan, 2002, Berretti et al., 2004].

Expression de la requête

Dans le cas où la requête s'exprime à partir d'un ensemble d'images, le moteur cible calcule alors des attributs sur ces images, puis effectue la recherche. Dans le cas où la

requête est une mesure de similarité, par exemple un SVM, il faut que celle-ci puisse être exprimée dans l'espace de représentation des données disponibles localement.

Fusion des résultats

Ensuite, il faut fusionner les tris renvoyés par les différents moteurs en un unique tri [Baumgarten, 1997]. Une manière naïve de faire est de procéder à un tourniquet (*round-robin*), c'est à dire de sélectionner la première image de chaque moteur, puis la seconde, etc. Dans le cas où des valeurs de similarité sont disponibles pour chacun des résultats, on peut aussi attribuer des pondérations aux résultats de chaque moteur afin de mieux représenter les moteurs donnant des résultats pertinents [Si and Callan, 2002]. Enfin, on peut rapprocher la fusions de listes triées des méthodes de vote alternatives proposées par Borda ou Condorcet, et citer les méthodes d'apprentissage supervisé de fonction d'ordonnancement [Vu, 2008].

1.3.3 Réseau pair-à-pair

Un réseau pair à pair est un ensemble de machines reliées entre elles et partageant des données. On sort alors du cadre de la traditionnelle relation *client/serveur*, puisque chaque ordinateur peut être à la fois client et serveur. Ce type de réseau peut disposer d'un moteur de recherche centralisé qui va indexer les données à chaque connexion d'un pair, ou bien être totalement décentralisé. C'est ce deuxième cas qui nous intéresse particulièrement ici.

Propagation de la requête

Dans le cas où la recherche sur le réseau pair-à-pair est complètement décentralisée, la requête est propagée de pair en pair. Chaque pair étudie la requête calcule les résultats puis la propage à ses voisins. La méthode la plus utilisée est celle de l'inondation (*flooding*) qui consiste à propager la requête à tous les pairs voisins jusqu'à ce qu'un nombre d'itérations soit atteint (*time to live*). Plus récemment, des algorithmes ont été proposés spécifiquement pour la recherche d'images afin de propager la requête uniquement vers les pairs contenant des images similaires [Lu and Callan, 2003, King et al., 2004]. Cependant, ces méthodes ne tiennent pas compte d'un schéma intégrant un apprentissage en ligne et donne par conséquent des résultats médiocres par rapport aux systèmes de recherche d'images non distribués.

Fusion des résultats

La manière la plus courante de fusionner les résultats dans les systèmes pair-à-pair est une présentation par ordre chronologique. Ainsi les pairs ayant répondu le plus rapidement verront leurs résultats en haut du tri. Cependant, si cette présentation à des avantages indéniables en terme de réactivité, elle ne permet pas d'avoir de bons résultats du point de vue de la sémantique recherchée. Encore une fois, les systèmes de recherche d'images

sur les réseaux pair-à-pair ne s'inspirent pas des systèmes traditionnels, en particulier en ce qui concerne l'apprentissage en ligne d'une mesure de pertinence basée sur le contenu, et offrent donc des résultats très éloignés des performances de ceux-ci.

1.4 Recherche à long-terme

À la fin d'une session de recherche, toutes les annotations données par l'utilisateur sont oubliées. De fait, chaque nouvelle session part du même point que les précédentes. L'idée de la recherche à long terme est de sauvegarder l'information fournie par l'utilisateur lors d'une session afin de la réutiliser pour les sessions suivantes.

Le contexte est le suivant : on dispose d'un grand nombre de sessions, c'est-à-dire de lot d'images et de leurs annotations respectives, avec les restrictions d'un contexte faiblement supervisé rapportées par Vasconcelos [Vasconcelos and Kunt, 2001], à savoir que l'on ne connaît pas la catégorie correspondant aux images annotées positivement, et que les images annotées négativement peuvent appartenir à plusieurs catégories. En d'autre terme, il est d'une part impossible de regrouper les sessions correspondant aux mêmes catégories recherchées, et d'autre part, il est impossible de regrouper l'information des images annotées négativement au sein d'une même session.

Il existe deux grandes familles d'optimisation à long-terme, la première travaille sur la représentation de la base et modifie les signatures des images afin de les faire correspondre aux annotations obtenues à l'issue des sessions ; alors que la seconde travaille sur la *"perception"* de la base en modifiant la mesure de similarité.

1.4.1 Apprentissage des signatures

L'idée générale derrière l'apprentissage des signatures est de déplacer les images dans l'espace de représentation de manière à rapprocher les images ayant eu un label positif lors d'une même session, et d'éloigner les images annotées négativement de celles annotées positivement.

La méthode proposée par P. Gosselin [Cord and Gosselin, 2006, Gosselin and Cord, 2006a] consiste à rapprocher les images positives de leur barycentre. Lors d'une session, le barycentre des images annotées positivement est une approximation du barycentre de la catégorie recherchée. Cette approximation s'améliorant au cours des sessions, les images appartenant à une même catégorie vont se rapprocher petit à petit du barycentre de la catégorie.

De même, les images annotées négativement sont éloignées du barycentre des annotations positives de manière à se rapprocher d'un point à une distance fixe du barycentre des images positives et située dans la direction opposée. Cette distance est calculée de manière à ce que les barycentres des différentes catégories soient tous équidistants.

On peut voir cette méthode comme une déformation de l'espace de représentation appliquée localement aux images de la base. Cependant, puisque la déformation est

concentrée sur les images de la base, la généralisation à de nouvelles images n'est pas immédiate.

1.4.2 Apprentissage de fonctions de similarité

La deuxième approche consiste à travailler sur la matrice contenant toutes les similarités image à image. Elle consiste à augmenter de la similarité entre les images ayant obtenu des annotations positives, et à baisser la similarité des images ayant reçu des annotations différentes [Fournier and Cord, 2002]. Au bout de plusieurs sessions, les images ayant conjointement reçu des labels positifs se retrouvent avec des valeurs de similarité proches de 1.

Avec le formalisme noyau, cela revient à apprendre la matrice de Gram $\{k(\mathbf{x}_i, \mathbf{x}_j)\}_{i,j}$ pour toutes les images de la base [Cristianini et al., 2001, Gosselin and Cord, 2004, Gosselin and Cord, 2005b]. On peut voir cette approche comme une déformation de la fonction ϕ de manière à mieux rapprocher les images proches entre elles. Cependant, puisque l'on opère directement sur la matrice de similarité, on ne possède plus l'expression analytique du noyau k (Avec ce formalisme, la transformation ϕ n'a jamais été explicitée). Ainsi, cette approche travaille sur une base fermée. De même que pour la méthode précédente, les déformations appliquées à la fonction d'injection sont locales aux images de la base et ne peuvent pas se généraliser facilement à de nouvelles images.

1.5 Positionnement de nos travaux

Nos travaux se situent à la fois dans la continuité des avancées faites dans ce champ de recherche ces dernières années, mais aussi sur des thématiques novatrices jusqu'alors peu explorées. Ils sont dans la continuité des travaux effectués au laboratoire ETIS, d'une part parce que nous nous sommes attachés à réutiliser les outils qui ont fait la réussite des systèmes développés par l'équipe, mais d'autre part parce que nous avons conservé les mêmes méthodologies, notamment en terme d'interaction avec l'utilisateur et de schéma d'apprentissage.

Cependant, nos travaux se positionnent dans un contexte qui n'a pas réellement été étudié, puisqu'il s'agit de recherche d'images par le contenu dans des bases distribuées. Comme on l'a vu dans ce premier chapitre, dès que l'on considère un ensemble de collections réparties sur un réseau tel qu'Internet ou bien les réseau pair-à-pair, les outils développés dans la littérature sont loin d'être aussi aboutis que ceux développés dans le cadre de la recherche dans une base unique.

Ces différences tiennent en quatre points :

Bases ouvertes distribuées

Les collections d'images que nous considérons sont réparties sur plusieurs ordinateurs d'un réseau. Nous considérons que les images stockées sont propres aux bases et qu'il est

impossible de les centraliser sur une unique machine (défaut de moyens soit techniques soit juridiques). Nous n'avons pas d'*a priori* sur le nombre et le contenu de ces bases, ces deux paramètres pouvant varier au cours du temps. Ainsi, nous nous interdisons toute possibilité d'adapter le système à la base comme c'est souvent le cas dans la littérature des systèmes de recherche d'images par le contenu.

Recherche interactive

Nous nous plaçons dans le cas de la recherche interactive, c'est-à-dire que le système va affiner la recherche à l'aide de l'information fournie par l'utilisateur dans le cadre du bouclage de pertinence. Contrairement à de nombreux systèmes de recherche d'images, nous n'utilisons pas d'information *a priori*. La qualité des résultats repose uniquement sur les signatures des images et les algorithmes d'apprentissage. Lesdits algorithmes sont d'ailleurs utilisés hors du cadre classique de l'apprentissage, pour lequel on dispose ordinairement d'une base d'apprentissage servant à l'entraînement, d'une base de test différente servant à l'évaluation et de la base des données réelles à traiter. Nous n'avons à disposition qu'une seule collection (étant la réunion des images disponibles sur le réseau) contenant les données à traiter, qui va aussi nous servir à l'entraînement.

Apprentissage actif

De part le contexte un peu particulier dans lequel nous utilisons des algorithmes d'apprentissage, le but de ces derniers change légèrement. Il ne s'agit plus de construire la meilleure fonction de similarité à partir d'un ensemble d'entraînement prédéterminé, mais de trouver le plus rapidement parmi les images à traiter, celles qui une fois ajoutées à l'ensemble d'apprentissage, permettent d'obtenir une bonne mesure de similarité. En d'autres termes, le cœur du problème est sur la stratégie qui sélectionne à chaque itération les exemples à faire annoter à l'utilisateur de sorte que la mesure de similarité s'améliore le plus rapidement possible. Cette problématique que l'on appelle couramment *apprentissage actif*, est ici à généraliser au contexte de collections distribuées qui est le notre.

Apprentissage à long terme

Nous sommes dans un contexte de réseau, où plusieurs utilisateurs différents utilisent le système au même instant. De part la nature réactive des solutions que nous employons, l'information fournie au système par les utilisateurs peut facilement se partager. L'idée de l'apprentissage à long-terme est alors d'utiliser l'information des sessions de recherche de tous les utilisateurs (ou bien des sessions passées d'un seul utilisateur) afin d'améliorer les résultats de futures recherches. Cependant, dans notre cas, l'apprentissage à long-terme va se porter sur le côté réseau et non pas sur la représentation de la base ou bien sur les fonctions de similarité comme on peut le voir dans la littérature.

1.6 conclusion

Dans ce chapitre nous avons survolé le domaine de la recherche d'images par le contenu. Nous avons détaillé les points clés de ce domaine, à savoir l'extraction de caractéristiques, la construction de signatures et de mesures de similarité, puis les méthodologies de recherche. Nous avons indiqué que la recherche débute le plus souvent par une requête sous forme d'image ou de lot d'images, puis que les systèmes les plus efficaces utilisent un bouclage de pertinence. Le bouclage de pertinence, afin d'obtenir des images et des annotations associées, permet d'utiliser les techniques d'apprentissage automatique.

Nous avons ensuite vu comment la recherche d'images pouvait s'étendre à un contexte distribué, dans lequel les images sont réparties sur différentes machines d'un réseau. Nous avons détaillé les différents types de réseaux qui pouvaient être considérés et les techniques qui leur sont associées. La conclusion sur ce point est qu'il n'existe pas à notre connaissance de système de recherche d'images par le contenu distribué utilisant les techniques modernes de *CBIR* développées dans le cas non-distribué.

Enfin, une brève présentation des techniques d'apprentissage à long-terme utilisées en recherche d'images par le contenu a été faite. Ces techniques réutilisent l'information fournie lors du bouclage de pertinence de plusieurs sessions afin d'améliorer les résultats des suivantes. Séparées en deux grandes familles, il s'agit soit d'améliorer la représentativité de la base en modifiant les signatures de manière à les faire correspondre aux annotations apportées par l'utilisateur, soit de modifier la mesure de similarité de manière à rendre plus similaires les images annotées conjointement. Nous avons conclu sur la difficulté d'utiliser ces techniques dans le cas de bases ouvertes.

Nous avons terminé ce chapitre par le positionnement de nos travaux. Ceux-ci se placent dans le cadre d'une utilisation interactive sur des bases distribuées, dans laquelle nous proposons un apprentissage actif, d'une part. Et d'autre part, nous exposons une nouvelle technique d'apprentissage à long terme pour un contexte distribué.

Chapitre 2

Systèmes multi-agents

La seconde question soulevée par la problématique de cette thèse est celle de la localisation d'une donnée recherchée sur un réseau de machines. Cette axe de recherche consiste à savoir comment, à partir d'une requête donnée, opérer un routage de manière à aboutir aux machines contenant les résultats intéressants. Si de nombreuses méthodes ont été proposées pour résoudre ce type de problème, nous nous sommes intéressés pour notre part aux systèmes multi-agents qui ont montré des capacités prometteuses dans ce domaine. La construction d'un système complet nous permet de répondre à la fois à cette question sur la localisation des données pertinentes et à la question de la similarité basée contenu décrite au chapitre précédent, en créant un bouclage entre ces deux problèmes.

Afin de positionner nos travaux sur ce deuxième axe qu'est la question de la localisation des données pertinentes, nous allons détailler dans ce chapitre l'état de l'art sur les systèmes multi-agents, et tout particulièrement ceux qui nous intéressent, c'est-à-dire les agents mobiles d'une part, et les agents d'inspiration éthologique d'autre part. Les systèmes multi-agents forment un domaine de recherche très vaste allant de l'application à des visées industrielles à la modélisation de comportements sociaux. Afin de ne pas nous perdre dans l'immensité de cette problématique, nous allons découper ce chapitre en trois parties, allant du plus général vers ce qui nous intéresse le plus dans cette thèse. Nous allons commencer par décrire le paradigme posé par les systèmes multi-agents, ainsi que les nouveaux modes de conception d'algorithmes qu'ils proposent. Puis nous détaillerons les agents mobiles, qui sont des agents autonomes ayant la capacité de se déplacer, afin de voir là encore quelles sont les possibilités offertes par cette technologie et l'impact que cela a sur l'architecture des systèmes d'une part et sur la conception d'algorithmes d'autre part. Enfin, nous conclurons par une partie sur les agents d'inspiration éthologique, c'est-à-dire d'agents qui miment le comportement animal. Ceux-ci permettent l'émergence de comportements de groupes intéressants sur lesquels nos travaux s'appuient.

2.1 Systèmes multi-agents

Selon la définition donnée dans son livre "Les Systèmes Multi-Agents - vers une intelligence collective", J. Ferber introduit l'agent comme étant *"une entité physique ou virtuelle qui est capable d'agir dans un environnement, qui peut communiquer directement avec d'autres agents, qui est mue par un ensemble de tendances (sous la forme d'objectifs individuels ou d'une fonction de satisfaction, voire de survie, qu'elle cherche à optimiser), qui possède des ressources propres, qui est capable de percevoir (mais de manière limitée) son environnement, qui ne dispose que d'une représentation partielle de cet environnement (et éventuellement aucune), qui possède des compétences et offre des services, qui peut éventuellement se reproduire, dont le comportement tend à satisfaire ses objectifs, en tenant compte des ressources et des compétences dont elle dispose, et en fonction de sa perception, de ses représentations et des communications qu'elle reçoit"* [Ferber, 1995].

Autrement dit, un agent est une entité *située*, c'est à dire qu'elle est en interaction avec un environnement qu'elle peut percevoir et sur lequel elle peut agir, et *autonome* en ce qu'elle est capable d'agir sans l'intervention d'un tiers (Fig. 2.1). Dans le cas d'une population d'agents constituant un système multi-agents, l'agent doit en plus être *social*, c'est-à-dire qu'il doit être capable d'interagir avec les autres agents présents dans l'environnement. Ces trois attributs forment le minimum de capacités pour avoir la notion d'agent [Wooldridge and Jennings, 1995].

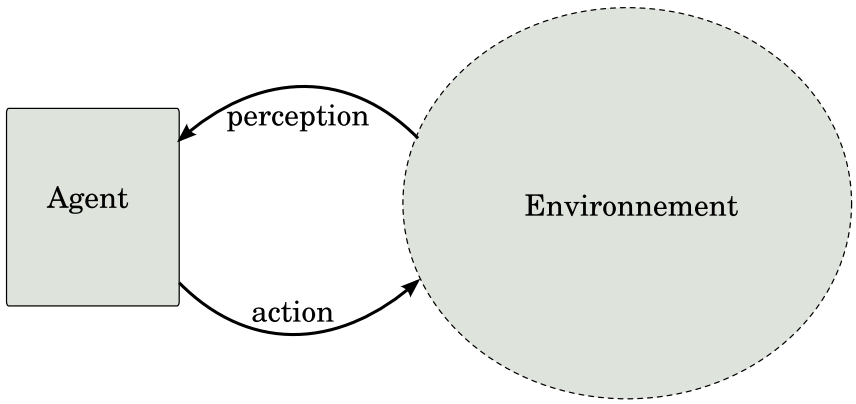


Fig. 2.1. Un agent est une entité située dans un environnement qu'elle observe et sur lequel elle peut agir de façon autonome.

2.1.1 Paradigme agent

Un système multi-agents (SMA) est *"un système distribué composé d'un ensemble d'agents"* [Jarras and Chaib-draa, 2002]. Ainsi, l'idée des systèmes multi-agents est de réduire un problème à l'ensemble des participations élémentaires et simples réalisées par chacun des agents. Les caractéristiques générales que l'on peut dégager des SMA sont que chacun des agents n'a qu'un point de vue incomplet sur le problème (soit parce qu'il n'en perçoit pas la globalité, soit parce que ses moyens de le résoudre sont limités)

que l'ensemble de la population est nécessaire à sa résolution, qu'il n'y a aucun contrôle centralisé et que les calculs sont asynchrones (ces deux derniers points étant du fait de l'autonomie des agents).

Pour la résolutions de problèmes facilement adaptables à ce paradigme, ces systèmes offrent des bénéfices considérables tels que la rapidité (due à la distribution des calculs entres les agents), la robustesse (due à la redondance) et l'adaptabilité (il n'est souvent pas nécessaire de changer tout les agents pour traiter un nouveau problème), ce qui en a fait un sujet de recherche privilégié à la fin des années 90 [Jennings et al., 1998]. Cependant, derrière ces avantages indéniables se cachent aussi de nouvelles problématiques liées à l'architecture distribuée des SMA. En effet, la formulation et la décomposition d'un problème de telle sorte que celui-ci soit traitable par un ensemble d'agents est loin d'être triviale. En particulier, quels sont les objectifs de chacun des agents ? Comment doivent-ils interagir ? Et surtout, comment être certain que le problème puisse être et sera résolu par le SMA ? Tout ceci fait que la conception d'un SMA est particulièrement difficile à mettre en œuvre [Jennings et al., 1998].

2.1.2 Interaction

Toute la problématique de la conception d'un système multi-agents réside dans l'*interaction* que celui-ci a d'une part avec l'environnement, mais aussi d'autre part avec les autres agents. L'interaction que celui-ci a avec l'environnement concerne l'information à laquelle il a accès et les modifications qu'il peut y apporter. Dans le cas d'un agent logiciel, par exemple, les informations à sa disposition peuvent être des flux de fichiers, des entrées au clavier, le réseau, une webcam, et ses actions peuvent modifier des fichiers, afficher quelque chose à l'écran, ou envoyer des trames sur le réseau [Ferber, 1995].

L'interaction entre agents est autrement plus complexe en ceci qu'elle doit mener à une coopération entre les agents menant à la résolution du problème, ce qui fait la différence entre un SMA et une simple collection d'agents indépendants [Jarras and Chaib-draa, 2002]. Cette interaction peut prendre différentes formes : elle peut être directe, c'est-à-dire que les agents s'envoient directement des messages les uns aux autres. Ces messages peuvent être hautement symboliques et comporter des types de messages telles des requêtes et des réponses construites dans un langage adapté (tels que KQML ou ACL) [Finin et al., 1994], ce qui implique que les agents soient capables de manipuler des symboles.

L'interaction peut être partagée entre tous les agents comme par exemple dans le cas du tableau noir [Corkill, 1991], qui constitue une mémoire partagée dans laquelle les agents peuvent lire et écrire des données. Les agents peuvent ainsi écrire leurs résultats partiels ou bien des indications et en obtenir de la part des autres agents. On peut considérer cette mémoire partagée comme une partie de l'environnement que les agents modifient afin de communiquer, ce qui nous amène à la façon la plus simple qu'on les agents pour interagir entre eux, à savoir de modifier leur environnement.

2.1.3 Émergence

Comme nous l'avons signalé, chaque agent du SMA n'a qu'une vue partielle du problème, en ceci qu'il est incapable à lui tout seul de le résoudre (soit par manque de moyen, soit par manque d'information). Ainsi, la solution apportée par le SMA émerge de l'ensemble de la population.

L'interaction entre les agents stimule cette émergence. Par exemple, dans le cas où un agent modifie son environnement à des fins d'interaction, un autre agent peut se retrouver stimulé par l'observation des modifications et adapter son comportement de telle sorte que l'effet global amenant à la résolution du problème s'en trouve amélioré. On appelle ce phénomène la *stigmergie*, introduite par le zoologiste P.-P. Grassé, qui consiste en la stimulation d'un agent par le résultat de l'action d'un autre agent.

Un bon exemple d'émergence peut se trouver dans [Drogoul and Ferber, 1992], où le problème est de ramener à un endroit donné un ensemble d'objets se trouvant dans un environnement quelconque. A. Drogoul considère deux types d'agents : une première population d'agents *Petit Poucets*, et une seconde population de *Dockers*. La première population dite *Petit Poucet* est composée d'agents qui explorent l'environnement de manière pseudo aléatoire en déposant des marquages sur le sol, de sorte que les autres agents, attirés par les marquages puissent trouver plus facilement les éléments recherchés. Ce comportement, fortement inspiré par les fourmis à la recherche de nourriture, est un exemple de stigmergie. En effet, l'action des agents modifiant l'environnement mène à une stimulation des autres agents améliorant le résultat final.

La seconde population dite *Dockers* est composée d'agents qui sont capables de détecter si un autre agent transporte un élément recherché et de le prendre. Le résultat obtenu est une chaîne d'agents allant de la sources d'objets jusqu'à la destination dans laquelle les objets sont transportés d'agent en agent. Là encore, le comportement collectif est bien plus efficace que le comportement individuel.

On trouve beaucoup d'exemples dans la littérature de systèmes multi-agents dans lesquels le système produit de meilleurs résultats que la somme de ces parties. Beaucoup de ces systèmes sont inspirés du vivant, comme nous allons le voir dans la dernière partie de ce chapitre.

2.2 Agents mobiles

Les agents mobiles sont une extension des systèmes multi-agents, dans laquelle certains des agents peuvent se déplacer sur un réseau de machines. Pour ce faire, chaque machine du réseau fait s'exécuter une plateforme d'agents mobiles. Lors d'un déplacement, le code d'un agent est téléchargé sur la machine cible et s'exécute au sein de la plateforme (Fig. 2.2).

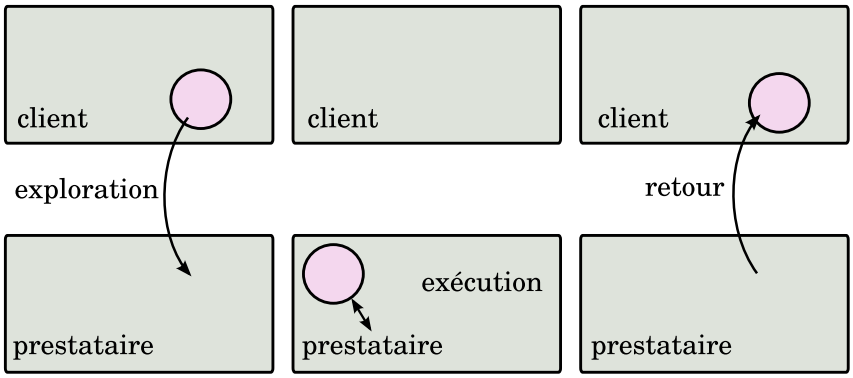


Fig. 2-2. Dans un premier temps le code mobile est téléchargé sur la machine client, puis il s'exécute pour effectuer la tâche requise (recherche d'information, négociation, etc.), avant de se télécharger de nouveau vers la machine de l'utilisateur afin de rapatrier les résultats.

2.2.1 Code mobile

L'idée de transférer du code d'une machine à une autre n'est pas tout à fait récente [Ghezzi and Vigna, 1997]. Les principales difficultés à la mise en place de code mobile est la nécessité d'installer des plateformes spécifiques souvent incompatibles entre elles [Gray et al., 2000]. Ces plateformes sont souvent inadaptées aux utilisations faites du réseau en terme de protocoles et de méthodologies [Magnin et al., 2002]. Ainsi, le web fonctionne sur le protocole *HTTP* en transférant des pages en langage *HTML*, or ces technologies n'autorisent pas d'exécution de code ni de communications complexes telles que requises par la mise en œuvre d'une plateforme d'agents mobiles.

D'autre part, l'utilisation d'agents mobiles mène à des problèmes de sécurité auxquels les administrateurs et les concepteurs des plateformes doivent faire face [Roth, 2004].

Cependant, il y a de bonnes raisons pour utiliser du code mobile dans de nombreux cas [Lange, 1998]. Parmi ceux-ci, on peut citer l'absence de besoins de connexions robustes (une fois que le code s'exécute sur la machine cible, il n'est pas nécessaire de garder la connexion ouverte, la machine de départ peut même être éteinte), la parallélisation naturellement massive des calculs effectués par les agents, l'économie de bande passante dans le cas où les données sont plus volumineuses à transférer que le code qui va les traiter, l'absence de latence du réseau puisque le code s'exécute en local (ce qui peut être très important dans les systèmes temps-réel). Tout ceci en fait une technologie de choix pour un large panel d'applications telles que l'*e-commerce*, la dissémination d'informations, la négociation en ligne, l'encapsulation de protocole réseau, les calculs distribués et bien sûr, ce qui nous intéresse plus particulièrement ici : la recherche d'informations distribuées [Lange and Oshima, 1999].

2.2.2 Plateformes et caractéristiques

Il existe de nombreuses plateformes d'agents mobiles dans diverses langages de programmation. Le projet *Aglets* d'IBM [Clements et al., 1997] est à notre connaissance le premier projet d'agents mobiles utilisant java et fut l'un des plus utilisés. Le langage java semble bien adapté aux agents mobiles, puisqu'il permet de facilement exécuter du code mobile dans un environnement hétérogène, pour peu que chaque élément du réseau possède une machine virtuelle java avec des capacités suffisantes pour faire fonctionner la plateforme.

Plus récemment, L'Institut Fraunhofer a développé une plateforme d'agents mobiles axée sur la sécurité du nom de *SeMOA* (Secure MObile Agents) [Roth and Jalali-Sohi, 2001], toujours programmée en java. *SeMOA* inclut des techniques de *sandbox* et de signature de code de manière à sécuriser l'utilisation des agents mobiles afin de protéger la machine recevant les agents. Les communications entre agents sont chiffrées afin d'éviter la fuite de données sensibles.

La plateforme multi-agents que nous avons utilisée est *Jade*. Toujours en java, elle a l'avantage d'être libre et très complète, notamment sur les méthodes de communication entre agents. *Jade* repose sur le langage *FIPA-ACL* (ACL pour Agent Communication Language) ce qui permet de concevoir des protocoles d'échanges entre agents assez complexes. Malheureusement, *Jade* n'a qu'un support limité pour la mobilité inter-machine, ce qui nous a amené à développer notre propre support de mobilité au sein de *Jade*.

Nous avons par ailleurs développé notre propre plateforme multi-agents pour le besoin de certaines simulations. Dans ce cas ci, le réseau est simulé par un ensemble de destinations virtuelles. Les agents ne quittent donc jamais physiquement l'ordinateur sur lequel est lancée la simulation. L'avantage de cette plateforme est de pouvoir simuler facilement un graphe de réseau assez complexe.

2.2.3 Agents mobiles de recherche

Une utilisation des agents mobiles notée comme étant prometteuse est la recherche d'informations distribuées [Lange and Oshima, 1999]. Les agents mobiles montrent de très bonnes performances du point de vue du réseau pour cette tâche [Jiao and Hurson, 2004]. En effet, la recherche d'informations *via* une architecture classique de type *client-serveur* nécessite une connexion réseau de bonne qualité afin de faciliter le transfert des flux de données, alors que le modèle à agents mobiles se comporte quand même convenablement dans le cas de réseau congestionnés.

Plusieurs exemples de systèmes à agents-mobiles pour la recherche d'informations ont été énoncés [Roth et al., 2005] et en particulier dans le cas de données multimédia [Groot et al., 2005]. En effet, dans le cas de données multimédia, des problèmes de propriété des données se posent. Une vidéo, par exemple, peut être protégée par des droits d'exploitation rendant sa diffusion sur le réseau interdite. Dans ce cas, l'indexation et la recherche doivent se négocier directement avec le propriétaire des droits. L'approche agent est particulièrement intéressante pour ce cas, en ceci qu'elle permet aux données de rester

à l'endroit de leur stockage et à leur propriétaire de restreindre l'analyse qui en est faite par les agents venus indexer à des fins de recherche [Brazier et al., 2004].

Dans le cadre du laboratoire ETIS, des travaux avaient été menés sur les agents mobiles de recherche d'informations et en particulier de texte [Revel, 2003]. Ces agents mobiles sont particuliers puisqu'ils ont un comportement mimant celui d'insectes sociaux. L'idée générale est de marquer le réseau à l'aide de phéromones virtuelles de telle sorte que les agents peuvent repérer les sites contenant de l'information pertinente grâce aux explorations précédentes du réseau. L'émergence d'une sorte de *carte routière* globale déposée sur le réseau indiquant les chemins pertinents provient de la somme des comportements de chaque agent [Revel, 2005]. Puisque nos travaux sont l'extension de ceux-ci à la recherche d'images par le contenu, nous allons maintenant présenter les étapes menant à la conception d'agents inspirés par le comportement d'insectes sociaux.

2.3 Agents d'inspiration éthologique

Les insectes offrent des modèles d'ensembles d'agents collaboratifs entraînant l'émergence d'un comportement à l'échelle de la colonie à partir du comportement de chacun des individus la composant. Il existe de nombreux exemples dans la nature de populations d'insectes résolvant un problème qu'un seul insecte est incapable de résoudre. L'observation du vivant permet d'analyser, modéliser et adapter ces comportements afin de résoudre des problèmes d'optimisation. Nous allons rappeler dans un premier temps quelles sont les particularités de ces familles d'insectes afin de mieux comprendre comment leurs comportements ont été adaptés à des agents dans le but de résoudre des problèmes complexes.

2.3.1 Insectes sociaux

Certaines sociétés d'insectes offrent des propriétés particulièrement intéressantes pour la résolution de problèmes de manière élégante et naturellement distribuée. Si chaque insecte est relativement simple, ou en tout cas limité du point de vue cognitif (par rapport à la cognition humaine que nous considérons évoluée), à l'échelle de la colonie, il en est tout autrement [Theraulaz and Gervet, 1992].

Dans le cas qui nous intéresse, nous prendrons essentiellement appui sur certaines espèces de fourmis. Une colonie de fourmis peut être particulièrement efficace au tri alors que chaque fourmi suit un comportement simpliste [Deneubourg and Goss, 1989] : prendre un objet (un cadavre par exemple) là où il y en a peu et le déposer là où il y en a beaucoup, ce qui conduit à la construction de tas.

Nous nous intéressons plus particulièrement à la manière dont les fourmis partent à la recherche de nourriture et aux optimisations que leur comportement entraîne. Sur les chemins qu'elles empruntent pour revenir au nid après avoir trouvé de la nourriture, les fourmis déposent des marqueurs chimiques appelés *phéromones*. De plus, elle préfèrent

suivre les chemins qui ont une forte quantité de phéromones. Il s'en suit que la quantité de phéromones sur le chemin le plus rapide augmente plus rapidement que sur les autres, puisque les fourmis sont plus rapides à faire des aller-retours sur celui-ci. Ainsi, si chacune des fourmis est incapable de déterminer à elle seule le plus court chemin vers la source de nourriture, l'ensemble de la colonie y arrive très bien [Deneubourg et al., 1992] (Fig. 2.3).

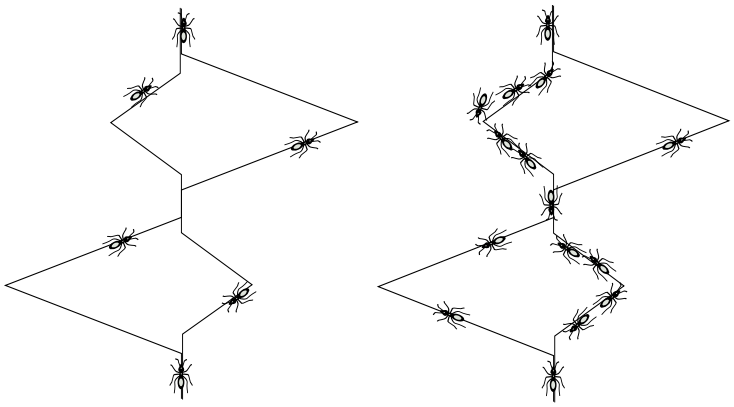


Fig. 2.3. Expérience dans laquelle les fourmis ont plusieurs chemins possibles pour aller du nid en bas à la source de nourriture en haut. Au début, tous les chemins sont explorés de la même manière. Puis rapidement, comme les fourmis prenant le plus court chemin mettent moins de temps à faire l'aller-retour, les phéromones sur ce chemin sont déposées en plus grande quantité et presque toutes les fourmis ne suivent que celui-ci.

Ces travaux ont pour but la modélisation du vivant, et ont pour but de fournir une explication acceptable aux comportements observés chez ces colonies d'insectes.

2.3.2 Algorithmes d'optimisation

Les travaux de modélisation sur les colonies d'insectes ont su inspirer les informaticiens, qui ont su alors adapter les modèles afin de produire des algorithmes génériques. Cet analyse de la découverte du plus court chemin a ainsi été formalisée et exploitée dans une collection d'algorithmes regroupés sous le nom d'*ACO* (Ant Colony Optimisation) [Dorigo et al., 1996, Botee and Bonabeau, 1998]. Une colonie de fourmis est représentée sous la forme d'un ensemble d'agents computationnels, et l'objectif de cette colonie est de trouver la solution optimale en terme de coût à un problème donné. Les agents se déplacent sur un graphe représentant les coûts associés aux étapes du problème et sont partagés entre suivre le chemin de plus bas coût (approche gloutonne) ou suivre le chemin marqué par les phéromones.

Par exemple, le problème du voyageur de commerce se prête bien à l'utilisation de l'*ACO*. Dans ce problème, un commerçant doit parcourir N villes une seule fois afin d'y faire fortune. L'objectif du problème est de trouver le chemin le plus court qui passe par toutes les villes. En utilisant *ACO*, chaque agent parcourt les villes en fonction de la distance à laquelle elles se trouvent les unes des autres (approche gloutonne) et en fonction

d'un niveau de phéromones déposé sur chaque ville (approche globale). Une fois que chaque agent de la colonie a terminé son tour, ils sont triés par ordre de distance parcourue, et des phéromones sont déposées sur les chemins les plus courts [Dorigo and Gambardella, 1997]. La stratégie gloutonne permet une optimisation locale, alors que les phéromones permettent de sortir d'un minimum local (Fig. 2-4).

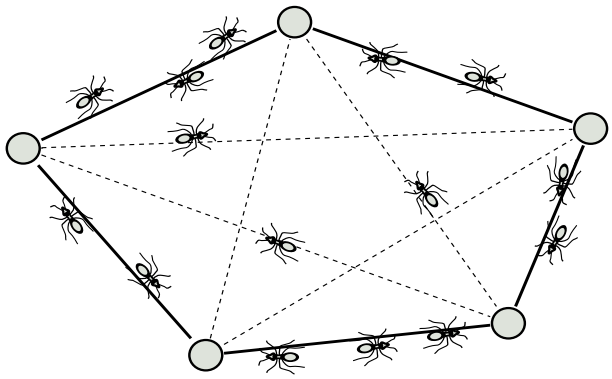


Fig. 2-4. Dans le problème du voyageur de commerce, chacune des fourmis logicielles parcourt l'ensemble des villes en fonction de leur distance et de la quantité de phéromones qui y sont déposées. Les fourmis logicielles ayant fait un chemin plus court déposent une plus grande quantité de phéromones, ce qui localement entraîne les autres fourmis logicielles à suivre leur chemin. Au bout de plusieurs itérations, le plus court chemin est marqué par la plus grande quantité de phéromones (trait plein).

Les algorithmes de type ACO ont été utilisés dans de nombreux autres problèmes d'optimisation combinatoire tels que l'affectation quadratique (*Quadratic Assignment Problem*), l'ordonnancement, la sélection ou la classification [Dorigo et al., 2006].

2.3.3 Application au routage

Une des applications à succès de l'optimisation par algorithme fourmis est le routage de paquets sur un réseau. Des système basés sur cette idée on été développé dès le début de la formalisation de ces algorithmes [Caro and Dorigo, 1997].

L'idée est d'envoyer des agents parcourir le réseau avec une destination choisie au hasard et de renforcer les phéromones sur le chemin choisi en fonction du temps mis pour le parcours [R.Schoonderwoed et al., 1997]. Les phéromones sont ainsi mises à jour périodiquement par des agents. Les paquets de données, sont quant à eux routés de manière déterministe : ils suivent le chemin de plus forte valeur de phéromones. Il y a diverses optimisations à l'utilisation de l'*ACO* à ce domaine, mais l'idée est globalement la même pour produire la table de routage de chaque nœud.

Plus récemment, l'utilisation de l'*ACO* s'est vue étendue au cas des réseaux mobiles, dans lesquels les tables de routage ne peuvent pas être fixes du fait de la grande variabilité des connexions entre les différents nœuds du réseau [Ducatelle et al., 2005]. Du fait que

ces algorithmes sont capables de s'adapter en temps réel à la topologie du réseau ainsi qu'à sa congestion, les tables produites gardent une grande efficacité au cours du temps.

2.4 Synthèse

Les systèmes multi-agents sont souvent utilisés pour paralléliser un processus, ou bien pour améliorer la robustesse d'un programme (redondance). Nous nous intéressons maintenant à l'intérêt de ces systèmes dans le cadre de la recherche d'information sur un réseau, afin de positionner nos travaux dans ce domaine. Nous utilisons un système multi-agents afin d'avoir un système collaboratif où l'optimisation émerge de l'ensemble des comportements de chacun des agents.

Agents mobiles

Dans notre contexte, qui est celui de la recherche sur un réseau, les agents mobiles sont particulièrement bien adaptés. En effet, la recherche implique une exploration du réseau afin de trouver des sites contenant de l'information. Les agents se déplaçant de proches en proches peuvent ainsi dynamiquement s'adapter à la topologie du réseau (ajout ou retrait de nœuds) et découvrir les nouveaux sites riches en informations. D'autre part, ils se prêtent particulièrement bien à une parallélisation massive des calculs, puisque ceux-ci sont effectués sur les machines qui stockent les données, au lieu d'être faits sur une seule et unique machine collectant toutes les données. Notre système tire donc parti de tous ces avantages en utilisant des agents mobiles.

