N° : D'ORDRE 12/2010 – M/INF
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

UNIVERSITE DES **S**CIENCES ET DE LA **T**ECHNOLOGIE **H**OUARI **B**OUMEDIENE (U.S.T.H.B)
Faculté d'Electronique et d'Informatique

MEMOIRE

Présenté pour l'obtention du diplôme de MAGISTER
En INFORMATIQUE

Option : Intelligence Artificielle et Ingénierie de logiciels

par : M^elle^ MOSTEGHANEMI Hadia

Les essaims d'abeilles pour la recherche d'information à grande échelle

M. LARABI Slimane	***Professeur à l'USTHB***	***Président***
Mme DRIAS Habiba	***Professeur à l'USTHB***	***Directrice de Mémoire***
Melle BOUGHACI Dalila	***Maître de conférence/A à l'USTHB***	***Examinatrice***
M. KACHID Samir	***Maître de conférence/B à l'USTHB***	***Invité***

Remerciements

En premier lieu Je remercie ALLAH le tout puissant de m'avoir permis de mener à bien ce travail.

Je tiens à exprimer mes plus vifs remerciements et ma profonde gratitude à ma directrice de recherche Mme Drias Habiba pour m'avoir soumis un sujet si intéressant. Je la remercie également pour son orientation et ses conseils tout au long de mon parcours. Je lui suis entièrement reconnaissante d'avoir guidé mes premiers pas dans le monde de la recherche, et de m'avoir fait bénéficier de sa riche expérience qui m'a été d'un grand apport, aussi bien sur le plan scientifique que sur le plan humain.

Je remercie vivement Mr Larabi Slimane pour l'honneur qu'il m'a fait en acceptant de présider le jury de ma soutenance.

Mes remerciements vont aussi à Mlle Boughaci Dalila et Mr Kechid Samir d'avoir accepté de faire partie du jury.

Dédicaces

Je tiens à dédier ce travail aux personnes qui me sont chères

A ceux qui m'ont donné la vie, je leur témoigne de mon amour et ma reconnaissance.

A mes deux sœurs.

A mes grands-parents qui ont toujours été là pour moi

A l'ensemble de ma famille

A tous mes amis qui sauront se reconnaître

J'exprime mes sentiments les plus profonds et leur dédie ce modeste travail.

HADIA

Sommaire

Chapitre 2 : Etat de l'art sur les essaims d'abeilles

Partie II : Les essaims d'abeilles pour la recherche d'information à grande échelle.

Chapitre 3 : Une approche basée sur les essaims d'abeilles pour une recherche dans la collection de documents

Chapitre 4 : Une approche basée sur les essaims d'abeilles pour une recherche dans le fichier inverse

Partie III : Expérimentation

Chapitre 5 : Implémentation et résultat expérimentaux

Introduction générale

Grâce à son expansion, Internet occupe toujours de plus en plus de place dans l'économie mondiale, mais également dans notre vie quotidienne. Toutes les entreprises ont aujourd'hui compris qu'Internet n'est pas un simple effet de mode, mais qu'il va probablement continuer à changer nos habitudes de consommation. En l'espace de quelques années, les applications d'Internet n'ont eu de cesse de s'élargir, que ce soit pour rechercher une information, réserver un voyage, payer les impôts dans certains pays, consulter les offres d'emplois ou encore débattre de sujets divers. Ce n'est probablement rien en comparaison de ce que les entreprises du secteur nous préparent. Déjà source d'actualité préférée des jeunes, demain, Internet constituera certainement notre principale source d'informations, devant la télévision, la presse et la radio.

Le développement d'Internet a vu l'émergence de la recherche d'information à coté d'une pléthore d'autres applications comme le commerce électronique (B2B, B2C), l'enseignement à distance…etc. Actuellement la recherche d'information est devenue un vaste champ de recherche très actif, ceci est dû à l'énorme quantité d'information disponible sur le réseau Internet, et aussi parce que les concepteurs de ce dernier n'ont rien défini quant à l'organisation des sites, de leurs pages et des contenus publiés en général.

La recherche d'information en elle-même constitue l'un des problèmes fondamentaux en informatique. Elle consiste à localiser une information donnée sur la base d'un ou plusieurs mots-clés fournis pour cette recherche. Ainsi défini, le problème de recherche s'apparente avec n'importe quel problème de recherche en optimisation.

Hormis ce fait, la recherche d'information porte en elle également un problème de représentation de l'information pour permettre de s'y référencer et d'y accéder. Ce problème n'est autre que l'indexation qui constitue un élément essentiel pour la recherche puisque celle-ci s'y base principalement. Depuis l'apparition de la recherche d'information en tant que discipline, les méthodes d'indexations n'ont pas cessé d'évoluer au fil du temps. Le choix d'une méthode adéquate pour l'indexation et la représentation de l'information advient de la performance et de la rapidité de la recherche.

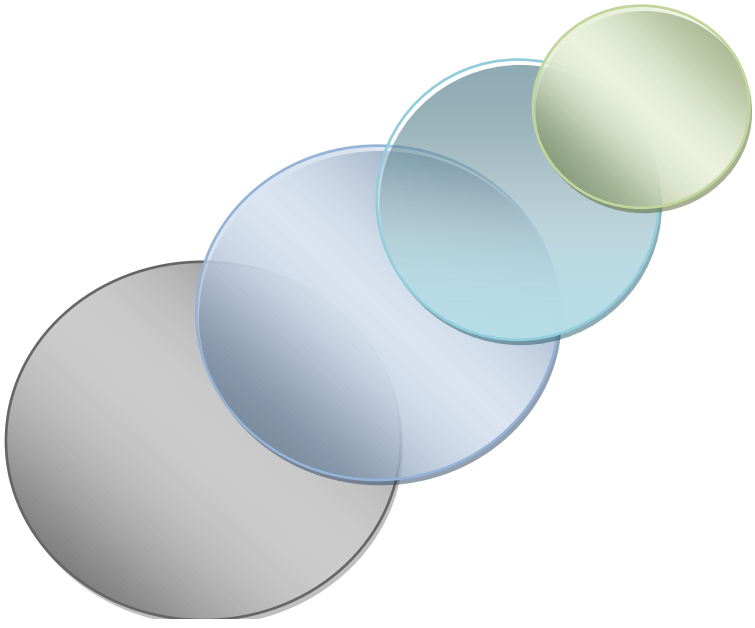
Ce problème peut donc être résolu grâce à l'utilisation d'une des nombreuses méthodes heuristiques et métaheuristiques qui ont montré leur efficacité pour la résolution des problèmes d'optimisation.

Les techniques de l'intelligence artificielle ont été très peu abordées en littérature pour la recherche d'information. C'est la raison pour laquelle nous suggérons une voie d'investigation nouvelle portant sur l'étude de la métaheuristique issue de l'intelligence en essaim : « essaim d'abeilles » et de son adaptation au problème de recherche d'information à grande échelle. Cette approche de résolution semble assez prometteuse car son fonctionnement peut conduire à une modélisation proche du domaine d'application.

Ce présent mémoire décrit notre contribution au problème de recherche d'information à grande échelle par l'exploitation des essaims d'abeilles comme mécanisme de recherche. Ce document est organisé en trois parties. La première inclut l'état de l'art de la recherche d'information et des essaims d'abeilles. La seconde partie comporte nos deux contributions, une première contribution correspondant à l'adaptation de l'essaim d'abeilles pour une recherche dans la collection de documents et la seconde qui est également le développement de l'approche basée sur les essaims d'abeilles pour une recherche d'information sur le fichier inverse. Enfin la troisième partie englobe les tests expérimentaux de validation des deux approches proposées ainsi que les résultats obtenus.

Partie I :

Etat de l'art



Chapitre I

Etat de l'art sur La recherche d'information

1. Introduction

La croissance explosive de l'information, a fait de la recherche d'information une question de survie pour les compagnies qui nécessitent d'avoir dans leurs dispositifs de bons instruments de recherche d'information.

Le stockage d'information n'est plus contraint par l'espace mémoire qui ne pose plus de problème et ce depuis plusieurs années, puisqu'il existe de plus en plus de dispositifs de stockages, disponibles, et à des prix raisonnables. Et donc l'information est devenue de plus en plus volumineuse.

A cause de cette croissance en quantité d'information, les informations inutilisées ou bien non récemment consultées disparaissent parce qu'il n'existe pas de possibilité d'y accéder efficacement. Ce problème a été étudié depuis plusieurs années et est connu sous l'appellation ***de recherche d'information***. Ce problème peut être décrit par le biais de l'expression suivante : « ***Par quelle voie se distinguent les informations pertinentes des informations hors de propos pour un certain besoin d'information*** »

2. Les systèmes de recherche d'information

La recherche d'information est un processus qui met en correspondance un besoin exprimé par l'utilisateur avec l'information associé à partir d'une large, voire même très vaste collection d'informations. Un système de recherche d'information (Noté ***IRS*** pour *Information Retrieval System*) est un mécanisme qui prend en charge ce processus de recherche.

2.1.Définitions

Plusieurs définitions de la recherche d'information existent dans la littérature, en voici quelque unes :

- La recherche d'information, de manière générale est l'art et la science de construire des systèmes qui aident l'humain à retrouver l'information recherchée **[CHE07].**
- Un système de recherche d'information consiste en une base de données, contenant un certain nombre de document, d'index, et qui associe chaque document aux termes relatés, et un mécanisme d'appariement qui associe la requête utilisateur, constituée de termes à un ensemble de documents qui leurs sont associés. Un but typique de la recherche d'information est de trouver un ensemble de documents qui contiennent l'information nécessaire en recherchant dans la base de données indexée **[SON06].**

- Dans le contexte des systèmes d'informations, la recherche d'information peut être définie comme un processus qui s'occupe de la représentation, du stockage et de l'accès aux documents ou aux documents représentatifs (substitut de document). L'information en entrée peut inclure le langage naturel, des textes de documents ou des résumés. Le résultat de la recherche d'information est la réponse à la demande de recherche, chaque demande contenant un ensemble de références. Ces références tentent de fournir à l'utilisateur des items qui devraient potentiellement l'intéresser **[SEL07]**.

- La recherche d'information est de trouver un 'matériel' (souvent un document) de nature non-structurée (souvent dans un format de texte), qui satisfait le besoin en information à l'intérieur d'une large collection (souvent sauvegardée sur des ordinateurs) **[MAN08]**.

Tous ces auteurs s'accordent sur le fait, que la recherche d'information est un processus qui tente de déterminer dans une large collection d'informations, les documents qui devraient correspondre à l'information exprimée par le demandeur. Le schéma général de la recherche d'information est donné par la figure qui suit :

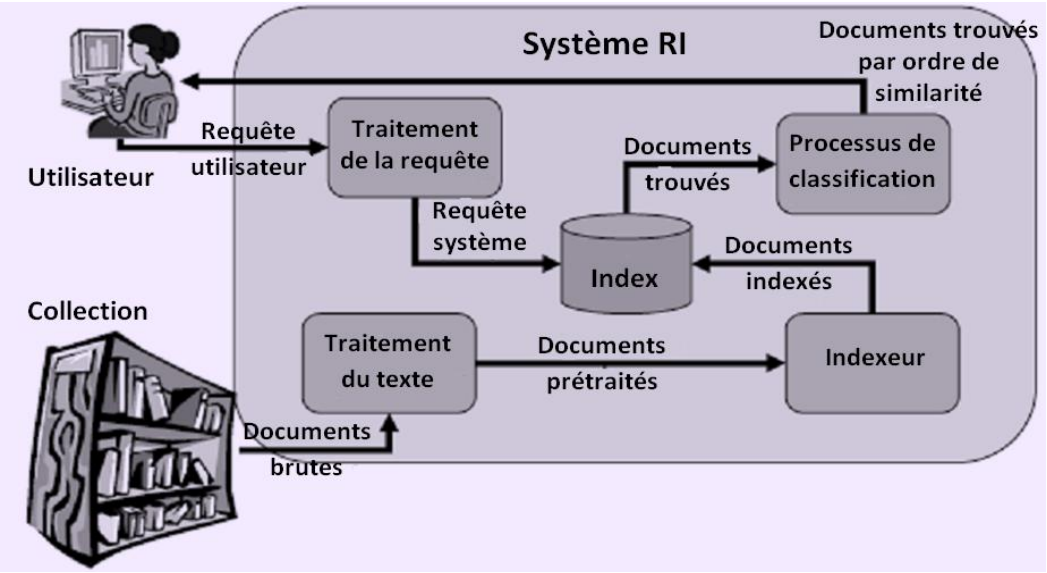


Figure 1: schéma général d'un système de recherche d'information **[ZIV05]**

Ce schéma montre que le processus de recherche se déroule en deux phases **[ZIV05]** :

- ***La première*** consiste à procéder à un découpage des documents contenus dans la collection dans le but d'en extraire des items figurant dans chaque document, et ensuite les indexer

- ***La seconde*** consiste à traiter la requête de l'utilisateur en se basant sur les index pour rechercher les documents qui s'apparentent avec cette requête.