Inspiration éthologique

Contrairement à nombre d'application mettant en œuvre des systèmes multi-agents, la tâche dévolue à un agent est relativement *"simple"*. Il s'agit de se déplacer sur un réseau et de récupérer des images pertinente (à l'aide d'outils de recherche par le contenu qui, eux, peuvent être *"complexes"*). Nous proposons ainsi de doter nos agents de comportements *"simple"*, comme ceux observés chez les fourmis à la recherche de nourriture. Nos agents ne sont donc pas particulièrement *"intelligents"*. Il ont un ensemble limité de comportements très simples (se déplacer, demander des images, renvoyer les résultats) choisis de manière déterministe en fonction de la tâche à effectuer. Ainsi, ils n'apprennent ni n'évoluent au fil du temps, et tous les agents sont identiques.

Système coopérative - stigmergie

Si le problème que doit résoudre chaque agent semble simple (trouver une source d'image et ramener des images pertinentes), le contexte de données massivement distribuées rend l'optimisation de la recherche de solution complexe. Nous proposons alors

de limiter les agents à la résolution d'une tache simple, et de faire émerger l'optimisation du problème entier à travers une coopération entre les agents. En effet, de part leur interaction avec l'environnement (les machines présentes sur le réseau), les agents vont partager de l'information. Ce partage de l'information s'opère non pas de manière directe par une communication entre les agents, mais par un marquage de l'environnement, appelé stigmergie, qui sert alors de sorte de mémoire commune. Nous nous intéressons à l'émergence d'un marquage consistant par rapport à la tâche du système (trouver les images appartenant à la catégorie recherchée) par les comportements de la population d'agents.

2.5 Conclusion

Nous avons rapidement survolé le vaste domaine des SMA et rappelé leur intérêt dans la résolution de problèmes complexes. L'émergence d'un comportement global à la population d'agents à travers l'interaction que ceux-ci ont entre eux (soit directement, soit par leurs modifications de l'environnement) nous est apparue comme cruciale dans l'efficacité du SMA. Nous nous sommes intéressés aux agents mobiles comme extension des agents à une problématique distribuée, et avons montré qu'ils pouvaient apporter un angle d'approche nouveau dans certains domaines, et notamment dans le cas de la recherche d'informations distribuée. Nous avons aussi détaillé l'influence qu'a eu l'étude des insectes sociaux sur les systèmes multi-agents, ce qui a conduit à la conception d'algorithmes d'optimisation combinatoire, avec des applications particulièrement intéressantes dans le domaine du routage réseau, qui forme le deuxième axe de la problématique de cette thèse. La présentation des agents-mobiles et des algorithmes inspirés de l'éthologie, et en particulier des agents au comportement stigmergique, nous permet de détailler plus particulièrement comment se situent nos travaux.

Chapitre 3

Architecture du système de recherche distribuée

Notre objectif est de développer un système de recherche d'images par le contenu opérant sur un réseau. Ce chapitre présente l'aspect système de nos travaux. Nous introduisons dans un premier temps les hypothèses que nous faisons sur le contexte de répartition des images et la topologie du réseau. Puis nous détaillons l'architecture qui a été développée, en décrivant son utilisation, ainsi que ses caractéristiques.

3.1 Hypothèses sur le réseau

Considérons un ensemble d'ordinateurs reliés par un réseau. Sur chacun de ces ordinateurs se trouve une collection d'images. On souhaite mettre à disposition d'un utilisateur travaillant sur l'un des ordinateurs du réseau, un système lui permettant de retrouver le plus grand nombre d'images disponibles dans les collections des autres ordinateurs et appartenant à un certain concept. Afin de nous aider dans cet objectif, nous formulons deux hypothèses : une sur la distribution des données sur le réseau et l'autre sur la topologie des connexions entre les machines du réseau.

3.1.1 Répartition des images sur le réseau

Notre première supposition est que la distribution des images dans les différentes collections n'est pas uniforme. Certaines des catégories que l'utilisateur va rechercher se trouvent majoritairement concentrées dans un sous-ensemble restreint des bases exploitables. Pour formaliser ceci, notons, C une catégorie, et $\{\mathbf{x}_C\}_n$ l'ensemble des images de la base n appartenant à la catégorie C. L'hypothèse sur la distribution des catégories dans N bases s'exprime alors :

$$\forall C \quad \exists \{B_n\} \neq \emptyset, |\{\mathbf{x}_C\}_n| > \frac{|\{\mathbf{x}_C\}|}{N} \tag{3.1}$$

Où $|\{\mathbf{x}_C\}_n|$ représente le nombre d'images de la catégorie C dans la base n, $|\{\mathbf{x}_C\}|$ le nombre total d'images de la catégorie C, et N le nombre total de bases. Autrement dit, pour chaque catégorie, il existe un ensemble non vide de bases sur lesquelles le nombre d'images de la catégorie concernée est supérieur à une distribution uniforme sur toutes les bases.

Cette hypothèse est très souple. Elle permet des cas aussi extrêmes qu'une distribution entière de chaque catégorie sur une seule et unique base, ou bien une distribution uniforme sur l'ensemble des bases du réseau sauf une (cas où les catégories sont très diluées). Nous affinerons dans les expériences plusieurs cas de distribution.

Cette hypothèse est loin d'être irréaliste. En effet, des bases spécialisées sont d'ores et déjà disponibles comme par exemple des bases d'imagerie aérienne[1], d'imagerie médicale[2], de clip-arts (illustrations de documents)[3], etc. De même, à l'échelle du web, on peut penser que les images appartenant à un certain concept sémantique sont regroupées sur un sous ensemble de sites relativement restreint par rapport à la taille du web.

Si on fait une analogie avec une colonie de fourmis cherchant de la nourriture, cette hypothèse pourrait se traduire par le fait que la nourriture n'est pas uniformément répartie autour du nid de la colonie. Il existe alors des zones de l'espace explorable qui sont plus riches en nourriture que les autres, ce qui semble parfaitement réaliste.

3.1.2 Topologie du réseau

Afin d'optimiser le routage de la requête et des résultats relativement au renforcement donné par l'utilisateur, nous devons faire quelques hypothèses sur la topologie du réseau. Nous supposons ainsi qu'en chaque nœud du réseau, le nombre de voisins direct est relativement faible. D'un point de vue plus pratique, puisque nous proposons un routage statistique qui en tout nœud du réseau associe une probabilité de routage vers chaque voisin direct, nous ne voulons pas que ces probabilités soient écrasées par le grand nombre des destinations possibles.

Dans le cadre de réseaux, cette hypothèse n'est pas totalement irréaliste. En effet, dans les réseaux pair-à-pair, ce taux de connectivité est assez faible. Par exemple, le réseau GNutella possède une connectivité en puissance où le nombre de nœud ayant L voisins est de l'ordre de L^{-k} [Ripeanu et al., 2002], où k est une constante du système. Pour le web, le nombre de pages décroît exponentiellement avec le nombre de liens présents sur celles-ci [Broder et al., 2000]. Le cas d'un réseau local (LAN) est plus compliqué. Pour un réseau d'entreprise, par exemple celui du laboratoire où ont été menées les expérimentations, toutes les machines sont reliées entre elles, partageant le même espace d'adressage *ip*. Pour palier cette absence de topologie, nous réorganisons le réseau en décidant arbitrairement des connexions entre machines à un niveau applicatif indépendamment des

[1] http ://www.imagerie-aerienne.com/
[2] http ://www.hon.ch/HONmedia/
[3] http ://openclipart.org/

connexions physiques et protocolaires. Ainsi le réseau considéré pour déterminer le routage à apprendre est différent du réseau physique, il s'agit d'un réseau applicatif ayant les propriétés de faible connectivité que nous recherchons.

3.2 Architecture du système

Dans cette partie, nous rappelons le positionnement de nos travaux ainsi que les choix que nous avons effectués pour l'architecture de notre système. Nous donnons ensuite le synopsis de l'utilisation du système au cours d'une session de recherche. Enfin, nous détaillons les caractéristiques visuelles utilisées ainsi que les mesures de similarités qui y sont associées.

3.2.1 Synthèse

En rapport avec le contexte décrit en 3.1, nous proposons plusieurs choix dans l'architecture de notre système.

Système multi-agents - agents mobiles

Les images étant réparties sur plusieurs machines formant un réseau, nous proposons d'utiliser une population d'agents mobiles afin d'explorer le réseau à la recherche de sites contenant des images pertinentes. Chaque agent se déplace alors de machine en machine afin de trouver des images pertinentes, puis retourne sur la machine de l'utilisateur une fois celles-ci trouvées.

Agents collaboratifs - stigmergie

Afin de partager l'information concernant les sites contenant des images pertinentes, les agents marquent leur environnement. Ils augmentent ainsi les chances des agents suivants de se déplacer vers ces sites. Le marquage étant très proche des algorithmes ACO présentés en 2.3.3, le but est d'obtenir des chemins optimaux menant aux sites les plus pertinents, par stigmergie.

Signature des images - mesure de pertinence

Les images sont représentées par des vecteurs correspondant à des caractéristiques de couleur et de texture. À partir de là, la mesure de similarité permettant de déterminer quelles sont les images pertinentes est apprise en interaction avec l'utilisateur dans un bouclage de pertinence.

3.2.2 Fonctionnement de l'ensemble

La figure 3.2.2 donne le déroulement d'une session de recherche. L'utilisateur commence par donner une image étant un exemple de la catégorie recherchée (1) à l'aide de l'interface de l'agent d'interface (voir A). Le système construit la signature associée à cette image de la même façon que cela à été fait pour les images des collections disponibles sur le réseau. En se basant sur ces signatures, une fonction de similarité (2) est entraînée (un estimateur de densité de type *One-Class SVM* [Chen et al., 2001]) et passée à plusieurs agents mobiles. Ces agents sont lancés sur le réseau à la recherche de machines contenant des collections d'images (3). Leurs déplacements sont régis par des règles d'inspiration éthologique à partir des valeurs fournies par les différents agents compteurs. Dès qu'une collection d'images est trouvée, l'agent envoie un message contenant la mesure de similarité ainsi qu'une stratégie de sélection à l'agent d'indexation afin de récupérer les images à annoter (4). Une fois ces images récupérées, l'agent mobile retourne sur la machine de l'utilisateur (5) et envoie les images à l'agent d'interface. L'agent d'interface sélectionne les images à être annotées par l'utilisateur parmi le lot d'images ramenées par les agents mobiles (6).

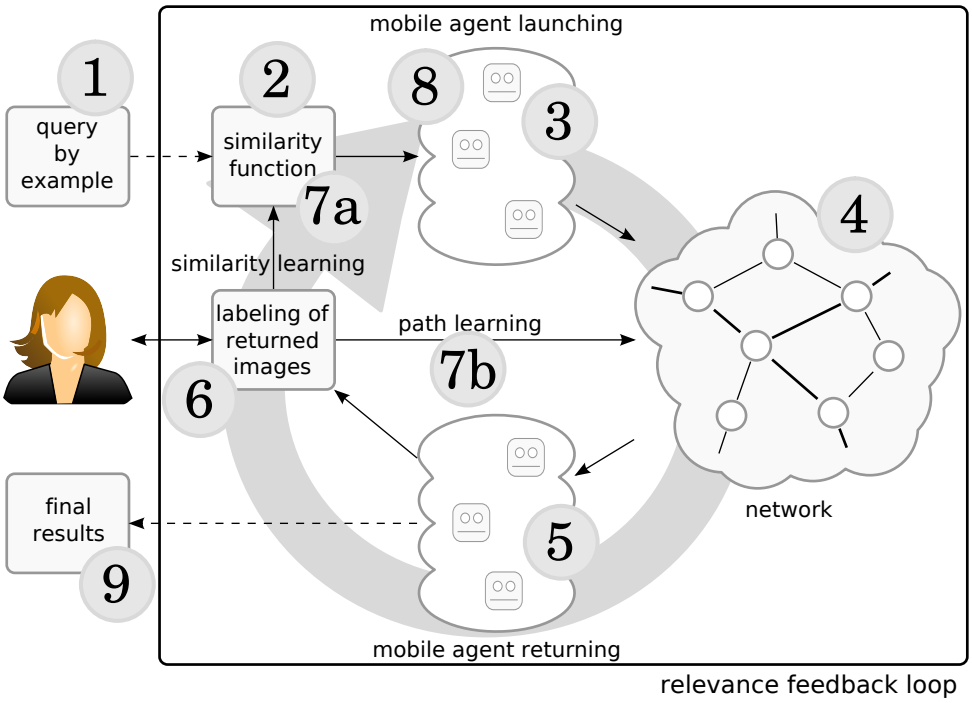


Fig. 3.1. Schéma décrivant le fonctionnement du système du point de vue de l'utilisateur. Les étapes 2 à 8 constituent le bouclage de pertinence : lancement des agents, récupération des images pertinentes, annotation par l'utilisateur, amélioration de la mesure de pertinence et des marqueurs sur le réseau.

L'utilisateur annote les images qui lui sont présentées, et ces annotations sont utilisées à la fois pour améliorer la mesure de similarité (7a) par apprentissage et pour améliorer les chemins qui sont utilisés par les agents mobiles pour se déplacer de manière à les

guider vers les sites contenant des collections d'images intéressantes du point de vue de l'utilisateur. Les agents mobiles sont relancés avec la mesure de similarité mise à jour et vont utiliser les chemins mis à jour (8).

Cette boucle, constituée de l'annotation par l'utilisateur et du parcours des agents mobiles, forme le bouclage de pertinence et est répétée plusieurs fois jusqu'à ce que l'utilisateur soit satisfait de la recherche. Les agents sont alors lancés une dernière fois pour retrouver les meilleures images au sens de la dernière amélioration de la mesure de similarité et en utilisant les chemins les plus pertinents au sens de la dernière mise à jour des compteurs (9), puis les résultats finaux sont affichés à l'utilisateur.

3.2.3 Signatures et similarité

Les primitives visuelles utilisées sont issues des travaux effectués par l'équipe ETIS pour le système de recherche RETIN [Fournier et al., 2001b, Gosselin, 2005]. Nous avons utilisé des caractéristiques de couleur et de texture.

Couleur

Selon le schéma expliqué en 1.2.3, les caractéristiques de couleurs proviennent d'un histogramme global à toute l'image sur l'espace colorimétrique *Lab*. Des pixels sont pris aléatoirement dans les images d'une base de référence afin d'estimer la population de l'espace colorimétrique. À partir de cette collection de pixel, un clustering de l'espace est effectué avec un algorithme de nuées dynamiques. Nous avons utilisé un clustering en 25 classes ou en 32 classes en fonction des bases d'images considérées dans nos tests. Les centroïdes des classes constituent la palette de couleurs de références que nous utilisons pour décrire les images.

Un histogramme est effectué sur la totalité de l'image en utilisant ce dictionnaire visuel colorimétrique. Chaque classe de l'histogramme est ensuite normalisée par rapport au reste de la base, de sorte que les variances de chacune des classes sur toute la base soient identiques. Ceci permet d'éviter qu'une classe soit plus discriminante qu'une autre du seul fait de sa grande variance.

Les valeurs de l'histogramme normalisé sont rangées dans un vecteur qui constitue la signature couleur de l'image.

Texture

Les caractéristiques de textures proviennent d'un histogramme global à toute l'image des réponses de l'image à un banc de filtre de Gabor. Nous avons utilisé quatre orientations et trois échelles différentes comme paramètres, donnant lieu à douze filtre de Gabor. De même que pour les couleurs, un échantillonage aléatoire des pixels des images de la base est effectué afin d'obtenir une collection de pixels représentative de la population de l'espace des sorties des filtres. Un clustering adaptatif en 25 ou 32 classes en fonction des bases

que nous avons testé est appliqué avec le même algorithme de nuées dynamiques. Les centroïdes constituent le dictionnaire de textures de références que nous utilisons pour décrire les images.

Un histogramme est calculé sur chacune des images en utilisant ce dictionnaire. Chacune des classes est de la même manière que pour les couleurs normalisée sur la base. Les valeurs de l'histogramme sont rangées dans un vecteur qui constitue la signature texture de l'image.

Les vecteur couleur et texture sont alors concaténés en un unique vecteur afin de former la signature unique de l'image, selon une méthode de fusion précoce.

Similarité

La mesure de similarité est basée sur un *SVM* appris sur la base des signatures annotées par l'utilisateur. Puisque ces signatures proviennent d'images distribuées sur différentes machines du réseau, la requête transportée par les agents mobiles est constituée à la fois des coefficients α du *SVM* mais aussi des signatures des images qui y sont associées.

Nous avons utilisé un noyau gaussien dans toutes nos expérimentations, avec une distance soit du χ^2, soit du "χ^1" en fonction de ce qui donnait les meilleurs résultats. L'algorithme d'optimisation du SVM est *SMO* tel que décrit par [Platt, 1999], et est recalculé sur l'ensemble d'apprentissage à chaque lancement d'un agent.

3.3 Parcours du réseau

Le réseau est une composante importante de nos travaux de part le fait qu'il contient à la fois les sources de données images, mais est aussi une donnée en lui même. Ainsi, il faut pouvoir explorer le réseau afin de trouver les sources d'images intéressantes, mais aussi exploiter les sources déjà connues. Ce compromis exploitation/exploration est très important dans un système dynamique comme le notre, comme il sera étudié au chapitre 4.

3.3.1 Définitions

Dans cette thèse, le réseau que nous utilisons est étudié au niveau applicatif de notre système ; les couches physiques et protocolaires sortent du cadre de notre étude. Dans ces conditions, on donne une définition simple d'un réseau :

Définition On appelle *réseau*, un ensemble d'ordinateurs perçus par le système, reliés par un ensemble de connections permettant le transfert de données.

Le réseau physique et/ou protocolaire peut être largement plus complexe que cela. Il peut par exemple contenir des connections et des machines (routeurs, passerelles, *etc*) qui ne sont pas vues par le système et qui par conséquent n'appartiennent pas à ce que nous appelons le réseau. L'ensemble des ordinateurs et des connections forment un graphe. Sur un réseau donné, on peut distinguer trois types de machines différentes :

- l'ordinateur de l'utilisateur.
- une machine contenant une base d'images.
- une machine n'étant pas l'ordinateur de l'utilisateur, et ne contenant pas de base d'images (que l'on appelle machine *nue*).

L'ordinateur de l'utilisateur est unique sur le réseau (du point de vue d'une session de recherche), alors que les autres machines peuvent être en grand nombre. La figure 3-2 montre un exemple de réseau sur lequel nous avons travaillé. On peut y distinguer l'ordinateur de l'utilisateur, quatre machines contenant des bases d'images, et quatre autres machines. Le taux de connections par machine est assez faible (de l'ordre de trois).

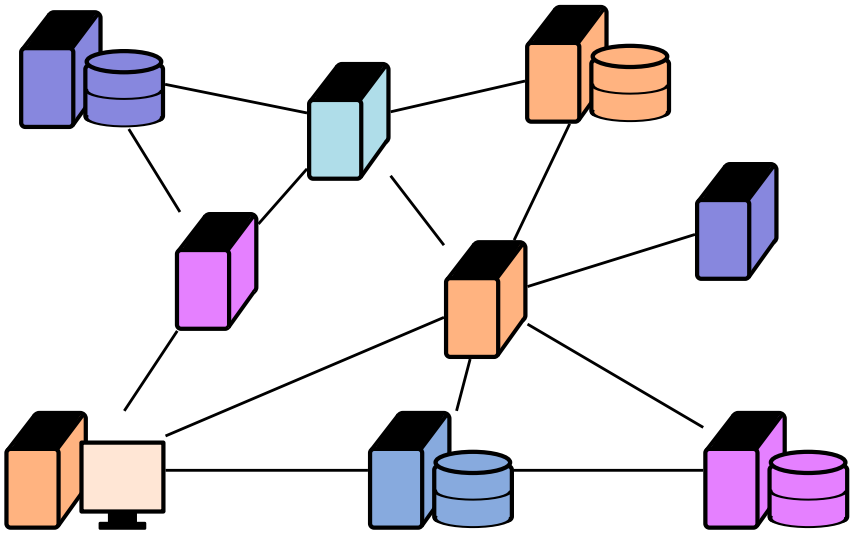


Fig. 3-2. Exemple de *réseau* contenant l'ordinateur de l'utilisateur (en bas à gauche), quatre ordinateurs munis de bases d'images (deux en haut et deux en bas) ainsi que quatre ordinateurs nus (au milieu).

Afin de pouvoir manipuler les déplacements des agents sur le réseau, définissons un chemin du réseau :

Définition On appelle *chemin* la suite de machines reliées par des connections commençant par l'ordinateur de l'utilisateur, et se terminant par n'importe quelle machine du réseau.

Ainsi, on peut dire qu'un agent emprunte un chemin lorsqu'il se déplace sur le réseau. Le chemin le plus simple est celui reliant l'ordinateur de l'utilisateur à un de ses voisins.

Ainsi il peut exister plusieurs chemins menant à la même machine, certains plus long (en nombre de connections) ou plus rapides (en temps mis pour effectuer tous les déplacements du chemin) que d'autres. Il peut aussi y avoir des chemins qui ne contiennent aucune base d'images. La figure 3-3 illustre la définition des chemins par un exemple issu du réseau de la figure 3-2.

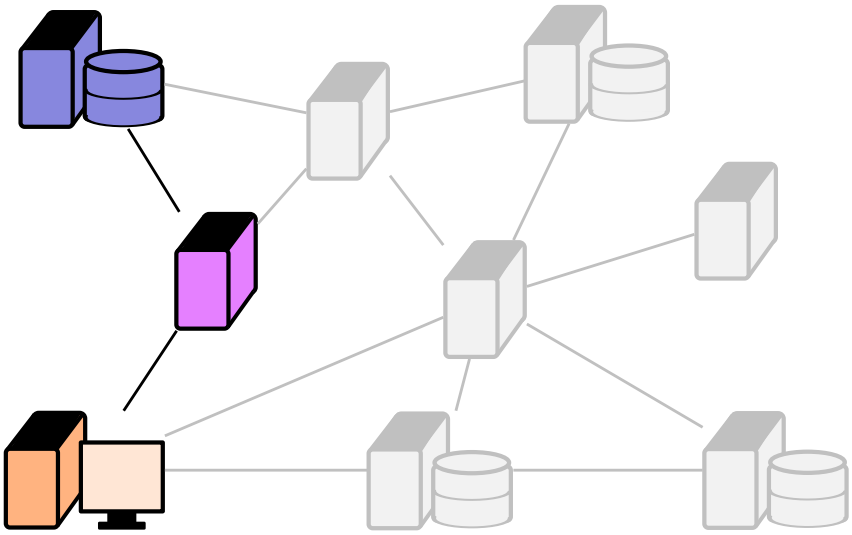


Fig. 3-3. Exemple de chemin composé de l'ordinateur de l'utilisateur, un ordinateur intermédiaire et se terminant par un ordinateur muni d'une base d'images. Le reste du réseau est grisé.

3.3.2 Parcours

Les agents mobiles partent de la machine de l'utilisateur et parcourent le réseau en empruntant des chemins. On conçoit les agents mobiles comme une machine à état. Pour effectuer des déplacements, ils sont pourvus d'un comportement décrit dans l'algorithme suivant : soit $prev_{host}$ une pile contenant des noms de machines, et E l'état interne de l'agent.

Comportement d'un agent mobile

1. L'état interne E est mis à *FORWARD*.
2. Si E est *FORWARD* aller en 3, s'il est *HOMING* aller en 2.
3. Chercher une base d'image locale. Si une telle base est présente, aller en 4, sinon en 5.
4. Copier les images recherchées de la base, passer l'état interne E à *HOMING*, puis aller en 6.
5. Empiler la machine locale dans $prev_{host}$. Choisir une machine parmi les destinations possibles et s'y téléporter. Aller en 2.
6. Si la machine locale est l'ordinateur de l'utilisateur, renvoyer les images et aller en 7. Sinon, dépiler une machine de $prev_{host}$ et s'y téléporter. Aller en 2.
7. Fin du programme.

Comme on peut le voir dans l'étape 4, les agents s'arrêtent dès qu'ils ont trouvé une base d'images. De fait, les chemins utilisés par les agents ne peuvent pas contenir une base d'images ailleurs que sur la machine terminale. Les agents parcourent donc un chemin dans un sens jusqu'à la base d'images, puis dans l'autre sens jusqu'à l'ordinateur de l'utilisateur. Le choix de la machine choisie parmi les destinations possibles est détaillé au chapitre 4.

3.4 Conclusion

Dans ce chapitre nous avons détaillé les hypothèses faites sur le contexte et l'architecture de notre système. Ceci nous a permis de définir le cadre et la portée de nos travaux en les limitant à des réseaux de faible connectivité et sur lesquels les images sont réparties de manière non-uniforme.

Nous avons ensuite présenté une synthèse de l'architecture de notre système de recherche d'information distribuée et de son fonctionnement, avant de détailler le parcours proprement dit des agents. Il ressort de cette présentation que ce schéma exploite une forte dépendance entre l'optimisation du réseau pour la recherche de sites pertinents effectuée par les agents, et l'optimisation de la mesure de similarité des images ramenées par les agents.

Nous allons, dans les deux parties qui suivent, nous attacher à décrire précisément nos stratégies d'apprentissage dans ce contexte particulier.

Deuxième partie

Apprentissage actif distribué

Chapitre 4

Apprentissage actif distribué

Dans ce chapitre, nous détaillons nos travaux dans le cadre de la problématique de l'apprentissage actif distribué pour la recherche d'images par le contenu. Ceci se base sur le système que nous avons décrit au chapitre précédent, avec une requête image (*query by example*), et un bouclage de pertinence constitué d'annotations fournies par l'utilisateur sur les images ramenées par les agents mobiles.

4.1 Introduction

Comme nous l'avons décrit au chapitre 1, la recherche d'informations distribuées consiste tout d'abord en une première étape de sélection des sites pertinents, puis une sélection locale et enfin une fusion des différents résultats. À la différence de la recherche classique de texte, notre système fonctionne en interaction avec l'utilisateur via un bouclage de pertinence afin de construire par apprentissage la mesure de pertinence servant à évaluer les images. Il s'agit alors de concevoir une stratégie d'apprentissage active minimisant le nombre d'annotations que l'utilisateur doit fournir pour obtenir des résultats satisfaisants. Cette succession de trois étapes va donc devoir être répétée un certain nombre fois dans le cadre du bouclage de pertinence.

4.1.1 Apprentissage actif distribué

L'idée de l'apprentissage actif est de sélectionner les éléments de l'ensemble d'apprentissage de manière à réduire le nombre d'annotations nécessaires à l'obtention de résultats "convenables" pour l'utilisateur. Dans le cadre d'une construction interactive de la mesure de pertinence, cela revient à "soigneusement" sélectionner les exemples à faire annoter à une itération donnée de manière à minimiser le nombre d'itérations nécessaires pour obtenir des résultats "satisfaisants".

Il existe deux grande familles de stratégies de sélection active : celles qui sont basées sur la maximisation de la performance de la classification, et celles qui sont basées sur la

minimisation de l'incertitude sur la mesure de similarité (*cf* partie 1.1.4.5). C'est à cette dernière famille que nos travaux appartiennent et tentent d'étendre au cas de la recherche d'images distribuées.

Dans ce contexte distribué, il y a deux processus de décision qui peuvent être optimisés :

– La sélection des sites depuis lesquels récupérer les images.
– La sélection des images sur les sites choisis.

Nous étendons alors l'apprentissage actif à ce schéma, ce qui revient à choisir les sites lors du premier processus de manière à améliorer la vitesse ou la qualité de l'apprentissage de la mesure de similarité, puis à opérer la sélection des images sur ces sites avec le même objectif.

L'extension de la stratégie basée sur l'incertitude devient alors la suivante : les sites à sélectionner sont préférablement ceux contenant des images annotées positivement. En effet, les sites n'ayant que des images annotées négativement n'ont qu'une faible chance de contenir des images pertinentes et n'ont donc pas d'intérêt pour la recherche. Les images à sélectionner sont les plus incertaines du point de vue de la mesure de pertinence. La sélection des images se faisant sur des sites contenant à la fois des images pertinentes et non-pertinentes, il s'agit alors d'optimiser la mesure de pertinence sur ces sites en sélectionnant les images de ces sites qui l'améliorent le plus.

4.2 Pertinence des sites

Dans cette partie, nous formalisons l'étape de sélection des sites. Nous introduisons dans un premier temps la notion de pertinence d'un site, puis nous montrons comment cette notion peut être utilisée pour la construction de l'ensemble d'apprentissage nécessaire à l'entraînement de la mesure de similarité.

4.2.1 Pertinence des sites

Notre stratégie consiste à apprendre conjointement la localisation des sites contenant la catégorie recherchée et la mesure de similarité déterminant la pertinence des images. Un site i est dit pertinent, s'il contient *"suffisamment"* d'images de la catégorie recherchée du point de vue de l'utilisateur. Définissons une grandeur $r_i \in \mathbb{R}^+$ pour représenter la pertinence :

Définition Si un site i n'est pas pertinent (noté $\bar{\mathcal{R}}(i)$) , sa pertinence r_i est nulle. À l'inverse, si un site est pertinent (noté $\mathcal{R}(i)$), nous proposons que sa pertinence soit définie par la proportion d'images pertinentes qu'il contient, en rapport avec ce que lui attribue l'utilisateur :

$$r_i = \frac{\text{Card}(\{\mathbf{x}_j \in B_i \,|y_j = 1\})}{\text{Card}(\{\mathbf{x}_j \in B_i\})} \tag{4.1}$$

Avec $\mathbf{x}_j$ les images de la base B_i. Alors :

$$\bar{\mathcal{R}}(i) \Rightarrow r_i = 0 \tag{4.2}$$

$$\mathcal{R}(i) \Rightarrow r_i > 0 \tag{4.3}$$

Afin de pouvoir comparer les pertinences des sites, nous définissons aussi la pertinence normalisée :

Définition On appelle r'_i *pertinence normalisée* le rapport de la pertinence du site i sur la somme des pertinences de tous les sites disponibles sur le réseau depuis i :

$$r'_i = \frac{r_i}{\sum_k r_k} \tag{4.4}$$

Cette grandeur représente en quelque sorte la part de pertinence du site i pour la catégorie recherchée.

4.2.2 Sélection des sites

Puisque les images pertinentes ne sont pas connues *a priori* mais vont être déterminées par apprentissage à l'aide de l'expertise de l'utilisateur, les r_i ne le sont pas non plus. Nous proposons alors d'approcher la pertinence des sites en nous basant sur l'ensemble d'annotations disponibles à l'issue d'une itération du bouclage de pertinence.

Soit N sites disponibles, ayant chacun pour pertinence r_i telle que perçue par l'utilisateur. Soit un ensemble $\mathcal{A}$ constitué d'un ensemble de I images $\{\mathbf{x}_j\}$ annotées par l'utilisateur et l'ensemble des labels $\{y_j\}$ correspondant. Nous proposons alors d'approximer la pertinence des sites par la répartition des labels sur les différents sites :

Proposition 1. *On approxime la pertinence r_i d'un site i par $\hat{r}_i$ la proportion d'images provenant de la base B_i annotées positivement par rapport au nombre total d'images de la base B_i annotées.*

$$\hat{r}_i = \frac{\mathrm{Card}(\{\mathbf{x}_j \in B_i \ \cap \mathcal{A} | y_j = 1\})}{\mathrm{Card}(\{\mathbf{x}_j \in B_i \cap \mathcal{A}\})} \tag{4.5}$$

L'idée est alors, au fil des itérations du bouclage de pertinence faisant s'accroître la taille de $\mathcal{A}$, de faire tendre cette valeur approchée $\hat{r}_i$ vers la vraie valeur de pertinence r_i. On peut de la même manière définir une valeur approchée de la pertinence normalisée :

Proposition 2. *On approxime la pertinence normalisée r'_i d'un site i par $\hat{r'}_i$ le rapport de $\hat{r}_i$ sur la somme des $\hat{r}_k$ de tous les sites disponibles :*

$$\hat{r'}_i = \frac{\hat{r}_i}{\sum_k \hat{r}_k} \tag{4.6}$$

Une fois la pertinence de chaque site calculée, I images sont sélectionnées à chaque tour dans le bouclage de pertinence afin d'être annotées. Le nombre d'image I_i choisies dans chaque site i est relatif à la pertinence normalisée $\hat{r'}_i$ de ce site :

$$I_i = \hat{r'}_i \cdot I \tag{4.7}$$

$$= \frac{\hat{r}_i}{\sum_k \hat{r}_k} I \tag{4.8}$$

Les images sont sélectionnées sur chaque site en suivant une stratégie active locale détaillée en 4.5.3. Ainsi, l'ensemble d'entraînement $\mathcal{A}$ est enrichi à chaque tour par un nombre d'images de chaque site proportionnel à la dernière estimation de la part de pertinence du site.

4.3 Synthèse

Une session de recherche commence par une ou plusieurs images annotées par l'utilisateur (*query by example*). Cet ensemble constitue l'état initial de d'un ensemble d'apprentissage $\mathcal{A}_{t(0)}$ qui va servir à la fois à l'entraînement d'un classifieur $f_{\mathcal{A}}$ déterminant les images pertinentes, mais aussi au calcul d'une valeur approchée $\hat{r}_i$ de la pertinence de chaque site i.

Ces deux outils, $f_{\mathcal{A}}$ et les $\hat{r}_i$, permettent de sélectionner un ensemble de I images qui sont présentées à l'utilisateur afin d'être annotées. L'ensemble des I images et de leurs annotations ainsi obtenues constitue le résultat d'une itération du bouclage de pertinence. Cet ensemble est alors ajouté à l'ensemble $\mathcal{A}$, ce qui permet de mettre à jour $f_{\mathcal{A}}$ et les $\hat{r}_i$ afin de préparer l'itération suivante.

4.3.1 Entraînement du classifieur

À partir de l'ensemble d'images annotées $\mathcal{A} = \{\mathbf{x}_j, y_j\}$, on peut entraîner un classifieur. Nous utilisons les systèmes à vaste marge (SVM) comme classifieur. Plutôt que d'utiliser le signe de la sortie du classifieur, nous utilisons directement la sortie normalisée comme valeur de pertinence :

$$f_{\mathcal{A}}(\mathbf{x}) = \sum_{\mathbf{x}_j \in \mathcal{A}} \alpha_j y_j k(\mathbf{x}_j, x) \in \{-1, 1\} \tag{4.9}$$

Ainsi, notre mesure de pertinence dépend de l'ensemble d'apprentissage $\mathcal{A}$. Or, d'une part cet ensemble varie au fil des itérations du bouclage de pertinence, ce qui veut dire que notre mesure de pertinence va s'affiner avec l'enrichissement de $\mathcal{A}$. D'autre part, cet ensemble est construit dans des proportions relatives à la pertinence normalisée des sites.

Notre mesure de pertinence est donc d'autant plus précise sur un site que celui-ci est pertinent.

4.3.2 Détail d'une itération du bouclage

Soit N sites i de bases B_i. Pour la recherche d'une catégorie donnée, les N r_i sont fixés (le contenu des bases ne change pas et l'expertise de l'utilisateur non plus) et le processus de passage de l'itération n à l'itération $n+1$ est le suivant :

1. Soit $\mathcal{A}(n)$ l'ensemble d'images $\{\mathbf{x}_j\}_n$ et de leurs annotations $\{y_j\}_n$ à l'itération n.
2. À partir de $\mathcal{A}(n)$ on calcule une valeur approchée de la pertinence $\hat{r}'_i$ (*cf* équation 4.6), ainsi que la mesure de pertinence $f_{\mathcal{A}(n)}$ (*cf* équation 4.9).
3. À l'aide de $f_{\mathcal{A}(n)}$ et de la pertinence $\hat{r}'_i$, on sélectionne I images $\{\mathbf{x}_j\}_{n+1}$, selon la proportion définie en 4.7.
4. On construit l'ensemble $\mathcal{A}(n+1) = \mathcal{A}(n) \cup \{\mathbf{x}_j, y_j\}_{n+1}$, où les y_j sont les annotations données par l'utilisateur aux I images $\{\mathbf{x}_j\}_{n+1}$.

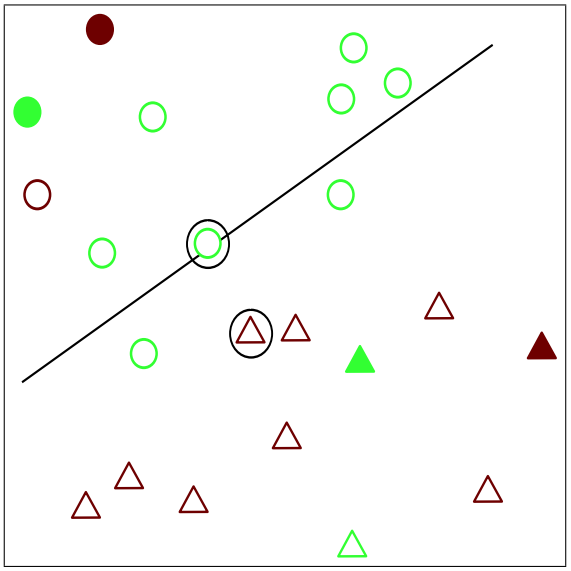
4.3.3 Stratégie active globale

Un lot d'images est sélectionné depuis les sites et annoté à chaque itération du bouclage de pertinence. Les annotations ainsi obtenues sont utilisées de deux manière :

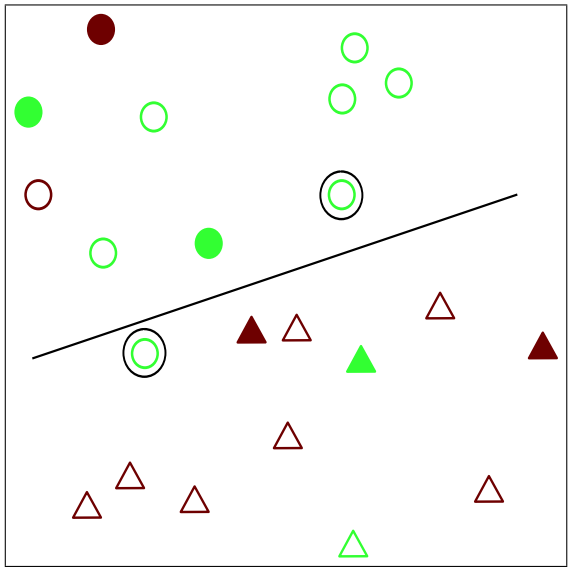
- D'un côté, elle permettent la mise à jour de l'estimation de la pertinence $\hat{r}_i$ de chaque site. Puisque cette pertinence est prise en compte pour la sélection des sites dans lesquels sont choisies les images à annoter, l'amélioration de l'estimée des r_i a une influence sur le classifieur par le biais de la construction de l'ensemble d'apprentissage (*cf* figure 4.1).
- De l'autre côté, elles permettent la mise à jour de la fonction de similarité. Or, cette fonction détermine le tri utilisé par la stratégie active de sélection des images à annoter (comme vu en 4.3.2). De fait, l'amélioration de $f_{\mathcal{A}}$ a une influence sur l'estimation des $\hat{r}_i$ par le biais de la sélection des images à annoter. Les deux mécanismes d'apprentissage faisant évoluer $f\mathcal{A}$ et $\hat{r}_i$ sont alors en interaction l'un avec l'autre, l'amélioration de l'un permettant l'amélioration de l'autre.

4.4 Exploration du réseau

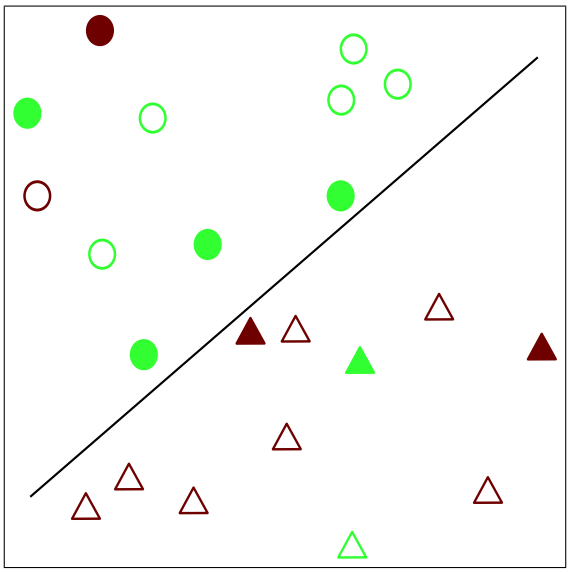
Dans la stratégie que nous venons de décrire, il reste à déterminer comment les pertinences $\hat{r}_i$ des sites sont calculées. En effet, l'étape 2 (voir 4.3.2) de la stratégie suppose la connaissance des $\hat{r}_i$ de toutes les destinations, données qui ne sont pas *a priori* disponibles.



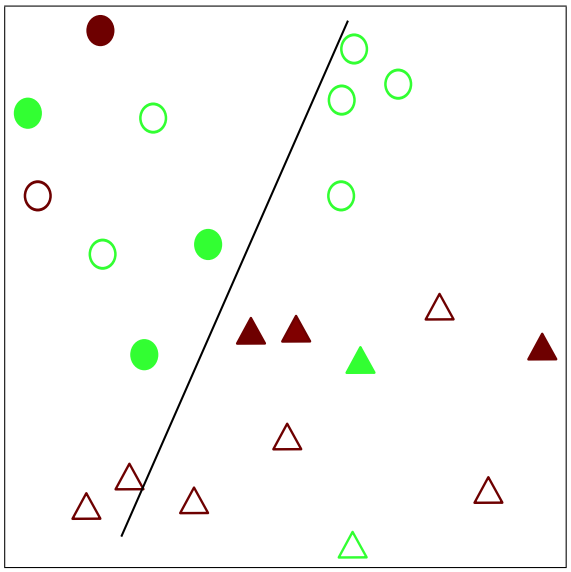
(a) La première itération sélectionne deux images à annoter. La plus proche de la frontière est sélectionnée sur chaque site, puisque les $\hat{r}_i$ sont égaux.



(b) Comme il y a plus d'annotations positives sur A que sur B, les deux images de l'itération suivante sont choisies sur A.



(c) La stratégie globale permet d'améliorer la classification sur la base contenant le plus d'images pertinentes.



(d) En effectuant un échantillonage uniforme sur les collections, la classification est moins bonne.

Fig. 4.1. Exemple de la stratégie globale avec deux bases A (vert clair) et B (rouge foncé). A contient beaucoup d'images pertinentes alors que B n'en a que peu. Un symbole plein représente une image annotée, les autres étant les images non annotées. La requête initiale est constituée de deux images pertinentes (cercle) et deux non pertinentes (triangles) venant de A et B. La stratégie se concentre sur A puisque celle-ci contient plus de bons exemples, et améliore ainsi la classification.

En fait, deux problèmes se posent : il s'agit d'une part de trouver des sites comportant une base d'images, c'est à dire d'explorer un réseau dont le contenu exact n'est pas connu ; et d'autre part d'optimiser le parcours sur ce réseau afin de converger vers les sites qui contiennent les images pertinentes. De plus, la résolution de ces deux problèmes doit se faire au cours d'une session, avec pour seule aide l'interaction avec l'utilisateur. Il s'agit donc de satisfaire un compromis entre exploration et exploitation du réseau.

Comme on l'a vu en 2.2.1, les agents mobiles se prêtent bien à l'exploration d'un réseau pour la découverte de services. En effet, ceux-ci se déplacent de proche en proche, jusqu'à ce que la ressource désirée soit trouvée. Dans notre cas, il s'agit à chaque itération de trouver des sites contenant des bases d'images et d'en effectuer un échantillonage afin d'évaluer les $\hat{r}_i$ qui y sont associés à l'aide des annotations de l'utilisateur. Cet échantillonage doit être optimisé de manière à ce que les évaluation des $\hat{r}_i$ soient bonnes, c'est à dire que la manière dont le réseau est exploré et un sous-ensemble de sites (parmi l'ensemble des sites accessibles sur le réseau) est sélectionné à chaque itération doit mener à des $\hat{r}_i$ proches des r_i.

Pour résoudre ce problème d'optimisation, les algorithmes inspirés des insectes sociaux semblent particulièrement adaptés (*cf* 2.3). En effet, on dispose d'un ensemble de chemins du réseau auxquels il faut associer un ensemble de valeurs (les $\hat{r}_i$). Ce sont les parcours dont les valeurs associées sont les plus élevées (car les plus pertinents) qui nous intéressent principalement.

Apprentissage par renforcement

En s'inspirant des algorithmes de renforcement, on associe à chaque connexion du réseau une valeur liée au coût de la transition d'une machine à une autre. Les agents effectuent différents parcours de certains chemins du réseau, en utilisant ces coûts de transition de sorte qu'ils optimisent le coût global, et en ramènent des images. On se sert alors des annotations fournies par l'utilisateur comme signal renforcement afin de faire évoluer les coûts de transition.

Ces coûts de transition sont finalement très proches des valeurs de $\hat{r}_i$ que l'on cherche à obtenir. On propose d'utiliser les $\hat{r}_i$ comme valeurs associées aux transitions sur le réseau, afin d'explorer celui-ci, et d'utiliser des règles de renforcement proches de celles des algorithmes de la famille *ACO* afin de les optimiser.

Le détail de l'exploration/exploitation du réseau par les agents mobiles de manière à obtenir une évaluation des $\hat{r}_i$ s'inscrivant dans le cadre du bouclage de pertinence est présenté ci-après.

4.4.1 Apprentissage des chemins pertinents par SMA

L'idée générale est de doter nos agents d'un comportement similaire à celui observé par les fourmis (*cf* 2.3). C'est à dire qu'il vont marquer leur environnement de déplacement afin de se rappeler les chemins intéressants. À cet effet, on dispose sur chaque machine

d'un compteur numérique, qu'on appelle *marqueur*, et qui va jouer le même rôle que les phéromones chez les fourmis. Ces marqueurs sont utilisés dans le processus de décision qui régit les mouvements des agents. Ils évoluent au cours du temps afin d'influencer les déplacements des agents vers les bases pertinentes.

Considérons l'évolution des marqueurs sur un chemin. Cette évolution se fait par rapport aux agents parcourant ce chemin et est donc indépendante du reste du réseau. Elle est alors relative à un temps local n que nous définissons :

Définition On nomme n le temps discret relatif à un site i donné (ou instant local).

Ce temps est incrémenté à chaque fois qu'un agent a effectué un aller-retour vers ce site et que l'utilisateur a donné l'annotation correspondant au résultat retourné. En toute rigueur, on devrait noter ce temps n_i puisqu'il est défini uniquement pour le site i, cependant, par souci de simplification des notations on le notera n en se rappelant bien que lorsque nous parlons d'une grandeur dépendant de n, cela n'est valable que pour le site concerné. En cas d'ambiguïtés, on reviendra à la notation n_i.

Définition On appelle *marqueur* $m_i(n)$ la variable associée au site i évoluant avec le temps discret n, permettant de déterminer les déplacements des agents vers ce site.

Pour chaque destination i de l'ensemble des destinations possibles $dest_s$ à partir du site s, on calcule une probabilité de déplacement (ou saut) à partir de la valeur courante des marqueurs (*cf* figure 4.2) :

$$p_{s\to i} = \frac{m_i(n_i)}{\sum_{d\in dest_s} m_d(n_d)} \tag{4.10}$$

Cette probabilité de saut est donc proportionnelle à la valeur des marqueurs. Pour simplifier les notations, on la notera simplement p_i. Pour chaque agent voulant se déplacer, un tirage est effectué selon les p_i et l'agent se déplace vers la destination ainsi sélectionnée. Le but est de faire évoluer les probabilités de saut, et les marqueurs associés, vers la pertinence normalisée des sites. Ainsi, si on considère l'ensemble d'une population d'agents, ceux-ci se déplaceront en moyenne proportionnellement à la pertinence des sites, et les images qu'ils ramèneront seront alors en nombre comparable à la pertinence normalisée des sites dont elles proviennent, conformément à ce qui a été décrit en 4.2.2.

Règles de renforcement

À l'initialisation, tous les marqueurs ont la même valeur non nulle de sorte que les probabilités de saut soient identiques pour tous les sites. À chaque incrémentation du temps local n, la valeur du marqueur est mise à jour avec les règles suivantes :

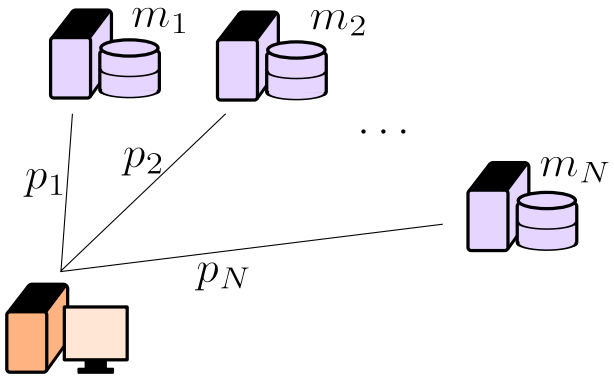


Fig. 4-2. Chacun des N sites dispose d'un marqueur m_i. Une probabilité de saut p_i est associée à chacune de ces destinations, calculée à partir des m_i, et les agents effectuent un tirage parmi les p_i pour déterminer leur destination.

1. Déplacement

Lorsque l'agent se déplace vers le site, le marqueur est décrémenté :

$$m_i \longleftarrow \alpha m_i \tag{4.11}$$

Avec α une constante positive inférieure à un. Cette règle sert à modéliser l'évaporation des phéromones chez les fourmis et permet d'oublier les chemins qui ne sont plus pertinents.

2. Retour

Lorsque l'agent revient du site, le marqueur est incrémenté :

$$m_i \longleftarrow m_i + \beta a \tag{4.12}$$

Où a vaut 1 si une base d'images a été trouvée, et 0 sinon, et β est une constante positive. Cette règle permet de renforcer les chemins menant à des bases d'images et donc de ne router les agents que vers des collections.

3. Annotation

Quand une image est annotée, les marqueurs correspondant au site dont elle est issue sont renforcés :

$$m_i \longleftarrow m_i + \gamma u \tag{4.13}$$

Où u vaut 1 si l'annotation est positive et 0 sinon, et γ est une constante positive. On peut faire le lien entre u et y l'annotation associée à une image : $u = 1 \Leftrightarrow y = 1$ et $u = 0 \Leftrightarrow y = -1$.

4.4.2 Évolution des marqueurs

Ce renforcement vise à faire converger les probabilités de déplacement p_i vers l'estimation de la pertinence normalisée $\hat{r}'_i$ telle que décrite précédemment. m_i est un processus aléatoire dépendant des deux variables aléatoires u_i et a_i· L'équation globale de renforcement s'écrit :

$$m_i \longleftarrow \alpha m_i + \beta a + \gamma u \tag{4.14}$$

À un instant local n, l'expression du marqueur m_i s'écrit alors :

$$m_i(n) = \alpha m_i(n-1) + \beta a_i(n-1) + \gamma u_i(n-1) \tag{4.15}$$

Ce qui peut se déduire des conditions initiales de la sorte :

$$m_i(n) = \alpha^{n-1} m_i(0) + \beta \sum_{0}^{n-1} \alpha^{n-1-k} a_i(k) + \gamma \sum_{0}^{n-1} \alpha^{n-1-k} u_i(k) \tag{4.16}$$

Or, puisque a et u sont supposés indépendants du temps, leur espérance est la même à n'importe quel instant. Alors l'espérance du marqueur s'écrit :

$$E[m_i(n)] = \alpha^{n-1} m_i(0) + \beta E[a_i] \sum_{0}^{n-1} \alpha^{n-1-k} + \gamma E[u_i] \sum_{0}^{n-1} \alpha^{n-1-k} \tag{4.17}$$

$$= \alpha^{n-1} m_i(0) + \beta E[a_i] \sum_{0}^{n-1} \alpha^{k} + \gamma E[u_i] \sum_{0}^{n-1} \alpha^{k} \tag{4.18}$$

$$= \alpha^{n-1} m_i(0) + (\frac{1-\alpha^{n-2}}{1-\alpha})(\beta E[a_i] + \gamma E[u_i]) \tag{4.19}$$

Et sa limite à l'infini est :

$$\lim_{n\to\infty} E[m_i] = \frac{\beta E[a_i] + \gamma E[u_i]}{1-\alpha} \tag{4.20}$$

Pour étudier les liens entre les marqueurs et la pertinence des sites, nous avons tout d'abord établi le théorème suivant :

Théorème 1. *Si $m_i(n)$ est le marqueur de la machine i au temps n, alors il converge en probabilité vers $E[m_i]$:*

$$\forall \varepsilon, \forall \delta, \exists N \mid \forall n \geq N \Rightarrow P(|m_i(n) - E[m_i(n)]| > \varepsilon) < \delta \tag{4.21}$$

Démonstration. On utilise l'inégalité de Markov :

$$\forall \Delta > 0, P(Y > \Delta) \leq \frac{E[Y]}{\Delta} \tag{4.22}$$

appliquée sur la variable err définie par :

$$err = |m_i(n) - E[m_i(n)]| \tag{4.23}$$

Alors

$$\begin{aligned}
\forall \varepsilon, P(err > \varepsilon) &\leq \frac{E[err]}{\varepsilon} && (4.24)\\
&\leq \frac{E[m_i(n) - E[m_i(n)]]}{\varepsilon} && (4.25)\\
&\leq \frac{E[\alpha^{n-1} m_i(0) + \sum_0^{n-1} \alpha^{n-1-k}\Big(\beta a_i(k) + \gamma u_i(k)\Big) - E[m_i(n)]]}{\varepsilon} && (4.26)\\
&\leq \frac{\alpha^{n-1} m_i(0) + \sum_0^{n-1} \alpha^{n-1-k}\Big(\beta E[a_i] + \gamma E[u_i]\Big) - E[m_i(n)]}{\varepsilon} && (4.27)
\end{aligned}$$

Or :

$$\forall n, \sum_0^{n-1} \alpha^{n-1-k} < \frac{1}{1-\alpha} \tag{4.28}$$

Donc :

$$P(err > \varepsilon) < \frac{\alpha^{n-1} m_i(0) + \frac{\beta E[a_i] + \gamma E[u_i]}{1-\alpha} - E[m_i(n)]}{\varepsilon} \tag{4.29}$$

Soit en remplaçant $E[m_i(n)]$ d'après 4.19 :

$$P(err > \varepsilon) < \alpha^{n-2} \frac{\beta E[a_i] + \gamma E[u_i]}{\varepsilon(1-\alpha)} \tag{4.30}$$

Alors, $\forall \delta$, on peut toujours choisir N tel que :

$$\alpha^{N-2} < \frac{\varepsilon(1-\alpha)}{\beta E[a_i] + \gamma E[u_i]} \delta \tag{4.31}$$

Et on en déduit que $\forall n > N$:

$$\begin{aligned}
P(err > \varepsilon) &< \alpha^{N-2} \frac{\beta E[a_i] + \gamma E[u_i]}{\varepsilon(1-\alpha)} && (4.32)\\
P(err > \varepsilon) &< \delta && (4.33)
\end{aligned}$$

On a donc bien montré que :

$$\lim_{n\to\infty} P\big(m_i(n) = \frac{\beta E[a_i] + \gamma E[u_i]}{1-\alpha}\big) = 1 \tag{4.34}$$

□

Puisque les marqueurs convergent en probabilité, on peut en déduire de même une estimation asymptotique de la probabilité de saut p_i :

$$\lim_{\substack{\forall d \in dest \\ n_d \to \infty}} p_i = \lim_{\substack{\forall d \in dest \\ n_d \to \infty}} \frac{m_i(n)}{\sum_k m_k(n_d)} \tag{4.35}$$

C'est à dire :

$$\lim_{\substack{\forall d \in dest \\ n_d \to \infty}} p_i = \frac{\beta E[a_i] + \gamma E[u_i]}{\sum_{d\in dest} \beta E[a_d] + \gamma[u_d]} \tag{4.36}$$

Dans le cas particulier où $\beta = 0$, on trouve alors la convergence de la probabilité p_i comme étant :

$$\lim_{\substack{\forall d \in dest \\ n_d \to \infty}} p_i = \frac{\gamma E[u_i]}{\sum_{d\in dest} \gamma E[u_d]} \tag{4.37}$$

On en déduit le corollaire suivant :

Corollaire. *Considérant un ensemble de destinations dest. Si $\beta = 0$, alors la probabilité de déplacement vers chacune de ces destinations est asymptotiquement :*

$$\lim_{\substack{\forall d \in dest \\ n_d \to \infty}} p_i = \frac{E[u_i]}{\sum_{d\in dest} E[u_d]} \tag{4.38}$$

On peut ensuite estimer $E[u_i]$ par l'estimateur empirique, que l'on a précédemment nommé $\hat{r}_i$:

$$\hat{r}_i = \frac{\sum_{y_j=1, \mathbf{x}_j \in B_i} 1}{N_i} \tag{4.39}$$

$$\approx E[u_i] \tag{4.40}$$

Avec $N_i = \text{Card}(\{\mathbf{x}_j \in B_i \cap \mathcal{A}\})$.

Et par conséquent :

$$\hat{r}'_i = \frac{\hat{r}_i}{\sum_{d \in dest} \hat{r}_d} \tag{4.41}$$

$$\approx \frac{E[u_i]}{\sum_{d \in dest} E[u_d]} \tag{4.42}$$

$$\approx \lim_{\substack{\forall d \in dest \\ n_d \to \infty}} p_i \tag{4.43}$$

En probabilité, les probabilités de saut p_i et les pertinences normalisées approchées $\hat{r}'_i$ attribuées par l'utilisateur convergent donc bien vers la même valeur. La figure 4.4.2 donne une comparaison entre une moyenne sur un ensemble de 1000 réalisations des p_i et l'estimateur $\hat{r}'_i$. Comme on peut le voir, les résultats sont très proches pour un nombre d'annotations supérieur à 10.

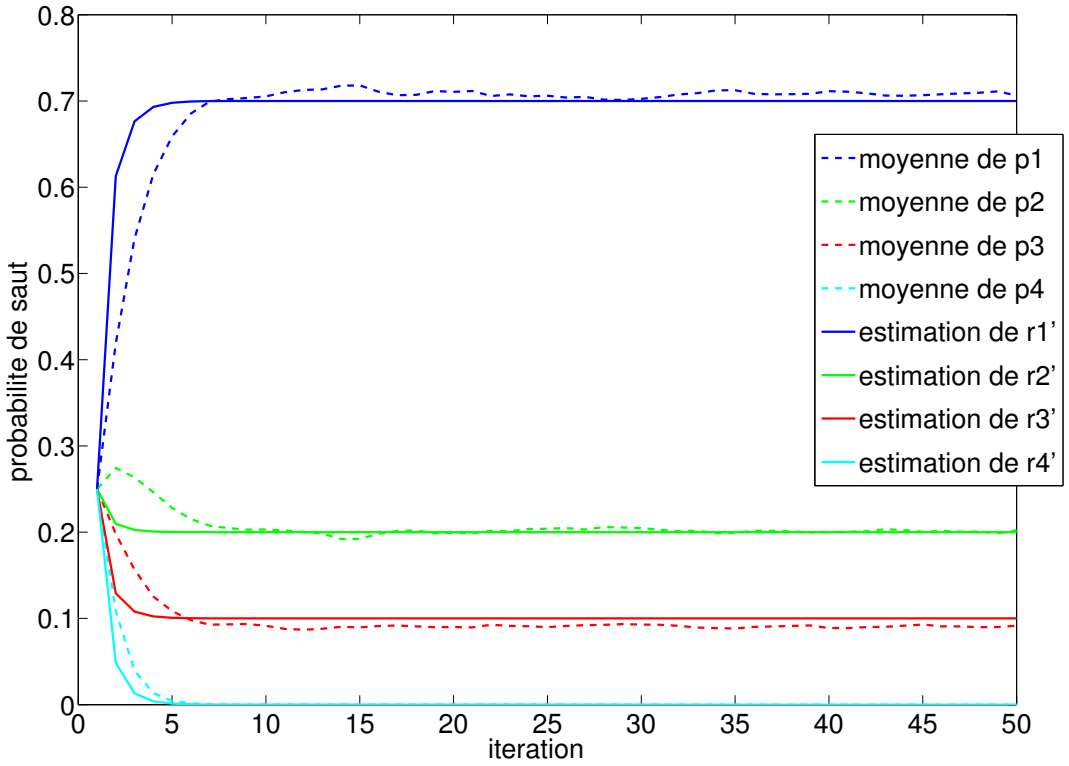


Fig. 4.3. Comparaison de l'estimation de la pertinence normalisée avec les probabilités de saut moyennes obtenues sur 1000 tests. On a choisit $E[u_1] = 0.7$, $E[u_2] = 0.2$, $E[u_3] = 0.1$ et $E[u_4] = 0$.

Temps local

Dans l'implémentation que nous présentons, l'échelle de temps sur laquelle se fixe les variations de chaque marqueur est locale. Les marqueurs n'évoluent que lors du passage d'un agent, ce qui veut dire que les chemins sur lesquels passent beaucoup d'agents ont un temps qui défile plus vite que les chemins sur lesquels peu d'agents circulent. Or le passage des agents dépend de la valeur des marqueurs. Ainsi, les chemins menant à des bases non pertinentes ont une dynamique d'évolution des marqueurs plus lente (de part la faible valeur de leur probabilité) que les chemins menant à des bases pertinentes.

4.5 Sélection des images

Pour chaque site, un lot d'images est sélectionné afin d'être ajouté à l'ensemble d'entraînement. Pour d'améliorer la mesure de pertinence $f_{\mathcal{A}}$ en un minimum d'interactions, cette sélection doit être optimisée.

4.5.1 Apprentissage actif

L'idée est de sélectionner les images à ajouter à l'ensemble d'apprentissage dans le but de réduire le nombre d'interactions nécessaires à l'obtention du classifieur optimal (où bien d'obtenir un meilleur classifieur avec un nombre d'annotation fixé). Il existe principalement deux approches dans la littérature (*cf* partie 1.1.4.5 ; la première se basant sur la minimisation de l'erreur du classifieur [Roy and McCallum, 2001]. Elle consiste à sélectionner l'exemple qui, ajouté à l'ensemble d'apprentissage, minimise l'erreur de classification sur l'ensemble des images non annotées. C'est à dire l'exemple qui augmente le plus le maximum de la probabilité d'appartenir à une classe, pour toutes les images non annotées.

Dans la deuxième approche, les images sont sélectionnées selon une stratégie basée sur l'incertitude. La plus populaire est celle développée par Tong [Tong and Chang, 2001]. Considérant l'ensemble d'apprentissage $\mathcal{A}$, il existe un ensemble d'hyperplans séparant les éléments de $\mathcal{A}$ relativement à leurs annotations, occupant un volume dans l'espace des signatures des images. La stratégie consiste alors à ajouter à l'ensemble d'apprentissage l'image qui divise par deux la taille de ce volume, c'est à dire l'images dont l'hyperplan associé se trouve au centre du volume. Pour les *SVM* cela revient à choisir l'image la plus proche de la marge du classifieur et [Tong and Chang, 2001]).

Bien sûr, la frontière est relative aux éléments présents dans la base, et un classifieur optimisé de la sorte pour une base n'est pas aussi bon sur des bases contenant des images différentes.

Correction de frontière

Du fait que le nombre d'images pertinentes est très petit devant le nombre d'images non pertinentes, ces stratégies ont tendance à sélectionner un grand nombre d'exemples non pertinents. On peut alors appliquer un mécanisme de correction de la frontière [Gosselin and Cord, 2006b] afin d'équilibrer l'ensemble d'apprentissage. Ainsi, si trop d'annotations négatives ont été données, la sélection active se déplacera de la marge vers le cœur de la classe. Considérant un tri des images de la base par pertinence décroissante, la sortie du classifieur est modifiée comme suit :

$$f(\mathbf{x}) = f(\mathbf{x}) - f(\mathbf{x}_{k(n)}) \tag{4.44}$$

avec $\mathbf{x}_{k(n)}$ l'image de position $k(n)$ dans le tri, et $k(n)$ est calculé à chaque itération n de la manière suivante :

$$k(n+1) = k(n) + 2(pos(n) - neg(n)) \tag{4.45}$$

Avec une telle correction, la frontière est déplacée sur l'image $\mathbf{x}_k$, *pos* et *neg* étant respectivement le nombre d'images annotées positivement et négativement.

Diversité

Dès lors que l'étape de sélection ne consiste pas en la sélection d'une seule image, mais d'un lot d'images, il faut trouver un moyen d'étendre la stratégie. Produire un classement des meilleurs exemples du point de vue de la stratégie et choisir les I premiers est une approche simple à mettre en œuvre, mais qui à le désavantage de sélectionner des exemples ayant des effets similaires sur le classifieur (car très proches du point de vue des signatures). L'idée est alors de diversifier le lot d'images à annoter en y ajoutant l'exemple qui offre le meilleur compromis entre la stratégie active et une grande distance par rapport aux autres exemples du lot [Brinker, 2003].

4.5.2 Contraintes et limitations

Ces stratégies actives ont le risque de fausser l'estimation de la pertinence des sites basée sur l'estimateur empirique $\hat{r}_i$. Cependant, elles restent aveugle, c'est-à-dire qu'elles ont le même comportement sur chaque site (elles ne vont pas sélectionner plutôt au cœur de la classe sur un site et à l'inverse sur d'autres). On peut ainsi espérer que, quand bien même la pertinence d'un site soit mal évaluée à cause d'un processus de sélection assez complexe, la pertinence normalisée n'en soit finalement que peu influencée car tous les sites ont subit le même processus.

D'autre part, notre système utilise des agents mobiles afin de propager la requête de site en site, et les déplacements des agents sont probabilistes. Il est alors possible que plusieurs agents arrivent sur le même site. Cela est d'autant plus vraisemblable que

notre système est prévu pour fonctionner avec un grand nombre d'agents mobiles. Ainsi plusieurs agents sélectionnant des images sur un même site obtiennent avec ces stratégies les mêmes résultats, et n'enrichissent pas l'ensemble d'apprentissage $\mathcal{A}$.

4.5.3 Stratégies proposées

Nous proposons une extension de la stratégie active basée sur l'incertitude incorporant la correction de frontière afin de respecter les contraintes de notre système multi-agents. Nous ajoutons une exploration stochastique autour de la marge de manière à, d'une part, éviter que les agents arrivant sur le même site ne ramènent les mêmes images, et d'autre part, pour inclure de la diversité à l'ensemble d'apprentissage (sans avoir à calculer un autre critère que l'incertitude). Nous proposons deux stratégies nouvelles, se différenciant sur la manière dont est effectuée l'exploration stochastique autour de la marge.

probabilité gaussienne

Les images de la base sont triées par ordre de proximité à la marge. Puis, une probabilité $g_{\mathbf{x}}$ d'être tirée est attribuée à chaque image $\mathbf{x}$ selon une loi gaussienne calculée sur $r(\mathbf{x})$ le rang de l'image $\mathbf{x}$ dans le tri :

$$g_{\mathbf{x}} = e^{\frac{-r(\mathbf{x})^2}{\sigma^2}} \tag{4.46}$$

σ est déterminé de telle sorte que l'énergie se concentre à 95% dans les I_i premières images :

$$\sum_{1 \leq i \leq I_i} g_{\mathbf{x}_i} = 0.95 \tag{4.47}$$

On effectue alors un tirage selon cette loi et l'image ainsi sélectionnée est retirée de la liste. Les probabilités sont recalculées sur le nouveau tri de la liste ainsi privée de l'image sélectionnée. On effectue I tirages de la sorte.

L'avantage de cette stratégie est d'être très concentrée sur la marge tout en permettant un exploration de la totalité de la base. L'inconvénient majeur est qu'elle est très coûteuse en calcul, car à chaque tirage, il faut réaffecter les probabilités à toutes les images de la base.

probabilité uniforme

Le second algorithme développé est destiné à simplifier le premier. Les images sont aussi triées par ordre de proximité à la marge. La sélection de I images est répartie en I sélections de 1 image. Chacune de ces sélections est faite sur un ensemble contenant n images avec une probabilité uniforme sur les images. Pour le premier tirage, l'ensemble

contient les images avec un rang compris entre 1 et n. L'image sélectionnée à l'itération i est retirée de l'ensemble et l'image avec le rang $n+i-1$ est ajoutée à l'ensemble afin de garder une taille de n.

Notons $p = \frac{1}{n}$ la probabilité d'une image dans l'ensemble à une itération donnée d'être sélectionnée et $q = 1 - p$ la probabilité de l'évènement inverse. Soit $P_i(\mathbf{x})$ la probabilité de l'image $\mathbf{x}$ avec le rang $r(\mathbf{x})$ d'être sélectionnée à l'itération i. P_i suit une sorte de loi géométrique :

$$P_i(\mathbf{x}) = \begin{cases} p\prod_{j=1}^{i-1} q & , \quad 1 \leq r(\mathbf{x}) \leq n \\ p\prod_{j=1}^{i-r_{\mathbf{x}}+n-1} q & , \quad n+1 < r(\mathbf{x}) < n+i \\ 0 & , \quad n+i \leq r(\mathbf{x}) \end{cases} \tag{4.48}$$

$$P_i(\mathbf{x}) = \begin{cases} pq^{(i-1)} & , \quad 1 \leq r(\mathbf{x}) \leq n \\ pq^{(i-r_{\mathbf{x}}+n-1)} & , \quad n+1 < r(\mathbf{x}) < n+i \\ 0 & , \quad n+i \leq r(\mathbf{x}) \end{cases} \tag{4.49}$$

La probabilité d'une image d'être sélectionnée après I itérations est la somme des probabilités d'être sélectionnée à chaque itération à partir du moment où elle est entrée dans l'ensemble de sélection :

$$P(\mathbf{x}) = \begin{cases} \sum_{i=1}^{I} pq^{(i-1)} & , \quad 1 \leq r_{\mathbf{x}} \leq n \\ \sum_{i=r_{\mathbf{x}}-n+1}^{I} pq^{(i-r_{\mathbf{x}}+n-1)} & , \quad n < r_{\mathbf{x}} < n+I \\ 0 & , \quad n+I \leq r_{\mathbf{x}} \end{cases} \tag{4.50}$$

$$P(\mathbf{x}) = \begin{cases} 1-q^{I} & , 1 \leq r_{\mathbf{x}} \leq n \\ 1-q^{(I-r_{\mathbf{x}}+n)} & , n < r_{\mathbf{x}} < n+I \\ 0 & , n+I \leq r_{\mathbf{x}} \end{cases} \tag{4.51}$$

Cette probabilité est uniforme pour les images avec un rang entre 1 et n, puis décroît exponentiellement de paramètre q pour les images entre n et $n+I$. Elle est nulle pour les images plus éloignées de la marge.

L'avantage de cette stratégie est qu'elle restreint l'exploration à un voisinage réduit autour de la marge et qu'elle est très rapide à calculer.

4.5.4 Fin de la session de recherche

Lorsque l'utilisateur a annoté suffisamment d'images, un agent est lancé vers chaque source de données avec pour objectif de ramener les meilleures images au sens de la mesure

de similarité entraînée par bouclage de pertinence. Sur chacun des sites i, l'agent récupère un nombre d'images N_i proportionnel à la probabilité de déplacement vers ce site, et donc à la part de pertinence du site dans le concept recherché. Si N est le nombre total d'images qu'on veut récupérer, alors :

$$N_i = p_i \cdot N \tag{4.52}$$

Les images ainsi récupérées depuis chaque site sont de nouveau triées par la fonction de pertinence avant d'être affichées à l'utilisateur. Ainsi, les images ayant le plus de chance d'appartenir au concept sont récupérées depuis les sites les plus pertinents, et les sites contenant moins d'images du concept (au sens de la pertinence $\hat{r}_i$ apprise) ramènent moins d'images que ceux en contenant beaucoup. Puisque la stratégie qui nous a conduit à construire le classifieur a été plus exploratoire de l'espace de représentation des images sur les collections ayant une forte pertinence normalisée, il semble normal de récupérer plus d'images en provenance de ces sites.

4.6 Conclusion

Dans ce chapitre nous avons explicité notre stratégie active distribuée. Nous avons introduit la notion de pertinence d'un site afin de formaliser le principe de notre stratégie. Nous avons détaillé les deux étapes de la stratégie que nous avons développée, la sélection des sites se basant sur un estimation de la pertinence de ceux-ci et la sélection des images se basant sur la mesure de similarité. Nous avons montré comme celles-ci s'incluaient dans le bouclage de pertinence, puis nous avons détaillé l'implémentation à l'aide du système multi-agents d'inspiration éthologique à l'origine de cette stratégie.

Du point de vue théorique, un théorème a été établi pour nous aider à discuter le comportement asymptotique de notre système. Ceci nous a permis de montrer que les marqueurs utilisés pour sélectionner les sites desquels ramener les images sont une bonne estimation de la pertinence de ces sites.

Enfin, nous avons introduit deux nouvelles stratégies actives locales permettant la sélection des images sur les sites. Ces stratégies sont bien adaptées au caractère stochastique de notre système et permettent d'ajouter de la diversité à l'ensemble d'apprentissage.

Chapitre 5

Expériences sur l'actif distribué

Dans ce chapitre, nous présentons les expériences menées sur la stratégie proposée au chapitre précédent. Nous détaillons dans un premier temps les différents protocoles expérimentaux utilisés. Nous observons alors la qualité des résultats retrouvés en fonction de la localisation des images recherchées sur un ou plusieurs sites. Puis nous mesurons la qualité de l'apprentissage des chemins à l'aide des marqueurs. Enfin, nous comparons notre stratégie à une approche naïve dans le cas où les images pertinentes sont effectivement très localisées. Nous concluons ce chapitre par une expérience préliminaire à l'utilisation des marqueurs sur plusieurs sessions de recherche, en guise d'introduction à la prochaine partie.

5.1 Exemples de résultats

Nous montrons ici quelques captures d'écran montrant les résultats obtenus par le système après une interaction de 50 annotations. Les images sont tirées de la base *Trec-Vid'05* (*cf* 5.2.1). Elles ont été réparties de manière à ce que chaque base soit spécialisée dans un petit nombre de catégories. Visuellement, les résultats semblent satisfaisants (voir figures 5-2, 5-3, 5-4 et 5-5). Une note en particulier une très bonne capacité à retrouver des images dans des contextes très différents (par exemple, la recherche sur les matchs de tennis donne aussi bien des plans de joueurs que des plans du terrain vu de dessus).

Le réseau consiste en quatre ordinateurs munis de bases d'images, trois ordinateurs intermédiaires (nus) et l'ordinateur de l'utilisateur, selon la topologie présentée en 5-1. L'apprentissage des chemins pertinents est correct sur ce réseau (c'est à dire que les probabilités de saut sont plus élevées pour la machine contenant la catégorie recherchée que pour les autres). Cet exemple illustre la capacité de notre système à fonctionner sur ce type de réseau *ad hoc* d'une dizaine de machines à la structure non triviale (il existe plusieurs chemins pour arriver à la même base). Les machines utilisées sont celles de l'intranet du laboratoire.

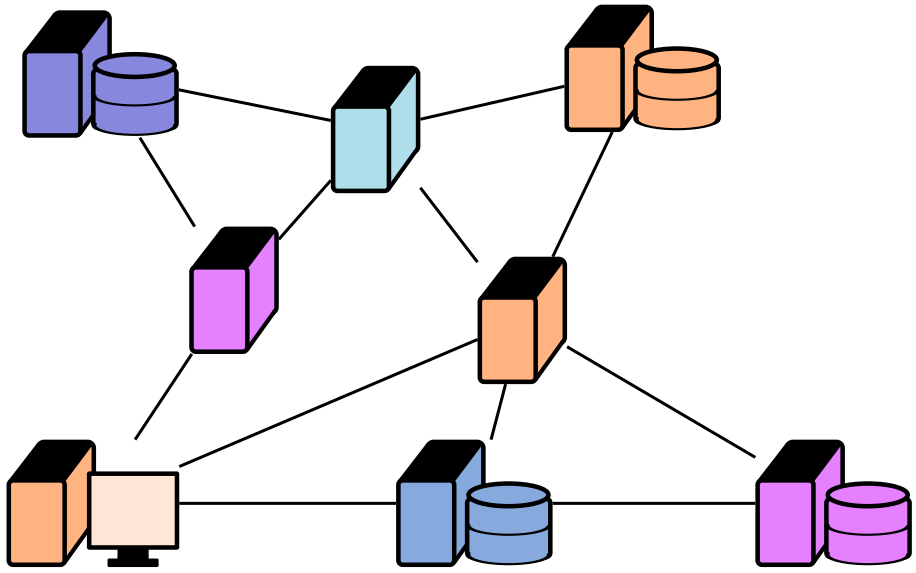


Fig. 5-1. Topologie du réseau utilisé pour l'exemple. Il comporte huit machines en tout : l'ordinateur de l'utilisateur, trois machines nues, et quatre bases d'images.

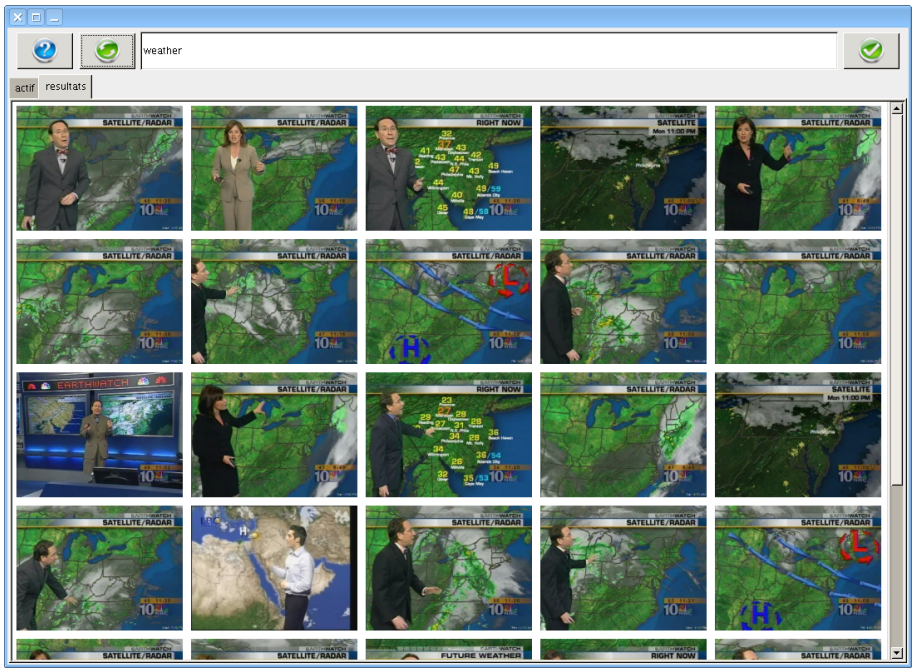


Fig. 5-2. Résultat d'une recherche d'images de présentation de la météo sur le réseau de la figure 5-1 après 50 annotations.