De manière plus formelle, un système de recherche d'information est défini comme suit **[ZIV05]**:

- Considérons D une collection de documents, et Q un ensemble de requêtes.
- La fonction f : D x Q → R est la *fonction du score de pertinence*, qui associe à chaque paire constituée d'un document et d'une requête un score qui correspond au niveau de pertinence du document pour cette requête. Ceci est fait pour chaque document dans D et pour chaque requête dans l'ensemble Q.
- Lors de la première phase l'ensemble D est prétraité, et un index I est crée en conséquence. Durant la seconde phase, et ayant une requête q dans Q, I est utilisé pour produire une réponse à partir de l'ensemble D.

Le principal objectif de la recherche d'information réside dans la production de la réponse qui correspond à la requête utilisant un index compacté et avec un temps de réponse rapide (raisonnable). A titre d'exemple, nous ne pouvons pas accomplir la haute exactitude en utilisant un énorme index ou en traversant l'index entier pour chaque requête. Ceci produirait un temps de réponse très important voire même énorme en fonction de la taille de l'ensemble D.

En bref, un système de recherche d'information permet de trouver les documents pertinents à une requête utilisateur, à partir d'une base de documents volumineuse. Cette expression permet d'identifier les éléments clefs de la recherche d'information.

2.2.Approches possibles pour la réalisation de RI **[JYN06b]**

Il est possible d'imaginer quelques approches pour réaliser un système de RI :

- ***1ère approche*** : consiste à considérer une requête comme un ensemble de mots clés et un document pertinent comme un document qui contient plusieurs de ces mots clés. C'est une vue simpliste dont le processus de recherche n'est autre qu'un balayage des documents séquentiellement, et en recherchant les mots clés de la requête. Si certains mots clés figurent dans un document, alors il sera sélectionné comme faisant partie de la réponse. Cette approche est évidemment simple à réaliser, néanmoins elle nécessite un temps d'exécution très long. La plupart des systèmes existants s'appuis plutôt sur une approche différente basée sur l'indexation.

- ***2ème approche*** : basée sur l'indexation. L'indexation est un prétraitement effectué sur les documents et les requêtes en vu de construire une structure qui permet de retrouver très rapidement les documents incluant les mots demandés. Par rapport à l'approche précédente, elle est plus rapide puisqu'avec la structure d'index on peut directement accéder aux documents qui contiennent un mot donné. Grâce à cette approche également, il est possible d'exprimer des requêtes complexes exprimant des besoins d'informations complexes.
- ***3ème approche*** : plus efficace, basée sur le fichier inverse. Ce dernier est une structure de données permettant d'engendrer à partir d'un mot clé, les documents qui contiennent ce mot.

Dans les trois cas, un document et une requête sont représentés par un ensemble de mots clés ou termes.

A travers ces trois approches possible nous pouvons distingués quelques notions essentielles pour la définition d'un système de recherche d'information à savoir :

- Document, requête, pertinence, et structure d'index et fichier inverse.

2.3. Notions élémentaires

Dans sa définition la plus grossière, un système de recherche d'information est un mécanisme dont le but est de trouver les documents les plus pertinents face à un besoin en information exprimé par l'utilisateur sous forme d'une requête.

2.3.1. Document

Un document peut être un texte, un morceau de texte, une page web, une image, une bande vidéo, …etc. De manière générale, on appelle document toute unité qui peut constituer une réponse à un besoin utilisateur.

2.3.2. Requête

La requête c'est l'expression du besoin en information que l'utilisateur a à un moment donné. Cette requête est souvent formulée en langage naturelle.

2.3.3. Structure d'index

Un index est une structure de données permettant une organisation et un accès rapides, souvent modélisé sous forme de liste chainée. L'index permet pour chaque document d'identifier la liste des termes ou mots clés qui le comportent, ainsi que leurs fréquences

respectives d'apparition dans le document. Il respecte la structure illustrée par la figure qui suit :

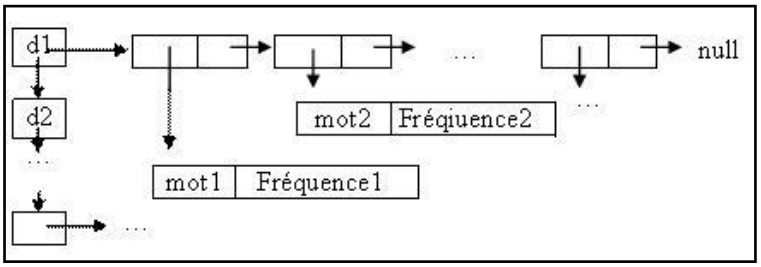


Figure 2: Structure d'index.

2.3.4. Fichier inverse

La plupart de systèmes de recherche d'information se basent sur les fichiers inverses. Ces fichiers sont représentés par une liste de mots clés. Chaque mot clé est suivi par la liste des documents qui le comporte et la fréquence respective d'apparition de ce mot clés dans chacun des documents.

Les fichiers inverses sont très performants en terme de temps de réponse, néanmoins ils consomment énormément d'espace de stockage (ces fichiers sont parfois aussi volumineux que le fichier de données).

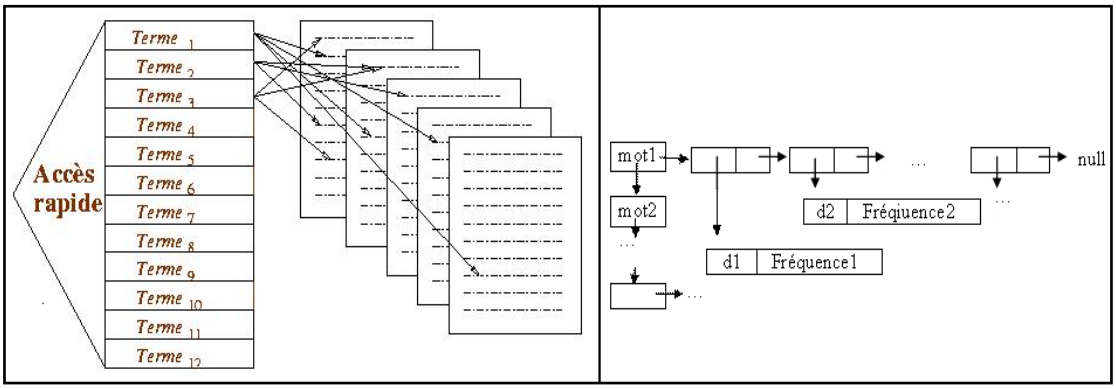


Figure 3: Structure de l'index inverse.

2.3.5. Similarité entre un document et une requête **[RAJ98]**

Évaluer des similarités entre entités textuelles est un des problèmes centraux dans plusieurs disciplines comme l'analyse de données textuelles, la recherche documentaire ou l'extraction de connaissances à partir de données textuelles (Text Mining). Dans chacun de ces domaines, les similarités sont en effet utilisées pour une large variété de traitements :

- en analyse de données textuelles (ADT), les similarités sont utilisées pour la description et l'exploration de données, pour l'identification de structures cachées et pour la prédiction ;
- en recherche documentaire (RD), l'évaluation des similarités entre documents et requêtes est utilisée pour identifier les documents pertinents par rapport à des besoins d'information exprimés par les utilisateurs;
- en Text Mining (TM), les similarités sont utilisées pour produire des représentations synthétiques de vastes collections de documents, dans le cadre de procédures d'extraction d'information à partir de données textuelles.

Les techniques mises en œuvre pour calculer les similarités varient bien évidemment selon les disciplines, mais elles s'intègrent cependant le plus souvent dans une même approche générale en deux temps :

1. Les entités textuelles sont tout d'abord associées à des représentations spécifiques qui vont servir de base au calcul des similarités ; La nature précise des représentations utilisées dépend fortement du domaine d'application considéré. Dans tous les cas, les structures associées sont représentées sous la forme d'éléments d'un espace vectoriel de grande dimension : espace de représentation.
2. Un modèle mathématique est choisi pour mesurer, dans l'espace de représentation, les proximités qui seront utilisées pour estimer les similarités entre entités textuelles.

Une technique d'évaluation ou une mesure de similarité permet donc de mesurer la ressemblance entre la requête et un document quelconque de la collection. C'est sur la base de cette mesure que se définit la pertinence calculée (niveau système). Il existe dans la littérature plusieurs formules ou mesures de calcul de la similarité :

Similarité Sim(X, Y)	Vecteur de termes	Vecteur de termes pondérés
Coefficient de Dice	$2\frac{\lvert X \cap Y \rvert}{\lvert X \rvert + \lvert Y \rvert}$	$\frac{2\sum_{i=1}^{t} x_i \cdot y_i}{\sum_{i=1}^{t} x_i^2 + \sum_{i=1}^{t} y_i^2}$
Coefficient de Cosinus	$\frac{\lvert X \cap Y \rvert}{\lvert X \rvert^{1/2} \lvert Y \rvert^{1/2}}$	$\frac{\sum_{i=1}^{t} x_i \cdot y_i}{\sqrt{\sum_{i=1}^{t} x_i^2 \cdot \sum_{i=1}^{t} y_i^2}}$
Coefficient de Jaccard	$\frac{\lvert X \cap Y \rvert}{\lvert X \rvert + \lvert Y \rvert - \lvert X \cap Y \rvert}$	$\frac{\sum_{i=1}^{t} x_i \cdot y_i}{\sum_{i=1}^{t} x_i^2 + \sum_{i=1}^{t} y_i^2 - \sum_{i=1}^{t} x_i \cdot y_i}$

Tableau 1 : Mesure de similarité.

Ce tableau synthétise les trois mesures de similarités des plus utilisées dans la recherche d'information et particulièrement pour le modèle vectoriel. La première colonne du tableau comporte les formules pour chaque mesure dans le cas de vecteurs binaires tandis que la seconde concerne le cas ou les termes sont pondérés. Dans les deux cas l'évaluation de ces formules donne des valeurs dans l'intervalle [0 – 1].

2.3.6. Le système SMART

2.3.6.1. Principe

Le système SMART (*System for the Mechanical Analysis and Retrieval of Text*), aussi appelé *Salton's Magic Automatic Retrieval Technique*, est un système de RI expérimental. Il utilise le modèle vectoriel. Ce système a été construit entre 1968 et 1970. Dans les années 1980, il a été réécrit et réorganisé. Les travaux sur SMART ont été dirigés par Prof. G. Salton. Dans ce système, nous pouvons remarquer les faits suivants:

- C'est le premier système de RI expérimental qui est robuste. Il peut manipuler un grand nombre de documents. Pour la version actuelle, SMART peut traiter jusqu'à 500 M octets de documents. Pour beaucoup d'applications, cette capacité est suffisante.
- SMART a été programmé avec un souci d'efficacité. Les opérations d'indexation et de recherche ont été optimisées. Le temps de traitement est raisonnable. Par exemple, pour indexer un corpus de 150 M octets, il faut environ 1 heure. Pour effectuer une recherche sur ce corpus, il faut 2-3 seconds en fonction de la taille de la requête. Cette vitesse est tout à fait acceptable.

- SMART est un système flexible. Il a été conçu dans le but de tester des approches en pratique. Les codes sont écrits de telle manière qu'on peut les modifier assez facilement.
- Dans SMART, certaines nouvelles techniques ont été testées, telles que, la rétroaction de pertinence, la classification de documents, etc.

Beaucoup de systèmes de RI actuels ont grandement bénéficié de SMART. Certains ont repris les mêmes idées, alors que d'autres sont développés directement à partir de SMART.

SMART a été étudié dans le but de comprendre un système de RI concret et de s'y inspirer pour concevoir nos propositions. Ce système sera présenté à travers ses différents traitements : indexation de document, indexation de requête et l'évaluation d'une requête.

2.3.6.2. Processus d'indexation

L'indexation des documents et des requêtes est illustrée par la figure suivante :

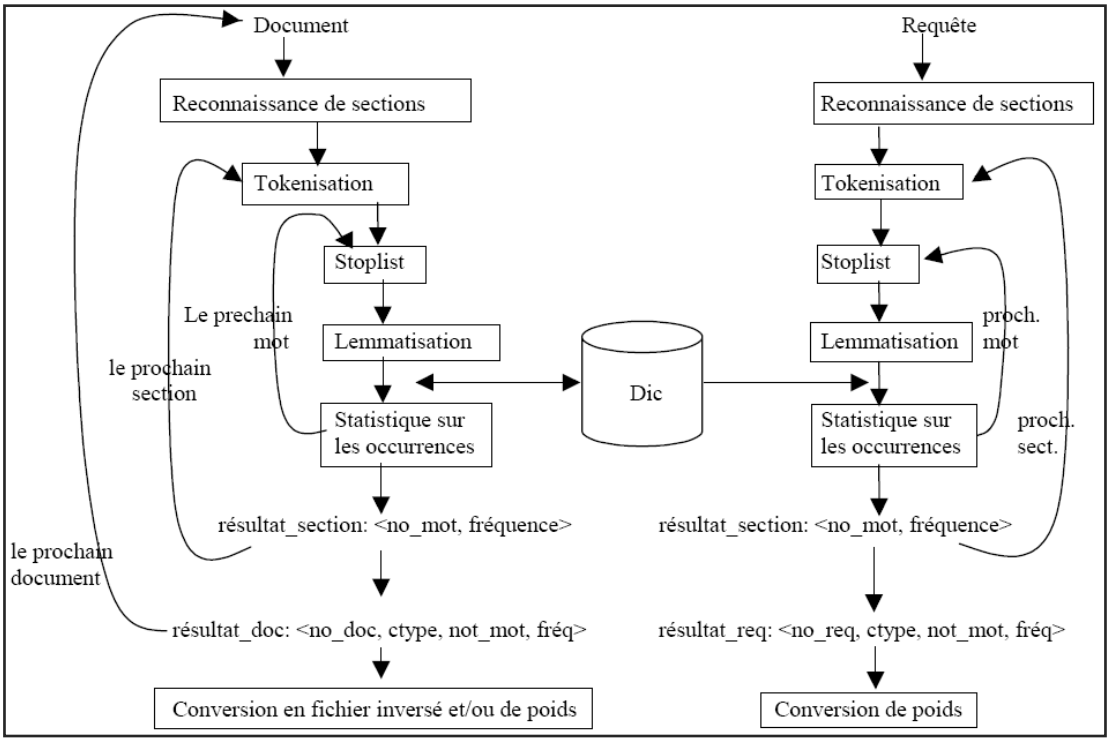


Figure 4 : Schéma d'indexation

La partie gauche correspond à l'indexation des documents, et la droite à l'indexation de la requête. Les deux traitements sont très similaires, à quelques détails près.

- **Reconnaissance de sections**: De même pour les documents et la requête, le premier traitement est la séparation de champs (ou la reconnaissance de section). Cela se fait

par des marqueurs qui annoncent le début de différentes sections. Par exemple, on peut avoir le marqueur <Title> pour annoncer que les lignes suivantes correspondent au titre. <Date> marque une date, <Body> marque le début du corps du document, etc. Dans Smart, on peut définir un ensemble de marqueurs, et les actions d'indexation correspondantes.

- **Tokenisation**: Pour chaque section à indexer, il faut d'abord la découper en mots. Cela consiste à reconnaître les séparateurs de mots, comme ",", ";", l'espace, etc. Le résultat de ce traitement est une suite de "tokens" ou mots (dans un sens large).
- **Stopliste**: Les mots trouvés sont comparés aux éléments de la « stoplist » qui contient l'ensemble des mots vide de sens. Les autres mots sont soumis aux traitements suivants. Ce traitement est facultatif. On peut très bien décider que tous les mots sont significatifs, et doivent être gardés.
- **Lemmatisation**: Ce traitement consiste à transformer une forme de mot en une certaine forme standard. Trois options sont offertes dans SMART: aucune lemmatisation, enlever le "s" à la fin, ou utiliser un arbre trie pour enlever la terminaison. Après la lemmatisation, la forme standard obtenue est comparée avec un dictionnaire. Dans ce dictionnaire, on stocke tous les index, et à chaque index, on attribue un numéro d'identification. Ainsi, on peut avoir la structure suivante pour le dictionnaire: <no_mot, nature, mot>, où no_mot est le numéro d'identification, mot est le mot correspondant, et nature désigne la nature de ce mot (un chiffre, un mot normal, un nom propre, …etc.). La comparaison du résultat de la lemmatisation a pour but d'obtenir le no_mot pour le mot. Si ce mot a été déjà rencontré (et donc stocké dans le dictionnaire), on reprend simplement son no_mot; sinon, le système créera un numéro automatiquement pour ce mot, et ajoute une entrée dans le dictionnaire pour ce nouveau mot. Dans le cas des requêtes, si ce mot n'existe pas dans le dictionnaire, il n'y sera pas rajouté, il sera considéré "inconnu" par le système.
- **Statistiques**: Pour chaque mot, on doit calculer la statistique de sa fréquence d'occurrence dans le document. Ainsi, à chaque occurrence d'un mot, on ajoute 1 dans sa fréquence.
- **Itérations:** Dans la figure, il y a 3 itérations dans l'indexation des documents: itération pour traiter chaque mot, itération pour chaque section, et itération pour chaque

document. Ce schéma traite d'une requête uniquement, mais il est possible de réitérer le processus pour tenir compte de plusieurs requêtes.

- **Résultat:** Le résultat d'indexation pour un document est un ensemble de <no_doc, ctype, no_mot, fréq>. Si le document contient 100 mots significatifs distincts, alors il aura 100 entrées.
- **Conversion du résultat**: La conversion se fait seulement quand tous les documents ont été indexés. On peut faire deux sortes de conversions: convertir des vecteurs en fichier inversé (cette conversion implique uniquement les documents), ou convertir les poids de termes. La première conversion consiste à trier à nouveau les résultats de l'indexation. Initialement, ils sont triés dans l'ordre croissant de no_doc. Dans un fichier inversé, ils seront triés dans l'ordre croissant de no_mot. La seconde conversion consiste à transformer le poids de chaque terme dans chaque document en un nouveau poids. La conversion dans SMART se fait en 3 étapes:
  - transformation de tf: on peut garder la fréquence obtenue comme la tf (*term frequency*), ou bien on peut choisir de la normaliser en la divisant par la fréquence maximale dans le document, ou bien on peut la convertir en une valeur binaire (0 ou 1).
  - La tf obtenue dans l'étape 1 est multiplié par idf (*inversed document frequency*). À ce stade, on peut choisir différentes façons de calculer idf. Par exemple, log (N/n), ou bien 1/n, etc.
  - La troisième étape est la normalisation. On peut normaliser le poids obtenu en le divisant par la valeur maximale, ou bien par la formule Cosinus.

2.3.4.3. Evaluation des requêtes

L'évaluation d'une requête utilise le fichier inverse et le résultat d'indexation de la requête qui est une suite de no_mot et de son poids. SMART utilise le produit interne comme mesure de similarité:

$$\text{Sim}(d, q) = \Sigma_i (p_i * q_i)$$

Où p_i et q_i sont respectivement les poids d'un mot dans le document d et dans la requête q.

SMART implante un calcul global pour tous les documents à la fois. L'algorithme est comme suit:

Début

Initialiser Sim (d_j, q) à 0 pour tout d_j;

Pour chaque mot (no_mot_i) dans la requête q (avec un poids q_i)

Faire

Trouver dans le fichier inverse tous les documents d_j incluant ce mot, avec le poids p_i;

Pour chaque d_j dans cet ensemble

Faire

$Sim(d_j, q) = Sim(d_j, q) + p_i * q_i$

Fait.

Fait.

Fin.

À la fin de cet algorithme, on obtient la valeur de Sim pour chaque document. Cette méthode d'évaluation globale est plus efficace qu'une évaluation séquentielle. Dans la méthode séquentielle, on doit comparer chaque document séparément avec la requête, et utiliser la formule théorique de Sim pour calculer la similarité. Le plus grand désavantage est qu'on doit considérer tous les documents, même ceux qui ne contiennent aucun mot de la requête. Dans l'évaluation globale utilisant le fichier inverse, on ne considère que les documents qui contiennent au moins un mot de la requête. Le nombre de documents à considérer est beaucoup plus réduit.

2.3.7. Pertinence

Le but de la RI est de trouver uniquement les documents pertinents. La pertinence est une notion très complexe. De façon générale dans un document pertinent, l'utilisateur doit pouvoir trouver les informations dans il a besoin. C'est sur cette même notion que le système devra juger si un document doit être donné à l'utilisateur comme réponse.

La pertinence peut être définie au moyen de ces quelques définitions :

- La correspondance entre un document et une requête, une mesure d'informativité du document à la requête;
- Le degré de relation (chevauchement, relativité, …) entre le document et la requête;
- Le degré de satisfaction qu'apporte un document, qui a un rapport avec le besoin de l'utilisateur;
- Une mesure d'utilité du document pour l'utilisateur;

Même dans ces définitions la notion de pertinence reste vague, ceci est dû au fait que les utilisateurs des systèmes de RI ont des besoins très variés mais aussi des critères différents pour juger si un document est pertinent ou non. Donc, la notion de pertinence est utilisée pour

recouvrir un très vaste éventail de critères et de relations. Beaucoup de travaux ont été menés sur cette notion pour mettre à l'évidence qu'elle n'est pas une relation isolée entre la requête et le document mais elle fait aussi appel au contexte de jugement. Ainsi, Tefko Saracevic propose la définition unificatrice suivante :

La pertinence est la A de B existant dans C et D, jugé par E

Où *A* est un intervalle de mesure pour le degré de pertinence (qui peut être binaire ou multi-value comme de 1 à 10), *B* est l'aspect de la pertinence absolue, *C* un document, *D* le contexte dans lequel la pertinence est mesurée, et *E* le juge. Donc, pour Saracevic, il y a une mesure de pertinence (*A*), et une notion de pertinence absolue (*B*). Cette relation de pertinence absolue peut exister, ne pas exister, ou exister à un certain degré, entre un document et le contexte de recherche. Il faut comprendre ici que le contexte *D* contient non seulement l'expression du besoin d'information qui est la requête, mais aussi tous les facteurs contextuels qui influencent le jugement de pertinence.

Bien que cette définition ne fournit pas plus de précision sur la nature de la pertinence, ni la façon dont elle doit être déterminée, elle identifie au moins les acteurs sur cette notion. Nous pouvons voir que ces acteurs sont assez nombreux et souvent mal identifiés (notamment les facteurs contextuels). Pour mieux cerner la nature de la pertinence, celle-ci est souvent étudiée dans le processus global de la recherche de l'information ou de communication.

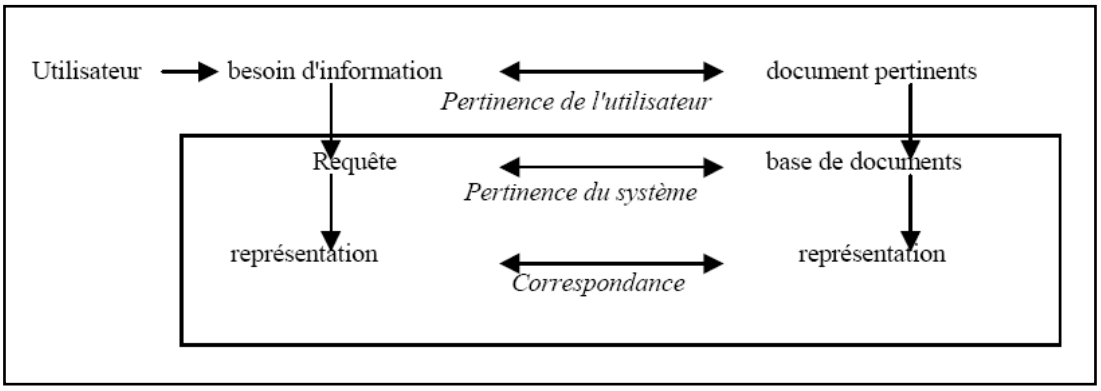


Figure 5 : Opération et environnement de la RI.

A travers ce schéma, le processus de recherche est initié par un besoin d'information de l'utilisateur exprimé sous forme d'une requête qui va subir une transformation à l'intérieur du système pour qu'elle puisse être considérée. De même que les documents de la base qui eux même subissent la même transformation, les relations qu'on peut déterminé à chaque niveau ne sont pas identiques. Un bon système de RI devra donner une évaluation de

correspondance qui reflète bien la *pertinence système*, et qui à son tour correspond bien au jugement de *pertinence de l'utilisateur*.

2.3.7.1. Niveau utilisateur

A ce niveau l'utilisateur a un besoin d'information dans sa tête qu'il exprime à l'aide de sa requête tout en espérant obtenir les documents qui répondent à ce besoin. La relation entre le besoin d'information dans la tête de l'utilisateur et les documents attendus constitue la relation de pertinence absolue (idéale).