5.2 Plan expérimental

L'objectif principal de ces tests est double : il s'agit tout d'abord de vérifier que notre système à marqueurs est bien capable d'apprendre la localisation de la catégorie recherchée sur le réseau, et ensuite de mesurer l'impact de l'apprentissage de cette locali-

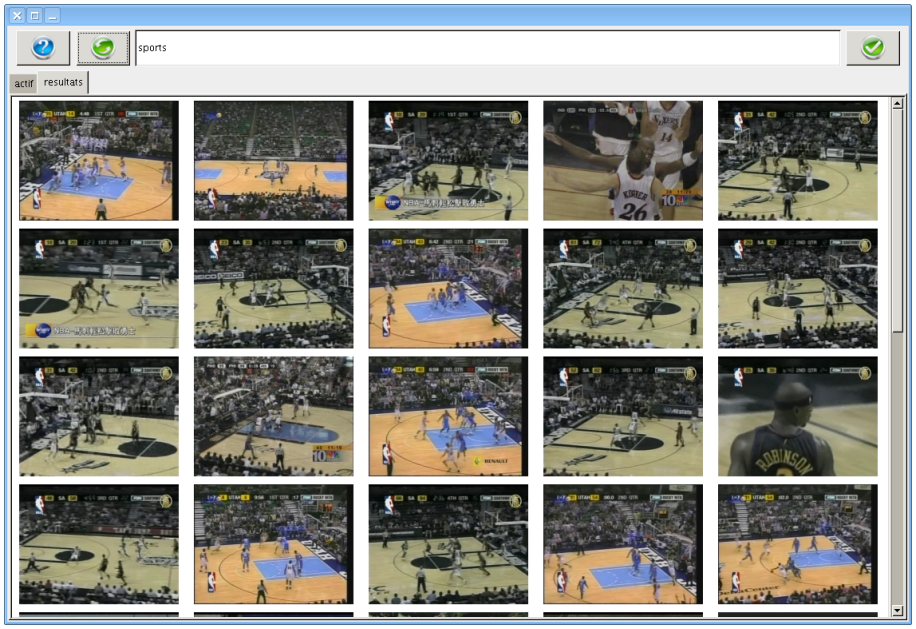


Fig. 5-3. Résultat d'une recherche d'images de matchs de basket sur le réseau de la figure 5-1 après 50 annotations.

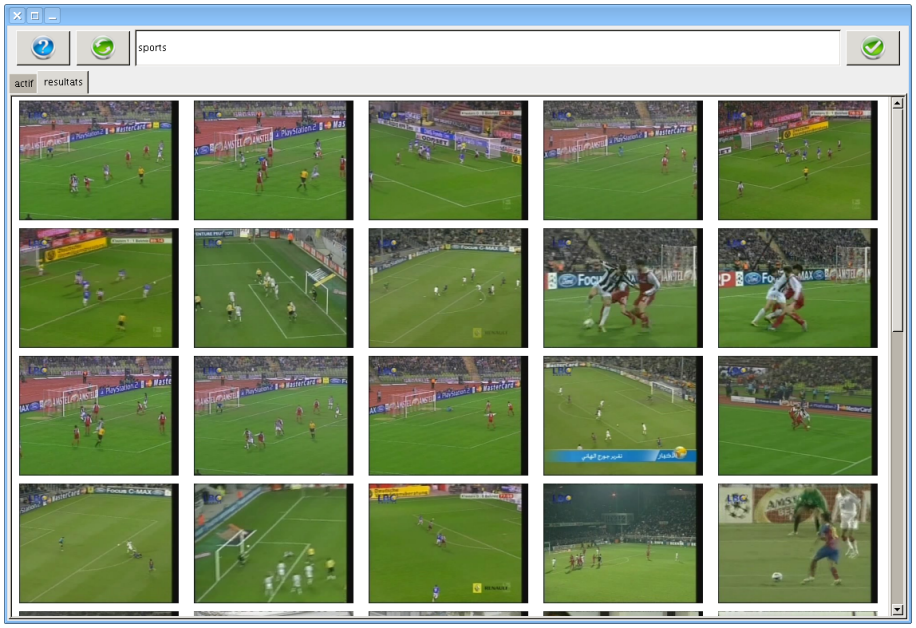


Fig. 5-4. Résultat d'une recherche d'images de matchs de football sur le réseau de la figure 5-1 après 50 annotations.

sation (profondément lié à la dynamique du bouclage de pertinence) sur les performances du système. Nous détaillons d'abord les bases qui ont servies à effectuer nos expériences, puis les critères d'évaluation. Enfin, nous détaillons les paramètres du système et les valeurs que nous avons utilisées pour l'obtention de nos résultats.

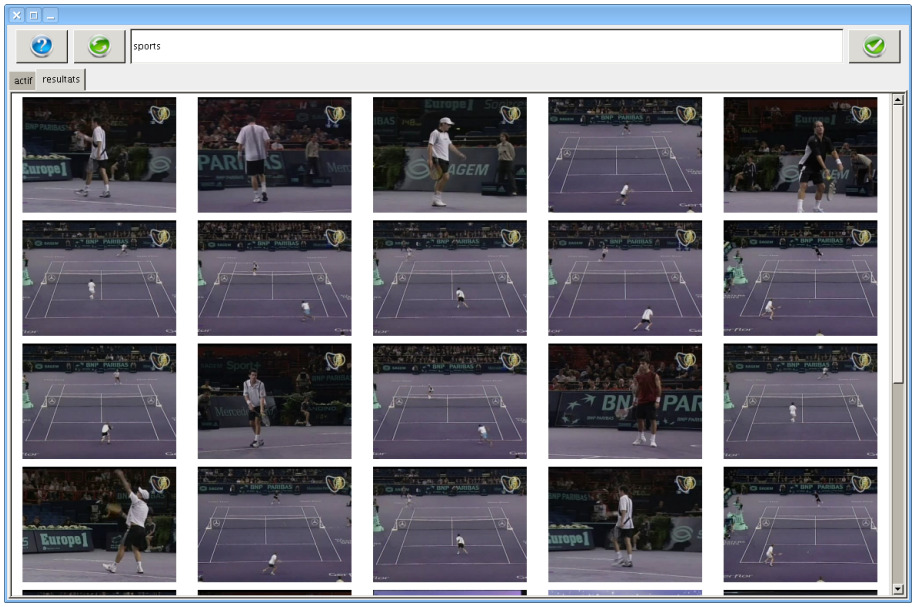


Fig. 5-5. Résultat d'une recherche d'images matchs de tennis sur le réseau de la figure 5-1 après 50 annotations.

5.2.1 Bases

Nous utilisons deux jeux de test pour nos expérimentations. Les bases d'images utilisées sont des standards de l'évaluation en matière de recherche d'image par le contenu.

5.2.1.1 Base *Corel*

Dans ce jeu de test, les images considérées sont un sous-ensemble de la base *Corel* contenant 5909 images de photographies professionnelles. La taille des catégories varie de 50 à 500 images environ, avec une moyenne de 100. Ces images contiennent des catégories variées comportant des paysages (*Mountains*, *Sunsets*), des concepts abstraits (*Europe*, *Finland*), ou des objets (*Doors*, *Dogs*). La liste des catégories que nous avons testé se trouve dans le tableau 5-6.

catégorie	cardinal
african	230
antiquity	83
asia	103
dogs	99
doors	199
finland	50
objects	100
people	29

Fig. 5-6. Nombre d'images par catégorie pour la base *Corel*.

La difficulté à retrouver une catégorie est très variable et peut aller de très facile (par exemple la catégorie *doors* qui sont des portes en gros plans) à très compliquée (par exemple la catégorie *people* dont la variabilité à l'intérieur de la catégorie est très grande). Les signatures utilisées sont constituées de distributions de 25 couleurs et 25 textures. On notera que les signatures utilisées, qui sont basées sur des attributs globaux, sont mal adaptées à la recherche d'objet de petite taille en rapport à la taille de l'image, et peuvent ainsi expliquer les médiocres résultats sur certaines catégories.

5.2.1.2 Base *TrecVid'05*

Dans ce jeu de test, les images considérées sont les *keyframes* extraites des vidéos de la base d'entraînement de la compétition TrecVid 2005. Les keyframes sont des images extraites d'une séquence vidéo afin de la résumer. Il y a 69128 images réparties en 21 catégories de tailles très variables (de quelques dizaines d'images à plusieurs dizaines de milliers). Le tableau 5-7 résume le cardinal des catégories que nous avons testées.

catégorie	cardinal
bus	190
charts	107
corporate-leaders	748
maps	418
mountain	467
road	2359
sports	2725
truck	107
urban	3637
weather	649

Fig. 5-7. Cardinal des catégories testées pour la base *TrecVid'05*.

Les signatures sont constituées de distributions de 32 couleurs et de 32 textures, telles que décrites au chapitre 3.

5.2.2 Critères d'évaluation

Pour évaluer l'apprentissage des chemins pertinents, nous mesurons directement la probabilité de saut p_i à la fin de la session pour chacune des destinations. Ces probabilités étant fixées à $\frac{1}{N}$, où N est le nombre de destinations (*cf* chapitre 4), au début de la session. On considère que l'apprentissage est bon lorsque la probabilité des machines contenant les collections pertinentes (respectivement non-pertinentes) est supérieure (respectivement inférieure) à cette valeur initiale.

L'évaluation de la performances du système se fait avec les outils classiques de la recherche d'images par le contenu. On mesure le *rappel*, c'est à dire le nombre d'images

de la catégorie ramenées sur le cardinal de la catégorie, et la *précision*, c'est à dire le nombre d'images de la catégorie ramenées sur le nombre d'images ramenées. C'est deux grandeurs sont comprises entre zéro et un. On peut ainsi tracer la courbe *précision* en fonction du *rappel*, l'idéal étant d'avoir une précision de un pour n'importe quelle valeur de rappel. Pour obtenir un scalaire permettant de comparer numériquement plusieurs méthodes, on peut calculer le *MAP* (*Mean Average Precision*) qui est l'intégrale de la courbe de précision-rappel. Pour évaluer nos méthodes, nous avons utilisé les valeur du *MAP* et du rappel calculé lorsque 500 images ont été récupérées.

5.2.3 Paramétrage

Chaque jeu se compose d'une base d'images associée à une topologie de réseau particulière. Les topologies utilisées sont volontairement simples afin de mettre en évidence l'influence mutuelle de l'apprentissage de chemins et de l'apprentissage de la mesure de pertinence.

5.2.3.1 Test *Corel*

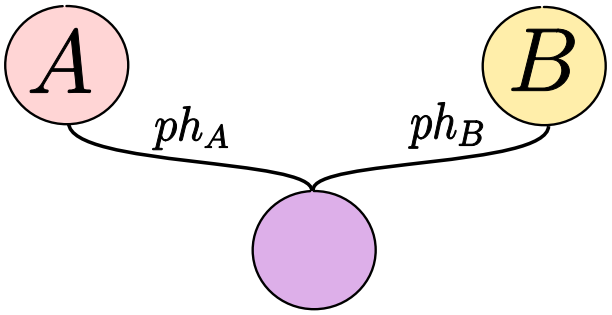


Fig. 5.8. Topologie du test Corel : deux destinations A et B sont disponibles et contiennent chacune une collection d'images. La localisation de la catégorie recherchée sur ces deux destinations varie d'également répartie à entièrement localisée sur A.

Nous créons un réseau simple avec deux bases comme destinations directes pour les agents (voir Fig. 5.8) que nous appelons A et B. Ce réseau à été simulé et ne correspond pas à des machines physiques distinctes. La catégorie recherchée est systématiquement répartie selon une certaine distribution sur les deux bases à chaque test. Le reste des images (non pertinentes, donc) est réparti aléatoirement entre A et B.

Pour chaque catégorie testée, les paramètres du système sont les suivants :

- p agents mobiles utilisés.
- q images ramenées par chaque agent grâce à la stratégie active locale *gaussienne* (*cf* 4.5.3).

Ces paramètres ont été fixés de manière empirique suite à différents tests à $p = 8$ et $q = 2$. Les tests ont porté pour p allant de 1 à 16 et q allant de 1 à 10, et nous ont montré

qu'il valait mieux beaucoup d'agents ramenant peu d'images afin d'avoir une évolution rapide des marqueurs, mais que trop d'agents menait à trop peu d'itération du bouclage de pertinence. $p = 8$ réalise un bon compromis dans ce sens. Les paramètres du test, sont quant à eux les suivant :

- Chaque image de la catégorie a été utilisée comme requête initiale donnant lieu à une session.
- Pour chaque session, les agents ont ramené des images de A et B jusqu'à ce que 100 annotations aient été obtenues.

5.2.3.2 Test *TrecVid'05*

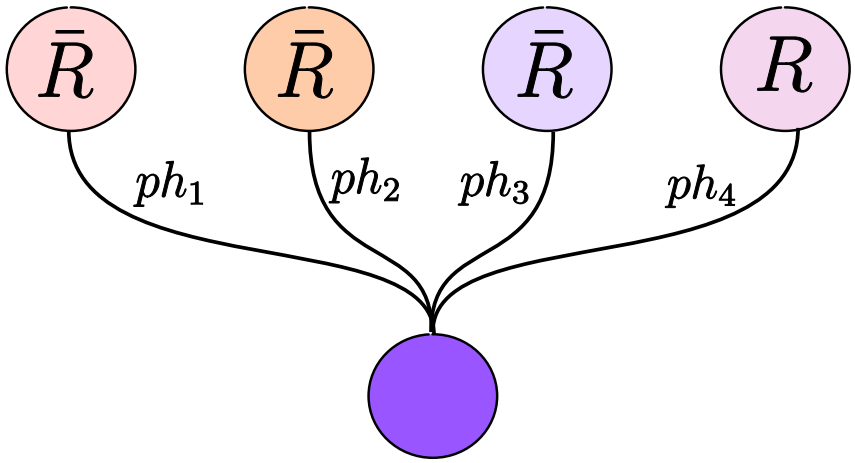


Fig. 5-9. Topologie du test *TrecVid'05*, pour lequel quatre destinations sont disponibles. La catégorie recherchée est toujours contenue par la quatrième destination.

Quatre machines sont utilisées comme destinations possibles des agents (voir figure 5-9). La catégories recherchée est systématiquement répartie sur la dernière machine. Ce réseau est constitué de machines appartenant à l'intranet du laboratoire.

Pour chaque catégorie testée, les paramètres du système sont les suivants :

- p agents mobiles utilisés.
- q images ramenées par chaque agent grâce à la stratégie active locale *uniforme* (*cf* 4.5.3).

De même que pour le test *Corel*, ces paramètres ont été fixés de manière empirique aux mêmes valeurs $p = 8$ et $q = 2$. Cela montre la force du système, puisque les mêmes paramètres peuvent être utilisés dans deux contextes complètement différents. Les paramètres du test, sont quant à eux les suivants :

- Pour chaque session, cinq images de la catégorie recherchée et cinq images non pertinentes ont été utilisées comme requête initiale.
- Pour chaque session, les agents ont ramené des images des quatre destinations jusqu'à ce que 100 annotations aient été obtenues.

– 100 sessions ont été faites pour chaque catégorie.
– le *rappel* a été calculé proportionnellement à la répartition des marqueurs sur les quatre machines.

5.3 Performances du système

Dans ce test, nous voulons observer l'influence de la concentration en un nombre restreint de sites de la catégorie recherchée sur l'apprentissage de la mesure de pertinence. Nous avons utilisé les jeux de test *Corel* et *TrecVid'05*.

5.3.1 Précision moyenne - test *Corel*

Pour chacune des catégories recherchées, nous avons fait varier la répartition des images y appartenant de 50% sur la base A et 50% sur la base B à 100% sur A et rien sur B, par paliers de 10%.

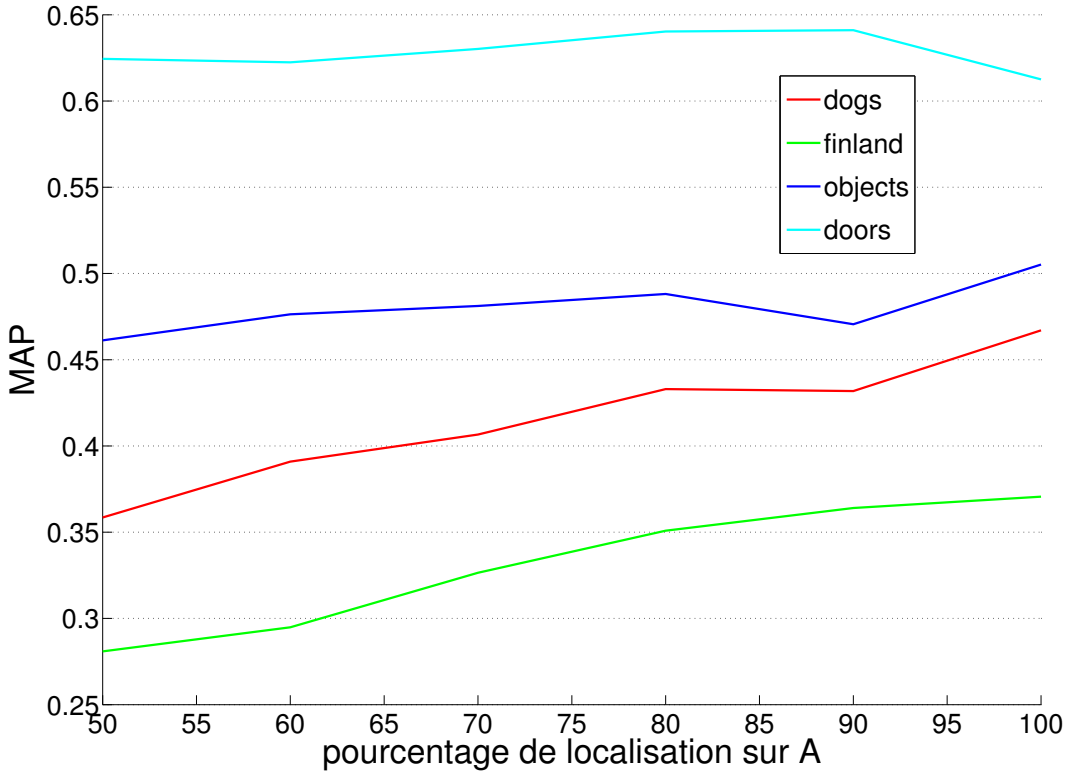


Fig. 5-10. ***MAP*** **pour les catégories** ***finland*****,** ***dogs*****,** ***doors*** **et** ***objects*** **de la base** ***Corel*****. Les catégories les plus difficiles profitent d'un gain de l'ordre de dix pour cent si elle sont bien localisées.**

La figure 5-10 montre les variations du *MAP* en fonction de la concentration de la catégorie recherchée sur la base A pour les catégories *finland*, *dogs*, *doors* et *objects*. Comme on peut le constater, le fait que la catégorie soit plus concentrée sur un seul site améliore la qualité des résultats retrouvés d'environ 10% pour les catégories de difficulté moyenne, par rapport à une dilution totale de la catégorie entre A et B. Pour les catégories très difficiles, présentées sur la figure 5-11 (catégories *african*, *antiquity*, *asia* et *people*), les résultats sont plus mitigés (entre 2% et 5%). La catégorie *doors* ne tire aucun gain de la stratégie active distribuée, ce qui peut s'expliquer par sa facilité, une optimisation de l'ensemble d'apprentissage n'étant alors pas nécessaire pour obtenir de bons résultats.

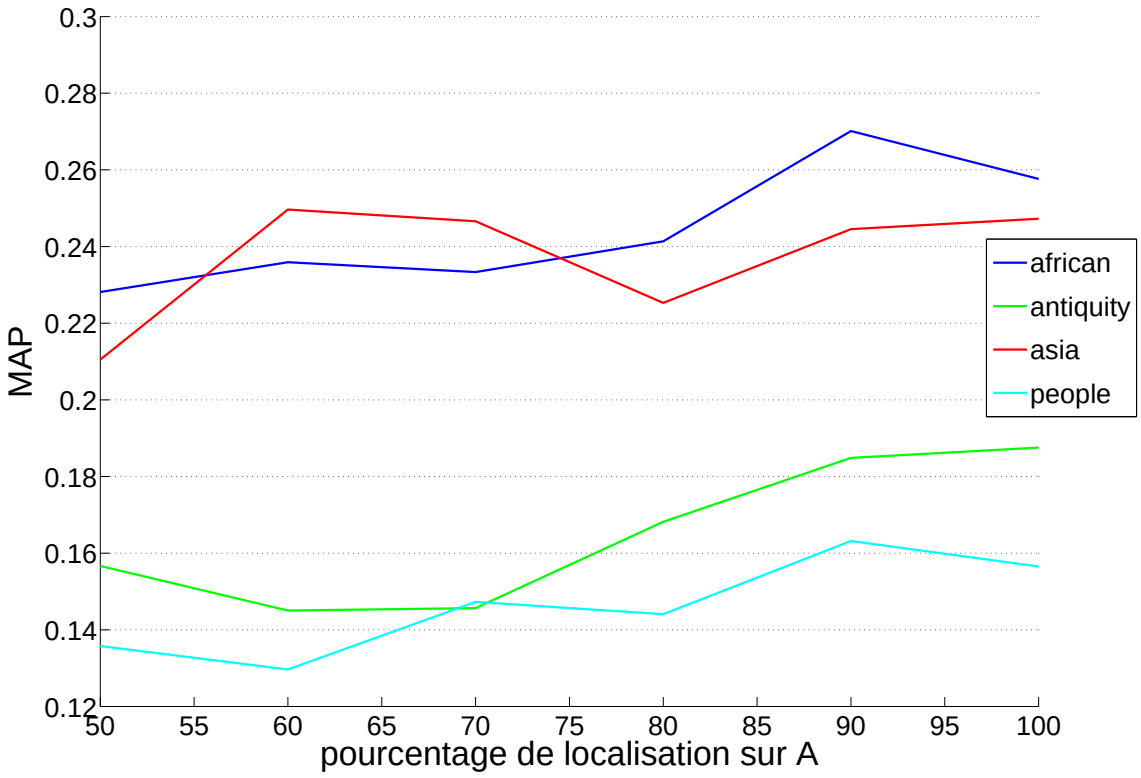


Fig. 5-11. *MAP* pour les catégories *african*, *antiquity*, *asia* et *people* de la base *Corel*. Pour ces catégories très difficiles, le gain de la localisation est de l'ordre de quelques pour cent.

5.3.2 Rappel - test *TrecVid'05*

Dans ce test, nous voulons conforter les résultats du premier test en comparant notre architecture sur une autre base à une méthode centralisée. Le jeu *TrecVid'05* a été utilisé et un sous-ensemble des catégories disponibles ont été testées. Nous avons testé deux cas de distribution de la catégorie recherchée : un premier cas où celle-ci est intégralement contenue sur la quatrième machine ; ainsi qu'un second cas où celle-ci est contenue à 80%

sur la quatrième machine, les 20% de surplus étant équitablement répartis sur les trois machines restantes. Ce premier cas est nommé *forte localisation* par la suite, alors que le second est appelé *faible localisation*.

La méthode centralisée consiste à réunir sur une seule base l'intégralité des images pour toutes les sessions. Cette méthode ne fait donc aucun usage de l'information de localisation qu'avaient les images avant d'être centralisées.

La figure 5.12 montre les valeurs de *rappel* pour les catégories testées lorsque 500 images sont récupérées. Les résultats de l'approche centralisée sont comparés à ceux des tests à *faible localisation* et à *forte localisation*. Confirmant les résultats obtenus sur le jeu de test *Corel*, l'architecture distribuée est en moyenne deux fois meilleure que l'approche centralisée. Le cas à *forte localisation* est quant à lui meilleur que le cas à *faible localisation* d'environ 5% à 10% en moyenne.

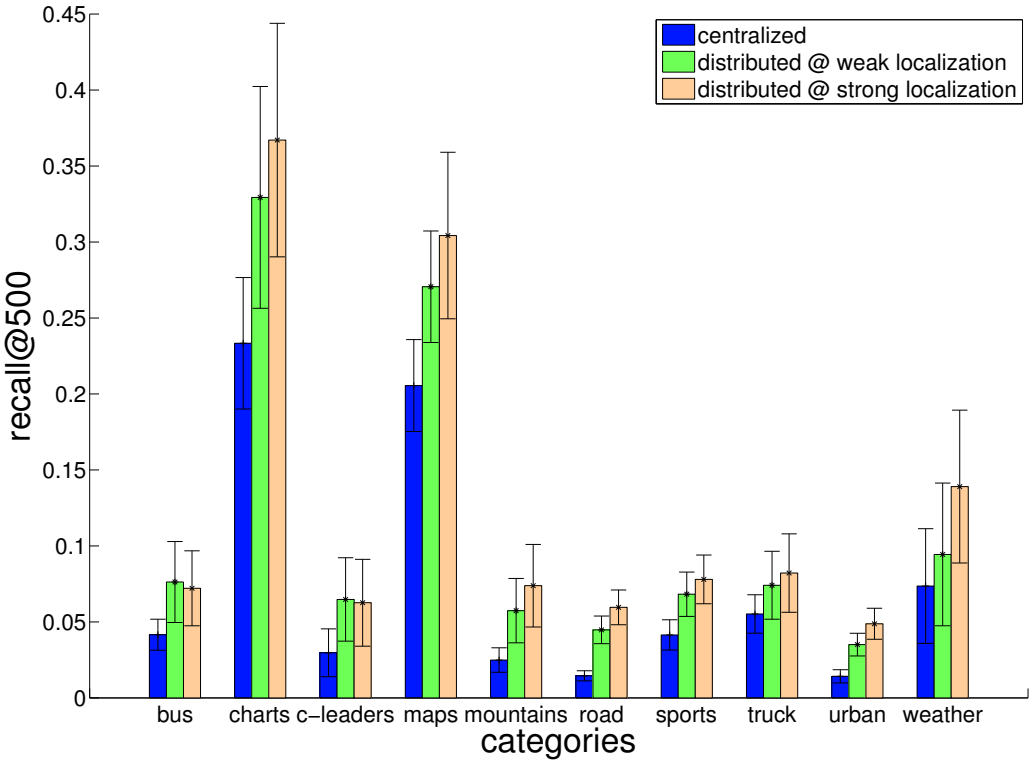


Fig. 5.12. Valeurs de *rappel* pour les catégories testées de la base *TrecVid'05* comparant l'approche centralisée de référence et les cas à *faible localisation* et à *forte localisation*. Dans ces deux derniers cas, le gain de la stratégie active distribuée est très significatif (jusqu'à doubler la valeur de *rappel* pour certaines catégories).

Même si la complexité des catégories varie beaucoup, les gains obtenus par le système distribué par rapport à la méthode centralisée sont relativement stables.

5.4 Apprentissage des chemins

Dans cette série de tests, nous voulons voir à quel point la stratégie à marqueurs est capable d'apprendre les chemins pertinents.

5.4.1 Apprentissage des marqueurs - Test *TrecVid'05*

La figure 5-13 montre les probabilités de saut vers chacune des quatre destinations pour toutes les catégories testées dans le cas à *forte localisation*. Comme on peut le voir, la quatrième machine obtient toujours une probabilité de saut supérieure à 25%, ce qui indique que la destination pertinente a toujours été correctement apprise (allant de 50% à 85% environ).

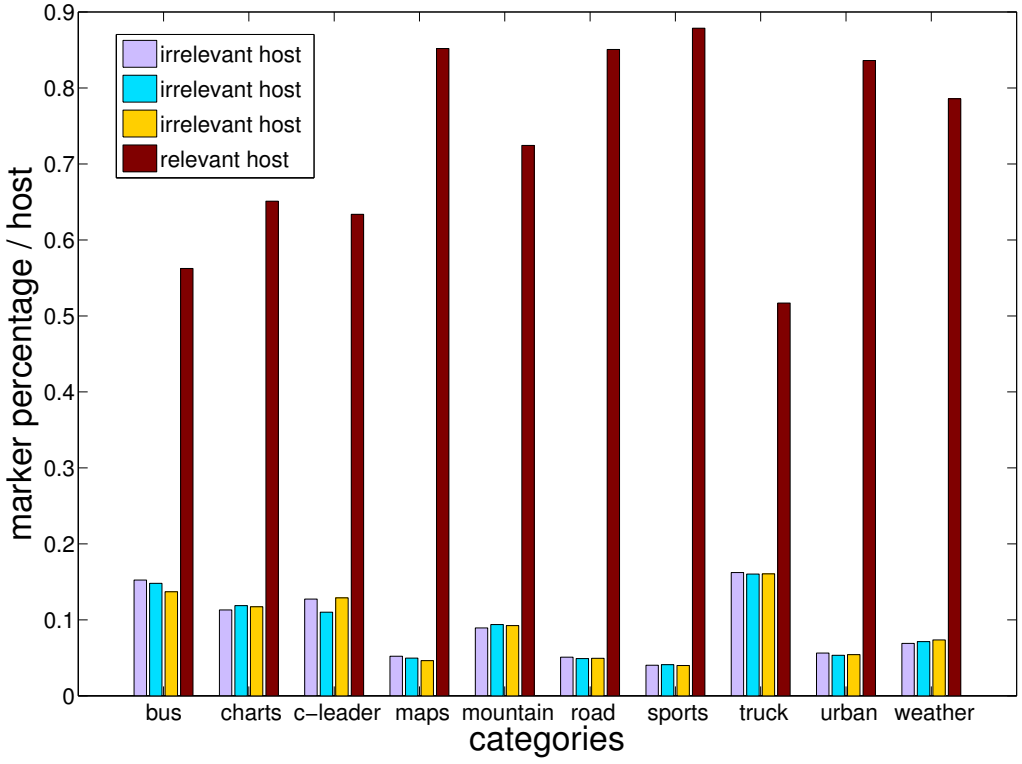


Fig. 5-13. Valeur des probabilités de saut vers chacune des quatre destinations pour le cas à *forte localisation*. La probabilité de se diriger vers la quatrième destination (destination pertinente) est largement plus élevée que pour les autres dans toutes les catégories, ce qui montre que la localisation a été correctement apprise.

Les probabilités de saut vers chacune des destinations pour toutes les catégories dans le cas à *faible localisation* sont montrées sur la figure 5-14. Là encore, le chemin menant à la machine contenant les images pertinentes a été correctement appris, même si cela est moins flagrant que dans le cas à *forte localisation* (de 0.45 à 0.62 contre de 0.51 à 0.88).

On remarque que dans les deux cas, la difficulté de la catégorie n'a pas d'influence sur la qualité de l'apprentissage des chemins. Ainsi, la catégorie *charts*, qui est facile, obtient un apprentissage des chemins moins bon que *sports* qui est difficile. À l'inverse, *maps*, qui est facile, obtient un bon apprentissage de chemins, alors que *truck*, catégorie difficile, obtient le moins bon apprentissage de chemins.

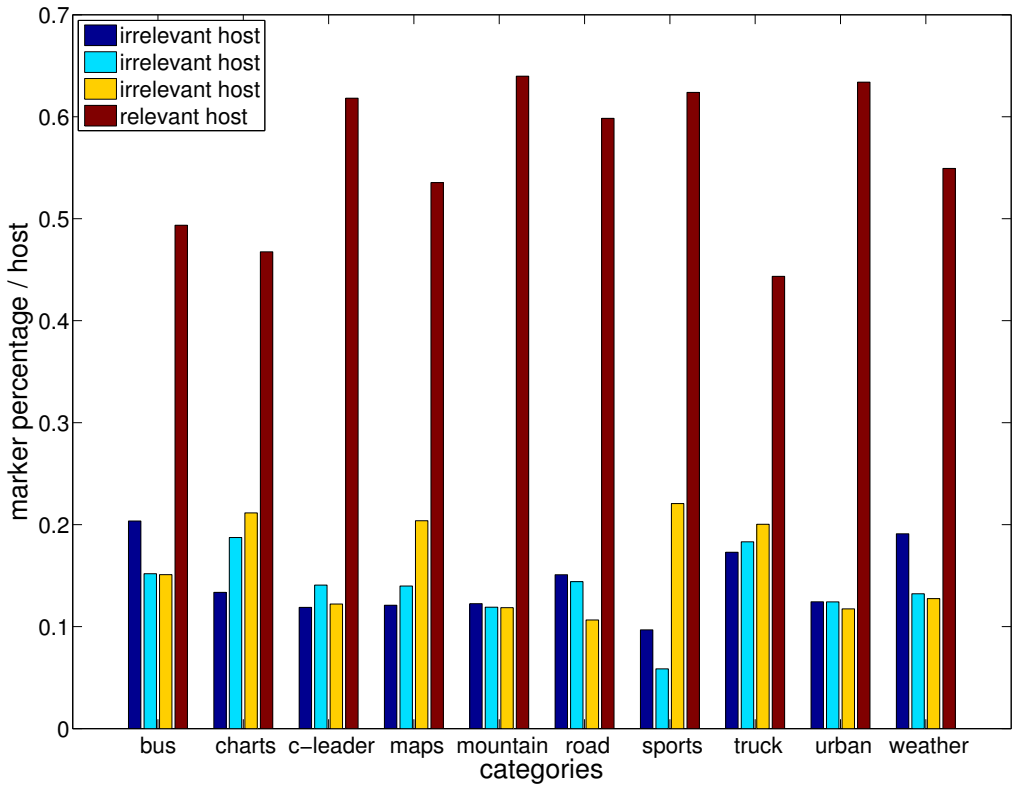


Fig. 5-14. Valeur des probabilités de saut vers chacune des quatre destinations pour le cas à *faible localisation*. La probabilité de se diriger vers la quatrième destination (destination pertinente) est largement plus élevée que pour les autres dans toutes les catégories, ce qui montre que la localisation a été correctement apprise, même si cela est moins flagrant que pour le cas à *forte localisation*.

5.4.2 Gains dus à l'apprentissage avec marqueurs - Test *Corel*

Nous voulons vérifier que le gain obtenu en 5.3 émerge bien de l'apprentissage des chemins. Pour cela, nous avons lancé la même série de test, mais sans évolution des marqueurs, c'est à dire que les agents ont systématiquement 50% de chance de se déplacer soit vers A soit vers B, à tout moment de la session. Comme le montre la figure 5-15, les résultats se dégradent à mesure que la catégorie se concentre sur une seule collection.

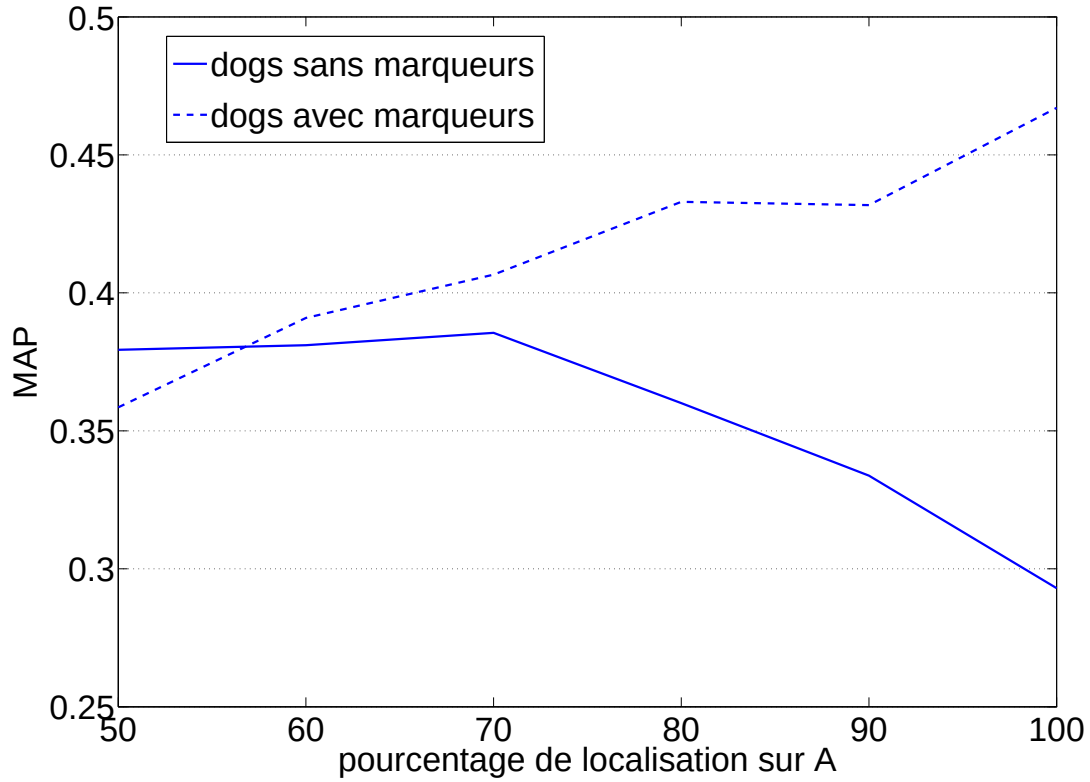


Fig. 5-15. Comparaison du *MAP* sur la catégorie *dogs* de la base *Corel* avec et sans la stratégie active distribuée. Les résultats sont comparables dans le cas où la catégorie est également distribuée sur les deux destinations. Dans le cas où la catégorie est concentrée sur une seule des destinations, le *MAP* sans la stratégie active distribuée s'effondre, tandis que celui avec la stratégie active distribuée augmente.

On peut expliquer ceci par le fait que puisque 50% des agents se dirigent en moyenne vers B et ramèneront d'autant plus d'images non pertinentes que la catégorie est faiblement concentrée sur cette base, alors la composition de l'ensemble d'apprentissage est fortement biaisé. Il contient beaucoup d'annotations négatives, dont une bonne partie qui ne sont vraisemblablement pas informatives, puisque choisies sur un sous-ensemble ne contenant que des images non pertinentes.

5.5 Réutilisation des marqueurs

Les marqueurs étant réinitialisés à chaque nouvelle session, toute l'information donnée par l'utilisateur et utilisée pour l'apprentissage de chemins se trouve perdue. L'utilisation à plus long terme de l'information fournie par l'utilisateur est un problème complexe. La question de l'apprentissage à long terme (*cf* partie III) est de voir si les informations accumulées durant plusieurs sessions de recherche peuvent être réutilisées et comment. À

titre d'expérience, nous nous plaçons dans le cas simple où une seule et unique catégorie est cherchée durant toutes les sessions.

Ce jeu de test est assez irréaliste, car il est fort improbable qu'un utilisateur ne cherche qu'une seule et unique catégorie. Cependant, il a le mérite d' être très simple à mettre en œuvre et de tester l'influence d'un meilleur apprentissage des marqueurs (dû à une plus grande quantité d'annotations) sur les résultats images obtenus. Précisons également que la fonction de similarité est contrairement aux marqueurs remise à zéro à chaque nouvelle recherche.