2.3.7.2. Niveau système

A ce niveau, le système répond à la requête formulée par l'utilisateur par un ensemble de documents trouvés dans la base de documents qu'il possède. La requête formulée par l'utilisateur n'est qu'une description partielle de son besoin d'information dans la majeure partie des cas. Beaucoup d'études ont montré qu'il est très difficile, voire impossible, de formuler une requête qui décrit complètement et précisément un besoin d'information. Du côté des documents, il y a aussi un changement entre les deux niveaux: les documents qu'on peut retrouver sont seulement les documents inclus dans la base de documents. On ne peut souvent pas trouver des documents parfaitement pertinents à un besoin. Il arrive souvent qu'aucun document pertinent ne soit inclus dans la base.

2.3.7.3. Niveau interne : Correspondance

La requête formulée par l'utilisateur (souvent en langue naturelle) ne peut pas se comparer directement avec des documents décrits en langue naturelle eux aussi. Il faut donc créer des représentations internes pour la requête et pour les documents. Ces représentations doivent être manipulables par l'ordinateur. Le processus de création de ces représentations est appelé l'indexation. Pour déterminer si la représentation d'un document correspond à celle de la requête, un processus d'évaluation est nécessaire dans le but de mesurer le degré de correspondance du document et de la requête. Différentes méthodes d'évaluation ont été développées, en relation avec la représentation des documents et des requêtes.

2.3.8. Technique d'indexation

L'utilisation des index remonte au 15ème siècle, peu après que l'imprimerie fut inventée. Depuis lors, ils sont utilisés notamment pour des livres et dans des bibliothèques. Les index (ou termes d'indexation) jouent un rôle primordial dans la RI. Ils déterminent avec quels mots on peut retrouver un document.

Le premier problème dans l'indexation fut de déterminer les éléments que l'on doit choisir comme index. Dans les premiers projets sur la RI, on s'interrogeait sur les questions suivantes:

- indexation manuelle ou automatique ?
- vocabulaire libre ou contrôlé ?
- quels mots à rajouter dans la stopliste ?
- quelles méthodes de troncature ?

La première approche à l'indexation automatique KWIC ou *KeyWord In Context*, fut introduite à l'*International Conference on Scientific Information* (ICSI) en 1958 par Luhn. On s'aperçut très tôt (dans le projet Cranfield) que les mots vides de sens (ceux de la stopliste) devraient être systématiquement éliminés. La fréquence d'occurrence de mots a été utilisée comme critère de sélection d'index.

Il fut ensuite question de pondérer les index. Les méthodes statistiques furent exploitées dès le début de la RI. Luhn proposa une méthode basée sur la fréquence de termes dans le document (term frequency –tf). Plus tard, cette mesure a été étendue à tf*idf (idf – inversed document frequency) afin de tenir compte de la spécificité d'un terme pour un document. D'autres méthodes de pondération, telles que 2-Poisson, se sont développées plus tard.

Dans la sélection et la pondération des index, deux aspects ressortent : spécificité (est-ce que les index sont spécifiques à un document ?) et exhaustivité (est-ce que les index couvrent tout le contenu d'un document ?). Une bonne méthode d'indexation doit faire un compromis entre ces deux aspects. Cet équilibre se retrouve notamment dans le schéma de pondération tf*idf.

Dans les premières études en RI, les techniques de traitement de la langue naturelle utilisées se sont limitées à l'analyse morphologique simple, plus spécifiquement pour la troncature ou la lemmatisation de mot. Durant le développement ultérieur, il y a eu souvent des questionnements sur le rôle du traitement automatique de langue naturelle (TALN) en RI, et plus spécifiquement dans l'indexation. Le débat à ce sujet était intense dans les années 1980, où d'une part, la popularité de l'intelligence artificielle incitait les chercheurs à utiliser des techniques du TALN dans la RI, et d'autre part, il manquait de tests avec des collections donnant des résultats convaincants.

3. Evaluation des systèmes de recherche d'information [JYN06b]

Le but de la RI est de trouver des documents pertinents à une requête, et donc utiles pour l'utilisateur. La qualité d'un système doit être mesurée en comparant les réponses du système avec les réponses idéales que l'utilisateur espère recevoir. Plus les réponses du système correspond à celles que l'utilisateur espère, mieux est le système.

3.1. Corpus de test

Pour arriver à une telle évaluation, on doit connaître d'abord les réponses idéales de l'utilisateur. Ainsi, l'évaluation d'un système s'est faite souvent avec certains corpus de test. Dans un corpus de test, il y a:

- Un ensemble de documents (collection)
- Un ensemble de requêtes
- La liste des documents pertinents pour chaque requête.

Pour qu'un corpus de test soit significatif, il faut qu'il possède un nombre de documents assez élevé. Les premiers corpus de test développés dans les années 1970 renferment quelques milliers de documents. Les corpus de test plus récents (par exemple, ceux de TREC) contiennent en général plus 100 000 documents (considérés maintenant comme un corpus de taille moyenne), voir des millions de documents (corpus de grande taille).

L'évaluation d'un système ne doit pas se reposer seulement sur une requête. Pour avoir une évaluation assez objective, un ensemble de quelques dizaines de requêtes, traitant de sujets variés, est nécessaire. L'évaluation du système doit tenir compte des réponses du système pour toutes ces requêtes.

Finalement, il faut avoir les réponses idéales de l'utilisateur pour chaque requête. Le dernier élément d'un corpus de test fournit cette information. Pour établir ces listes de documents pour toutes les requêtes, les utilisateurs (ou des testeurs simulant des utilisateurs) doivent examiner chaque document de la base, et juger s'il est pertinent. Après cet exercice, on connaît exactement quels documents sont pertinents pour chaque requête. Pour la construction d'un corpus de test, les jugements de pertinence constituent la tâche la plus difficile.

3.2.Précision et Rappel

Plusieurs mesures de performance peuvent être appliquées aux processus et aux algorithmes de recherche d'information dans le but de connaitre l'efficience de ces derniers pour cette recherche. La plupart des mesures se basent principalement sur deux autres mesures : Le *rappel* et la *précision.* Ces deux mesures constituent des métriques de base pour la qualité d'un système de recherche d'information.

Soit D_q l'ensemble des documents de D qui sont pertinents pour la requête q. Cet ensemble peut être construit sur la base d'une évaluation booléenne de la pertinence ou bien sur une évaluation réelle en tenant compte d'une fonction d'évaluation comme pour le modèle vectoriel. Comme dans la majeure partie des cas, la recherche est face à une large collection de documents, les résultats seront exprimés dans une liste ordonnée selon le degré de pertinence des documents qu'elle comporte. Soit L_q cette liste.

Notons que D_q est un sous ensemble de la collection D, ainsi que L_q qui est aussi un sous ensemble de la collection D.

Le rappel est défini selon la formule suivante : $\frac{|L_q \bigcap D_q|}{|D_q|}$

La précision quant à elle est définie par cette formule : $\frac{|L_q \bigcap D_q|}{|L_q|}$

Idéalement, on voudrait qu'un système donne de bons taux de précision et de rappel en même temps. Un système qui aurait 100% pour la précision et pour le rappel signifie qu'il trouve tous les documents pertinents, et rien que les documents pertinents. Cela veut dire que les réponses du système à chaque requête sont constituées de tous et seulement les documents idéaux que l'utilisateur a identifiés. En pratique, cette situation n'arrive que très rarement. Plus souvent, on peut obtenir un taux de précision et de rappel aux alentours de 30%.

Les deux métriques ne sont pas indépendantes. Il y a une forte relation entre elles. Quand l'une augmente, l'autre diminue. Il n'est pas opportun de juger la qualité d'un système en utilisant seulement une des deux métriques. En effet, il est facile d'avoir 100% de rappel: il suffirait de donner toute la base comme réponse à chaque requête. Cependant, la précision dans ce cas-ci serait très basse. De même, on peut augmenter la précision en donnant très peu de documents en réponse, mais ça sera fait au détriment du rappel. Il faut donc utiliser les deux métriques ensemble.

Les mesures de précision-rappel ne sont pas statiques non plus (c'est-à-dire qu'un système n'a pas qu'une mesure de précision et de rappel). Le comportement d'un système peut varier en faveur de précision ou en faveur de rappel (au détriment de l'autre métrique). Ainsi, pour un système, il existe une courbe de précision-rappel qui a en général la forme suivante:

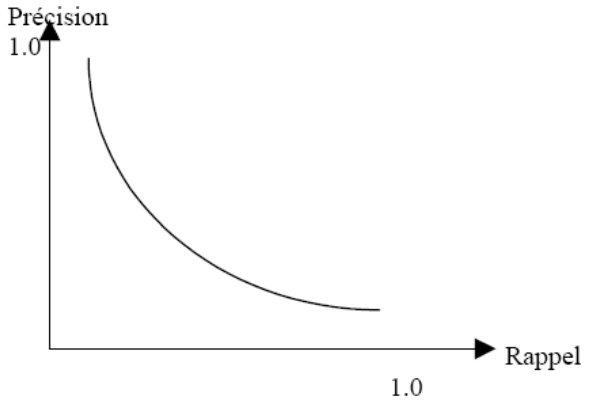


Figure 6 : Courbe Précision-Rappel

4. Relation entre domaines

4.1. Recherche d'information vs recherche dans le web **[ZIV05]**

Il est nécessaire de distinguer la recherche d'information classique de la recherche d'information dans le web, malgré le fait que les deux processus se ressemblent étroitement. La principale différence réside dans la taille de l'espace de recherche. Aussi grande qu'elle puisse être dans la recherche classique, elle ne pourra atteindre l'énorme volume de celui considéré pour la recherche dans le web. Cette quantité d'information disponible sur le web et aussi sa distribution sur plusieurs localités et ainsi sur plusieurs ordinateurs, influe considérablement sur les algorithmes de recherche pour assurer une efficacité en termes de temps et d'exploitation de l'espace. Une autre différence réside dans la qualité des informations. Dans le processus de recherche classique, les informations sont considérées « propres » contrairement aux informations dans le web qui sont fortement bruitées. Même l'issue de la recherche diffère dans les deux processus. Le premier met à la disposition du demandeur un ensemble de documents qui contiennent l'information ou le besoin exprimé. Par contre, le résultat de la recherche dans le web est exprimé par un ensemble de liens vers des pages qui correspondent au besoin.

4.2. Recherche d'information vs recherche de données **[ZIV05]**

Un processus de recherche d'information n'est pas un processus de recherche de données. Les systèmes de recherche d'information se basent sur un besoin en information

exprimé par l'utilisateur sous forme de mots clés, souvent exprimés en langage naturel, pour déterminer les documents ou les contenus qui s'apparient approximativement avec ce qui est demandé, et ordonnés par rapports à leurs pertinences pour cette requête. La recherche de données, contrairement à cela, travaille sur des données structurées en table, accessibles par des requêtes en langage SQL ou en langage algébrique pour retrouver l'appariement exact entre la requête et son résultat.

4.3. Recherche d'information et système de question – réponse **[JYN06b]**

Un système QR permet de répondre aux questions relatives à un petit domaine. Par exemple, on peut poser la question "quelle version de Word est disponible sous Windows 98?" à un système spécialisé sur le marché de logiciel. Pour cela, il faut créer une modélisation du domaine d'application dans lequel les concepts ou objets sont reliés par des relations sémantiques. Ce modèle permettra de retrouver le concept ou l'objet et ainsi donner une réponse directe à la question. Pour notre exemple, la réponse peut être "Word 95 et Word 98", par exemple.

On voit ici qu'il y a une différence sur la nature de réponse entre les deux types de système. Dans la RI, c'est une réponse indirecte à une question: on identifie les documents dans lesquels l'utilisateur peut trouver des réponses directes à sa question. Tandis que dans un système QR, on fournit une réponse directe.

Il y a des tentatives de rapprocher la RI vers des systèmes QR, mais cela s'avère très difficile. La raison principale est que la RI s'applique en général à tous les domaines sans restriction. Il est impossible, dans ce cas, de créer un modèle nécessaire pour déduire la réponse directe à une question dans un système QR. Dans certains contextes très spécialisés, la RI incorpore une base de connaissances. Elle utilise aussi des raisonnements pour déduire si un document peut être pertinent ou pas. Donc, le fonctionnement de ce type de RI ressemble un peu plus à celui d'un système QR.

Une tentative plus restreinte consiste à raffiner la notion de document dans la réponse: au lieu de fournir un document complet comme une réponse, on essaie d'identifier un passage dans le document (passage Retrieval). C'est une étape qui diminue un peu la distance entre la RI et la QR. Mais la différence fondamentale reste la même.

5. Historique sur la recherche d'information **[JYN06a]**

5.1. Naissance

Le domaine de la recherche d'information remonte au début des années 1950, peu après l'invention des ordinateurs. Comme plusieurs autres domaines informatiques, les pionniers de l'époque étaient enthousiastes à utiliser l'ordinateur pour automatiser la recherche des informations, qui dépassaient la capacité humaine : il y avait une explosion d'information après la deuxième guerre mondiale.

Le nom de « recherche d'information » (information Retrieval) fut donné par Calvin N. Mooers en 1948 pour la première fois quand il travaillait sur son mémoire de maîtrise **[MOO48]**. La première conférence dédiée à ce thème – International Conference on Scientific Information - s'est tenue en 1958 à Washington. On y comptait les pionniers du domaine, notamment, Cyril Cleverdon, Brian Campbell Vickery, Peter Luhn, etc.

Les premiers problèmes qui intéressaient les chercheurs portaient sur l'indexation des documents afin de les retrouver. Déjà à la « International Conference on Scientific Information », Luhn avait fait une démonstration de son système d'indexation KWIC qui sélectionnait les index selon la fréquence des mots dans les documents, et filtrait des mots vides de sens en employant des « stoplistes ». C'est à cette période que le domaine de RI est né.

5.2. Quelques projets expérimentaux de l'histoire de la RI

Dès les premiers travaux, l'aspect d'expérimentation occupait une place particulière. Pour n'importe quelle méthode, on voulait la tester expérimentalement afin de connaître son effet en réalité. Cette tradition bien ancrée dans la communauté de la RI a l'avantage de se prémunir des spéculations, mais elle a aussi souvent teinté la communauté de pragmatisme.

5.2.1. Projet Cranfield **(dirigé par Cyril Cleverdon, 1957-1967) [CLE67]**

Dans la première phase de ce projet, il visait à tester l'efficacité de différentes façons d'indexer et de rechercher des documents. Ces tests sont vigoureusement contrôlés. Une collection de test est constituée d'un ensemble d'articles (18 000 dans Cranfield I) et un ensemble (1 200) de requêtes. Ces requêtes sont évaluées par des experts afin de déterminer les réponses souhaitées-les articles pertinents. Les résultats d'une recherche automatique sont comparés avec les réponses souhaitées pour mesurer la performance en termes de précision et de rappel.

Le projet Cranfield a une influence marquante sur toute l'histoire de la RI. On utilise encore aujourd'hui les mêmes principes d'évaluation pour les systèmes de RI.

5.2.2. *Projet MEDLARS* MEDical Literature Analysis and Retrieval System (F. Wilfrid Lancaster, complete en 1968) [LAN68]

Comme l'indique son nom, les documents de la collection appartiennent au domaine biomédical. Ces documents sont indexés manuellement, avec un vocabulaire contrôlé. Les résultats sont évalués en termes de précision et de rappel. Les résultats de ce projet montrent qu'en utilisant une approche automatique, il est possible d'atteindre la même performance qu'avec une indexation manuelle et un vocabulaire contrôlé. Une analyse des résultats a aussi montré que l'utilisation d'un vocabulaire contrôlé et de l'indexation manuelle étaient largement responsable des cas d'échecs dans la recherche de documents pertinents, qui peuvent être évités par l'approche automatique.

5.2.3. SMART (Gerard Salton, 1ière version 1961-1965) [SAL71]

Dans ce projet, une série d'expérimentations a été menée, portant sur divers sujets comme :

- la comparaison entre l'indexation manuelle et l'indexation automatique ;
- le problème de recherche d'information interactive et la rétroaction de pertinence (relevance feedback);
- l'architecture des systèmes de RI ;
- l'utilisation du modèle vectoriel ;
- le regroupement de documents (ou clustering), … etc.

Le système SMART fut réécrit dans les années 1970 et 1980 par E. Fox et C. Buckley. Ce système a été, et est encore, utilisé par de nombreux chercheurs pour des expérimentations en RI. Le système SMART est sans doute le système qui a eu le plus grand impact sur l'histoire de la RI.

5.2.4. Projet STAIRS STorage And Information Retrieval System (Blair et Maron) [BLA85]

Les documents dans ce cas ont trait au domaine du droit. L'indexation automatique utilise la troncation de suffixes, et la recherche exploite une liste de synonymes. Contrairement aux expérimentations antérieures qui utilisaient de petites collections, les tests de Blair et Maron portent sur une collection de taille réaliste – 40 000 documents totalisant

350 000 pages. Le résultat montre que la performance de moins de 20% de rappel est insuffisante dans le domaine du droit pour lequel le rappel est très important.

5.2.5. TREC Text REtrieval Conference, (D. Harman, 1992 -) [HAR92]

Cette série de conférences a pour objectif de tester des méthodes et des systèmes de RI avec des collections de plus grandes tailles. Elle est organisée annuellement. Les tâches (tracks) changent d'une année à une autre, mais elles reflètent bien les intérêts des chercheurs et les besoins réels. Au fil des années, il y a eu la RI ad hoc (la tâche classique de RI – soumettre des requêtes sur une collection statique), le filtrage de l'information, la RI non anglaise (en espagnol, français, chinois) et translinguistique (retrouver des documents dans une langue différente de celle de la requête), la question-réponse, la RI multimédia (vidéo et parole), etc. Ces conférences attirent chaque année des chercheurs universitaires et industriels. Les conférences TREC ont grandement contribué au développement récent de la RI, en fournissant des collections de tests réalistes, et en offrant une nouvelle méthodologie d'évaluation. Elles ont grandement stimulé le domaine de RI.

5.3. Améliorations techniques

De nombreuses études ont porté sur des améliorations possibles de techniques d'indexation et de recherche. Parmi les tentatives les plus marquantes, on retrouve notamment:

5.3.1. Rétroaction de pertinence (relevance feedback)

Cette technique vise à étendre la portée de la recherche en intégrant les termes issus des documents pertinents, ou des documents en tête de la liste de réponses trouvées automatiquement.

5.3.2. Expansion de requête

Cette technique vise à renforcer l'expression de la requête de l'utilisateur (qui est souvent très courte) par l'intégration des termes reliés (soit en exploitant un thésaurus, soit en utilisant un calcul basé sur des cooccurrences).

5.3.3. Regroupement (clustering) des documents

Il vise à créer une structure entre les documents selon leurs similarités. Cette structure peut aider à la fois la recherche et la présentation des résultats.

6. *Les modèles de recherche d'information* **[BRA02]**

Si c'est l'indexation qui choisit les termes pour représenter le contenu d'un document ou d'une requête, c'est au modèle de leur donner une interprétation. Étant donné un ensemble de termes pondérés issus de l'indexation, le modèle remplit les deux rôles suivants:

- créer une représentation interne pour un document ou pour une requête basée sur ces termes;
- définir une méthode de comparaison entre une représentation de document et une représentation de requête afin de déterminer leur degré de correspondance (ou similarité).

Le modèle joue un rôle central dans la RI. C'est le modèle qui détermine le comportement clé d'un système de RI

Baeza Yates et Ribeiro Neto **[BRA02]**, ont affirmé en 1999 que chaque modèle de recherche d'information est caractérisé par les quatre composants suivants :

- ✓ Un ensemble D qui représente la ou les collection(s) de documents considérées lors de la recherche.
- ✓ Un ensemble Q représentant les informations dont l'utilisateur peut avoir besoin, référencées comme des requêtes.
- ✓ Un Framework F dans lequel les documents et les requêtes peuvent être modelés.
- ✓ Une fonction *R* (*qi; dj*) qui attribue un nombre réel à chaque combinaison de requête q dans Q et une représentation d'un document d dans l'ensemble D.

6.1. Le modèle score matching **[WAL79], [JYN06c]**

C'est peut-être le premier "modèle" utilisé dans la RI. L'idée est assez primitive et intuitive: Un document est représenté par un **ensemble** de termes pondérés par leur fréquence. Une requête est aussi un ensemble de termes, pondérés à 1. Le degré de correspondance est la somme des fréquences des termes de la requête dans le document:

$$R(d, q) = \Sigma_i f_i$$

Où f_i est la fréquence d'un terme de q dans le document d.

La valeur R ainsi calculée est appelée le "matching score" ou score d'appariement. En réalité, cela est équivalent à parcourir le document, et de voir combien de fois les termes de la

requête apparaissent dans ce document. Plus ce matching score est élevé, plus on considère que le document correspond à la requête, et donc plus il sera classé haut dans la réponse.

Ce modèle est primitif car il utilise directement le résultat de l'indexation sans aucune réorganisation ou modélisation.

6.2. Le modèle booléen **[WAL79], [JYN06c]**

Dans ce modèle, un document est représenté comme une conjonction logique de termes (non pondérés), par exemple,

$$d = t_1 \wedge t_2 \wedge \ldots \wedge t_n$$

Une requête est une expression logique quelconque de termes. On peut utiliser les opérateurs et ($\wedge$), ou ($\vee$) et non ($\neg$). Par exemple:

$$q = (t_1 \wedge t_2) \vee (t_3 \wedge \neg t_4)$$

Pour qu'un document corresponde à une requête, il faut que l'implication suivante soit valide: $d \Rightarrow q$

Cette évaluation peut être aussi définie de la façon suivante:

- Un document est représenté comme un ensemble de termes, et une requête comme une expression logique de termes. La correspondance R (d, q) entre une requête et un document est déterminée de la façon suivante:
  - R (d, t_i) = 1 si $t_i \in$ d; 0 sinon.
  - R (d, $q_1 \wedge q_2$) = 1 si R (d, q_1) = 1 et R (d, q_2) = 1; 0 sinon.
  - R (d, $q_1 \vee q_2$) = 1 si R (d, q_1) = 1 ou R (d, q_2) = 1; 0 sinon.
  - R (d, $\neg q_1$) = 1 si R (d, q_1) = 0; 0 sinon.

On peut remarquer deux problèmes dans ce modèle :

1. La correspondance entre un document et une requête est soit 1, soit 0. En conséquence, le système détermine un ensemble de documents non-ordonnés comme réponse à une requête. Il n'est pas possible de dire quel document est mieux qu'un autre. Cela crée beaucoup de problèmes aux usagers, car ils doivent encore fouiller dans cet ensemble de documents non-ordonnés pour trouver les documents qui les intéressent. C'est difficile dans le cas ou beaucoup de documents répondent aux critères de la requête.
2. Tous les termes dans un document ou dans une requête étant pondérés de la même façon simple (0 ou 1), il est difficile d'exprimer qu'un terme est plus important qu'un autre dans

leur représentation. Ainsi, un document qui décrit en détail "informatique", mais mentionne un peu "commerce" se trouve être représenté par {informatique, commerce} dans laquelle les deux termes deviennent aussi important l'un que l'autre. Cela ne correspond pas à ce qu'on souhaite avoir.

3. Le langage d'interrogation est une expression quelconque de la logique de propositions (un terme étant une proposition). Cela offre une très grande flexibilité aux usagers d'exprimer leurs besoins. Cependant, un problème en pratique est que les usagers manipulent très mal les opérateurs logiques, surtout dans beaucoup de cas, les mots "et" et "ou" ne correspondent pas tout à fait aux opérateurs logique ∧ et ∨. Par exemple, quelqu'un qui cherche des documents en "logique des propositions et logique de prédicats" peut en réalité vouloir chercher des documents sur "la logique de prépositions" ou sur "la logique de prédicat". Ici, le mot "et" doit plutôt être traduit en ∨. En partie à cause de cela, les expressions logiques données par un usager correspondent souvent mal à son besoin. La qualité de la recherche souffre donc en conséquence.

Il faut cependant remarquer que ce modèle booléen standard n'est utilisé que dans très peu de systèmes de nos jours. Si on utilise un modèle booléen, c'est plutôt comme une extension de ce modèle qu'on utilise. Les extensions proposées essaient justement de corriger ces lacunes.

6.3. Le modèle binaire **[BRA02] [MAN08]**

Le modèle binaire se base sur l'algèbre de Boole, ce qui implique que le poids associé à chaque item *i* de l'index associé à un document *j* peut uniquement avoir les valeurs égales à '0' ou à '1'. Dans ce modèle, la requête est spécifiée comme une expression booléenne. Ainsi une requête est composée de termes ou d'items de l'index reliés par les opérateurs booléens : *And, Or, Not*. Il en résulte que chaque requête peut être réécrite en une forme normale disjonctive stricte (une disjonction de conjonctions). Selon le modèle booléen un document est soit pertinent ou non par rapport à une requête donnée. Ce qui implique que la similarité qu'un document d_j pour une requête q est binaire et est définie par :

$$sim(d_j, q) = \begin{cases} 1 \ \textit{Si} \quad \exists\, \vec{q}_{cc} \mid (\vec{q}_{cc} \in \vec{q}_{dnf}) \wedge (\forall k_i,\ g_i(\vec{d_j}) = g_i(\vec{q}_{cc})) \\ 0 \ \textit{Sinon} \end{cases}$$

Avec :

- q_{dnf} : Correspond à la requête q réécrite sous forme normale conjonctive

- q_{cc} : Une sous expression de la requête q_{dnf} (sous forme de conjonction de termes).
- La fonction $g_i(.)$ qui permet d'évaluer la pertinence (la présence ou l'absence dans ce cas) d'un terme k_i respectivement dans l'expression q_{cc} et dans le document d_j

Le modèle binaire a l'avantage d'avoir un formalisme clair et simple. Malheureusement, il présente également quelques inconvénients :

- L'absence de la notion du gradient de pertinence de documents prévient une bonne performance de la recherche.
- Beaucoup d'utilisateurs ont du mal à exprimer leurs besoins dans une expression booléenne.