5.5.1 Protocole expérimental

Le protocole expérimental est sensiblement le même que pour le jeu de test *Trec-Vid'05*. Chaque catégorie est recherchée durant 100 sessions successives. Cependant, d'une session à une autre, les marqueurs ne sont pas remis à un, mais les valeurs obtenues à la fin de la session précédente sont conservées.

5.5.2 Résultats

La figure 5.16 montre un comparatif des valeurs de rappel entre le système centralisé, notre système avec une ré-initialisation des marqueurs à chaque nouvelle requête et ce même système sans aucune ré-initialisation des marqueurs. Le gain observé par rapport à la ré-initialisation systématique des marqueurs est comparable à celui obtenu par ce dernier vis à vis du système centralisé. Puisque les marqueurs sont très bien appris sur le long terme, la stratégie active, n'étant appliquée qu'à la collection contenant les images recherchées, est tout particulièrement efficace.

Le résultat de l'apprentissage des marqueurs est visible sur la figure 5.17. Toutes les valeurs dépassent les 90%, ce qui montre que le système a très bien appris le chemin menant à la collection pertinente, et ceci quelque soit la difficulté de la catégorie recherchée. Or, la mesure de pertinence (obtenue à partir du classifieur SVM) est réinitialisé à chaque nouvelle requête, ce qui entraîne une baisse du nombre d'images pertinentes annotées périodiquement. À chaque nouvelle requête, le renforcement des marqueurs est par conséquent moins performant. Ceci montre l'efficacité de l'algorithme sur un plus long terme, et malgré les perturbations dues aux ré-initialisations systématiques du classifieur entre les sessions.

5.6 Conclusion

Les expériences présentées dans ce chapitre ont permis de valider la stratégie proposée au chapitre précédant sur plusieurs points. Tout d'abord, plus les images recherchées sont précisément localisées, plus notre stratégie offre un gain dans la qualité des résultats retrouvés. Cela est d'autant plus vrai comparé à une stratégie qui ignorerait la localisation

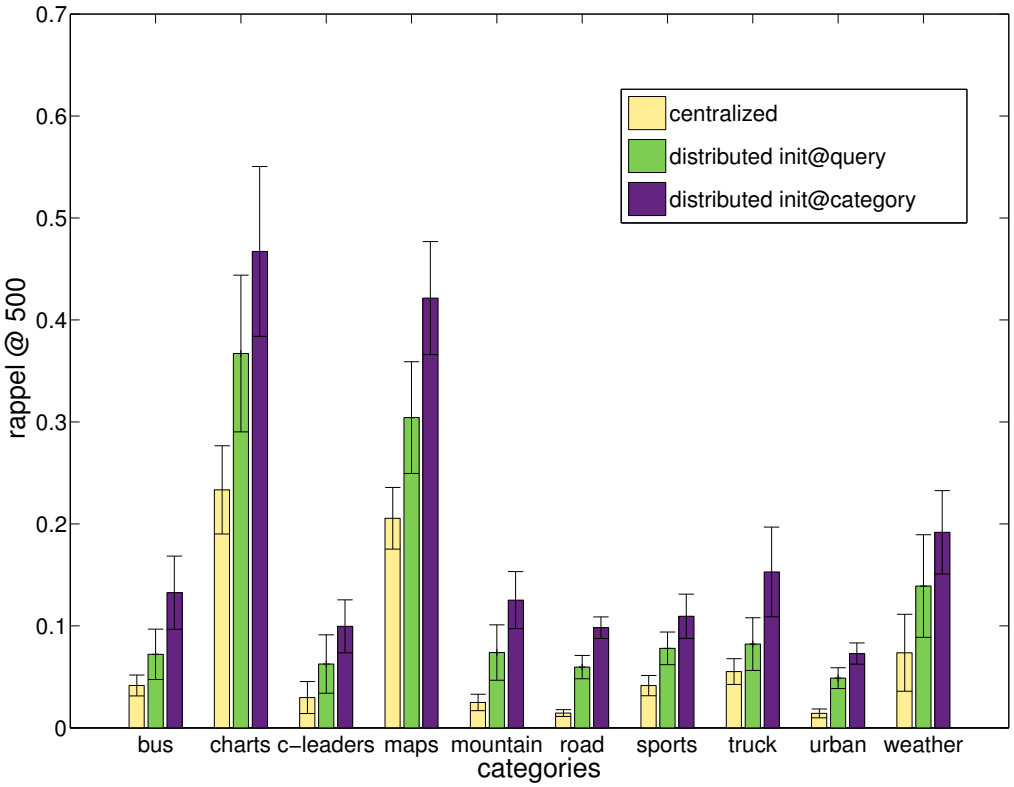


Fig. 5.16. ***Rappel*** **pour les catégories testées de la base** ***TrecVid'05*** **comparant l'approche centralisée de référence, l'approche active distribuée classique et l'approche active distribuée en conservant les marqueurs d'une session à l'autre. Les gains obtenus en gardant les marqueurs d'une session à l'autre par rapport à l'approche active distribuée classique sont comparables à ceux obtenus par l'approche active distribuée classique par rapport à l'approche centralisée.**

des données. Ensuite, l'apprentissage des chemins pertinents par notre système multi-agents s'effectue correctement, même dans le cas où la localisation des images recherchée n'est pas parfaite. Enfin, notre expérience préliminaire sur la réutilisation des marqueurs d'une session à une autre nous permet d'entrevoir un fort potentiel dans ce procédé, ce qui fait l'objet de la partie suivante.

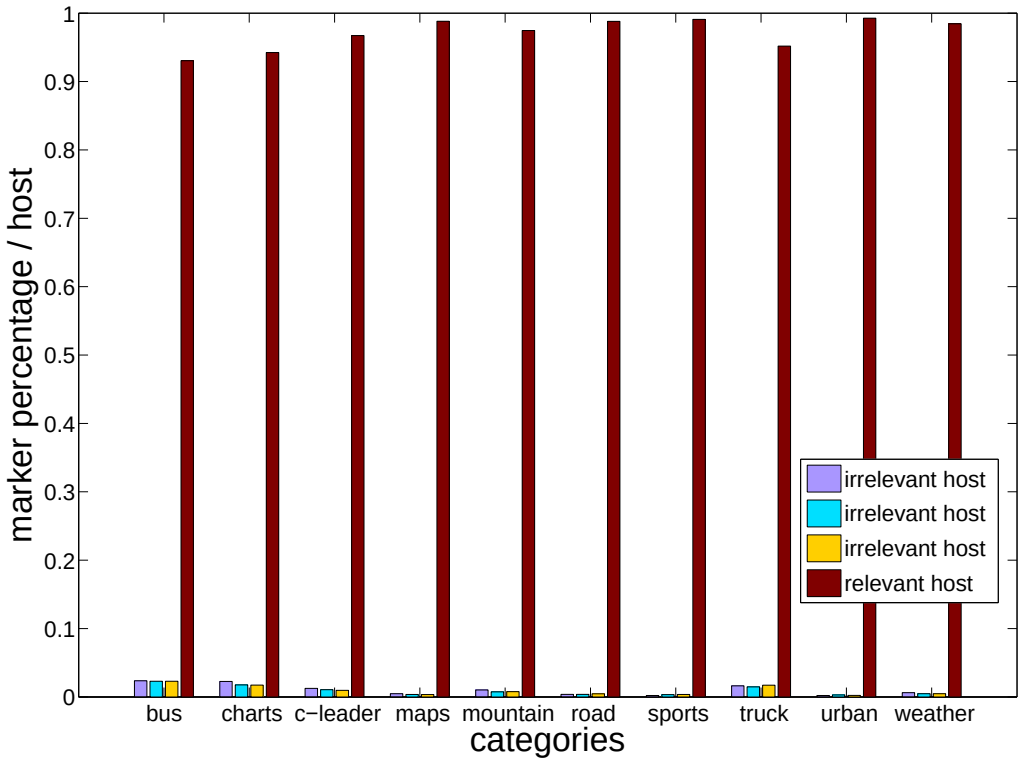


Fig. 5-17. Valeur des probabilités de saut vers les quatre destinations pour l'approche conservant les marqueurs d'une session à une autre pour chacune des catégories de la base *TrecVid'05* testées. Les probabilités de saut sont très proches de 1, ce qui montre que la localisation de la catégorie recherchée est très bien apprise.

Troisième partie

Apprentissage distribué à long terme

Chapitre 6

Apprentissage à long-terme

Ce chapitre est la continuation et l'extension du paragraphe 5.5 du chapitre précédent, dans lequel nous avons présenté des expériences réutilisant les marqueurs d'une session à une autre. Il s'agit donc d'optimiser le système non plus sur une seule session mais sur un ensemble de sessions, ce que l'on appelle optimisation à *long terme*. Dans un premier temps, nous faisons une présentation de l'apprentissage à long terme, afin de mieux définir la problématique qui découle de notre contexte distribué. Puis, nous présentons l'architecture de notre système, et en particulier les deux problèmes d'apprentissage que nous formulons et que nous tentons de résoudre. Enfin, nous explicitons l'optimisation effectuée sur ces deux problèmes.

6.1 Apprentissage à long terme

Il s'agit d'étendre notre système à une utilisation massivement multi-utilisateurs et renouvelée dans le temps. En effet, à la fin d'une session utilisant le système présenté dans la partie précédente, toute l'information fournie par l'utilisateur au cours du bouclage de pertinence est perdue. L'idée générale de cette partie est de réutiliser l'information obtenue au cours des sessions précédentes afin d'améliorer le moteur de recherche soit en rapidité (nombre de labels jusqu'à un résultat convenable), soit en qualité.

6.1.1 Introduction

Dans la partie précédente, l'utilisation du système démarre par une requête image, se poursuit par plusieurs itérations du bouclage de pertinence, et se conclut par l'affichage des résultats finaux. Dans le contexte auquel on s'attaque dans ce chapitre, on dispose de plusieurs réalisations de ce type. On appelle *sessions* ces réalisations :

Définition Une *session s* est l'ensemble des itérations du bouclage de pertinence obtenu à l'issue d'une recherche.

$$s = \{\mathbf{x}_k \in \mathcal{A}, y_k, u_k\} \tag{6.1}$$

Avec $\mathbf{x}_k$ les images annotées lors de la session, y_k les labels utilisés pour la construction de $f_{\mathcal{A}}$ et u_k les signaux de renforcement utilisés pour l'estimation de r'_i du réseau. Définissons alors un concept :

Définition Un concept (ou *catégorie*) est un sous-ensemble de l'ensemble des images disponibles sur le réseau.

À la fin d'une session, nous avons un ensemble d'images et d'annotations associées permettant à l'aide d'algorithmes d'apprentissage de déterminer quelles sont les images pertinentes au regard de la requête, mais aussi quels sont les sites pertinents desquels récupérer les résultats. À la fin d'un grand nombre de sessions, nous disposons d'un ensemble d'images et d'annotations associées beaucoup plus grand.

L'idée générale de l'apprentissage à long-terme est alors d'utiliser cet ensemble afin d'améliorer l'efficacité des recherches des sessions suivantes. C'est un problème d'apprentissage singulier, car les concepts relatifs à chacune sessions ne sont pas connus et donc le regroupement de sessions relatives à un même concept n'est pas possible *a piori*.

6.1.2 État de l'art

Plusieurs méthodes existent pour optimiser un système de recherche d'informations visuelles sur l'ensemble des sessions disponibles. Une catégorie importante de méthodes concerne la modification de la similarité entre images. Elle se divise principalement en deux types : la modification de la mesure de similarité, ou bien la modification de la représentation des images. La première consiste souvent à optimiser les paramètres d'une distance (associée à la fonction de similarité) entre les images de manière à ce que les images qui ont été annotées pareillement se trouvent plus proches. Le plus souvent, cela revient à optimiser un jeu de poids sur l'ensemble des sessions [Müller et al., 2000].

La seconde consiste à optimiser la représentation de la base soit en modifiant directement les signatures des images [Cord and Gosselin, 2006], soit en modifiant la matrice de similarité [Heisterkamp, 2002, Cord and Gosselin, 2006]. Dans le cas des machines à noyaux, modifier la matrice de similarité revient à modifier le noyau par une transformation inconnue (et se rapproche donc du premier type de méthodes). Ce second type de méthodes a le désavantage de ne pouvoir être utilisé sur des bases ouvertes, puisque la transformation modifiant la représentation est spécifique à chaque image et ne saurait être extrapolée à de nouvelles images. Il en est de même pour les bases distribuées, l'ensemble des images étant indisponible localement, il est impossible d'optimiser la représentation des images.

Dans le domaine de la recherche d'information distribuée, l'optimisation des recherches futures à partir des précédentes est un sujet quotidiennement abordé. Sa forme la plus simple est le système de *bookmarks* de navigateur qui permettent de retenir les

résultats des navigation précédentes. Les navigations effectuées par les utilisateurs peuvent être utilisées afin de concevoir des profiles facilitant les recherches futures. L'idée est alors d'adapter les résultats des moteurs de recherche en fonction du profile de l'utilisateur en se concentrant sur les pages les plus pertinentes pour le profile concerné [Chan, 1999, Speretta and Gauch, 2005].

D'un point de vue plus technique, les traces générées par les utilisateurs lors de leurs navigations peuvent être utilisées pour optimiser les stratégies de cache en retenant les résultats de requêtes associées à ces navigations [Amann and Constantin, 2007]. [Gasparetti and Micarelli, 2003] proposent d'utiliser un système multi-agents inspiré des algorithmes *ACO* vus au chapitre 2, dans lequel les agents parcourent le graphe de documents hypertextes et marquent les nœuds en fonction de leur pertinence de manière à optimiser le parcours des agents suivants. Cette idée consiste à utiliser un grand nombre de navigations afin d'optimiser les navigations suivantes en associant une plus grande importance aux parcours qui ont mené vers des documents pertinents.

À notre connaissance, il n'y a pas de travaux s'attaquant notre problème spécifique de recherche par le contenu dans des bases distribuées.

6.1.3 Problématique

Appliquée à la recherche d'images distribuées, on peut définir deux façons de réutiliser l'information fournie par les utilisateurs : soit en améliorant la représentation des images, soit en améliorant la sélection des sites dans lesquels la recherche est effectuée. Nos travaux portent sur cette deuxième partie, et nous faisons quelques hypothèses sur le contexte afin de formaliser notre problème d'apprentissage :

Distribution des images

La distribution des images sur le réseau est la même que pour le système présenté précédemment, à savoir que pour chaque catégorie, la répartition des images lui appartenant sur les différentes bases disponibles est différente d'une répartition uniforme. Autrement dit, les bases du réseau sont spécialisées : elle contiennent seulement quelques catégories par rapport à toutes les catégories disponibles. L'hypothèse supplémentaire que nous faisons est que cette répartition ne varie pas dans le temps.

Nombre de concepts

Nous nous limitons aussi sur le nombre de concepts contenus dans toutes les bases du réseau. Nous supposons ainsi qu'il n'y a en tout que P concepts. Chacune des images peut, par contre, appartenir à plusieurs de ces concepts. Ce nombre de concepts est invariant dans le temps, de même que la nature des concepts.

Annotations disponibles

L'information que nous allons utiliser est celle contenue dans les annotations données par les utilisateurs. Dès lors, et puisque nous voulons nous servir de l'information fournie par tous les utilisateurs et à n'importe quel instant, il faut que celle-ci soit cohérente. Cela se traduit par deux hypothèses, l'une sur l'utilisateur, et l'autre sur l'ensemble des utilisateurs.

Pour un utilisateur donné, il faut que sa définition d'une catégorie ne change pas trop dans le temps. Autrement dit, il faut que l'utilisateur donne environ les mêmes annotations pour un même concept recherché, et ce à n'importe quelle session.

Proposition. *Soit $u_n(\mathbf{x})$ l'annotation donnée par l'utilisateur à l'image $\mathbf{x}$ lors de la session n, alors :*

$$\forall n, k \quad u_n(\mathbf{x}) = u_k(\mathbf{x}) \tag{6.2}$$

Cette hypothèse paraît réaliste dans le sens où notre point de vue n'évolue que peu pour la grande majorité des concepts (une voiture restant étiquetée "voiture").

D'autre part, il faut que l'ensemble des utilisateurs soit cohérent dans leurs annotations. C'est à dire que pour chaque image, ils attribuent les mêmes catégories.

Proposition. *Soit $u^i(\mathbf{x})$ l'annotation donnée par l'utilisateur i à l'image $\mathbf{x}$, alors :*

$$\forall i, j \quad u^i(\mathbf{x}) = u^j(\mathbf{x}) \tag{6.3}$$

Cette hypothèse formulée de cette façon est très stricte, puisqu'elle suppose que les concepts sont exactement les mêmes pour tous les utilisateurs. Or, si cela peut sembler réaliste pour des catégories d'objets ("voitures", "chats", ...), cela l'est largement moins pour des concepts plus abstraits englobant des familles d'objets ("animaux"/"insectes", "maison"/"immeuble", ...). Cependant, nous sommes contraint à de telles hypothèses lors de nos expériences pour lesquelles nous ne disposons que d'une unique vérité-terrain par image. Il reste quand même assez réaliste de considérer que les variations sur les annotations données par les utilisateurs aux images sont faibles.

Synthèse

Considérant ces ressources et ces hypothèses, nous nous trouvons face à un problème d'apprentissage très particulier dans un contexte où les annotations disponibles ne sont pas directement exploitables. En effet, lors d'une session nous disposons d'annotations, mais nous ne savons pas avec quelles annotations des sessions précédentes nous pouvons les associer afin d'améliorer l'apprentissage d'un concept. On parle d'apprentissage *faiblement supervisé*.

Notre objectif est d'améliorer la localisation des concepts recherchés, en se basant sur l'estimation qui en a été faite aux itérations précédentes. Le problème d'apprentissage

se décompose alors en deux parties : la première consiste à trouver le concept qui est recherché lors d'une session, et la seconde vise à fusionner les informations obtenues lors d'une session à celles obtenues lors des sessions précédentes concernant le même concept.

6.2 Modélisation du problème

Comme nous l'avons vu en 6.1.2, il n'existe pas à notre connaissance de travaux s'intéressant à notre problématique très particulière, sur lesquels nous pourrions nous appuyer. Nous modélisons le problème en créant une nouvelle architecture basée sur celle que nous avons présentée dans la partie précédente. Cette architecture doit permettre de répondre à la problématique dans le cadre des hypothèses que nous nous sommes fixées.

6.2.1 Architecture

À la fin d'une session s_c, nous disposons d'un ensemble de marqueurs répartis sur le réseau, dont les valeurs permettent d'obtenir une estimation de la pertinence de chaque site par rapport au concept c recherché. Appelons cet ensemble de marqueurs un *plan* :

Définition On appelle *plan* un ensemble de marqueurs du réseau permettant d'évaluer la pertinence des sites en rapport avec un concept donné.

L'objectif de l'apprentissage à long-terme est de pouvoir réutiliser un plan appris lors de sessions précédentes pour une recherche portant sur le même concept. Cependant, puisqu'il existe P concepts, un seul plan ne peut suffire. Nous proposons une nouvelle architecture qui comporte P' plans, chacun étant associé à un concept donné, afin de pouvoir estimer la pertinence des sites pour chacun des P concepts.

Le cas le plus facile à imaginer est quand $P' = P$. Il y a alors une association directe entre chacun des plans et chacun des concepts. Cependant, on peut imaginer donner plus de richesse au système en prenant $P' > P$, auquel cas plusieurs plans peuvent être associés au même concept, ou bien à aucun concept. Ou encore restreindre la richesse du système en imposant $P' < P$, auquel cas il est possible que certains concepts n'aient pas de plan associé, ou bien qu'un plan soit associé à plusieurs concepts partageant la même localisation.

Cette architecture nous semble bien adaptée à la double problématique que nous avons énoncée. En effet, d'une part la détermination du concept recherché lors d'une session revient à sélectionner un plan. Nous proposons pour cela d'estimer les paramètres d'une fonction ψ de sélection de plan à l'aide des annotations fournies par l'utilisateur au cours de la session. Et d'autre part, la fusion des données de sessions correspondant à un même concept c se fait en améliorant l'estimation des pertinences des sites sur le plan correspondant à c.

Notre système possède alors deux ensembles d'éléments relatifs à chaque partie du problème d'apprentissage :

- ψ la fonction qui sélectionne le plan p correspondant au concept recherché parmi les P' plans.
- L'ensemble des $P' \times N$ estimations des pertinences des N sites relatives aux P' concepts.

Les seules données d'apprentissage disponibles étant les annotations fournies par l'utilisateur, elles sont utilisées pour les deux parties du problème d'apprentissage. Cependant, l'échelle de leur utilisation n'est pas la même : en effet, la fonction ψ est apprise sur une seule session (puisque le problème de la sélection de plan est restreint à la session courante), tandis que l'optimisation des valeurs de pertinence est prolongée à toute les sessions utilisant le même plan.

Une fois la fonction ψ déterminée, on dispose d'une unique valeur de pertinence pour chacun des sites, à la manière étudiée dans la partie II. La seconde partie du problème consistant à optimiser les valeurs de pertinence est donc très semblable à ce que nous avons présenté au chapitre 4. Ceci nous permet de nous appuyer sur les résultats théoriques et expérimentaux obtenus dans les chapitres 4 et 5.

6.2.2 Notations

Localement à un site, on peut ranger les pertinences de chaque plan dans un vecteur, dont la représentation vectorielle est illustrée sur la figure 6-1 :

Définition Soit $\mathbf{R}_i$ le vecteur de pertinence de la machine i contenant un ensemble de P' pertinences $R_i[j]_{1 \leq j \leq P'}$.

Les pertinences sont définies comme en 4.2.1 : elles reflètent la pertinence du site considéré en regard d'une certaine catégorie. De même, on peut définir la pertinence normalisée $\mathbf{R}'_i$:

Définition La pertinence normalisée $\mathbf{R}'_i$ de la machine i est égale au rapport de chacune des pertinences $\mathbf{R}_i[j]$ qui la composent sur la somme des pertinences $\mathbf{R}_k[j]$ de toutes les machines accessibles :

$$\mathbf{R}'_i = \left[\frac{\mathbf{R}_i[j]}{\sum_k \mathbf{R}_k[j]} \right]_{1 \leq j \leq P'} \tag{6.4}$$

On peut alors utiliser ces notations pour décrire un plan, qui est l'ensemble $\{\mathbf{R}_k[j]\}_k$ des j-ième pertinences de chaque machine k du réseau, et que l'on note $\mathbf{R}[j]$. Dans ce cas, le prototype de la fonction ψ sélectionnant le plan p correspondant au concept recherché lors de la session s est le suivant :

$$\begin{aligned} \psi_s : \quad & \mathbb{R}^{P'} \rightarrow \mathbb{R} \\ & \mathbf{R}_i \mapsto \mathbf{R}_i[p] \end{aligned} \tag{6.5}$$

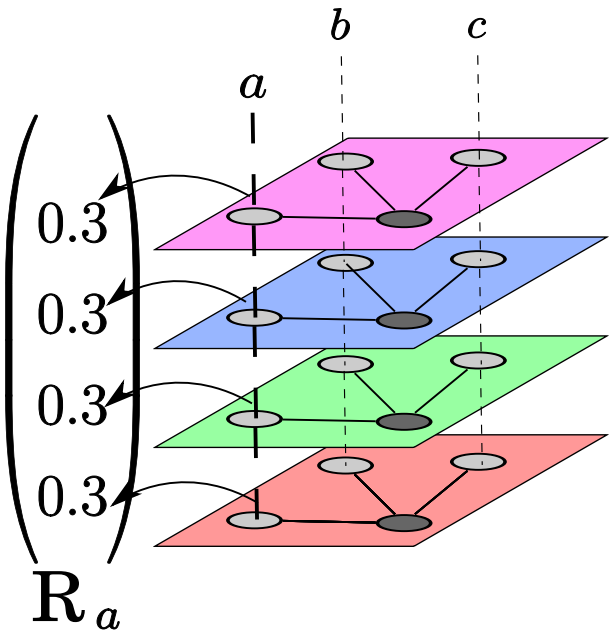


Fig. 6-1. Modèle à quatre plans pour un système comportant trois sites possibles a, b et c. Le vecteur de pertinence R_a donne la valeur de pertinence du site a pour chacun des quatre plans.

À partir de ces notations, nous pouvons formaliser la double problématique de l'apprentissage à long-terme en l'optimisation de deux jeux de paramètres, à l'aide des annotations fournies par l'utilisateur. Le premier, que l'on note θ_1, conduit à l'estimation de la fonction de sélection de plan ψ. Le second, que l'on note θ_2 conduit à l'estimation des vecteurs $\mathbf{R}_i$ de chaque machine du réseau de sorte que le calcul de la pertinence de chaque site lors d'une session tienne compte des informations des sessions passées. Le résultat attendu de l'apprentissage à long terme est illustré sur la figure 6-2.

6.2.3 Optimisation

Les optimisations de θ_1 et θ_2 sont dépendantes l'une de l'autre. En effet, afin de déterminer ψ, il peut être utile qu'un plan p offre une bonne estimation de la pertinence des sites par rapport au concept recherché. L'optimisation de θ_1 sera donc d'autant meilleure si l'optimisation de θ_2 est accomplie. À l'inverse, l'amélioration de l'estimation des pertinences des sites pour un plan p lors d'une session s suppose que ψ est déterminée. Donc, l'optimisation de θ_2 repose sur celle de θ_1.

Nous proposons une méthode itérative au sein du bouclage de pertinence pour résoudre ce double problème. Lors de l'itération $n+1$ du bouclage de pertinence d'une session s, on estime $\theta_1(n+1)$ à partir de $\theta_1(n)$, $\theta_2(n)$ et $\mathcal{A}(n)$ l'ensemble des annotations. Puis, $\theta_2(n+1)$ est estimé à partir de $\theta_1(n+1)$, $\theta_2(n)$ et $\mathcal{A}(n)$. Nous détaillons alors les méthodes d'obtention de $\theta_1(n+1)$ et $\theta_2(n+1)$.

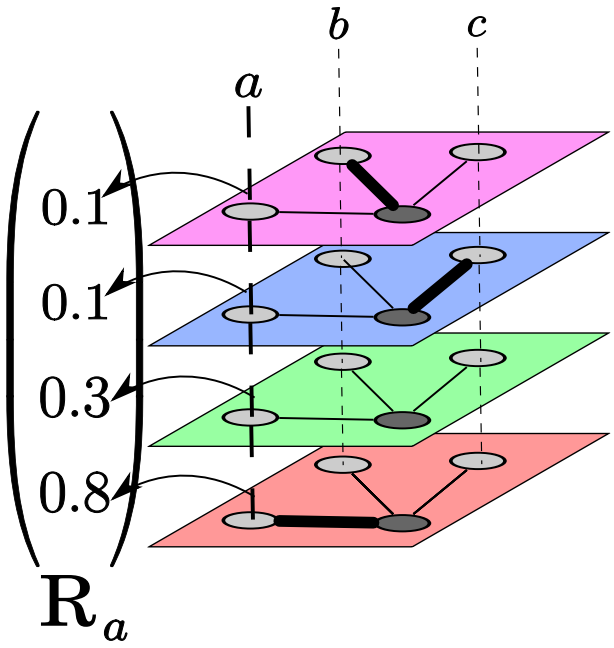


Fig. 6-2. Exemple de résultat de l'optimisation à long-terme pour un modèle à quatre plans d'un système comportant 3 sites a, b et c. Le premier plan est spécialisé vers b, le second vers c, le quatrième vers a et le troisième ne s'est pas spécialisé. Les valeurs de pertinence du vecteur R_a reflètent bien cette spécialisation.

Sélection du concept recherché

Partant de l'hypothèse que l'on possède pour chaque destination un ensemble d'estimation de la pertinence par rapport à tous les concepts, et sachant que l'on ne connaît pas le concept recherché, l'optimisation de θ_1 consiste à trouver la dimension j du vecteur $\mathbf{R}_i[j]$ qui correspond à l'évaluation de la pertinence du site i pour le concept recherché. La stratégie que nous proposons consiste à utiliser de nouveau les annotations fournies par l'utilisateur pour construire ψ, comme le montre la figure 6-3.

Nous associons à chaque valeur de pertinence j une probabilité w_j d'être la pertinence relative au concept recherché. À chaque itération, la sélection opérée par ψ correspond à un tirage effectué sur les plans selon les probabilités w_j de chacun d'être associé au concept recherché. Des images sont alors sélectionnées depuis chacune des collections proportionnellement à leur pertinence sur le plan sélectionné, exactement comme cela était fait dans la partie II.

L'optimisation de θ_1 correspond à l'optimisation de l'ensemble des $\{w_j\}_{1\leq j\leq P'}$, de sorte que les w_p des plans correspondant au concept recherché soient non-nuls, alors que les $w_{\bar{p}}$ des plans ne correspondant pas au concept recherché soient nuls.

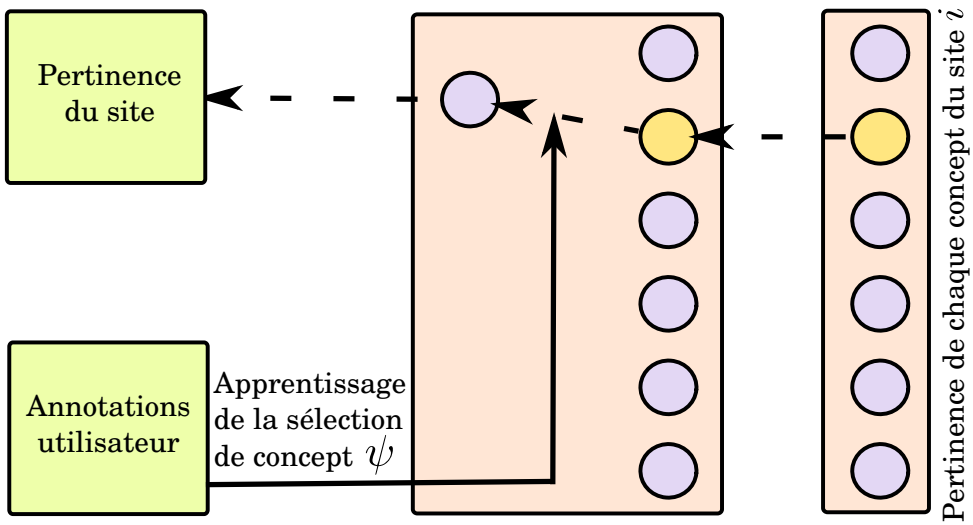


Fig. 6.3. La fonction de sélection ψ choisit le plan le plus en rapport avec la recherche. Elle est construite par apprentissage à l'aide des annotations de l'utilisateur.

Sélection des sites pertinents

Supposons que l'étape de sélection du plan associé au concept recherché ait été effectuée (voir 6.2.3), la pertinence qui lui est associée est récupérée de chaque destination. On se retrouve alors dans le cas de figure étudié au chapitre 4. Les pertinences sont utilisées afin de sélectionner les images à annoter. Comme en 4.4.1, le nombre d'images sélectionnées sur chaque site est proportionnel à la pertinence de ce site (équation 4.7).

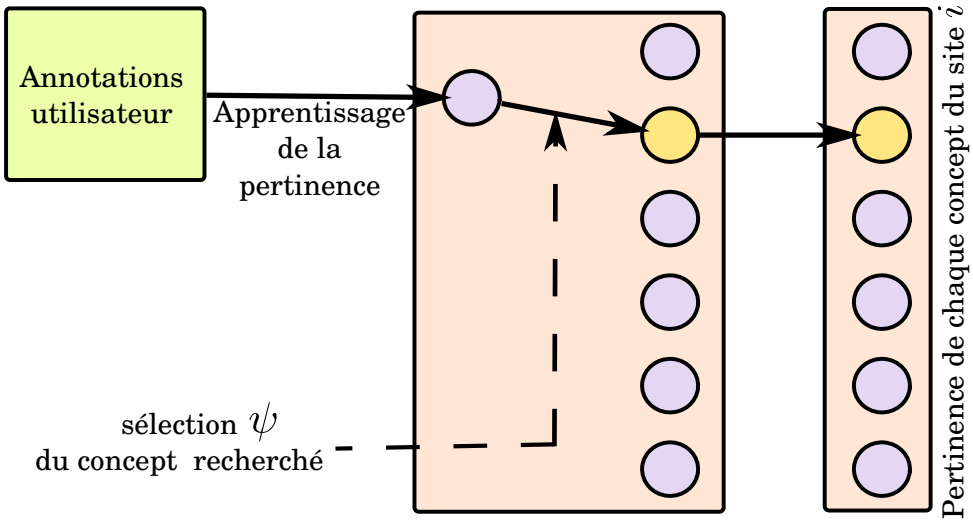


Fig. 6.4. La pertinence sélectionnée par la fonction de sélection est mise à jour à l'aide des annotations obtenues lors de l'interaction avec l'utilisateur. Cette opération étant commune à toute les sessions, les valeurs de pertinences contenues sur chaque site sont issues d'un processus d'évolution contenant l'ensemble des labels de toutes les sessions, c'est-à-dire sur le long-terme.

La pertinence est alors évaluée sur la base des annotations données par l'utilisateur, comme dans l'équation 4.5. La figure 6.4 illustre la propagation des annotations de l'utilisateur aux estimations de la pertinence de chaque site, pour le plan sélectionné. Cette évaluation de la pertinence tient alors compte non seulement des annotation fournies lors de la session en cours, mais aussi de toutes les annotations fournies, relatives à ce plan, lors des sessions précédentes.

6.2.4 Synthèse

Comme on peut le voir, notre système comporte trois tâches d'apprentissage :
- La sélection de plan ψ.
- L'évaluation de la pertinence $\mathbf{R}_i$ des sites i.
- L'apprentissage de la fonction de similarité $f_{\mathcal{A}}$.

On peut alors utiliser les deux méthodologies décrites précédemment pour réaliser ces tâches d'apprentissage au sein d'une itération du bouclage de pertinence lors d'une session. Soit N sites i de bases B_i. Pour la recherche d'une catégorie p donnée, les N r_i sont fixés et le système dispose de P' plans. Le processus de passage de l'itération n à l'itération $n+1$ est le suivant :

1. Soit $\mathcal{A}(n)$ l'ensemble d'images $\{\mathbf{x}_j\}_n$ et de leurs annotations $\{y_j\}_n$ à l'itération n.
2. On estime la fonction $\psi_{\mathcal{A}(n)}$ de sélection de plan à l'aide de $\mathcal{A}(n)$, et on sélectionne le plan p' à l'aide de $\psi_{\mathcal{A}(n)}$.
3. À partir de p' et de $\mathcal{A}(n)$ on calcule une valeur approchée de la pertinence $\hat{R}'_i[p']$ (*cf* équation 4.6).
4. On calcule la mesure de pertinence $f_{\mathcal{A}(n)}$ à partir de $\mathcal{A}(n)$ (*cf* équation 4.9).
5. À l'aide de $f_{\mathcal{A}(n)}$ et de la pertinence $\hat{R}'_i[p']$, on sélectionne I images $\{\mathbf{x}_j\}_{n+1}$, selon la proportion définie en 4.7.
6. On construit l'ensemble $\mathcal{A}(n+1) = \mathcal{A}(n) \cup \{\mathbf{x}_j, y_j\}_{n+1}$, où les y_j sont les annotations données par l'utilisateur aux images $\mathbf{x}_j$.

On remarque qu'une fois le plan p sélectionné, on se retrouve avec un seul plan, et le reste de l'itération est très similaire à celle vue en 4.3.2.

Dynamique du système

La sélection du concept recherché et l'évaluation de la pertinence de chaque site se basent toutes deux sur les mêmes informations (les annotations fournies par l'utilisateur). Or les annotations données par l'utilisateur sont relatives à la pertinence sélectionnée. Ainsi, pour déterminer le concept à sélectionner, cela suppose que la pertinence du site pour chaque concept soit correctement évaluée, et donc que les images ramenées soient représentatives du concept sélectionné. De même, pour évaluer la pertinence des sites cela

suppose que la sélection du concept recherché soit correct. Ces deux apprentissages sont donc dépendants l'un de l'autre.

Pour pouvoir bénéficier de l'optimisation à long-terme, il faut que la sélection de plan (qui est limitée à une seule session) soit plus rapide que l'apprentissage des pertinences sur un plan donné (qui se font sur plusieurs sessions). Il faut donc que la dynamique de l'apprentissage de la fonction de sélection soit plus rapide que celle du renforcement des pertinences des sites.

On supposera dans un premier temps que la pertinence associée à chaque concept est correctement évaluée, puisqu'issue d'une fusion de plusieurs sessions (le système a déjà été utilisé). On peut alors imaginer deux scénarios : dans un premier cas, nous proposons alors de découper la stratégie en deux étapes. Dans une première étape, nous déterminons le concept recherché et utilisons les annotations fournies par l'utilisateur afin d'apprendre la fonction de sélection. Dans une seconde étape, nous allons alors plutôt nous concentrer sur l'apprentissage de la pertinence. Les annotations fournies par l'utilisateur vont alors être utilisées pour apprendre la pertinence du site relativement au concept recherché.

Dans le second scénario, l'apprentissage de ψ se fait sur la totalité de la session, mais avec la contrainte d'avoir une dynamique rapide à l'échelle de la session (avoir convergé à 10% de la session permet d'utiliser 90% des annotations pour renforcer le bon plan).

En réalité, le démarrage du système ne pose pas de problème. Puisque la fonction de sélection choisit le plan le plus pertinent pour la requête, lors d'une session cherchant un concept encore jamais cherché, le plan le moins spécialisé devrait être choisi et appris pour ce concept.

6.3 Implémentation

Nous proposons une implémentation dans laquelle nous disposons sur chaque machine i d'un ensemble de P' marqueurs qui vont servir à déterminer les mouvements à la manière de ce qui était présenté en 4.4.1. Parmi ceux-ci, un marqueur va être sélectionné à l'aide de la fonction ψ sur chacun des sites afin d'en estimer la pertinence. On peut alors renforcer les marqueurs sélectionnés par ψ de la même manière que dans la partie II.

Afin de garder le formalisme vectoriel proposé précédemment, nous notons sous forme de vecteur l'ensemble de marqueurs présent sur chaque machine :

Définition On appelle un vecteur de marqueurs un vecteur $\mathbf{m}_i(n)$ composé de N variables aléatoires $\mathbf{m}_i[j]_{1\leq j\ \leq P'}(n)$ évoluant dans un temps discret local n.

6.3.1 Sélection de plan

Pour sélectionner la composante du vecteur de marqueurs en correspondance avec le concept recherché, on utilise un vecteur $\mathbf{\Psi}$ ayant 1 sur la composante p sélectionnée, et 0

ailleurs. L'application de la fonction ψ au vecteur de marqueurs $\mathbf{m}_i$ s'écrit alors sous la forme du produit matriciel suivant :

$$\psi(\mathbf{m}_i) = \mathbf{\Psi}^{\mathrm{T}} \cdot \mathbf{m}_i \tag{6.6}$$

Le vecteur $\mathbf{\Psi}$ est la projection sur la composante p sélectionnée :

$$\mathbf{\Psi} = [\delta(k, p)] \tag{6.7}$$

Pour calculer $\mathbf{\Psi}$, on utilise l'ensemble des poids w_j de θ_1 que l'on réunit dans un vecteur $\mathbf{w}$:

$$\mathbf{w} = [w_j]_{1 \leq j \ \leq P'} \tag{6.8}$$

Chaque poids w_j représente la probabilité de chaque plan d'être associé au concept recherché. $\mathbf{\Psi}$ est tiré selon la loi multinomiale $\mathcal{M}(1; w_1, \ldots, w_{P'})$ utilisant les poids de $\mathbf{w}$. Puisque nous n'avons aucun *a priori* sur la composante pertinente (c'est à dire le plan pertinent) pour le concept recherché, nous proposons d'apprendre les poids w_j à l'aide du bouclage de pertinence.