6.4. Le modèle à espace vectoriel **[BRA02]**

Le modèle à espace vectoriel, noté également le modèle vectoriel a été défini par Salton et al. en 1975. Ce modèle assigne des poids non binaires aux termes de l'index pour les documents et aussi pour les requêtes, dans le but d'obtenir une meilleure performance de recherche. Le poids associé, pour un terme k_i, à un document d_j est noté par $w_{ij,}$ tandis que le poids associé, pour un terme $k_{i,}$ à une requête q est noté par w_{iq}. Selon les travaux de Salton et Buckley en 1988, le poids du terme d'index apparaissant dans le document est calculé comme suit :

$$w_{ij} = f_{ij}\, idf_i$$

où: *fij* est la fréquence relative du terme k_i dans le document d_j

idf_i est l'inverse de la fréquence du document pour le terme k_i de l'index.

La fréquence relative et l'inverse de la fréquence du document pour le terme k_i sont respectivement calculés suivant ces formules :

$$f_{i,j} = \frac{freq_{i,j}}{max_l\, freq_{l,j}} \qquad\qquad idf_i = \log \frac{N}{n_i}$$

Où $freq_{i,j}$ est la fréquence d'apparition du terme k_i dans le document d_j, max_l est la fréquence maximale que peut obtenir ce terme pour tous les documents de la collection. N représente le nombre total des documents, et n_i le nombre de documents, où le terme considéré apparait.

Le modèle vectoriel a trois principaux avantages :

- ✓ En attribuant des pondérations aux termes, la recherche d'information est améliorée.
- ✓ Parce que les appariements approximatifs sont permis, même les requêtes approximatives peuvent être retrouvées
- ✓ En utilisant un degré de similarité, les documents peuvent être rangés en fonction de leur similarité aux requêtes.

Le principal inconvénient du modèle est que les termes de l'index sont supposés être mutuellement indépendants.

6.5.Modèle probabiliste **[BRA02]**

Le modèle probabiliste a été introduit par Robertson et Sparck Jones en 1976. L'essentiel de ce modèle est que le problème de recherche d'information est modélisé dans un Framework probabiliste. Selon ce modèle il existe un ensemble de réponses idéales noté R pour chaque requête, comportant tous les documents relevés qui respectent cette même requête. Un problème de recherche d'information peut, ainsi être vu comme un processus de détermination de la description parfaite d'une requête q pour l'ensemble de réponses idéales.

Au début du processus, les propriétés de l'ensemble R sont inconnues. Par conséquent, une supposition sur la description initiale p_1 de cet ensemble doit être faite. En tenant compte de la description initiale, un ensemble de documents R_1 est retrouvé, qui généralement ne correspond pas encore à l'ensemble idéal. Cependant, l'utilisateur détermine quels documents de R_1 sont actuellement pertinents, et avec cette information, la description initiale de l'ensemble R est adapté pour former P_2. P_2 est ensuite utilisé comme une nouvelle requête. Ce processus est répété jusqu'à ce que l'ensemble des documents trouvés (R_i) soit égal ou similaire à l'ensemble R.

Le modèle probabiliste est basé sur deux hypothèses :

- ✓ Le modèle suppose que la probabilité que l'utilisateur trouve le document pertinent pour une requête donnée, dépend uniquement de la représentation du document et de la requête.
- ✓ Suppose également, que pour chaque requête, il existe un ensemble de réponses idéales qui maximisent toutes les probabilités de pertinences de l'utilisateur.

Le modèle possède un avantage principal stipulant que les documents de la collection sont considérés selon leurs probabilités de pertinences. Malheureusement, ce même modèle présente au moins trois inconvénients :

- Il nécessite de mettre des hypothèses sur l'ensemble R
- Il considère des poids binaires
- Il adopte l'hypothèse d'indépendance entre les termes de l'index.

6.6. Le modèle d'ontologie **[VAL05]**

L'utilisation de l'ontologie pour couvrir les limites de la recherche basée mots-clés, a été prise comme une des motivations du web sémantique depuis son émergence vers la fin des années 90.

Le modèle ontologique consiste en une représentation ontologique de l'espace de recherche. Cette représentation n'est plus basée sur le document et les termes ou mots-clés qu'il comporte, mais plutôt par rapport à la valeur sémantique du document. Donc, dans ce modèle, on parle plus de recherche basée connaissance que de recherche par mots-clés.

Un processus d'extraction de connaissance devra d'abord être appliqué sur tous les documents de la collection pour associer chaque document aux connaissances qu'il englobe. Cette extraction peut être faite de manière automatique, semi-automatique, ou même manuelle au dépend des outils d'extraction disponible et de la taille de l'espace de recherche considéré.

Un système basé sur ce modèle, requiert la construction d'une base de connaissance sur la base des trois classes d'ontologie de base : *Concepts du domaine*, *la taxonomie*, et le *document* en lui-même. Ces trois classes sont complémentées par une *annotation* ontologique qui fournit une base pour l'indexation sémantique des documents qui ne comportent pas d'annotations incorporées. Les documents sont annotés par le concept d'instance de la base de connaissances. Une annotation a deux propriétés relationnelles :

- Instance et document : Ce par quoi un document et un concept sont reliés.
- Document et concept du domaine : qui peuvent avoir plusieurs propriétés d'annotation.

Une fois la base de connaissance construite, les annotations attribuées aux différents documents de la collection, tout comme pour le modèle vectoriel, des pondérations sont attribuées avec le même principe, aux annotations. Le processus de recherche se comporte comme n'importe quel processus basé mots-clés, sauf qu'ici les mots clés sont remplacés par les connaissances sémantiques.

Le principal avantage de ce modèle est le fait qu'il considère le document comme un ensemble de connaissances et la recherche également est faite en respectant les connaissances exprimées dans la requête, ainsi l'ensemble résultat est plus pertinent que pour les autres modèles. Néanmoins, cet avantage est en lui-même un inconvénient, puisque le processus d'extraction des connaissances est lui aussi très complexe et aussi couteux en temps.

6.7. Le modèle cognitif **[SEL07]**

Le modèle cognitif adopte une position holistique de la recherche d'information. Autre que les mécanismes de recherche utilisés pour apparier une requête avec un ensemble d'information stockée, il tient compte également des manières dont les besoins en informations de l'utilisateur peuvent être formulés en requêtes, des interactions hommes ordinateurs qui ont lieu pendant le processus de recherche, des environnements sociaux et cognitifs dans lesquels le processus a lieu et de la façon dont l'information sera employée par l'utilisateur pour avoir formulé cette requête ou ce besoin en information.

7. La recherche d'information dans l'architecture Peer To Peer **[ZEI03]**

Les avancées dans les réseaux publics et le déploiement de puissantes unités de calcul personnelles par des utilisateurs ont mis fin au modèle Client Serveur pour laisser la place au modèle *Peer To Peer* (P2P). HP définit le réseau P2P comme une manière, à la puissance très vaste, de fournir une large quantité de puissance de calcul, de stockage, et de connectivité à partir d'ordinateurs individuels, distribués partout dans le monde. Dans le modèle P2P chaque nœud a un rôle symétrique d'un client et d'un serveur en même temps. Un grand nombre de nœuds collaborent d'une façon dynamique et ad-hoc, et partagent des informations dans des environnements distribués à grande échelle sans aucune coordination centralisée.

La recherche d'information dans le P2P admet que chaque nœud dispose d'une base de données (ou une collection) de documents qu'il partage dans le réseau. Les documents peuvent être une collection de textes, audio, vidéo, ou autre format de documents semi-structurés. Un nœud qui recherche une information commence par envoyer une requête sous forme de message à tous ces égaux. Un système de recherche dans les infrastructures P2P admet que les requêtes soient formulées sous forme d'un ensemble de mots-clés. Un nœud recevant le message de requête, calcule la similarité de cette requête avec les informations contenues dans la collection de documents disponibles à son niveau. Typiquement, il recherche l'ensemble des documents qui comportent les mots-clés identifiés dans la requête (un processus de recherche d'information classique en se basant sur sa collection locale). Si

l'évaluation est un succès, ce nœud génère un message de réponse comportant les adresses d'accès aux documents associées à la requête (la réponse est généralement exprimée sous forme d'une liste d'URL).

Le problème de recherche d'information dans le P2P est plus complexe que le problème de recherche classique. Plusieurs algorithmes de recherche existent depuis plusieurs années, mais ne peuvent être directement appliqués aux réseaux P2P du fait de l'absence d'un répertoire ou d'une collection centralisée. Le principal défi de la recherche d'information dans le P2P est donc d'être capable de guider la requête vers la ou les sources qui contiennent l'information la plus pertinente de manière ***rapide et efficace***.

8. Conclusion

La recherche d'information est un processus qui consiste à trouver dans une large collection de documents, ceux qui sont pertinents pour le besoin en information exprimé par l'utilisateur à travers sa requête. Cette requête peut être formulée sous forme d'un ensemble de termes ou de mots-clés pour une recherche basée mots-clés, ou encore peut être exprimée sous forme d'un ensemble de connaissances pour faire l'objet d'une recherche sémantique.

Le domaine de recherche d'information fut créé à cause de *l'explosion de l'information* dans les années 1950. Ce terme ne décrit jamais aussi bien la réalité qu'aujourd'hui, dû à la popularisation de l'Internet et du Web. Cette nouvelle explosion a propulsé la RI au premier plan. Les techniques développées dans la RI sont de plus en plus demandées. Mais cette explosion apporte aussi de nouveaux problèmes auxquels la RI ne s'est jamais confrontées: collection gigantesque, dynamique et changeante, surabondance de l'information, données multimédia, données réparties, multilinguisme, et interaction entre l'utilisateur et le système.

Plusieurs travaux ont été entrepris dans ce domaine pour proposer des algorithmes qui soient efficaces et peu coûteux en temps de réponse.

L'infrastructure des réseaux Peer To Peer est venue pour rajouter encore plus de complexité au problème, puisque la recherche à présent ne se base pas uniquement sur une collection locale, mais elle considère, plutôt l'aspect distribué et non localisé des collections.

Les années 1980 ont été influencées par le développement de l'intelligence artificielle. Ainsi, on tentait d'intégrer des techniques de l'IA en RI, par exemple, système expert pour la RI, etc.

Chapitre II

Etat de l'art sur l'essaim d'abeilles

1. Introduction

Les problèmes complexes et particulièrement les problèmes d'optimisation combinatoires se présentent comme étant intraitables par l'algorithmique classique. La raison principale est que la topologie de tels problèmes est imprévisible. L'autre source de difficulté découle du volume immense d'informations qui composent le problème en termes de données et de contraintes. Les approches utilisées pour résoudre ce type de problème peuvent être classifiées en deux grandes classes : méthodes de recherche exactes et approchées. Une méthode exacte cherche à atteindre la solution coûte que coûte, ce qui induit des temps de recherches importants. L'autre classe comporte les méthodes heuristiques et métaheuristiques.

Au fil du temps les chercheurs ont proposé des méthodes de nature indépendante inspirée de phénomènes biologiques et bio-sociologiques. Ces méthodes ou techniques sont efficientes et flexibles, Elles peuvent être modifiées et/ou adaptées dans le but de convenir aux exigences d'un problème bien spécifique. L'intelligence en essaim est une branche des algorithmes bio-inspirés qui se focalise sur le comportement des insectes sociaux, qui malgré la simplicité de leur comportement individuel réalisent grâce à leur coopération, des tâches très complexes.

Cette manière de concevoir les solutions aux problèmes, a donné naissance à des métaheuristiques puissantes et très efficaces. Ce présent chapitre est dédié à l'étude des métaheuristiques, les fondements de base les régissant et leur applicabilité à différents domaines.

2. Heuristique et métaheuristique **[DRE04]**

Une heuristique est l'utilisation d'une information pour guider la recherche vers la solution. Une méthode heuristique est une méthode approchée se voulant simple, rapide, et adaptée à un problème donné. Sa capacité à résoudre un problème avec un minimum d'informations est contrebalancée par le fait qu'elle n'offre aucune garantie quant à l'optimalité de la meilleure solution trouvée. Les heuristiques trouvent cependant leur place dans les algorithmes qui nécessitent l'exploration d'un grand nombre de cas, car elles permettent de réduire leur complexité moyenne en examinant d'abord les cas qui ont le plus de chances de donner la réponse. Le choix d'une telle heuristique suppose de connaître déjà certaines propriétés statistiques sur l'ensemble d'instances du problème que l'on s'apprête à résoudre.

Si l'heuristique est bien choisie, la complexité moyenne de l'algorithme sur notre ensemble d'instances probabilisé peut même éventuellement être dans une classe inférieure (par exemple, polynômiale au lieu d'exponentielle) à celle de sa complexité au pire, ou à celle de la complexité moyenne du même algorithme où l'on explorerait les cas dans un ordre inapproprié.

Contrairement aux méthodes heuristiques, Les métaheuristiques ne sont pas conçues pour un problème donné. Elles sont plutôt générales, adaptables et applicables à plusieurs problèmes. Tout comme les méthodes heuristiques, elles permettent d'obtenir une solution réalisable dont le coût est proche de la solution en un temps raisonnable.

Les **métaheuristiques** forment une famille d'algorithmes visant à résoudre des problèmes difficile pour lesquels on ne connaît pas de méthode de l'algorithmique classique plus efficace.

Les métaheuristiques sont généralement des algorithmes stochastiques itératifs, qui progressent vers un optimum global, c'est à dire l'extremum global d'une fonction, par échantillonnage d'une fonction objective. Elles se comportent comme des algorithmes de recherche, tentant d'apprendre les caractéristiques d'un problème afin d'en trouver une approximation de la meilleure solution (d'une manière proche des algorithmes d'approximations).

Elles sont apparues dans les années 1980 pour résoudre des problèmes d'optimisation difficiles **[GLO86]**. Ces méthodes ont en commun certaines caractéristiques :

- Leur nature leur permet d'explorer plus facilement un espace des solutions de très grande taille ;
- Le calcul du gradient de la fonction objective n'est pas nécessaire à ces méthodes ;
- Elles sont inspirées par des analogies avec la nature ;
- Un réglage long de leurs paramètres est souvent nécessaire pour obtenir des résultats performants;
- Elles sont coûteuses en temps de calcul.

Le principe de base d'une métaheuristique est de parcourir l'espace des solutions à la recherche de son minimum global en utilisant des mécanismes lui permettant de s'extraire des minima locaux du paysage de recherche. L'un des atouts majeurs des métaheuristiques est leur facilité d'adaptation à de nouvelles fonctions objectives.

3. Principes communs des métaheuristiques

De façon générale les métaheuristiques consistent en une boucle itérative caractérisée par les quatre points suivants :

- Un état courant qui peut être soit une solution unique ou une collection de solutions appartenant à l'espace de recherche.
- Un ensemble d'opérateurs qui définissent la manière de construire le successeur de l'état courant. Ces opérateurs constituent les techniques de transition d'un état à un autre.
- Une règle de transition qui détermine le prochain état à choisir parmi l'ensemble des successeurs.
- Un critère d'arrêt représente généralement la rencontre de la solution ou tout simplement le nombre d'itérations au bout desquelles aucune amélioration sur la solution n'est observée, ou encore, le nombre d'itération limité par des contraintes physiques de la machine.

L'efficacité d'une métaheuristique est jugée sur la base de critères tels que la gestion du compromis grâce au maintien de la balance d'intensification et diversification, l'utilisation de mémoire et l'habilité à échapper aux optimums locaux. C'est au mérite de ces trois critères que se définira la qualité de la solution engendrée et le taux de succès de la méthode.

3.1. Maintien de la balance d'intensification/diversification

L'intensification décrit le processus par lequel l'approche vise la recombinaison des bonnes propriétés contenues dans des solutions trouvées antérieurement pour définir et parcourir des zones intéressantes. La diversification quant à elle vise à introduire de nouvelles propriétés inexistantes au niveau des solutions déjà visitées, ce qui peut mener la recherche vers des zones peu ou pas encore explorées.

Les notions d'intensification et de diversifications sont prépondérantes dans la conception des métaheuristiques, qui doivent atteindre un équilibre délicat entre ces deux dynamiques de recherche. Les deux notions ne sont donc pas contradictoires, mais complémentaires, et il existe de nombreuses stratégies mêlant à la fois l'un et l'autre des aspects.

3.2. Utilisation de la mémoire

Les métaheuristiques se distinguent aussi par l'utilisation ou non de la mémoire de stockage. Cette mémoire de stockage est souvent utilisée pour historiser les meilleures solutions déjà rencontrées ou encore tout le voisinage qui vient d'être exploité. Cet historique va servir à guider l'optimisation aux itérations suivantes. Beaucoup de métaheuristiques utilisent une mémoire plus évoluée, que ce soit sur le court terme (solutions visitées récemment, par exemple) ou sur le long terme (mémorisation d'un ensemble de paramètres synthétiques décrivant la recherche).

3.3. Risque de tomber sur un optimum local

Un optimum local est une solution qui correspond à la meilleure solution que la méthode a pu obtenir jusqu'à présent. Le principal inconvénient de tomber sur cette solution est qu'il n'est plus possible d'explorer d'autres zones de recherche qui pourraient être prometteuses en se basant sur des caractéristiques semblables à celle-ci. Et donc la recherche peut s'arrêter à ce niveau là et ne permet pas de garantir que c'est vraiment la solution optimale du point de vue de tout l'espace de recherche. La diversification est l'une des techniques utilisées pour pouvoir s'échapper de ces optimums locaux.

4. Classification des métaheuristiques

Les métaheuristiques peuvent être classées selon de nombreuses façons. Ce diagramme tente de présenter où se placent quelques unes des méthodes les plus connues. Un élément présenté à cheval sur différentes catégories indique que l'algorithme peut être placé dans l'une ou l'autre classe, selon le point de vu adopté.

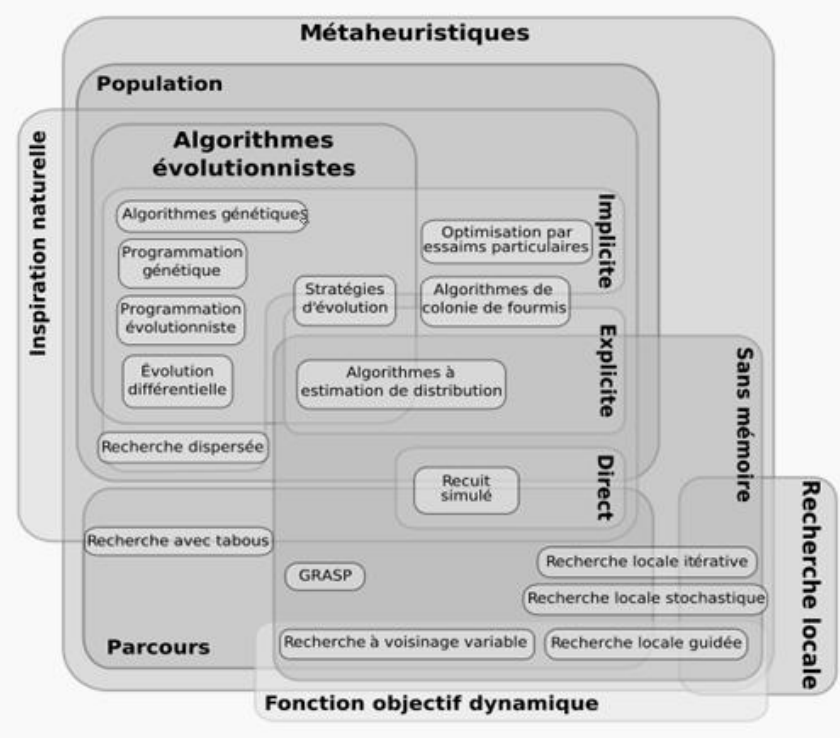


Figure 7 : Classification des métaheuristiques

La classification des métaheuristiques dépend du point de vu considéré lors de cette classification. La plus utilisée est celle qui les distingue par rapport aux solutions obtenues :

- Métaheuristiques de voisinage qui fait progresser une solution à la fois
- Métaheuristiques de population qui fait progresser toute une population de solutions à la fois.

4.1. Métaheuristiques basées sur une solution unique

C'est la classe des métaheuristiques les plus classiques, elle est fondée sur la notion de parcours. Dans cette optique, l'algorithme fait évoluer une seule solution sur l'espace de recherche à chaque itération. La notion de voisinage est alors primordiale.

Le voisinage d'une solution est constitué de toutes les solutions qui ressemblent en termes de propriétés et de caractéristiques à la solution actuelle. Parmi les métaheuristiques de solution nous pouvons citer :

La recherche locale

Appelée aussi la descente ou l'amélioration itérative, elle représente une classe de méthodes heuristiques très anciennes. Elle rentre dans la composition de la majorité des métaheuristiques, et constitue une arme très efficace pour solutionner les problèmes réputés difficiles tels que le voyageur de commerce **[ZHA05]**, le placement des VLSI Design

[FAR01], mais aussi pour la résolution de problèmes liés aux mathématiques discrètes tels que SAT et Max-W-SAT **[DRI00].** Elle ne constitue pas une métaheuristique en elle-même, ce n'est qu'une simple méthode itérative permettant d'obtenir de bonnes solutions approximatives **[VOU95].** En général, une procédure de recherche locale fonctionne comme suit:

- Elle génère un point initial (une solution initiale souvent aléatoire) de l'ensemble des solutions ;
- Ensuite, elle essaye d'améliorer la fonction objective f en faisant des transitions dans le voisinage de la solution courante.

La recherche taboue **[GEN02]**

La recherche taboue a été proposée en 1986 par Fred Glover, et indépendamment par Hansen, pour permettre à la méthode de recherche locale de surmonter les optima locaux. Cette méthode permet de poursuivre la recherche de solutions, y compris lorsqu'un optimum local est rencontré et ce en autorisant les déplacements non améliorants et en utilisant une liste taboue. La liste taboue enregistre l'historique de la recherche, et interdit tout déplacement qui mène à une position existante dans la liste. La liste taboue permet également d'éviter tous les cycles de longueur inférieure ou égale à K (K étant la taille de la liste taboue). La valeur de K dépend du problème à résoudre, et peut éventuellement évoluer au cours de la recherche.

Le recuit simulé (***S****imulated* ***A****nnealing*) **[ULU99]**

Cette méthode a été proposée par S. Kirkpatrick, C. D. Gelatt et M.P.Vecchi en 1983, et indépendamment par V.Cerny en 1985. Le recuit simulé est un algorithme local général de recherche, cherchant à réduire au minimum un objectif unique noté z(x). Il s'inspire du processus de recuit utilisé en métallurgie pour améliorer la qualité d'un solide, en cherchant un état d'énergie minimale.

En partant d'une haute température où le solide se trouve dans un état liquide, la phase de refroidissement conduit la matière liquide à retrouver sa forme solide par une diminution progressive de la température. Chaque température est maintenue jusqu'à ce que la matière trouve un équilibre thermodynamique. L'idée fondamentale de la méthode est d'accepter les mouvements non améliorants parce qu'ils peuvent aider à s'échapper d'un minimum local. L'acceptation d'un mouvement non améliorant se fait selon une règle, ou plus précisément, après avoir évaluer une certaine probabilité *p* qui est calculée en respectant la distribution de Boltzmann.

$$p = e^{(\Delta z/Tn)} ;$$

Où T_n représente la température à l'itération n. Et $\Delta z = z(y) - z(x)$

4.2. Métaheuristiques basées populations

Les méthodes basées population notées aussi méthodes distribuées, sont des méthodes qui permettent d'atteindre l'optimum par une population de solutions. Parmi ces méthodes et à titre d'exemple nous pouvons citer :

- ***Les algorithmes évolutifs*** **[DRI04]**

Ils sont basés sur le principe du processus d'évolution naturelle. Ils doivent leur nom à l'analogie avec les mécanismes d'évolution des espèces vivantes. Un algorithme typique est composé de trois éléments essentiels :

1. Une population constituée de plusieurs individus (solutions potentielles) d'un problème donné;
2. Un mécanisme d'évaluation de l'adaptation de chaque individu de la population à l'égard de son environnement extérieur ;
3. Un mécanisme d'évaluation composé d'opérateurs permettant d'éliminer des individus et de produire de nouveaux à partir des individus sélectionnés.

- ***Les algorithmes génétiques***

Les algorithmes génétiques sont inspirés de la génétique et de la théorie de la sélection naturelle citée par *Charles Darwin* au 19ème siècle et développée par John Holland de l'université de Michigan. Ils font parties des algorithmes évolutionnaires. Les GA introduits par Holland s'appuient souvent sur le codage universel sous forme de chaîne 0/1 de longueur fixe. Un individu dans ce codage est appelé chromosome. L'idée de base des algorithmes génétiques est de simuler un processus évolutionnaire afin d'obtenir de meilleurs individus (solutions).