À chaque fois qu'une image est annotée, le poids associé au plan utilisé pour retrouver cette image est mis à jour avec la règle suivante :

$$w_j \longleftarrow w_j + \varepsilon(u - w_j) \tag{6.9}$$

Avec $u = \frac{y+1}{2}$ dépendant de l'annotation de l'utilisateur et ε une constante d'apprentissage.

Puis le vecteur de poids $\mathbf{w}$ est normalisé à 1. Ainsi, les probabilités de sélection des plans croissent dans le cas où le plan mène à des images positives et sont progressivement mises à zéro dans le cas contraire.

L'avantage de cette manière de faire évoluer les probabilités de chaque plan est d'avoir des variations faibles lors des premières itérations. Nous avons essayé d'autre méthodes comme l'estimateur empirique (w_j est alors égal à la moyenne des u obtenus sur ce plan), mais ses variations se sont montrées trop chaotiques en début de session pour que le système arrive à se fixer sur un plan.

6.3.2 Renforcement des marqueurs

Pour calculer les mouvements des agents, on utilise la composante du vecteur de marqueurs sélectionnée afin de se ramener à une seule valeur réelle pour chaque destination comme dans la partie II :

$$p_i = \frac{\psi(\mathbf{m}_i)}{\sum_k \psi(\mathbf{m}_k)} \tag{6.10}$$

En notant p la composante sélectionnée, cela s'écrit alors :

$$p_i = \frac{\mathbf{m}_i[p]}{\sum_k \mathbf{m}_k[p]} \tag{6.11}$$

Les propriétés des déplacements des agents deviennent ainsi les mêmes que celles présentées dans l'algorithme de la partie II.

L'ensemble des marqueurs $\{\mathbf{m}_k[p]\}$ pour toutes les machines k du réseau forment alors un *plan* tel que définit dans la première partie de ce chapitre. Cette notation permet de voir les agents comme se déplaçant sur un plan particulier désigné par la fonction de sélection.

Renforcement

Puisque nous nous sommes ramenés à une seule grandeur par destination, les règles de renforcement sont les mêmes que dans le chapitre 4, mais appliquées aux marqueurs sélectionnés :

1. Déplacement

À chaque incrémentation du temps local à un site par le passage d'un agent, le marqueur sélectionné est décrémenté :

$$\mathbf{m}_i[j] \longleftarrow \alpha \mathbf{m}_i[j] \tag{6.12}$$

Avec α une constante positive inférieure à un. Cette règle sert à modéliser l'évaporation des phéromones chez les fourmis et permet d'oublier les chemins qui ne sont plus pertinents.

2. Retour

Ce marqueur est incrémenté si l'agent a trouvé une collection :

$$\mathbf{m}_i[j] \longleftarrow \mathbf{m}_i[j] + \beta \cdot a \tag{6.13}$$

Avec a valant 1 si l'agent a trouvé une collection, et 0 sinon, et β est une constante positive. Cette règle permet de renforcer les chemins menant à des bases d'images et donc de ne router les agents que vers des collections.

3. Annotation

Le marqueur sélectionné est mis à jour en fonction de l'annotation donnée par l'utilisateur aux images que l'agent a ramenées :

$$\mathbf{m}_i[j] \longleftarrow \mathbf{m}_i[j] + \gamma \cdot u \tag{6.14}$$

Avec u valant 1 si l'annotation est positive, et 0 sinon, et γ est une constante positive. On peut faire le lien entre u et y l'annotation associée à une image : $u = 1 \Leftrightarrow y = 1$ et $u = 0 \Leftrightarrow y = -1$.

6.3.3 Paramètres et dynamique du système

La fonction de sélection et les valeurs de marqueurs étant basées sur le même signal de renforcement, comme expliqué en 6.2.4, il faut paramétrer le système afin de favoriser sa convergence. Nous proposons deux moyens de régler le comportement du système :

- Le premier consiste en la séparation de l'attribution des annotations à l'un ou l'autre des algorithmes d'apprentissage. Pratiquement, les n premières annotations sont utilisées pour l'apprentissage de la fonction de sélection (typiquement les 10 ou 20 premières), et les suivantes sont utilisées pour l'évolution des marqueurs.
- Dans le second cas, nous fixons la dynamique de sélection de plans comme étant plus rapide que la dynamique d'évolution des marqueurs. Ceci revient à fixer la valeur de ε de manière à ce que $\mathbf{w}$ ait convergé en peu d'annotations (typiquement les 10 ou 20 premières, ce qui correspond à $\varepsilon \simeq 0.1$), et les valeurs de α, β et γ de sorte qu'il faille plusieurs sessions pour bien apprendre un plan (typiquement $\alpha \simeq 0.99$, et $\beta = \gamma \simeq 0.5$). Ainsi, la convergence des marqueurs se fera sur plusieurs

sessions, alors que celle de la fonction de sélection aura converger avant la fin de la session.

6.4 Conclusion

Dans ce chapitre, nous avons défini notre contexte d'apprentissage à long-terme, visant l'amélioration de la sélection des sites pertinents lors d'une nouvelle recherche en se basant sur l'information obtenue par interaction avec l'utilisateur accumulée lors des sessions passées. Nous avons remarqué que c'était un problème d'apprentissage singulier, puisque l'on ne dispose pas d'une information explicite permettant de regrouper les sessions entre elles.

Nous avons proposé une nouvelle architecture basée sur nos précédents travaux qui dispose d'un ensemble de plans. Ces plans sont associé à des concepts, et permettent d'estimer la pertinence de chaque site. Notre problématique est alors divisée en deux parties : il s'agit tout d'abord de sélectionner le plan relatif au concept recherché lors de la session courante, puis d'optimiser les estimations de la pertinence de chaque site portées par ce plan.

Nous avons proposé de résoudre ces deux problèmes par apprentissage en se basant sur les annotations fournies par l'utilisateur lors du bouclage de pertinence. Ceci porte à trois le nombre de tâche d'apprentissage effectuées par notre système :

- La sélection de plan.
- L'évaluation de la pertinence des sites.
- L'apprentissage de la fonction de similarité.

L'apprentissage de la fonction sélectionnant le bon plan et l'apprentissage de la mesure de similarité s'effectuent sur au cours de la session, tandis que l'apprentissage de la pertinence des sites est continué lors des sessions utilisant le même plan.

L'implémentation d'une telle architecture a été mise en œuvre à l'aide d'un système multi-agents similaire à celui développé dans la partie II, mais utilisant plusieurs niveaux de marqueurs (plusieurs *"phéromones"*), chacun de ces niveaux correspondant à un plan. La fonction de sélection de plan correspond alors à la sélection d'un niveau de marqueur, et est partagée entre tous les agents. Le paramétrage de cette fonction se fait alors par une exploration statistique des plans par l'ensemble de la population des agents, jusqu'à trouver le plan pertinent. Une fois le niveau sélectionné, les marqueurs sont renforcés de la même manière qu'au chapitre 4.

Nous avons souligné que tous les algorithmes d'apprentissage mis en œuvre se faisaient sur la base de la même information donnée par l'utilisateur lors du bouclage de pertinence, et donc qu'il y avait des phénomènes de dynamique entrant en jeu entre ces différents niveaux d'apprentissage (sélection des plans, apprentissage des plans, apprentissage de la mesure de pertinence).

En perspective, il serait intéressant d'étudier sous quelles conditions il est possible d'établir une convergence en probabilités de l'ensemble du système.

Chapitre 7

Expériences sur le long-terme

Dans ce chapitre, nous présentons les expériences que nous avons conduites sur le système décrit au chapitre précédent. Ce système possède un grand nombre de paramètres. Nous décrivons dans un premier temps le protocole expérimental commun à tous nos tests. Puis, nous nous attachons à observer l'influence du nombre de plans, P', afin de régler ce paramètre pour les expériences suivantes. Nous établissons ensuite trois réseaux de tests de difficultés croissantes, afin d'isoler et de tester différents aspects du système. Ces trois protocoles servent par la suite à tester le gain apporté par le long-terme, l'apprentissage des plans et l'apprentissage de la fonction de sélection. Enfin, nous étudions dans une dernière partie l'aspect dynamique de ce système. Nous nous intéressons à la stabilité des associations entre plans et catégories, en analysant les évolutions temporelles des marqueurs.

7.1 Protocole expérimental

Pour toutes les expériences, la base d'images utilisée est la même, à savoir la base *TrecVid'05* présentée au chapitre 5. Les signatures utilisées sont aussi les mêmes (32 couleurs et 32 textures). Dans le cadre du long-terme, les marqueurs n'ont bien sûr pas été réinitialisés au début de chaque nouvelle session. En revanche, la fonction de sélection, qui est dépendante de la catégorie recherchée, a été ré-initialisée à chaque session de sorte que la probabilité de choisir un plan soit uniforme sur les plans.

Pour chaque catégorie testée, les paramètres du système sont les suivants :

- p agents mobiles utilisés.
- q images ramenées par chaque agent grâce à la stratégie active locale *uniforme* (*cf* 4.5.3).

Ces paramètres ont été fixés de manière empirique suite à différents tests à $p = 8$ et $q = 2$, c'est-à-dire aux mêmes valeurs que dans la partie II. La stratégie *uniforme* a été sélectionnée pour sa rapidité supérieure à la stratégie *gaussienne*, tout en offrant des résultats similaires.

Les paramètres des sessions sont les suivants :

- l images annotées par session.
- L sessions effectuées pour chaque catégorie testée.
- *pos* images positives dans la requête initiale d'une session.
- *neg* images négatives dans la requête initiale d'une session.

Les paramètres l et L ont été fixés à $l = 100$ et $L = 70$ de manière à disposer d'un grand nombre d'échantillons pour observer l'évolution sur le long terme. La requête initiale correspond à $pos = 2$ (respectivement $neg = 2$) images tirées au hasard dans la catégorie recherchée (respectivement dans le reste des images disponibles). Pour chaque nouvelle session, la catégorie recherchée est tirée uniformément sur l'ensemble des catégories à tester.

7.2 Test préliminaire et paramétrage de P'

Nous présentons dans un premier temps une expérience préliminaire destinée à paramétrer le nombre de plans P'. Pour ce faire, nous reprenons un protocole proche de celui utilisé en 5.5.

Nous utilisons un réseau composé de quatre destinations, sur lesquelles des catégories de la base *TrecVid'05* ont été réparties, comme présenté sur la figure 7.1. La répartition des catégories a été choisie de manière à minimiser le recouvrement des catégories entre les différents sites.

Une catégorie supplémentaire, *entertainment*, à également été divisée en quatre et répartie aléatoirement sur les destinations. Elle contient des images d'émissions de loisirs, c'est à dire principalement du sport, des jeux télévisés, etc. C'est donc un catégorie très variée qui possède un grand recouvrement avec les autres catégories. Elle est ajoutée afin de compliquer la recherche. En effet, une image d'*entertainment* peut assez fréquement recevoir une annotation positive lorsque l'on cherche une autre catégorie. Outre le fait d'obtenir ainsi un contexte plus réaliste (c'est à dire avec des catégories plus mélangées), cela permet de tester la robustesse du système face aux annotations de l'utilisateur (et aux imprécisions que celui-ci peut faire).

50 sessions de 100 annotations ont été effectuées pour chaque catégorie testée, et pour chaque nouvelle session, le concept recherché a été sélectionné de manière aléatoire.

7.2.1 Paramétrage du nombre de plans

Afin d'analyser la spécialisation des plans au cours du temps, nous proposons une visualisation qui consiste à projeter les valeurs des marqueurs de chaque destination appartenant à un même plan sur un espace à deux dimensions. Les coordonnées sont calculées à l'aide de la formule suivante (N étant le nombre de destinations) :

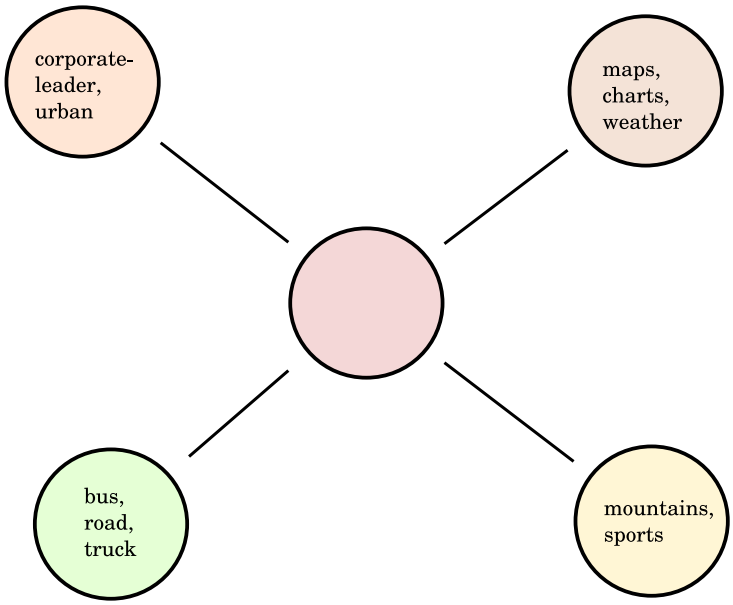


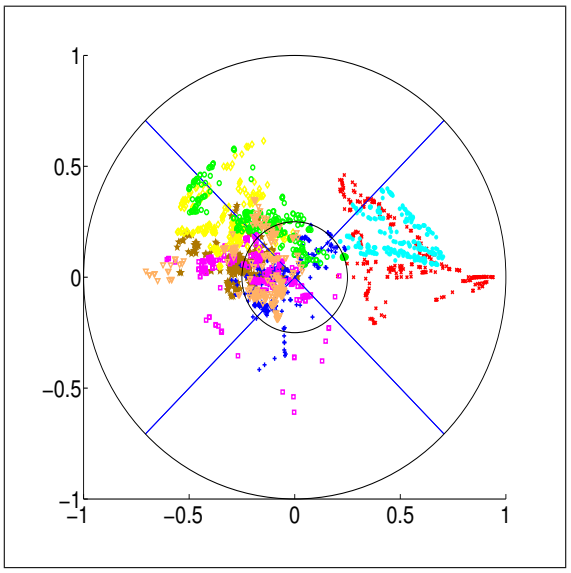
Fig. 7-1. Topologie du test *préliminaire*. Il y a quatre destinations sur lesquelles les catégories sont réparties. La catégorie entertainment a été divisée en quatre parties réparties sur chacune des bases.

$$(x_j, y_j) = (\sum_{i=0}^{N-1} \mathbf{m}_i[j] \cdot \cos(\frac{i \cdot 2\pi}{N}), \sum_{i=0}^{N-1} \mathbf{m}_i[j] \cdot \sin(\frac{i \cdot 2\pi}{N})) \tag{7.1}$$

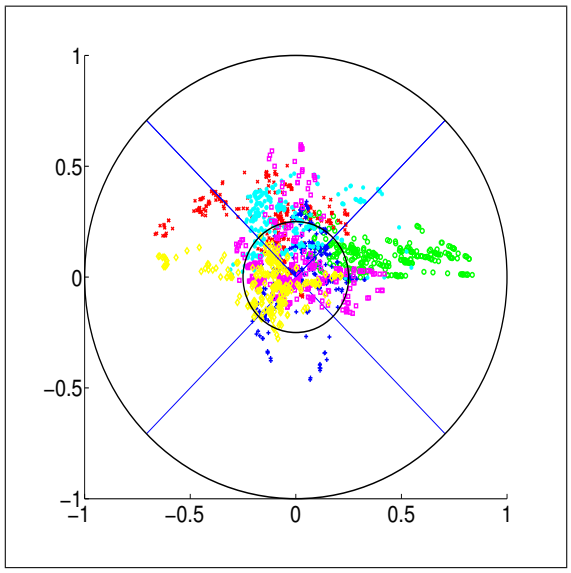
Ainsi, le cercle unité est découpé en secteurs angulaires, chacun représentant une destination. Plus un point est éloigné du centre, plus la probabilité que le plan associé mène à la destination associée au secteur est forte. Pour construire un graphique, un point pour chacun des plans est ajouté à la fin de chaque session, ce qui permet de voir la distribution des plans sur les destinations pour l'ensemble des sessions de l'expérience. Dans ces graphiques, chaque plan est représenté par une couleur différente. La distribution de la spécialisation d'un plan sur l'ensemble de l'expérience correspond donc à un ensemble de points d'une même couleur.

Nous faisons varier le paramètre P' de 2 à 8, alors que le nombre de catégories est $P = 10$ (figure 7-2). Nous limitons volontairement le nombre de plans, car les catégories présentes sur une même base sont indissociables du point de vue de la localisation (elle peuvent alors partager un même plan). Les tests à 6 et 8 plans ont menés à une bonne spécialisation des plans, permettant d'atteindre n'importe quelle destination en choisissant le bon plan. Les tests à 4 et 2 plans montrent, à l'inverse, que les plans ne se sont pas ou peu spécialisés.

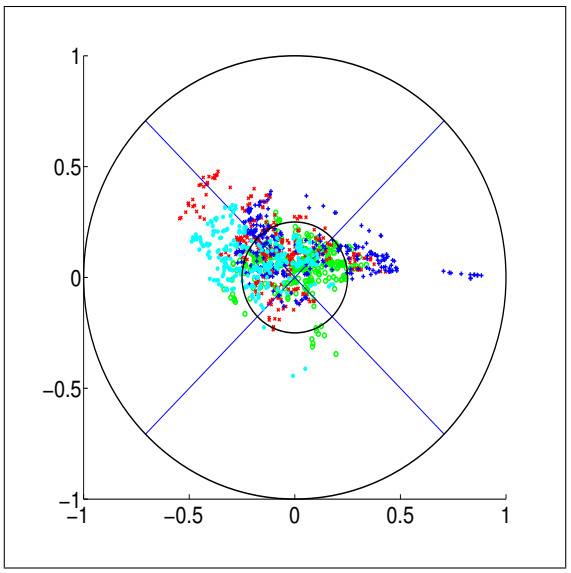
La plus faible spécialisation vers la quatrième destination contenant les catégories *sports* et *mountains* peut s'expliquer par le fait que les images de *sports* se trouvent aussi dans la catégorie *entertainment* pour 50% d'entre-elles, permettant aux autres destinations de répondre positivement à cette requête.



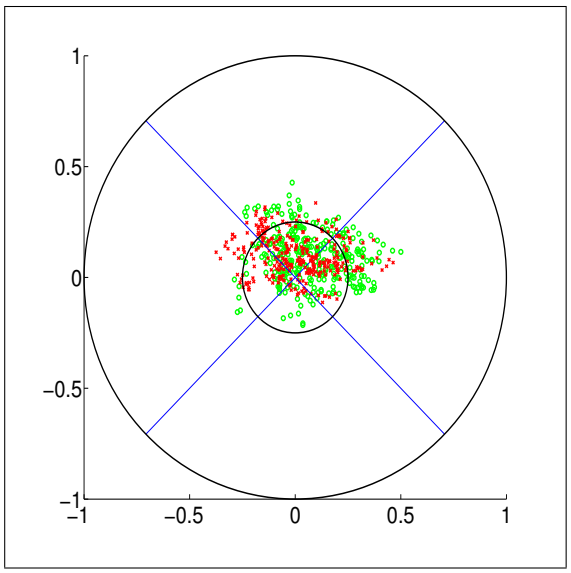
(a) Distribution des marqueurs pour le test à 8 plans.



(b) Distribution des marqueurs pour le test à 6 plans.



(c) Distribution des marqueurs pour le test à 4 plans.



(d) Distribution des marqueurs pour le test à 2 plans.

Fig. 7-2. Distribution des marqueurs pour P' allant de 2 à 8 lors du test préliminaire. Chaque plan est représenté par une couleur différente (de deux à huit couleurs, donc), et l'association entre couleur et plan n'est pas marquée pour des questions de lisibilité des figures. Les plans ne se spécialisent que lorsqu'ils sont en nombre supérieur (6 et 8) au nombre de destinations possibles (4).

On observe alors qu'il semble nécessaire d'avoir un nombre de plans supérieur au nombre de destinations. Ceci nous permet de paramétrer le système à P' légèrement supérieur au nombre de destinations par la suite.

7.3 Réseaux de tests

7.3.1 Réseau 1 : configuration simple sans bruit (SSB)

Le premier protocole de test simple consiste à offrir aux agents trois destinations, chacune ne contenant que les images d'une catégorie bien spécifique parmi *airplanes*, *maps* et *explosion fire*. Ces catégories n'ont aucune image en commun. Ainsi, un agent qui se déplace vers la bonne base ramène nécessairement une image qui va être annotée positivement, et, à l'inverse, un agent se déplaçant vers une mauvaise base ramènera nécessairement une image qui sera annotée négativement. L'apprentissage des routages va donc dépendre des labels indépendamment de la qualité de la mesure de similarité.

Le paramètre P' est fixé à $P' = 6$ suite à l'étude faite en 7.2.

7.3.2 Réseau 2 : configuration simple bruité (SB)

Le protocole de test que nous avons appelé *"simple bruité"* est une extension du précédent dans laquelle 4000 images de la catégorie *entertainement* ont été ajoutées sur chaque destination. Cette classe est très volumineuse et possède une très grande variabilité visuelle. Pour ces raisons, elle nous a semblé pertinente afin de tester la robustesse du système par rapport à l'apprentissage de la fonction de similarité. Ainsi, un agent arrivant à bonne destination n'est pas certain de ramener une image qui sera annotée positivement. L'apprentissage des routages dépend donc de la qualité de la fonction de similarité et de la capacité de la stratégie de sélection à trouver des images appartenant à la catégorie recherchée.

Le paramètre P' est fixé à $P' = 6$ suite aux observations faites en 7.2.

7.3.3 Réseau 3 : configuration complexe (C)

Le protocole de test dit *"complexe"* consiste à offrir aux agents cinq destinations contenant plusieurs catégories d'images, comme l'illustre la figure 7.3. La répartition des images est complexe, les catégories pouvant être réparties sur plusieurs bases, dans des proportions différentes.

hosts	catégories
host 1	bus, charts, road, urban 10%
host 2	corporate-leaders, road, urban 30%
host 3	mountains, sports, truck
host 4	urban 60%, weather
host 5	mountains, sports

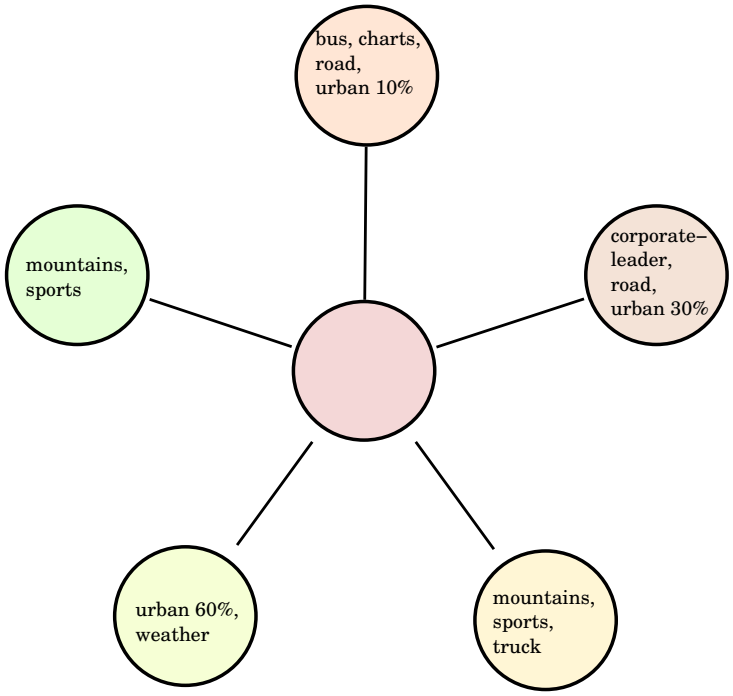


Fig. 7-3. Topologie du réseau pour la configuration *complexe*. Il y a cinq destinations possibles sur lesquelles les catégories sont réparties de manière complexe. Certaines destinations partagent des catégories dans des proportions qui peuvent aller de cinquante pour cent à seulement dix pour cent.

Ce réseau permet de voir dans quelle mesure le système est capable d'apprendre des répartitions complexes des catégories sur le réseau, et de quelle manière. Le paramètre P' est fixé à 8 suite aux observations en 7.2.

7.3.4 Synthèse

Nous disposons ainsi de trois réseaux de test permettant d'évaluer différentes parties du système. Le premier réseau, avec la configuration *simple sans bruit*, nous permet de vérifier la pertinence d'un système à plusieurs plans en compétition en éliminant toutes les imperfections dues aux hypothèses (la localisation est parfaite) ou bien à l'approche de type *recherche interactive* (la sélection d'images et les annotations associées sont toujours positives pour la bonne localisation).

Le second réseau, dans la configuration *simple bruité*, nous permet d'évaluer la robustesse du système par rapport à l'apprentissage de la fonction de similarité. En effet, les annotations qui servent à renforcer les plans sont fortement dépendantes de l'apprentissage de la fonction de similarité (par le biais de la sélection des images à annoter).

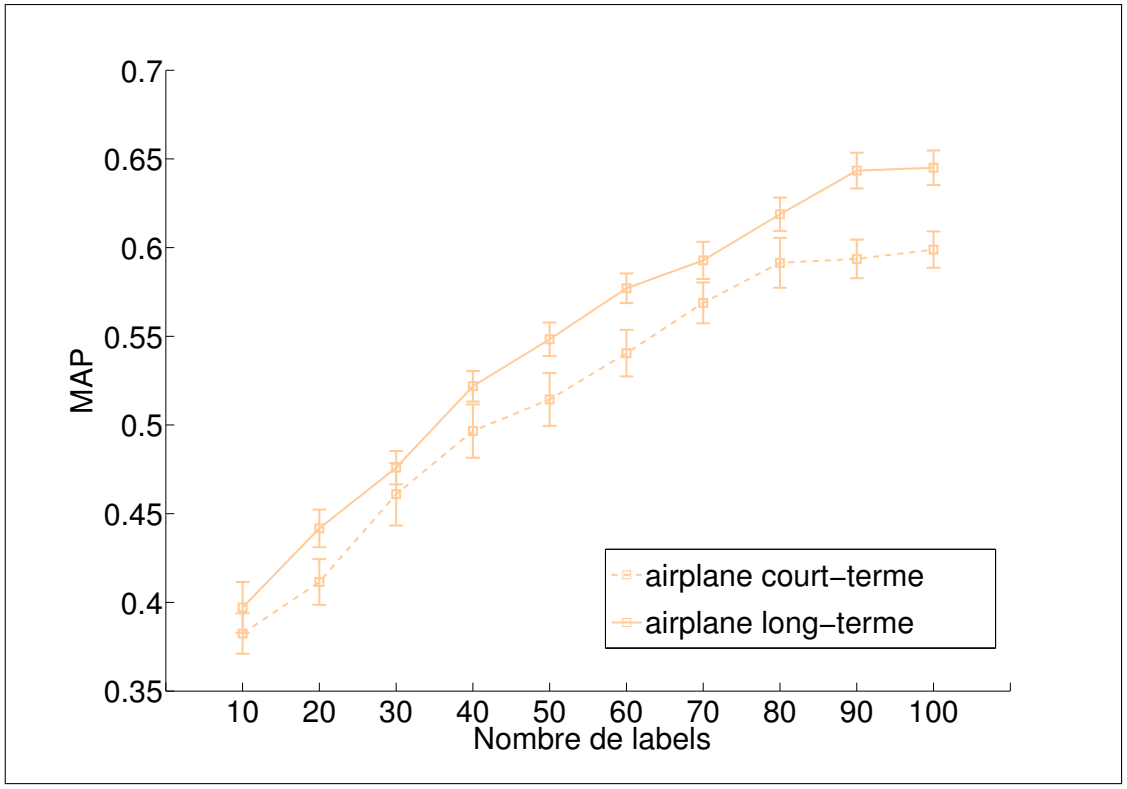
Le dernier réseau, à configuration *complexe*, nous permet d'évaluer la robutesse du système complet par rapport à l'hypothèse de distribution des images sur le réseau, en offrant une répartition complexe des catégories.

7.4 Performances du système

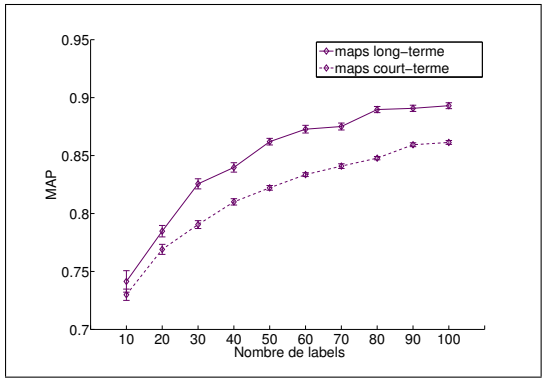
7.4.1 Évaluation intra-session

Nous mesurons le gain de performances du système, ce qui était notre première motivation à l'utilisation d'un contexte d'apprentissage à long terme. Nous calculons un MAP à chaque fois que 10 labels ont été obtenus et traçons l'évolution de ce MAP au cours de la session. Nous comparons les courbes de progression entre une approche à court terme et une fois l'optimisation à long-terme effectuée. Nous utilisons pour cela le réseau 2 (SB) à cause des temps de calcul nécessaires à l'obtention d'un MAP toutes les dix annotations pour chaque session (le nombre de MAP a calculer est alors de $P \times L \times \frac{l}{10} = 3000$). Ces calculs ont duré environ deux jours. Le réseau 3 (C) aurait été intéressant à tester, mais il contient 7 catégories de plus à (soit $P \times L \times \frac{l}{10} = 10000$ MAP à calculer), mais aussi un nombre plus important d'images à traiter lors d'un MAP (allongeant le temps de calcul de celui-ci), rendant les temps de calculs trop importants.

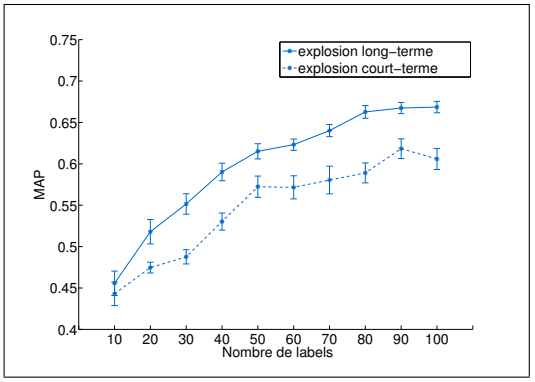
Pour les trois catégories présentées, *airplane*, *maps* et *explosion*, les résultats en fin de session avec l'optimisation à long-terme sont meilleurs (figure 7.4). De plus, la progression est plus rapide. Cela veut dire que pour obtenir un résultat équivalent à celui obtenu avec 100 labels sans le long-terme, il ne suffit que de 70 labels avec le long-terme pour la catégorie *airplane*. En quelque sorte, le long-terme accélère la stratégie active.



(a) *MAP* pour la catégorie *airplane*. En terme de gain en nombre de labels, le *MAP* obtenu avec le long-terme est le même avec environ trente annotations en moins que le court-terme.



(b) *MAP* pour la catégorie *maps*. En terme de gain en nombre de labels, le *MAP* obtenu avec le long-terme est le même avec environ cinquante annotations en moins que le court-terme.



(c) *MAP* pour la catégorie *explosion*. En terme de gain en nombre de label, le *MAP* obtenu avec le long-terme est le même avec environ cinquante annotations en moins que le court-terme.

Fig. 7-4. *MAP* en fonction du nombre de labels donnés par l'utilisateur pour les stratégies long-terme et court-terme. Le *MAP* augmente avec le nombre de labels, et le long-terme obtient à nombre de labels équivalent un meilleur *MAP*.

7.4.2 Évaluation globale sur *TrecVid'05*

Nous comparons maintenant les performances de notre système avec celles obtenues lors de la partie II. Nous nous attendons à ce que les résultats soit meilleurs que le système à un seul plan utilisé à court terme, mais moins bons que celui-ci utilisé dans un cas particulier (et très simple) de long terme présenté en 5.5.

La figure 7-5 montre les valeurs de *rappel* pour les dix catégories testées avec P' valant 6 ou 8 pour le réseau défini en 7.2. Les résultats sont conformes à nos attentes et illustrent le gain apporté par le long-terme, c'est à dire qu'ils se situent entre les valeurs du système à court terme présenté dans la partie II et ce même système utilisé dans un cas particulier du long terme.

Les valeurs de *rappel* ainsi présentées sont à nuancer légèrement, car seule la catégorie *entertainment* a été utilisée comme bruit (au contraire du tout le reste de la base en 5.5), ce qui rend la tâche plus facile. En effet, un découpage en quatre parties homogènes et quasi-orthogonales de la base était impossible. Ceci permet d'expliquer les très bons résultats du système à huit plans.

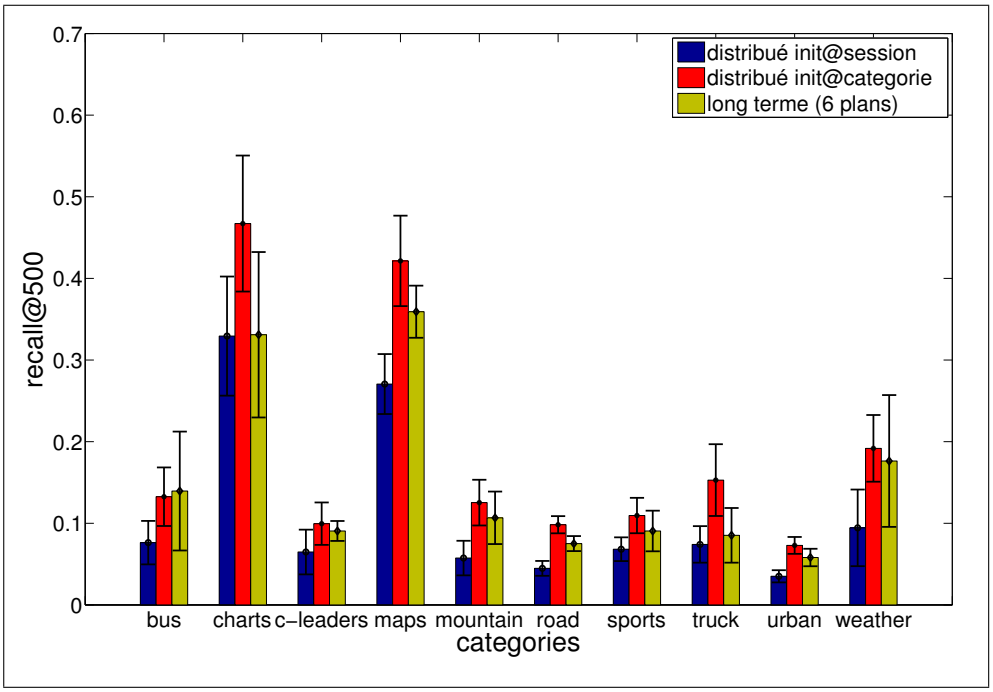
7.5 Apprentissage des plans

Nous observons maintenant la valeur relative des marqueurs (probabilité de saut liée au marqueur) sur chacune des destinations afin de voir si certains marqueurs se sont spécialisés. Nous avons deux visualisations nous permettant d'analyser les résultats. La première correspond à celle présentée lors des tests préliminaires en 7.2. La deuxième visualisation consiste à tracer pour chaque destination la probabilité d'y être mené par chaque plan. C'est en fait la moyenne des probabilités de saut qui est représentée, afin de voir, en moyenne, quel plan correspond à quel routage.

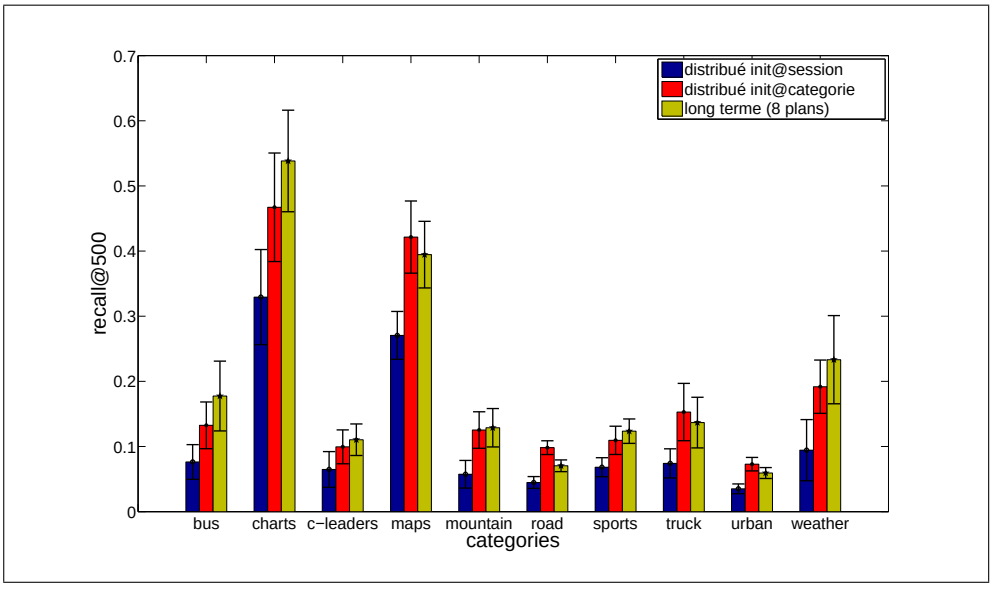
7.5.1 Résultats sur le réseau 1 (SSB)

La figure 7-6 présente la distribution des marqueurs sur les trois secteurs des destinations du réseau 1. Chacun des six plans disponibles s'est spécialisé vers l'une des trois destinations. Le premier plan a légèrement oscillé entre les destinations 1 et 3 mais s'est spécialisé vers la deuxième machine la majeure partie du temps. Les points s'accumulent majoritairement sur les trois cadrans dans des zones proches du cercle unité, ce qui signifie que le routage produit par les marqueurs a été très bien appris et que les probabilités de saut sont très élevées.

Nous avons également tracé sur la figure 7-7 les moyennes de probabilités de saut pour chacune des trois destinations. La première ligne correspond ainsi aux probabilités associées au vecteur $\mathbf{m}_1$ (*cf* 6.3). On retrouve de même les probabilités associées à $\mathbf{m}_2$ et $\mathbf{m}_3$ sur les lignes suivantes.



(a) Comparaison du *rappel* entre le système long-terme à 6 plans et les résultats du chapitre 5.



(b) Comparaison du *rappel* entre le système long-terme à 8 plans et les résultats du chapitre 5.

Fig. 7-5. Valeur de *rappel* pour le test à 6 et à 8 plans (*long terme* sur les figures). Les valeurs obtenues pour un protocole proche lors de la partie II sont présentées à titre de référence. Le système à plans multiples offrent de bons résultats situés entre le système à un seul plan utilisé à court terme (*init@session* sur les figures), et ce même système utilisé dans un cas particulier du long terme (*init@catégorie* sur les figures).

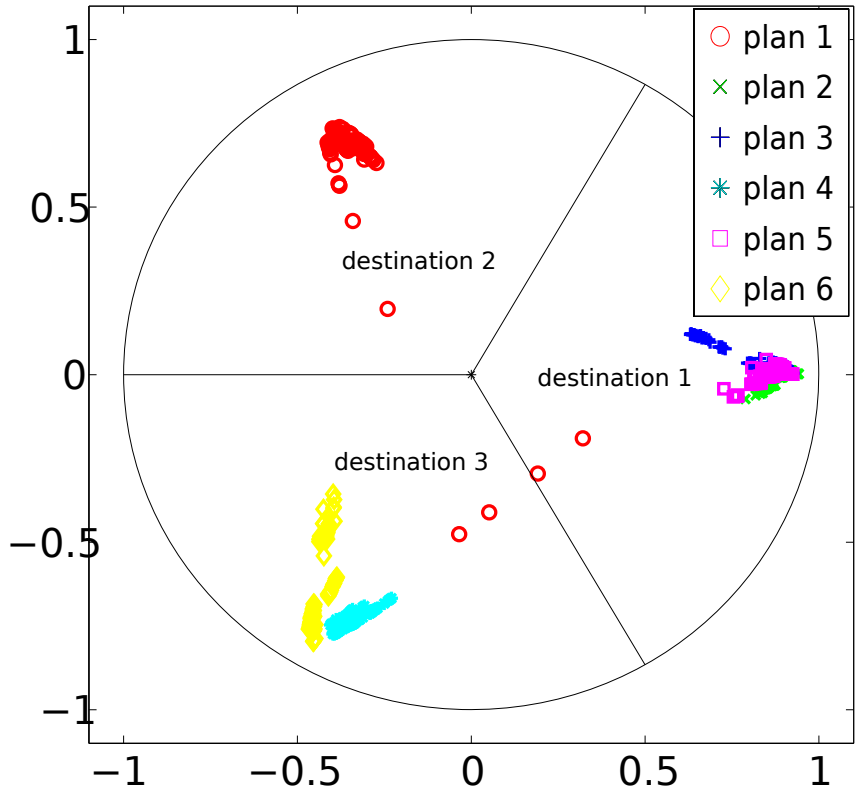


Fig. 7-6. Distribution des relevés de marqueurs sur les trois destinations pour le réseau 1 (*simple sans bruit*). Chacun des six plans se spécialise vers l'une des trois destinations et reste spécialisé dans celle-ci pour toute la durée de l'expérience.

Cette représentation complète celle de la figure 7.6. On peut y observer que les probabilités de saut dépassent 0.9 pour chacun des plans. On voit que le premier plan donne une forte probabilité d'aller vers la seconde destination, ce qui est cohérent avec l'accumulation observée en 7.6.

Cette expérience valide notre architecture en montrant que plusieurs plans en compétition peuvent se spécialiser sur un concept et mener à des routages différents et cohérents avec la distribution des données.

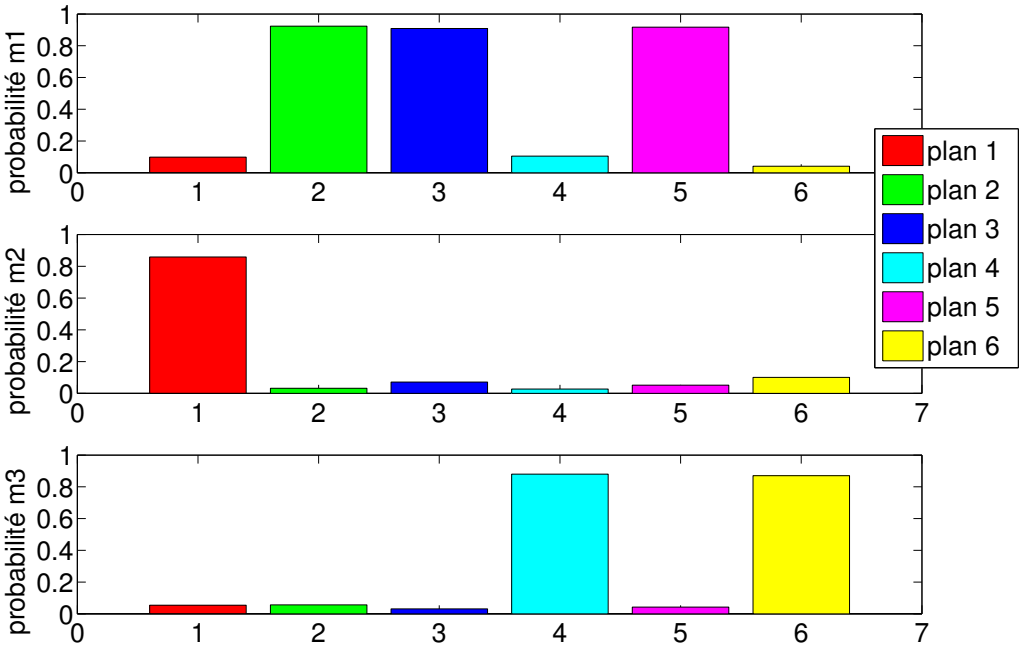


Fig. 7.7. Valeur moyenne des probabilités de saut vers chacune des trois destinations pour chacun des six plans dans le réseau 1 (SSB). On voit que le premier plan donne une forte probabilité d'aller vers la seconde destination. La moyenne des probabilités de saut est particulièrement élevée pour chacun des plans, ce qui montre que la localisation est bien apprise.

7.5.2 Résultats sur le réseau 2 (SB)

Dans le test sur le réseau 2 (SB), l'apprentissage des marqueurs dépend de l'apprentissage de la fonction de similarité. En effet, si la fonction est mauvaise et que la stratégie de sélection ne renvoie que des images annotées négativement, alors il est impossible d'apprendre quoi que ce soit. La stratégie de sélection (et par extension la mesure de similarité) n'est pas mauvaise puisque déjà validée dans la partie II, mais cependant, elle ne renvoie pas des résultats parfaits comme dans le test *simple*. En effet, il y a des annotations négatives pour la base contenant la catégorie recherchée et des annotations positives pour

les bases ne contenant pas la catégorie (c'est dans ce sens que l'on appelle ce test *bruité*, ces labels venant s'ajouter comme du bruit par rapport au signal de renforcement pur du test précédent). En somme, ce test permet de vérifier la robustesse de notre système par rapport au signal de renforcement.

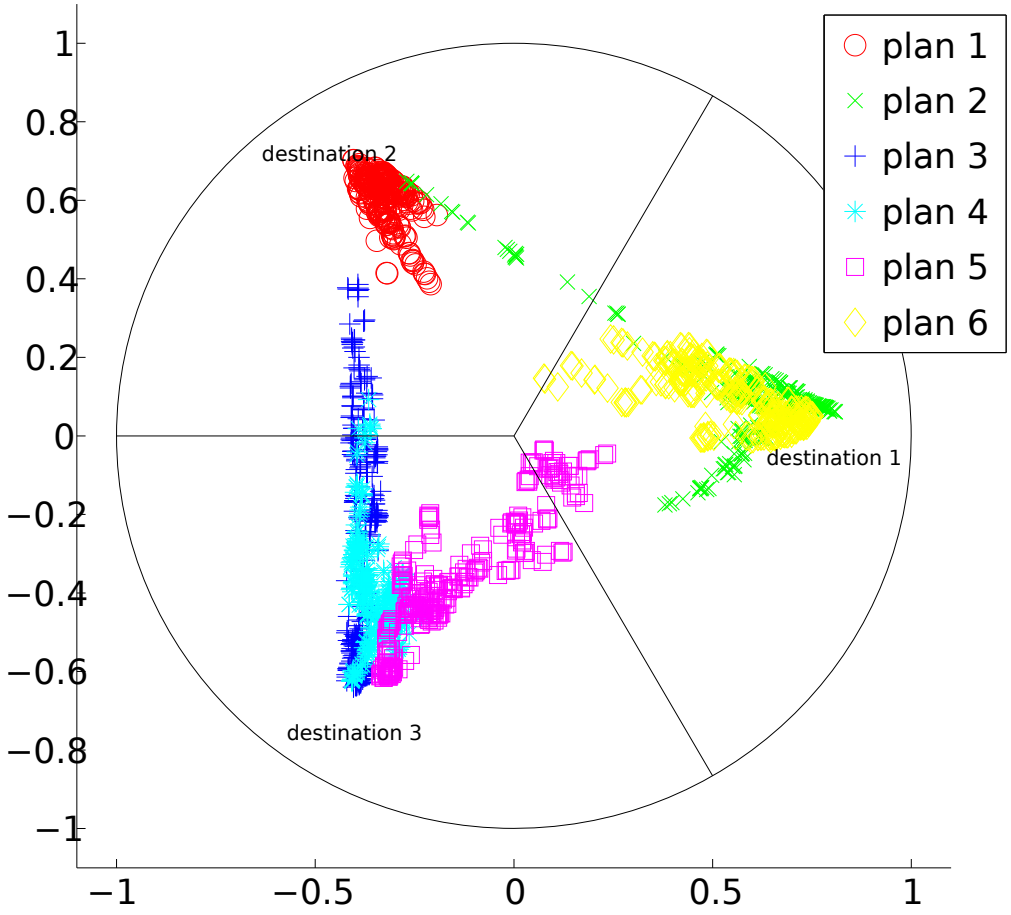


Fig. 7-8. Distribution des relevés de marqueurs sur les trois destinations pour chacun des six plans lors du test *simple bruité*. La concentration des plans est moins marquée que pour le test *simple*, cependant, on remarque que tous les plans se sont spécialisés vers l'une des trois destinations.

La figure 7-8 présente la distribution des marqueurs sur les trois destinations. On remarque que l'apprentissage des routages est satisfaisant. Même si les distributions des marqueurs sont moins concentrées qu'au test précédent sur le réseau 1, et que les points se rapprochent moins du cercle unité, tous les marqueurs se sont spécialisés.

La figure 7-9 présente la moyenne des probabilités de saut pour chaque destination et pour chaque plan. En moyenne, les probabilités de saut sont moins élevées que pour le

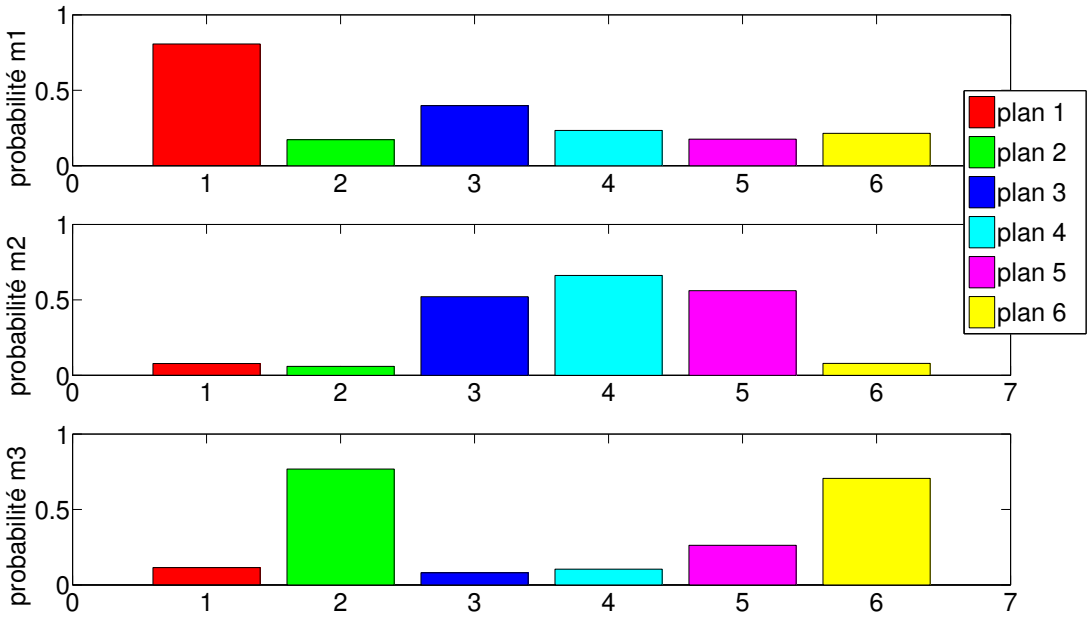


Fig. 7-9. Moyenne des probabilités de saut vers chacune des trois destinations pour les six plans lors de l'expérience *simple bruité*. Par exemple, le second plan a une probabilité majoritaire de diriger les agents vers la troisième machine.

test sur le réseau 1 (SSB), même si elles sont supérieures à plus de deux fois la probabilité uniforme ($2 \times \frac{1}{3}$ ici). Ces valeurs des moyennes des probabilités sont suffisamment élevées pour considérer que l'apprentissage de la localisation des trois catégories est validé.

Ceci nous permet de valider le bouclage entre l'apprentissage de la mesure de similarité et l'apprentissage des plans.

7.5.3 Résultats sur le réseau 3 (C)

Le test *complexe* diffère du précédent en ceci que les routages à apprendre sont plus difficiles. Il nous permet de tester la robustesse du système face à la distribution des données.

La figure 7-10 montre la distribution des marqueurs sur les secteurs angulaires des cinq destination du réseau 3 (C). Certains marqueurs se sont bien spécialisés dans une seule destination (par exemple les plans 4 et 6), alors que la destination 4 n'a pas été routée par un plan en particulier. Globalement, tout les marqueurs sont bien au-delà du cercle de probabilité uniforme (0.2 pour ce réseau), ce qui montre que les plans ont bien été appris.

La figure 7-11 nous montre la répartition moyenne des marqueurs sur les différentes catégories. Certains marqueurs se sont spécialisés vers plusieurs destinations, comme par exemple le plan 7 menant majoritairement vers la destination 4, un peu moins vers la

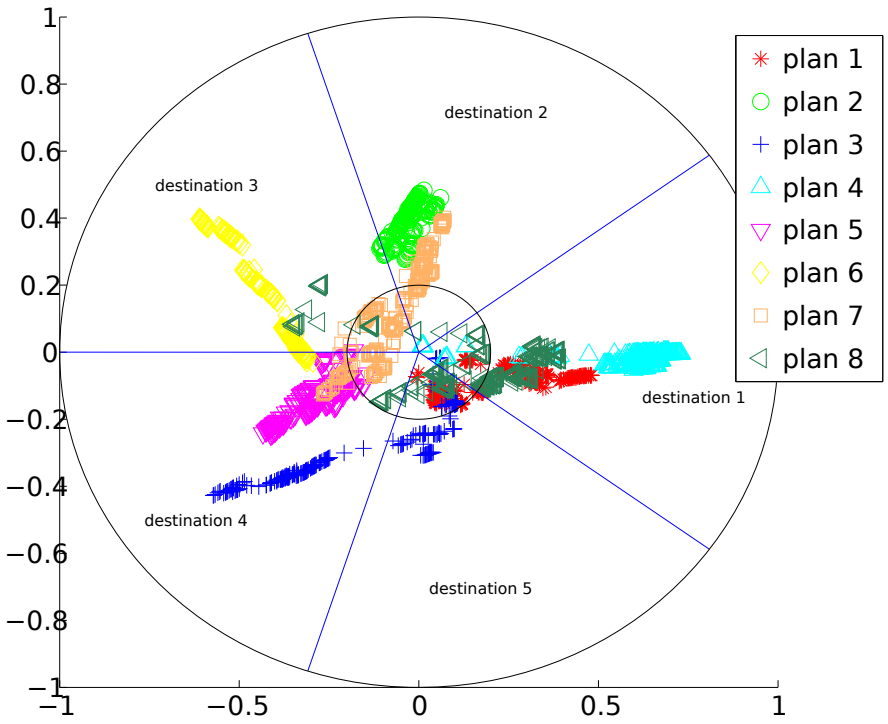


Fig. 7-10. Distribution des marqueurs pour chacun des huit plans pour le test complexe. Certains plans se sont bien spécialisés vers une destination (les plans 2, 3, 4 et 6 par exemple), même si la destination 5 semble avoir été ignorée par tous les plans.

destination 2 et encore un peu moins vers la destination 1, dans des proportions qui correspondent à la distribution de la catégorie *urban* sur le réseau.

Le plan 6 a routé majoritairement vers le site 3 et un peu vers le site 5. Ce résultat peut s'expliquer de la façon suivante : ces deux destinations contiennent les mêmes catégories, à ceci près que 3 posséde en plus la catégorie *truck* uniquement disponible à cet endroit. Du coup, la destination 3 offre des résultats toujours au moins aussi satisfaisants que la destination 5 pour n'importe quelle requête.

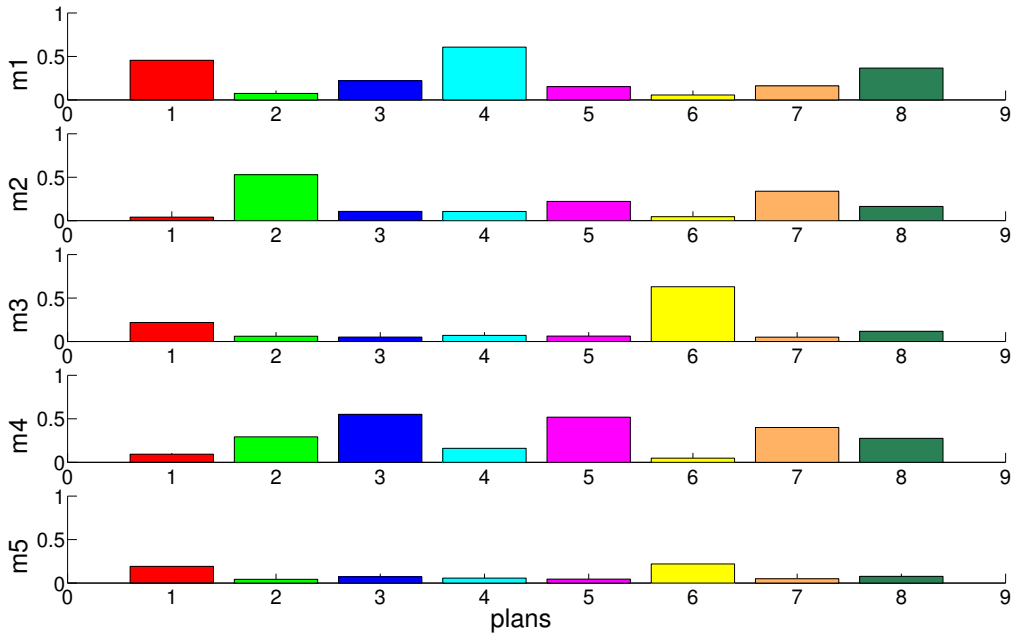


Fig. 7-11. Moyenne des probabilités de saut vers chacune des cinq destinations pour chacun des 8 plans sur le réseau 3 (C). Certains plans se sont spécialisés vers une seule destination (2, 3, 4 et 6), d'autres vers plusieurs (5 et 7). Les valeurs moyennes sont suffisamment élevées pour considérer que l'apprentissage des localisations est correct, sauf pour la cinquième destination qui n'a aucun plan ayant une probabilité élevée de déplacer les agents vers elle.

7.5.4 Conclusion sur l'apprentissage de plans

Ces expériences sur les trois réseaux de test nous permettent de valider l'apprentissage des plans. Cet apprentissage est moins marqué quand la sélection des images à annoter est plus difficile (réseau 2) et quand la localisation des catégories est plus complexe (réseau 3). Cependant, les destinations pertinentes obtiennent toujours un ou plusieurs plans se spécialisant vers elles.

7.6 Apprentissage de la fonction de sélection

Nous nous concentrons à présent sur l'apprentissage de la fonction de sélection de plan. Nous utilisons pour cela les réseaux 1 (SSB) et 2 (SB), et nous relevons les valeurs des poids w_j de chaque plan à la fin de chaque session pour en faire la moyenne.

La figure 7-12 montre les valeurs moyennes en fin de session des composantes $[w_j]_{1\leq j\leq 6}$ de la fonction de sélection ψ pour chacune des trois catégories testées, en utilisant le réseau 1 (SSB). Les résultats sur ce réseau montrent que la catégorie 2 a majoritairement été associée au plan 1, la catégorie 3 a été également répartie entre les plans 4 et 6 et que la catégorie 1 a utilisé de manière égale les trois plans restant. En somme, les associations entre les plans et les catégories sont cohérents avec les routages liés aux plans vus en 7.5.

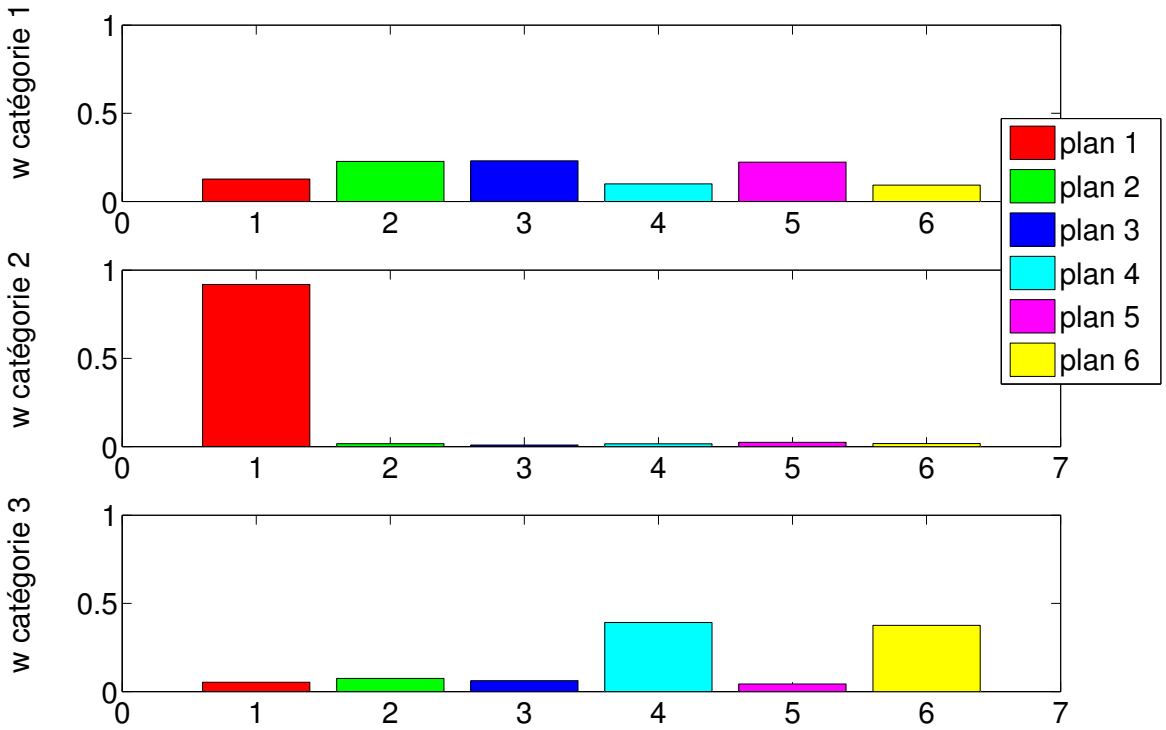


Fig. 7-12. Moyenne des probabilités d'utilisation de chacun des plans pour chacune des trois catégories lors du test sur le réseau 1 (SSB). Les plans utilisés correspondent bien à ceux menant à la machine contenant les images de la catégorie. Par exemple, la catégorie 2 a majoritairement utilisé le plan 1 qui menait bien à la seconde machine.

Pour le réseau 2 (SB) (Fig. 7-13), les associations ont été légèrement moins bien apprises. Cependant, la catégorie 2 a été bien plus associée aux plans 3, 4 et 5 qu'aux autres, par rapport à la catégorie 1 du test sur le réseau 1. Là encore, lorsque plusieurs plans sont pertinents, la probabilité de les utiliser est quasiment uniforme entre eux.

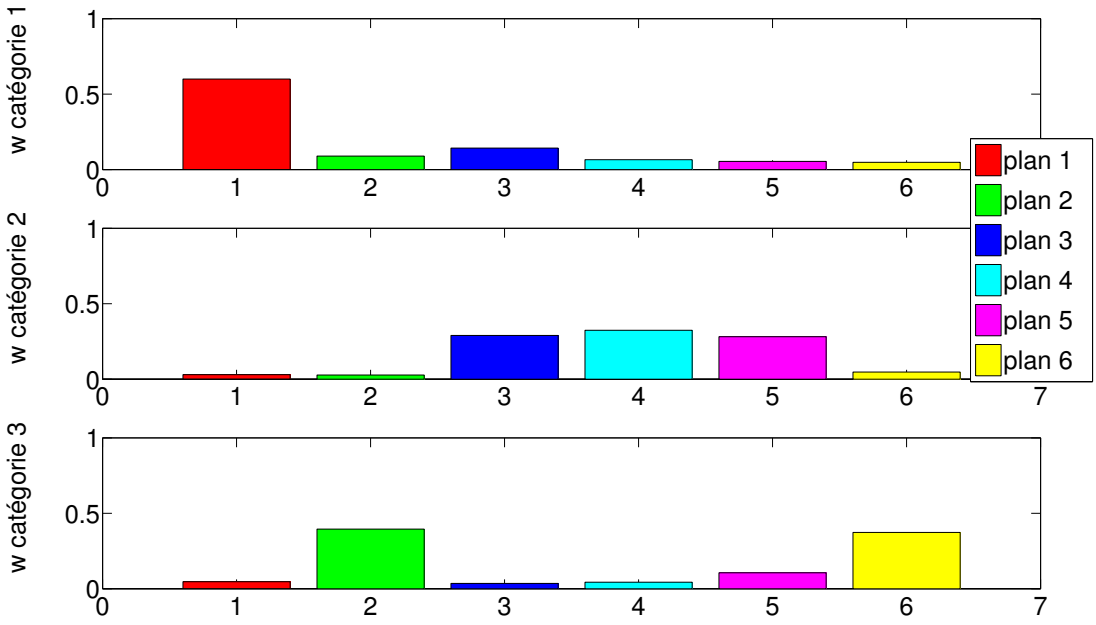


Fig. 7-13. Moyenne des probabilités d'utilisation de chacun des plans pour les trois catégories lors du test sur le réseau 2 (SB). Les probabilités d'utiliser un plan sont bien en accord avec la spécialisation des plans vers les destinations contenant les catégories pertinentes.

Ceci nous permet de valider l'apprentissage de la sélection de plan. En effet, les plans utilisés sont ceux qui ont été associés à la catégorie recherchée. Cet apprentissage est cependant moins marqué dans le cas où la sélection des images est plus difficile (cas du réseau 2).

7.7 Dynamique du système

Nous nous intéressons maintenant à la dynamique du système, c'est-à-dire à l'évolution temporelle des marqueurs. Pour cela, nous allons tracer la probabilité de saut associée à un marqueur sur une machine à la fin d'une session, en fonction de sa valeur à la fin de la session précédente (graphiques connus sous le nom de cartes de Poincaré). Pour chacune des machines, une trace est effectuée pour chaque marqueur présent sur celle-ci.

Pour ces expériences, nous utilisons les réseaux 1 (SSB) et 2 (SB).

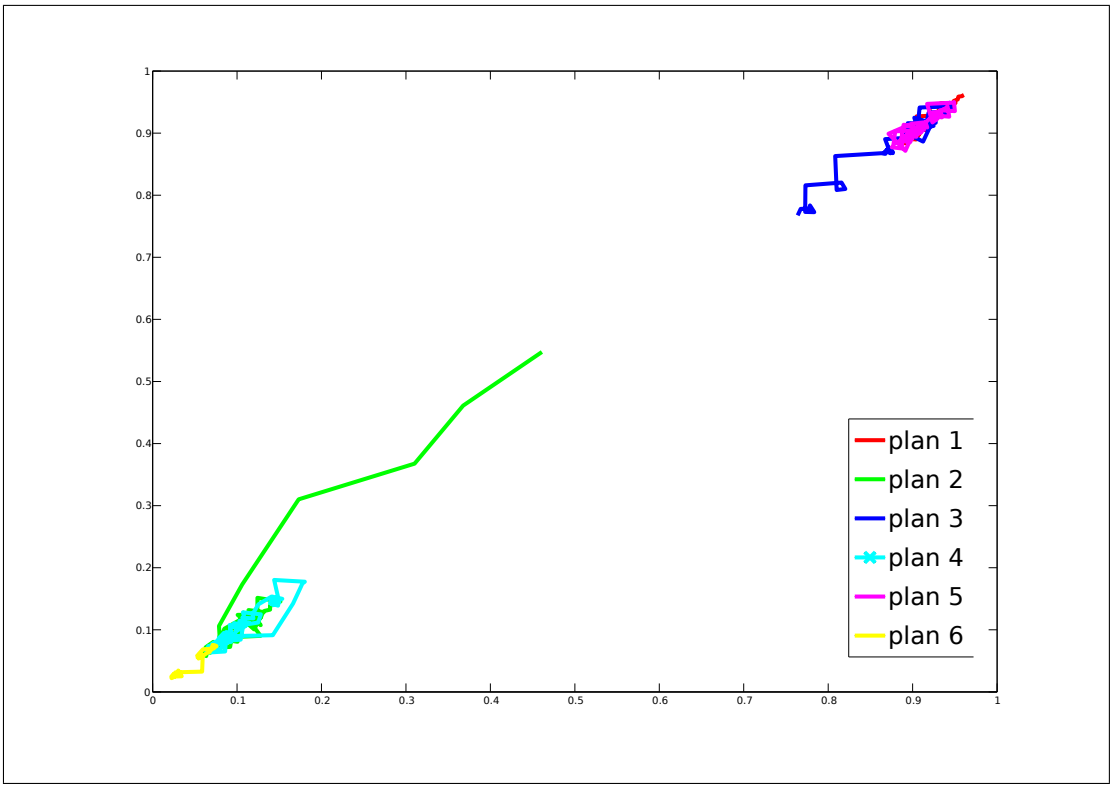
7.7.1 Dynamique sur le réseau 1 (SSB)

Nous observons ici le système dans le cadre du réseau 1 (SSB). La figure 7-14 montre les traces pour chaque marqueur, regroupées par destination. Une trace oscillant autour d'un point de ce plan montre que les valeurs du marqueur sont stables dans le temps. Une trace proche de la première bissectrice montre que les variations du marqueur d'une itération à l'autre sont faibles.

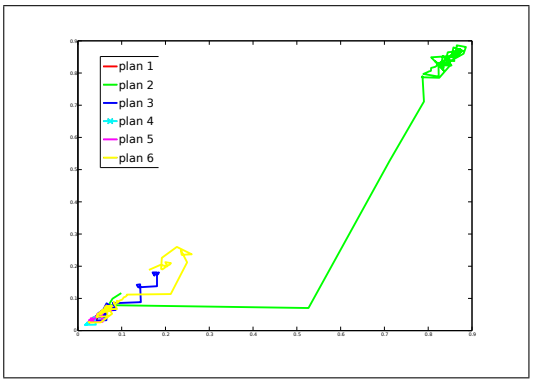
L'apprentissage des routages se montre extrêmement rapide : il ne faut que quelques itérations pour que les marqueurs soient proches des valeurs autour desquelles ils oscillent pour le reste de l'expérience. Les plans 2, 3 et 5 oscillent au delà de la zone $(0.9, 0.9)$, tout comme les autres oscillent en deçà de $(0.1, 0.1)$, ce qui montre la stabilité des routages appris dans le temps (*i.e.* un routage fort à une itération donnée reste fort à l'itération suivante). Le fait que les oscillations soient concentrées autour de la diagonale et restent de faible amplitude montre la grande stabilité de la spécialisation des marqueurs.

7.7.2 Dynamique sur le réseau 2 (SB)

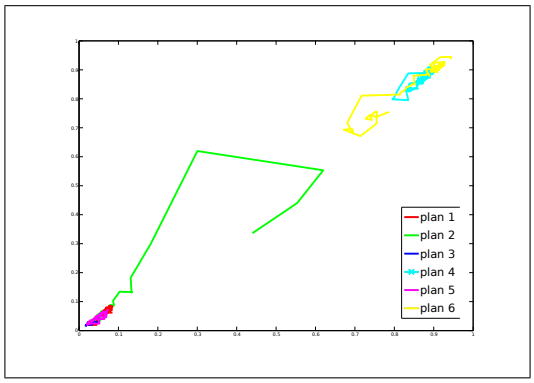
Nous observons ici le système dans le cadre du test sur le réseau 2 (SB). La figure 7-15 présente les traces pour chaque marqueurs regroupées par destination. Les traces sont très proches de la première bissectrice, ce qui veut dire que le système est lent dans ses évolutions (il n'y a pas de variations rapides des probabilités de saut). Le plan 1 tend à osciller autour de la zone $(0.8, 0.8)$, tandis que les autres marqueurs chutent vers 0, puis oscillent autour de la zone $(0.1, 0.1)$, ce qui veut dire que le routage appris par le marqueur est stable dans le temps. De même que sur le réseau 1, les oscillations étant concentrées autour de la diagonale et de faible amplitude, on en déduit la stabilité de la spécialisation des marqueurs.



(a) Trajectoires sur la première destination. Le marqueur des plans 1, 3 et 5 convergent vers le point $(0.9, 0.9)$ et oscille dans cette zone, tandis que les autres marqueurs convergent et oscillent autour de $(0.1, 0.1)$.

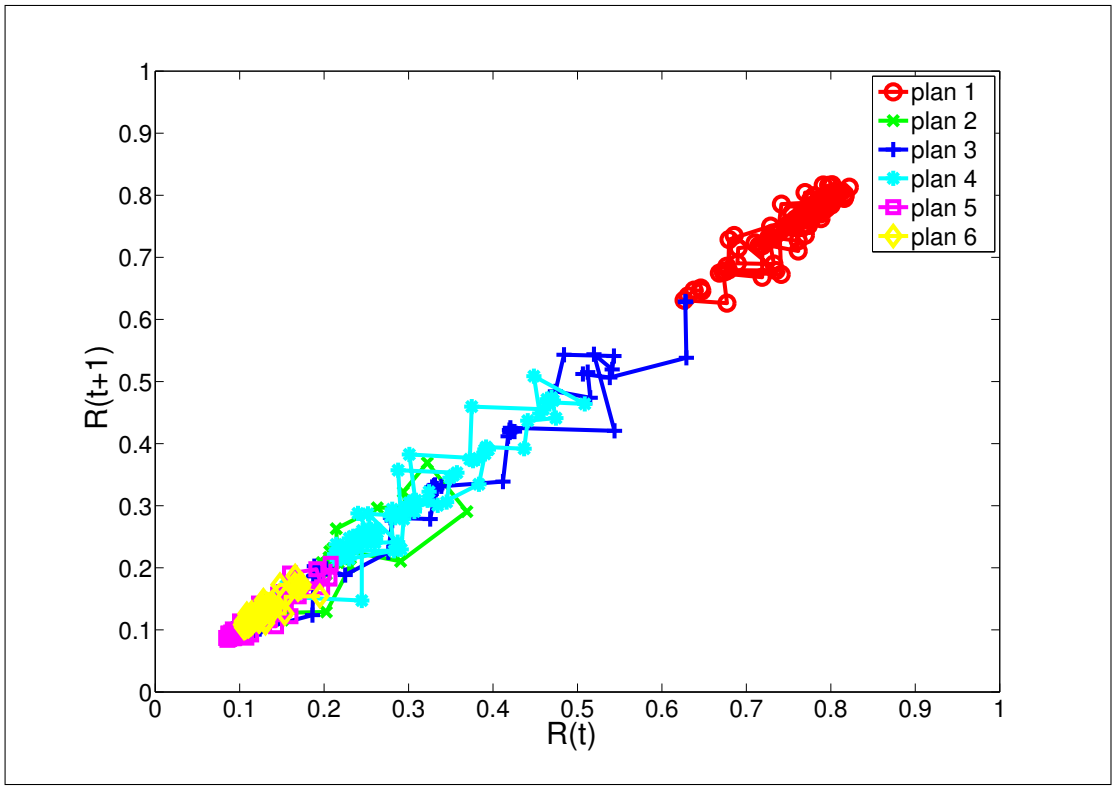


(b) Trajectoires sur la seconde destination. Le marqueur du plan 2 converge très rapidement vers le point $(0.9, 0.9)$.

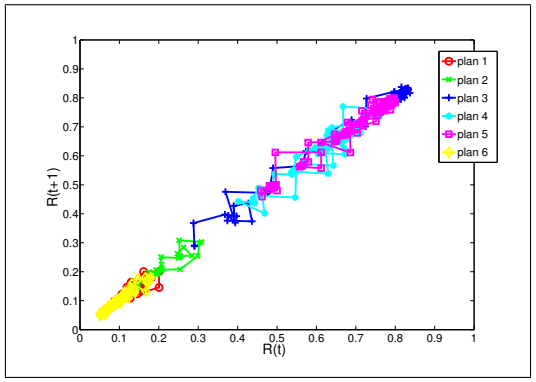


(c) Trajectoires sur la troisième destination. Les marqueurs des plans 4 et 6 convergent vers le point $(0.9, 0.9)$.

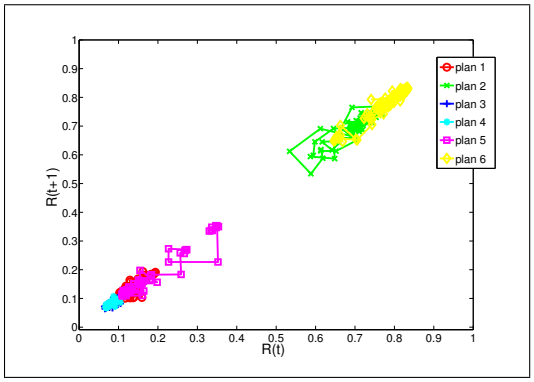
Fig. 7-14. Trajectoires des six marqueurs sur chacune des trois destinations lors du test sur le réseau 1 (SSB). Pour chacune, un ou plusieurs marqueurs convergent vers $(0.9, 0.9)$ et oscillent autour de ce point.



(a) Trajectoires sur la première destination. Le marqueur du premier plan converge vers le point $(0.8, 0.8)$ et oscille dans cette zone, tandis que les autres marqueurs convergent et oscillent autour de $(0.1, 0.1)$.



(b) Trajectoires sur la seconde destination. Le marqueur des plans 3, 4 et 5 convergent lentement vers le point $(0.8, 0.8)$.



(c) Trajectoires sur la troisième destination. Les marqueurs des plans 2 et 6 convergent vers le point $(0.8, 0.8)$.

Fig. 7-15. Trajectoires des six marqueurs sur chacune des trois destinations lors du test *bruité*. Pour chacune, un ou plusieurs marqueurs convergent vers $(0.8, 0.8)$ et oscillent autour de ce point.

7.7.3 Conclusion sur la dynamique des marqueurs

Ces différents expérimentations attestent de la stabilité de notre système.

En effet, les traces observées convergent vers des points représentatifs d'associations entre plan et localisation de catégories (points sur la diagonale, proche de $(0, 0)$ pour les sites non pertinents et au delà de $(0.8, 0.8)$ pour les sites pertinents). Les oscillations sont de faibles amplitudes, ce qui montre que les associations ont de faibles variations dans le temps.

Dans le cas du test *simple bruité*, les traces sont très proches de la première bissectrice, ce qui montre que l'apprentissage des associations se fait de façon progressive, par petites variations.

7.8 Conclusion

L'ensemble des expériences effectuées nous a permis de valider plusieurs aspects de notre système à long-terme. Tout d'abord, nous avons validé l'efficacité du système pour la recherche d'images. Nous avons en particulier montré une amélioration des résultats retrouvés par rapport à l'approche à court-terme (de l'ordre de 10% sur les rappels). Nous avons également montré le gain significatif obtenu en terme de nombre d'annotations nécessaires afin d'obtenir des résultats satisfaisants.

En terme d'apprentissage, nous avons vu que la localisation des catégories est correctement apprise, même dans le cas où l'apprentissage de la mesure de similarité est difficile. Il s'est avéré que cette spécialisation est correctement apprise même dans le cas de distributions complexes des catégories d'images sur le réseau.

Ensuite, nous avons confirmé le bon fonctionnement de l'apprentissage de la fonction de sélection de plans. En effet, les plans utilisés sont ceux qui sont se spécialisés dans la catégorie recherchée, et ceci sans l'influence de l'apprentissage de la fonction de similarité. Notons que même dans les cas les plus complexes (réseau 3) cette association entre les plans utilisés et la catégorie recherchée a toujours été apprise correctement.

Enfin, nous avons validé l'aspect dynamique du système en montrant que l'association entre les plans et les concepts est stable au cours du temps. Nous avons aussi observé que l'évolution des marqueurs est régulière, sans variation abrupte.

De manière plus générale, nous avons validé expérimentalement notre stratégie à trois niveaux d'apprentissage (la fonction de similarité, la sélection de plan, la pertinence des sites pour chaque plan). Ces apprentissages se basent tous sur le même signal de renforcement (les anotations de l'utilisateurs) et sont fortement liés les uns aux autres. Nos expériences ont cependant porté sur des configurations particulières pour lesquelles le système a été paramétré empiriquement. Il serait intéressant de voir dans quelle mesure cette stratégie peut s'étendre à des réseaux plus complexes, et à d'autres bases d'images.

Conclusion

Nos travaux ont porté sur la recherche d'images par le contenu dans un contexte distribué. Dans ce cadre, la problématique est double : il s'agit non seulement de construire une mesure de similarité basée sur le contenu des images comme dans la cadre classique de la recherche d'images par le contenu (*CBIR*), mais aussi de trouver les sites hébergeant les collections susceptibles de contenir les images pertinentes vis-à-vis de la requête. Le système que nous avons développé met en interaction ces deux axes au sein du bouclage de pertinence. Notre architecture s'appuie sur un système multi-agents d'inspiration éthologique, dans lequel les agents ont un comportement stigmergique permettant de faire émerger une sorte de cartographie du réseau relativement à la requête.

Nos résultats se divisent en deux grandes parties : l'apprentissage actif distribué, qui consiste à adapter l'apprentissage actif à un contexte distribué, et l'apprentissage à long terme, qui consiste à utiliser les résultats de plusieurs sessions afin d'optimiser le système pour les sessions futures (toujours dans un cadre distribué).

Apprentissage actif distribué

Nos travaux sur cet axe proposent une nouvelle stratégie active adaptée à la recherche dans des collections distribuées. Cette stratégie opère en deux étapes au sein d'une itération du bouclage de pertinence. Tout d'abord il s'agit de sélectionner les sites les plus pertinents. Cette pertinence est évaluée sur la base des annotations données par l'utilisateur aux images provenant de ces sites. Puis, il s'agit de choisir sur les sites sélectionnés les images qui vont être annotées. La sélection des images s'effectue à la marge du classifieur de manière à réduire l'incertitude sur la mesure de pertinence.

De nombreux tests ont été effectués sur deux bases d'images de référence (*Corel* et *TrecVid'05*), et avec différents protocoles. Nous avons testé l'apprentissage des chemins pertinents, le gain apporté par notre stratégie, et la robustesse du système à l'hypothèse de forte localisation de la catégorie recherchée.

Les résultats obtenus montrent que cette stratégie améliore les résultats par rapport à une approche centralisée (approche dans laquelle toutes les images auraient été concentrées en une seule collection) dans le cas où la catégorie recherchée est concentrée sur un sous-ensemble du réseau. Par rapport à une stratégie qui sélectionnerait les sites de manière

uniforme, les performances de notre système sont grandement supérieures (jusqu'à deux fois meilleurs).

Apprentissage à long terme

L'apprentissage à long terme consiste à utiliser les annotations fournies par l'utilisateur lors des sessions de recherche précédentes afin d'améliorer les sessions à venir. Dans ce cadre, nous proposons une extension de notre système développé précédemment de sorte à ce que plusieurs jeux de marqueurs soient présents sur chaque site, regroupés en plans. Notre stratégie opère alors en trois étapes. D'abord, il s'agit de sélectionner le marqueur de chaque site pertinent pour la requête (c'est à dire le plan pertinent), en utilisant les annotations données par l'utilisateur sur les images récupérées. Puis, il s'agit de sélectionner les sites pertinents à l'aide du plan choisi. Enfin, il s'agit de sélectionner les images à annoter sur les sites choisis.

Notre approche résout donc trois tâches d'apprentissage en utilisant uniquement les annotations données par l'utilisateur :

- La sélection du plan.
- La sélection des sites pertinents.
- L'entraînement de la mesure de similarité.

Nous avons mené de nombreuses expériences avec des protocoles variés (sur la distribution des images, la topologie du réseau) pour valider chacun de ces trois niveaux d'apprentissage séparément, puis nous avons évaluer la dynamique d'ensemble du système. Nous avons aussi testé le gain apporté par l'apprentissage à long terme par rapport à notre précédente stratégie.

Les résultats obtenus montrent une association correcte et stable entre les plans de marqueurs et les concepts disponibles sur le réseau. L'évolution des marqueurs se faisant sur plusieurs sessions, la sélection des sites est grandement améliorée. Le gain de l'apprentissage à long terme pour l'utilisateur s'observe sous la forme d'une réduction du nombre d'itérations du bouclage de pertinence nécessaire pour obtenir les meilleurs résultats.

Perspectives

Parmi les perspectives qui s'offrent à notre système, la première est l'extension à des réseaux de plus grande envergure. Cette extension peut se faire sur la connectivité (largeur du réseau) et sur la profondeur (nombre de sauts). Sur ce premier problème, la question est de déterminer comment conserver l'efficacité de notre système dans le cas où de nombreuses destinations sont disponibles (suffisamment grande pour que les probabilités de saut soient noyées dans l'erreur d'apprentissage). Pour le second problème, il s'agit de déterminer une méthode pour permettre aux agents de continuer leur chemin après le premier site contenant une collection. Cela implique de modifier à la fois les

stratégies de sélection des images mais aussi la manière dont sont renforcés les marqueurs. La seconde extension est de voir dans quelle mesure le système peut être amélioré afin de gérer plusieurs centaines de concepts. En effet, un grand nombre de concepts reviendrait à devoir utiliser une grand nombre de plans dans notre version à long terme, et l'étage de sélection des plans nécessiterait alors une interaction beaucoup trop longue. Dans ce cadre, nous envisageons de déplacer les marqueurs d'un espace de P' dimensions à l'espace de représentation des images. Les avantages d'une telle solution sont d'une part d'avoir une représentation riche pour les concepts (et cohérente avec les images), et d'autre part de pouvoir utiliser la fonction de pertinence $f_{\mathcal{A}}$ pour sélectionner les marqueurs, ce qui simplifie la complexité du système. Par contre, il faut adapter les règles de renforcement des marqueurs à cette nouvelle représentation.

Une piste que nous n'avons pas explorée dans nos travaux est l'optimisation à long terme en ce qui concerne la représentation des images. En effet, notre optimisation à long terme concerne la localisation des collections d'images pertinentes. Il serait envisageable d'essayer une optimisation à long terme, soit sur les signatures des images, soit sur la mesure de similarité afin de réduire le temps d'interaction nécessaire à la construction de résultats satisfaisants. Les travaux qui ont été faits dans ce sens (*cf* 6.1) se sont portés sur des bases uniques et fermées, il serait intéressant de voir comment ceux-ci peuvent s'adapter à notre contexte distribué. Nous pensons notamment à associer à chaque plan une représentation des images adaptées sur le long terme. Cette représentation adaptée peut se faire par le biais d'un noyau $k_{p,i}$ associé à chaque plan $p \in P'$ et à chaque site i. L'apprentissage de $k_{p,i}$ est un problème assez simple : d'une part le contexte est totalement supervisé (la sélection de plan ayant déjà été effectuée), et d'autre part il s'agit d'un problème à deux classes (le concept recherché contre le reste des images).

Enfin, il serait intéressant de voir comment nous pouvons étendre nos travaux à la recherche d'informations multimédia comprenant des images, du texte, de l'audio, de la vidéo, etc. Dans ce cas, les collections contiennent des types de données hétérogènes, pour lesquels les signatures sont différentes et les outils pour en mesurer la similarité aussi. Il serait nécessaire alors d'adapter le système pour prendre en compte toutes ces nouvelles chaînes de traitement de données, et de voir comment celles-ci peuvent être intégrées au sein d'une approche globale mettant en interaction les différents éléments. On peut par exemple penser à des colonies d'agents spécialisées dans certains types de média, partageant les mêmes jeux de marqueurs, lesquels seraient alors représentatifs de concepts multimédias.

Annexe A

Architecture du système

Pour des raisons d'uniformité au sein de la plateforme, tout à été développé sous forme d'agents. Notre système est donc un véritable système multi-agents où les agents mobiles communiquent avec d'autres agents afin d'obtenir des services. Nous décrivons ici chacun de ces agents.

A.1 Agents implémentés

Le premier agent que l'utilisateur rencontre est l'agent d'interface (UIA). Cet agent propose une interface à l'utilisateur à travers laquelle celui-ci peut interagir avec le système (commencer une recherche, annoter des images, observer les résultats). Cet agent est charger de construire la mesure de similarité à partir des annotation fournies à l'utilisateur et de lancer les agents mobiles sur le réseau.

Les agents mobiles (MA) sont les agents qui vont se déplacer sur le réseau à la recherche de collection d'images. Les agents mobiles utilisent les services de trois autres types d'agents :

- les agents compteurs (CA)
- les agents téléporteurs (TA)
- les agents d'indexation (IA)

La figure A.1 décrit les interactions entre les différents agents présents sur la machine de l'utilisateur.

L'agent compteur est un agent qui conserve en mémoire un ensemble de compteurs, c'est-à-dire un ensemble de valeurs numériques et propose un ensemble de services (modification, lecture, initialisation). Ces compteurs vont servir aux agents mobiles pour décider de leur déplacement tels que décrit au chapitre 2.

L'agent téléporteur est l'agent qui sert au déplacement de l'agent mobile. Celui-ci envoie un message contenant son code à l'agent téléporteur qui se charge de lancer son exécution sur la machine où il se trouve. L'agent mobile peut alors terminer son exécution sur la machine de départ.

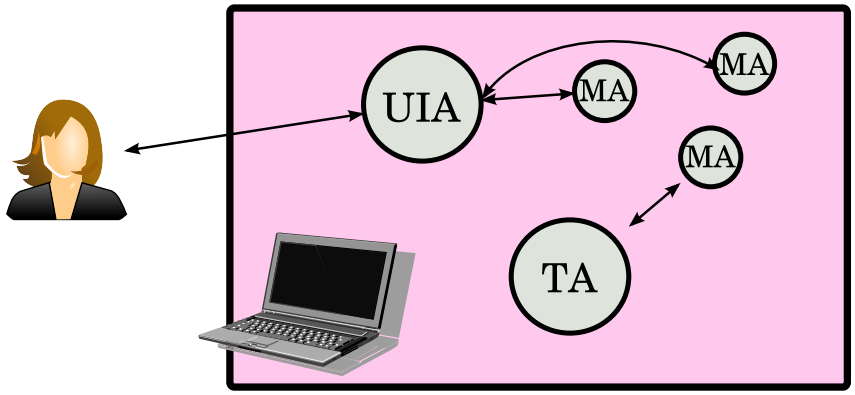


Fig. A-1. Schéma représentant les interactions entre les différents agents présents sur la machine de l'utilisateur. L'utilisateur communique avec l'agent d'interface (UIA). Les agents mobiles communiquent avec l'agent téléporteur (TA) pour pouvoir revenir sur cet ordinateur, et avec l'agent d'interface pour lui renvoyer les résultats.

La figure A.1 décrit les interactions entre les différents agents d'une machine comportant une base d'images.

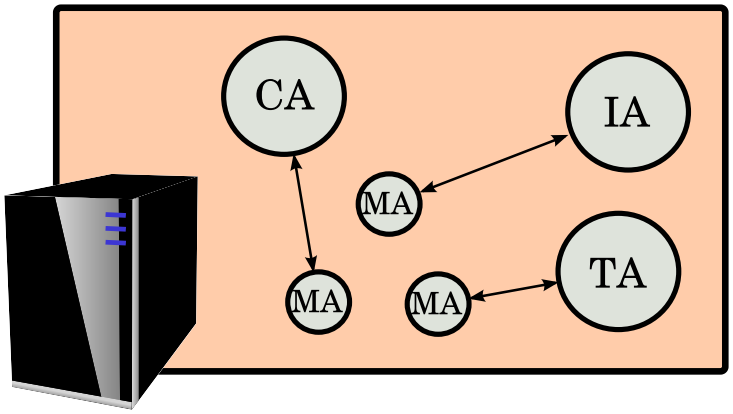


Fig. A-2. Schéma représentant les interactions entre les différents d'agents présents sur un ordinateur contenant une base d'images. Les agents mobiles communiquent avec l'agent compteur (CA) pour obtenir les valeurs de marqueurs, avec l'agent téléporteur pour pouvoir se télécharger sur cet ordinateur et avec l'agent d'indexation (IA) pour récupérer les images pertinentes.

L'agent d'indexation est chargé d'indexer les images localement à la machine où il se trouve et de garder en mémoire l'index des signatures. Il propose les services de recherche aux agents mobiles : l'agent mobile lui envoie un message contenant la mesure de pertinence et la stratégie de sélection, l'agent d'indexation les utilise pour sélectionner un lot d'images dans son index et renvoie ce lot dans un message à l'agent mobile.

Bibliographie

[Aksoy and Haralick, 1998] Aksoy, S. and Haralick, R. (1998). Textural features for image database retrieval. In *IEEE Workshop on Content-Based Access of Image and Video Libraries, in conjunction with CVPR'98*, pages 45–49, Santa Barbara, CA.

[Amann and Constantin, 2007] Amann, B. and Constantin, C. (2007). Collaborative cache based on path scores. In Benatallah, B., Casati, F., Georgakopoulos, D., Bartolini, C., Sadiq, W., and Godart, C., editors, *WISE*, volume 4831 of *Lecture Notes in Computer Science*, pages 87–98. Springer.

[Baumgarten, 1997] Baumgarten, C. (1997). Probabilistic modeling of distributed information retrieval. Technical Report TUD-FI97-01.

[Berretti et al., 2004] Berretti, S., Bimbo, A. D., and Pala, P. (2004). Merging results for distributed content based image retrieval. *Multimedia Tools Appl.*, 24(3) :215–232.

[Botee and Bonabeau, 1998] Botee, H. M. and Bonabeau, E. (1998). Evolving ant colony optimization. *Advanced Complex Systems*, 1 :149–159.

[Brazier et al., 2004] Brazier, F., Groot, D. d., Oskamp, A., and Wijngaards, N. (2004). Agent-based information retrieval : Legal and technical considerations in a simple case. pages 95–107.

[Brinker, 2003] Brinker, K. (2003). Incorporating diversity in active learning with support vector machines. In *In Proceedings of the 20th International Conference on Machine Learning*, pages 59–66. AAAI Press.

[Broder et al., 2000] Broder, A., Kumar, R., Maghoul, F., Raghavan, P., Rajagopalan, S., Stata, R., Tomkins, A., and Wiener, J. (2000). Graph structure in the web. *Comput. Netw.*, 33(1-6) :309–320.

[Brunelli and Mich, 2001] Brunelli, R. and Mich, O. (2001). Histograms analysis for image retrieval. *Pattern Recognition*, 34 :1625–1637.

[Callan, 2000] Callan, J. (2000). Distributed information retrieval. *Advances in information retrieval*, 5 :127–150.

[Caro and Dorigo, 1997] Caro, G. D. and Dorigo, M. (1997). AntNet : a mobile agents approach to adaptive routing. Technical Report IRIDIA/97-12, Université Libre de Bruxelles, Belgium.

[Carrilero, 1999] Carrilero, A.-C. (1999). Les espaces de représentation de la couleur. Technical Report 99D006, ENST, Paris.

[Chan, 1999] Chan, P. K. (1999). A non-invasive learning approach to building web user profiles. In *KDD-99 Workshop on Web Usage Analysis and User Profiling*.

[Chapelle et al., 1999] Chapelle, O., Haffner, P., and Vapnik, V. (1999). Svms for histogram based image classification. *IEEE Transactions on Neural Networks*, 9.

[Chen et al., 2004] Chen, Y., Wang, J. Z., and Geman, D. (2004). Image categorization by learning and reasoning with regions. *Journal of Machine Learning Research*, 5 :2004.

[Chen et al., 2001] Chen, Y., Zhou, X., and Huang, T. (2001). One-class svm for learning in image retrieval. In *International Conference in Image Processing (ICIP'01)*, volume 1, pages 34–37, Thessaloniki, Greece.

[Clements et al., 1997] Clements, P. E., Papaioannou, T., and Edwards, J. (1997). Aglets : Enabling the virtual enterprise. In *Managing Enterprises – Stakeholders, Engineering, Logistics and Achievement (ME-SELA'97)*, Loughborough University, UK.

[Cocquerez and Philipp, 1995] Cocquerez, J. and Philipp, S. (1995). *Analyses d'images : filtrage et segmentation*. Masson, Paris.

[Cohn, 1996] Cohn, D. (1996). Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4 :129–145.

[Cord et al., 2008] Cord, M., Cunningham, P., Cord, M., and Cunningham, P. (2008). *Machine Learning Techniques for Multimedia : Case Studies on Organization and Retrieval (Cognitive Technologies)*. Springer-Verlag TELOS, Santa Clara, CA, USA.

[Cord and Gosselin, 2006] Cord, M. and Gosselin, P.-H. (2006). Image retrieval using long-term semantic learning. In *IEEE International Conference on Image Processing*, Atlanta, GA, USA. IEEE.

[Corkill, 1991] Corkill, D. (1991). Blackboard Systems. *AI Expert*, 6(9).

[Cortes and Vapnik, 1995] Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3) :273–297.

[Cristianini et al., 2001] Cristianini, N., Shawe-Taylor, J., Elisseeff, A., and Kandola, J. S. (2001). On kernel-target alignment. In *NIPS*, pages 367–373.

[Deneubourg and Goss, 1989] Deneubourg, J. and Goss, S. (1989). Collective patterns and decision-making. *Ethology Ecology & Evolution*, 1 :295–311.

[Deneubourg et al., 1992] Deneubourg, J., Theraulaz, G., and Beckers, R. (1992). Swarm-made architectures. In Varela, F. and Bourgine, P., editors, *Proceedings of the first european conference on artificial intelligence*, pages 123–133. MIT Press.

[Dorigo et al., 2006] Dorigo, M., Birattari, M., and Stu ?tzle, T. (2006). Ant colony optimization artificial ants as a computational intelligence technique. *IEEE Comput. Intell. Mag.*, 1(4) :28–39.

[Dorigo and Gambardella, 1997] Dorigo, M. and Gambardella, L. M. (1997). Ant colony system : A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1) :53–66.

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

[Drogoul and Ferber, 1992] Drogoul, A. and Ferber, J. (1992). From tom thumb to the dockers : Some experiments with foraging robots. In *Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, pages 451–459.

[Ducatelle et al., 2005] Ducatelle, F., Caro, G. D., and Gambardella, L. M. (2005). Using ant agents to combine reactive and proactive strategies for routing in mobile ad-hoc networks. *International Journal of Computational Intelligence and Applications*, 5(2) :169–184.

[Ferber, 1995] Ferber, J. (1995). *Les systèmes multi-agents, vers une intelligence collective*. InterEditions, Paris (France).

[Ferecatu, 2005] Ferecatu, M. (2005). *Image retrieval with active relevance feedback using both visual and keyword-based descriptors*. PhD thesis, INRIA Rocquencourt.

[Ferecatu et al., 2005] Ferecatu, M., Boujemaa, N., and Crucianu, M. (2005). Hybrid visual and conceptual image representation within active relevance feedback context. In *MIR '05 : Proceedings of the 7th ACM SIGMM international workshop on Multimedia information retrieval*, pages 209–216, New York, NY, USA. ACM.

[Finin et al., 1994] Finin, T., Fritzson, R., McKay, D., and McEntire, R. (1994). KQML as an Agent Communication Language. In Adam, N., Bhargava, B., and Yesha, Y., editors, *Proceedings of the 3rd International Conference on Information and Knowledge Management (CIKM'94)*, pages 456–463, Gaithersburg, MD, USA. ACM Press.

[Fournier, 2002] Fournier, J. (2002). *Content based image indexing and interactive retrieval*. PhD thesis, UCP, Paris, France. Written in French.

[Fournier and Cord, 2002] Fournier, J. and Cord, M. (2002). Long-term similarity learning in content-based image retrieval. In *IEEE International Conference in Image Processing (ICIP'02)*, Rochester, New-York, USA.

[Fournier et al., 2001a] Fournier, J., Cord, M., and Philipp-Foliguet, S. (2001a). Retin : A content-based image indexing and retrieval system. *Pattern Analysis and Applications Journal, Special issue on image indexation*, 4(2/3) :153–173.

[Fournier et al., 2001b] Fournier, J., Cord, M., and Philipp-Foliguet, S. (2001b). Retin : A content-based image indexing and retrieval system. *Pattern Analysis and Applications Journal, Special issue on image indexation*, 4(2/3) :153–173.

[Gabor, 1946] Gabor, D. (1946). Theory of communication. *Journal of the Institution of Electrical Engineers*, 93(III) :429–457.

[Gasparetti and Micarelli, 2003] Gasparetti, F. and Micarelli, R. (2003). A. : Adaptive web search based on a colony of cooperative distributed agents. In *Cooperative Information Agents VII. Volume 2782 of Lecture Notes in Artificial Intelligence. Springer-Verlag*, pages 168–183.

[Geusebroek et al., 2001] Geusebroek, J.-M., van den Boomgaard, R., Smeulders, A. W., and Geerts, H. (2001). Color invariance. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(12) :1338–1350.

[Ghezzi and Vigna, 1997] Ghezzi, C. and Vigna, G. (1997). Mobile code paradigms and technologies : A case study. In *Proceedings of the First International Workshop on Mobile Agents*, Berlin, Germany.

[Gosselin and Cord, 2004] Gosselin, P. and Cord, M. (2004). Semantic kernel updating for content-based image retrieval. In *IEEE International Workshop on Multimedia Content-based Analysis and Retrieval (MCBAR)*, Miami, Florida, USA.

[Gosselin, 2005] Gosselin, P.-H. (2005). *Méthodes d'apprentissage pour la recherche de catégories dans des bases d'images*. PhD thesis, Université de Cergy-Pontoise. Direction : Sylvie Philipp-Foliguet et Matthieu Cord.

[Gosselin and Cord, 2005a] Gosselin, P.-H. and Cord, M. (2005a). Active learning techniques for user interactive systems : application to image retrieval. In *International Workshop on Machine Learning techniques for processing MultiMedia content*, Bonn, Germany.

[Gosselin and Cord, 2005b] Gosselin, P.-H. and Cord, M. (2005b). Semantic kernel learning for interactive image retrieval. In *IEEE International Conference on Image Processing*, Genoa, Italy. IEEE.

[Gosselin and Cord, 2006a] Gosselin, P.-H. and Cord, M. (2006a). Feature based approach to semi-supervised similarity learning. *Pattern Recognition*, (39) :1839–1851. Special issue : Similarity-Based Pattern Recognition.

[Gosselin and Cord, 2006b] Gosselin, P.-H. and Cord, M. (2006b). Precision-oriented active selection for interactive image retrieval. In *IEEE International Conference on Image Processing (ICIP'06)*, Atlanta, GA, USA.

[Gosselin et al., 2007a] Gosselin, P.-H., Cord, M., and Philipp-Foliguet, S. (2007a). Kernel on bags for multi-object database retrieval. In *ACM International Conference on Image and Video Retrieval (CIVR)*, Amsterdam, The Netherlands.

[Gosselin et al., 2007b] Gosselin, P.-H., Cord, M., and Philipp-Foliguet, S. (2007b). Kernel on bags of fuzzy regions for fast object retrieval. In *IEEE International Conference on Image Processing (ICIP 07)*, San Antonio, Texas, USA.

[Gray et al., 2000] Gray, R. S., Kotz, D., Cybenko, G., and Rus, D. (2000). Mobile agents : Motivations and state-of-the-art systems. Technical Report TR2000-365, Dartmouth College, Hanover, NH.

[Groot et al., 2005] Groot, D. d., Boonk, M., Brazier, F., and Oskamp, A. (2005). Issues in a mobile agent-based multimedia retrieval scenario. In *Proceedings of The 4th Workshop on the Law and Electronic Agents (LEA 2005)*, pages 33–43.

[Guigues et al., 2006] Guigues, L., Cocquerez, J. P., and Men, H. (2006). Scale-sets image analysis. *Int. J. Comput. Vision*, 68(3) :289–317.

[Harris and Stephens, 1988] Harris, C. and Stephens, M. (1988). A combined corner and edge detection. In *Proceedings of The Fourth Alvey Vision Conference*, pages 147–151.

[He et al., 2004] He, J., Tong, H., Li, M., Zhang, H.-J., and Zhang, C. (2004). Mean version space : a new active learning method for content-based image retrieval. In *MIR '04 : Proceedings of the 6th ACM SIGMM international workshop on Multimedia information retrieval*, pages 15–22, New York, NY, USA. ACM.

[Heisterkamp, 2002] Heisterkamp, D. R. (2002). Building a latent semantic index of an image database from patterns of relevance feedback. In *International Conference on Pattern Recognition (ICPR)*, Quebec City, Canada.

[Huang and Zhou, 2001] Huang, T. and Zhou, X. (2001). Image retrieval with relevance feedback : From heuristic weight adjustment to optimal learning methods. In *International Conference in Image Processing (ICIP'01)*, volume 3, pages 2–5, Thessaloniki, Greece.

[Jarras and Chaib-draa, 2002] Jarras, I. and Chaib-draa, B. (2002). Aperçu sur les systèmes multiagents. CIRANO Working Papers 2002s-67, CIRANO.

[Jennings et al., 1998] Jennings, N. R., Sycara, K., and Wooldridge, M. (1998). A roadmap of agent research and development. *Journal of Autonomous Agents and Multi-Agent Systems*, 1(1) :7–38.

[Jiao and Hurson, 2004] Jiao, Y. and Hurson, A. R. (2004). Performance analysis of mobile agents in mobile distributed information retrieval system - a quantitative case study. *Journal of Interconnection Networks*, 5(3) :351–372.

[Jurie and Triggs, 2005] Jurie, F. and Triggs, B. (2005). Creating efficient codebooks for visual recognition. In *ICCV '05 : Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, pages 604–610, Washington, DC, USA. IEEE Computer Society.

[King et al., 2004] King, I., Ng, C. H., and Sia, K. C. (2004). Distributed content-based visual information retrieval system on peer-to-peer networks. *ACM Trans. Inf. Syst.*, 22(3) :477–501.

[Lange, 1998] Lange, D. B. (1998). Mobile objects and mobile agents : The future of distributed computing ? In *ECOOP*, pages 1–12.

[Lange and Oshima, 1999] Lange, D. B. and Oshima, M. (1999). Seven good reasons for mobile agents. *Commun. ACM*, 42(3) :88–89.

[Lewis and Catlett, 1994] Lewis, D. and Catlett, J. (1994). Heterogenous uncertainty sampling for supervised learning. In *International Confenrence on Machine Learning*.

[Lindenbaum et al., 2004] Lindenbaum, M., Markovitch, S., and Rusakov, D. (2004). Selective sampling for nearest neighbor classifiers. *Machine Learning*, 54(2) :125–152.

[Liu and Picard, 1996] Liu, F. and Picard, R. W. (1996). Periodicity, directionality, and randomness : Wold features for image modeling nad retrieval. *IEEE Trans. Pattern Anal. Machine Intell.*, 18(7) :722–733.

[Lowe, 2003] Lowe, D. (2003). Distinctive image features from scale-invariant keypoints. In *International Journal of Computer Vision*, volume 20, pages 91–110.

[Lu and Callan, 2003] Lu, J. and Callan, J. (2003). Content-based retrieval in hybrid peer-to-peer networks. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management (CIKM'03)*, New Orleans.

[Ma and Manjunath, 1995] Ma, W. and Manjunath, B. (1995). Image indexing using a texture dictionnary. In *SPIE Conference on Image Storage and Archiving System*, volume 2606, pages 288–298, Philadelphia, Pennsylvania.

[Magnin et al., 2002] Magnin, L., Snoussi, H., Pham, V., and Dury, A. (2002). Agents need to become welcome.

[Manjunath and Ma, 1996] Manjunath, B. and Ma, W. (1996). Texture features for browsing and retrieval of image data. *IEEE Transactions on Pattern Analysis and Machine Intelligence (Special Issue on Digital Libraries)*, 18(8) :837–842.

[Müller et al., 2000] Müller, H., Müller, W., Squire, D. M., Müller, H., Marchand-Maillet, S., and Pun, T. (2000). Long-term learning from user behavior in content-based image retrieval. Technical report 00.04, Computer Vision Group, Computing Science Center, University of Geneva, Genève, Switzerland.

[Najjar et al., 2003] Najjar, N., Cocquerez, J., and Ambroise, C. (2003). Feature selection for semi supervised learning applied to image retrieval. In *IEEE ICIP*, Barcelena, Spain.

[Niblack et al., 1993] Niblack, W., Barber, R., Equitz, W., Flickner, M., Glasman, E., Petkovic, D., Yanker, P., Faloutsos, C., and Taubin, G. (1993). The QBIC project : Querying images by content, using color, texture, and shape. In *Storage and Retrieval for Image and Video Databases (SPIE)*, pages 173–187.

[Pala and Santini, 1999] Pala, P. and Santini, S. (1999). Image retrieval by shape and texture. *Pattern Recognition Journal*, 32(3) :517–527.

[Park, 2000] Park, J. (2000). On-line learning by active sampling using orthogonal decision support vectors. In *IEEE Neural Networks for Signal Processing*.

[Petkov and Kruizinga, 1997] Petkov, N. and Kruizinga, P. (1997). Computational models of visual neurons specialised in the detection of periodic and aperiodic oriented visual stimuli : Bar and grating cells.

[Philbin et al., 2007] Philbin, J., Chum, O., Isard, M., Sivic, J., and Zisserman, A. (2007). Object retrieval with large vocabularies and fast spatial matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[Philipp-Foliguet and Guigues, 2006] Philipp-Foliguet, S. and Guigues, L. (2006). Evaluation de la segmentation : état de l'art, nouveaux indices et comparaison. *Traitement du Signal*, 23(2) :109–125.

[Picard et al., 2006a] Picard, D., Cord, M., and Revel, A. (2006a). Cbir in distributed databases using a multi-agent system. In *IEEE International Conference on Image Processing (ICIP'06)*, Atlanta, GA, USA.

[Picard et al., 2008a] Picard, D., Cord, M., and Revel, A. (2008a). Image retrieval over networks : Active learning using ant algorithm. *ieee Transactions on Multimedia*. To appear.

[Picard et al., 2008b] Picard, D., Cord, M., and Revel, A. (2008b). Long term learning for image retrieval over networks. In *IEEE International Conference on Image Processing (ICIP'08)*, San Diego, CA, USA. To appear.

[Picard et al., 2006b] Picard, D., Revel, A., and Cord, M. (2006b). Performances of mobile-agents for interactive image retrieval. In *2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI'06)*, pages 581–586. IEEE Computer Society.

[Picard et al., 2008c] Picard, D., Revel, A., and Cord, M. (2008c). Image retrieval over networks : Ant algorithm for long term active learning. In *CBMI 2008, 6th International Workshop on Content Based Multimedia Indexing, June, 18-20th 2008, London, UK*. To appear.

[Platt, 1999] Platt, J. C. (1999). Fast training of support vector machines using sequential minimal optimization. pages 185–208.

[Randen and Husoy, 1997] Randen, T. and Husoy, J. (1997). Image content search by color and texture properties. In *Proc.IEEE Int. Conf. on Image Processing*, Santa Barbara, California.

[Revel, 2003] Revel, A. (2003). Web-agents inspired by ethology : a population of "ant"-like agents to help finding user-oriented information. In *IEEE WIC'2003 : International Conference on Web Intelligence.*, pages 482–485, Halifax, Canada. IEEE, IEEE Computer Society.

[Revel, 2005] Revel, A. (2005). From robots to web-agents : Building cognitive software agents for web-information retrieval by taking inspiration from experience in robotics. In *ACM International conference on Web Intelligence*, Université Technologique de Compiègne, Compiègne, France.

[Ripeanu et al., 2002] Ripeanu, M., Foster, I., and Iamnitchi, A. (2002). Mapping the gnutella network : Properties of large-scale peer-to-peer systems and implications for system design. *IEEE Internet Computing Journal*, 6 :2002.

[Roth, 2004] Roth, V. (2004). Obstacles to the adoption of mobile agents. In *Mobile Data Management, 2004. Proceedings. 2004 IEEE International Conference on*, pages 296 – 297.

[Roth and Jalali-Sohi, 2001] Roth, V. and Jalali-Sohi, M. (2001). Concepts and architecture of a security-centric mobile agent server. In *ISADS*, pages 435–.

[Roth et al., 2005] Roth, V., Pinsdorf, U., and Peters, J. (2005). A distributed content-based search engine based on mobile code. In *SAC '05 : Proceedings of the 2005 ACM symposium on Applied computing*, pages 66–73, New York, NY, USA. ACM Press.

[Roy and McCallum, 2001] Roy, N. and McCallum, A. (2001). Toward optimal active learning through sampling estimation of error reduction. In *International Confenrence on Machine Learning*.

[R.Schoonderwoed et al., 1997] R.Schoonderwoed, O.Holland, J.Bruten, and L.Rothkrantz (1997). Ant-based load balancing in telecommunications networks. *Adaptive Behavior*, 5 :169–207.

[Rubner et al., 1998] Rubner, Y., Tomasi, C., and Guibas, L. (1998). The earth mover's distance as a metric for image retrieval. Technical Report STAN-CS-TN-98-86, Department of Computer Science, Stanford University.

[Rui and Huang, 2000] Rui, Y. and Huang, T. (2000). Optimizing learning in image retrieval. In *Conf on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 236–243, Hilton Head, SC.

[Rui et al., 1998] Rui, Y., Huang, T., and Mehrotra, S. (1998). Relevance feedback techniques in interactive content-based image retrieval. In *SPIE Storage and Retrieval of Images Video Databases VI, EI'98*.

[Santini et al., 2001] Santini, S., Gupta, A., and Jain, R. (2001). Emergent semantics through interaction in image databases. *IEEE Transactions on Knowledge and Data Engineering*, 13(3) :337–351.

[Si and Callan, 2002] Si, L. and Callan, J. (2002). Using sampled data and regression to merge search engine results. In *Proc. of the 24 Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.*

[Smeulders et al., 2000] Smeulders, A., Worring, M., Santini, S., Gupta, A., and Jain, R. (2000). Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12) :1349–1380.

[Smith and Chang, 1996] Smith, J. and Chang, S. (1996). VisualSEEK : a fully automated content-based image query system. In *ACM Multimedia Conference*, pages 87–98, Boston, USA.

[Smola and Scholkopf, 2002] Smola, A. and Scholkopf, B. (2002). *Learning with kernels.* MIT Press, Cambridge, MA.

[Snoek et al., 2005] Snoek, C., Worring, M., and Smeulders, A. W. M. (2005). Early versus late fusion in semantic video analysis. In *ACM Multimedia*, pages 399–402.

[Speretta and Gauch, 2005] Speretta, M. and Gauch, S. (2005). Personalized search based on user search histories. *Web Intelligence, 2005. Proceedings. The 2005 IEEE/WIC/ACM International Conference on*, pages 622–628.

[Stricker and Orengo, 1995] Stricker, M. and Orengo, M. (1995). Similarity of color images. In *SPIE, Storage and Retrieval for Image Video Databases III*, volume 2420, pages 381–392.

[Suard, 2006] Suard, F. (2006). *Machines à noyaux pour la détection de piétons.* PhD thesis, INSA de Rouen.

[Tamura et al., 1978] Tamura, H., Mori, S., and Yamawaki, T. (1978). Texture features corresponding to visual perception. *IEEE Transactions on Systems, Man and Cybernetics*, 8(6).

[Theraulaz and Gervet, 1992] Theraulaz, G. and Gervet, J. (1992). Les performances collectives des sociétés d'insectes. *Psychologie Française*, 37(1) :7–14.

[Tieu and Viola, 2000] Tieu, K. and Viola, P. (2000). Boosting image retrieval. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 228–235.

[Tong and Chang, 2001] Tong, S. and Chang, E. (2001). Support vector machine active learning for image retrieval. In *ACM Multimedia.*

[Usunier, 2006] Usunier, N. (2006). *Apprentissage de fonctions d'ordonnancement : une étude théorique de la réduction à la classification binaire et deux applications à la Recherche d'Information.* PhD thesis, LIP6 UPMC.

[Vasconcelos, 2000] Vasconcelos, N. (2000). *Bayesian models for visual information retrieval.* PhD thesis, Massachusetts Institute of Technology.

[Vasconcelos and Kunt, 2001] Vasconcelos, N. and Kunt, M. (2001). Content-based retrieval from image databases : current solutions and future directions. In *International Conference in Image Processing (ICIP'01)*, volume 3, pages 6–9, Thessaloniki, Greece.

[Veltkamp, 2002] Veltkamp, R. (2002). Content-based image retrieval system : A survey. Technical report, University of Utrecht.

[Vu, 2008] Vu, H.-T. (2008). *Apprentissage d'ordonnancements pour la constitution de Corpus d'evaluation et pour l'Agregation de listes en Recherche d'Information*. PhD thesis, LIP6 UPMC.

[Wang and Ma, 2005] Wang, D. and Ma, X. (2005). A hybird image retrieval system with user's relevance feedback using neurocomputing. *Informatica (Slovenia)*, 29(3) :271–280.

[Wang et al., 2001] Wang, J., Li, J., and Wiederhold, G. (2001). SIMPLIcity : Semantics-Sensitive Integrated Matching for Picture LIbraries. *IEEE Transactions on pattern analysis and machine intelligence*, 23(9).

[Wolf et al., 2000] Wolf, C., Jolion, J.-M., and Bischof, H. (2000). Histograms for texture based image retrieval. In *OEAGM 2000*, pages 169–176, Oldenbourg. Robert Sablatnig and Christian Menard, editors.

[Wooldridge and Jennings, 1995] Wooldridge, M. and Jennings, N. R. (1995). Intelligent agents : Theory and practice. *Knowledge Engineering Review*, 10(2) :115–152.

[Zhang et al., 2001] Zhang, L., Lin, F., and Zhang, B. (2001). Support vector machine learning for image retrieval. In *ICIP (2)*, pages 721–724.