Sur l'ensemble des individus de la population actuelle, une phase de sélection est appliquée dans le but de choisir un sous ensemble de descendants pour la phase de reproduction.

A chaque itération, une nouvelle population est générée à partir d'un sous-ensemble issu de la population précédente après sélection, et aussi de l'ensemble des nouveaux individus C. L'ensemble C est obtenu par une phase de reproduction en utilisant des

opérateurs génétiques appliqués sur les individus sélectionnés de la population courante. Ces opérateurs génétiques sont **[MED03]** :

- *Le croisement :* la principale caractéristique du croisement est de préserver les meilleurs gènes des parents (les parents sont les individus sélectionnés), ainsi l'individu résultant du croissement a de grandes chances de ressembler aux parents ;
- *La mutation :* L'opération de mutation est un processus à deux phases. La première phase consiste à supprimer un certain nombre de gènes des parents de façon aléatoire. La seconde phase correspond à une phase de reconstruction pour le recouvrement des caractéristiques perdues.

La recherche dispersée (Scatter Search) **[DRK01], [DRI04]**

La recherche dispersée consiste, à chaque itération, à générer une population diversifiée à partir d'une solution initiale appelée *semence*. De cette population un ensemble *RefSet* est choisi à partir des meilleures solutions en termes de qualité et de diversité. L'ensemble *RefSet* est ensuite partitionné en sous-ensembles, sur lesquels des combinaisons linéaires seront appliquées dans le but de créer de nouvelles solutions.

Une solution améliorée peut remplacer la mauvaise solution dans *RefSet* si elle est meilleure en qualité sinon en diversité. La génération de nouvelles solutions s'arrête lorsque l'ensemble *RefSet* ne change plus. Dans ce cas, une nouvelle itération est commencée en utilisant la meilleure solution de *RefSet* comme semence.

5. Les méthodes bio-inspirées **[PAV05] [BON01]**

Les technologies « bio-inspirées » proviennent de la biomécanique et de l'informatique. Contrairement à la biomécanique, elles ne se limitent pas à construire un dispositif technologique ressemblant à une unité vivante, mais également à des processus, biologiques, écologiques et évolutifs. Il en va ainsi, par exemple, des algorithmes génétiques, qui miment des processus de mutation-sélection pour résoudre des problèmes d'optimisation. Ces relations entre biologie et technologie ne sont pas nouvelles ; elles remontent au moins aux années 1940, avec l'invention de la cybernétique par Norbert Wiener. Cette dernière a été principalement développée pour mettre en correspondance les processus de régulation physiologiques et technologiques (la célèbre notion de « rétroaction » vient de la cybernétique). Plus tard, on a parlé de « bionique », mais ce mot est tombé en désuétude. Résultats de près de quatre milliards d'années d'évolution, les systèmes biologiques exhibent

des originalités et inspirent des solutions qui peuvent être utiles dans nombre de domaines de l'activité humaine.

Plusieurs méthodes existent qui sont inspirées de domaines biologiques à savoir **[VDH07] :**

- Les algorithmes génétiques
- Les réseaux de neurones
- L'intelligence en essaim
- Le calcul à l'ADN
- Le système immunitaire artificiel, … etc.

6. L'intelligence en essaim

La métaphore des comportements collectifs des insectes pour la résolution des problèmes est devenue un domaine très actif ces dernières années. De plus en plus, des chercheurs s'intéressent à cette nouvelle forme d'intelligence, l'intelligence en essaim qui signifie l'intelligence collective et émergente de groupes d'agents relativement simples.

Cette approche met l'accent sur la distribution, la communication directe et indirecte entre des agents relativement simples et robustes à la fois pour la réalisation de tâches plutôt complexes. Ces tâches sont issues des problèmes quotidiens et de nature variée auxquels une même société d'insectes devra faire fasse, qu'ils s'agissent de la recherche de nourriture, construction du nid, distribution du travail et des tâches quotidiennes, ou encore de la rente de ces mêmes tâches entre les individus …etc.

Le nombre d'applications de l'intelligence en essaim est en pleine expansion, dans l'optimisation combinatoire, la robotique, l'analyse de données, la programmation des équipes d'une usine, les réseaux de télécommunication **[BDT99]**, et bien d'autres domaines.

Ainsi, L'intelligence en essaim offre une méthode alternative pour la conception des systèmes intelligents pour lesquels l'autonomie, l'émergence et le fonctionnement distribué remplacent contrôle, pré programmation et centralisation.

6.1. Caractéristiques de l'intelligence en essaim **[DRI06]**

6.1.1. L'Auto-organisation

Les théories de l'auto-organisation décrivent l'*émergence* des modèles macroscopiques à partir de processus et d'interactions définies au niveau microscopique. Les modèles basés sur l'auto-organisation présument qu'il est possible d'exprimer quelque chose

apparemment complexe en termes de processus simples qui interagissent entre eux. La question cruciale est donc de comprendre comment les composants d'un système interagissent entre eux pour produire un modèle complexe (plus complexe que les composants eux-mêmes).

6.1.2. La stigmergie

C'est un autre concept théorique important de l'intelligence en essaim, elle est à la base de la création des métaheuristiques des essaims. La stigmergie est définie comme une « forme de communication passant par le biais des modifications de l'environnement ». Sa grande force réside dans le fait que les individus échangent des informations via les travaux en cours et de l'état d'avancement de la tâche globale à accomplir.

La stigmergie fournit un mécanisme général qui relie les comportements individuels et le comportement collectif de la colonie : Le comportement individuel modifie l'environnement qui à son tour modifie le comportement des autres individus.

6.1.3. Contrôle décentralisé

Les systèmes auto-organisés permettent de s'affranchir d'un contrôle central et d'une programmation explicite de tous les cas potentiellement rencontrés. Dans ce genre de systèmes, il n'y a pas de prise de décision à un niveau donné. Chaque individu dispose d'une vision locale de son environnement, et ne connait donc pas le problème dans son ensemble.

6.1.4. Robustesse et flexibilité

Les sociétés d'insectes ont une capacité remarquable à résoudre des problèmes de manière *très flexible* puisque la colonie s'adapte aux brusques changements d'environnement ce qui leur donne une efficacité pour des problèmes dynamiques, et *robuste* puisque la colonie est fractionnée, ce qui fait que, même lorsque certains individus échouent à accomplir leurs tâches d'autres y parviendront sûrement.

6.2. Métaheuristiques issues de l'intelligence en essaim

6.2.1. Essaim de fourmis (ACO)

Les entomologistes ont analysé la collaboration qui s'établit entre les fourmis pour aller chercher la nourriture à l'extérieur de la fourmilière. Il est à remarquer que les fourmis suivent toujours le même chemin, et que ce chemin soit le plus court possible. Chaque fourmi dépose, le long de son chemin une trace chimique appelée *phéromone*. Tous les membres de

la colonie perçoivent cette trace et s'orientent préférentiellement vers les régions les plus odorantes. Il en résulte notamment la faculté de retrouver le plus court chemin.

L'optimisation par colonie de fourmi est une métaheuristique bio-inspirée **[SAM05]** du comportement des fourmis lors de la recherche de nourritures à l'extérieur de la fourmilière. Cette métaheuristique a été utilisée pour résoudre différents problèmes d'optimisation combinatoire réputés difficiles. Elle est due à Colorni, Dorigo et Maniezzo **[COL92]**. Elle s'efforce de simuler la capacité collective de résolution de certains problèmes, observée chez une colonie de fourmis, dont les membres sont pourtant individuellement dotés de facultés très limitées. La métaheuristique assimile la recherche de solutions comme la recherche de nourriture chez les fourmis, la quantité ou la qualité de la nourriture à la fonction objective du problème à optimiser et les traces de phéromone par une mémoire adaptative **[DOR99]**.

6.2.2. Essaim de particule (PSO)

L'optimisation par essaim de particules (PSO) est une métaheuristique d'optimisation développée, en 1995, par Dr. Russel Eberhart (ingénieur en électricité) et Dr. James Kennedy (socio psychologue). Celle-ci est inspirée par le comportement social des nuées d'oiseaux ou de bancs de poissons et s'appuie notamment, sur un modèle développé par le biologiste Craig Reynolds à la fin des années 1980, qui permet de simuler le déplacement d'un groupe d'oiseaux. **[BER01]**.

Au début des travaux J. Kennedy et R. Eberhart cherchaient à simuler la capacité des oiseaux à voler de façon synchrone et leur aptitude à changer brusquement de direction tout en restant en une formation optimale. Le modèle qu'ils ont proposé a ensuite été étendu en un algorithme simple et efficace d'optimisation **[TRA03]**. L'optimisation par essaim de particules se base sur la collaboration des individus au sein d'un groupe.

Observons un champ en train d'être labouré en automne **[DUT02].** Lorsque le soc de la charrue pénètre le sol pour la première fois, le champ est vide de tout goéland et quelques minutes après, une nuée accompagne le tracteur. Au début du labour, un oiseau découvre la source de nourriture et très rapidement un autre arrive et ainsi de suite. Que s'est-il passé ? L'information concernant un festin potentiel s'est largement diffusée au sein du groupe de goélands. Ces derniers volaient, à la recherche de nourriture de façon plus ou moins ordonnée. Leur rassemblement s'est effectué par un échange (volontaire ou non) social d'informations entre individus de la même espèce. L'un d'entre eux a trouvé une solution et les autres se sont

adaptés en copiant sa solution. Si nous faisons un rapprochement entre l'exemple précédent et l'approche PSO, les goélands représentent une population de solutions potentielles, interprétées comme des particules se déplaçant dans l'espace de recherche. Les particules volent à travers l'espace de recherche en suivant l'optimum courant.

7. L'optimisation par essaim d'abeilles (BSO)

La métaheuristique d'optimisation par essaim d'abeilles est une nouvelle métaheuristique qui a été élaborée par **[DRI04]** et qui s'inspire du comportement réel des abeilles durant leur recherche de nourriture. Les butineuses d'une même colonie visitent plusieurs zones potentielles d'exploitation, mais elles concentrent leurs efforts de récolte sur un petit nombre d'entre elles ; les plus riches et les plus faciles d'accès. Une colonie peut même déplacer rapidement son exploitation d'une source de nourriture à une autre.

7.1. L'abeille réelle **[BOT94], [BAY07]**

En 1946, l'éthologue autrichien *Karl Von Fris* avait présenté à Zurich, devant la société suisse des sciences naturelles, l'essentiel de ses conclusions : par l'orientation et la vitesse des mouvements qu'elle effectue sur les rayons de la ruche, la butineuse (L'ouvrière chargée de la récolte du nectar et du pollen) indique à ses congénères la direction, la distance et la richesse de la source de nourriture qu'elle a découverte.

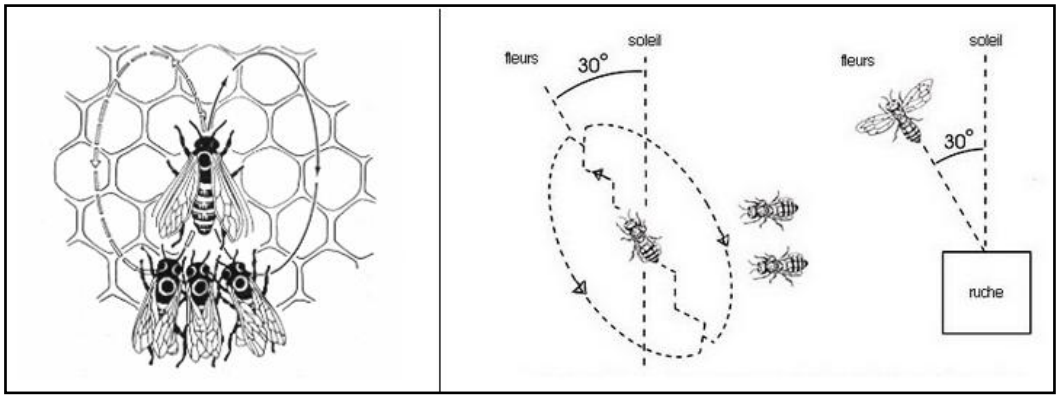


Figure 8: La danse des abeilles.

7.1.1. L'exploitation sélective de la source de nourriture

Il n'est pas rare qu'au cours d'une même journée, les butineuses d'une même colonie visitent plus d'une douzaine de zones d'exploitation potentielles, mais on observe que la colonie concentre ses efforts de récolte sur un petit nombre d'entre elles. Les plus riches et les plus faciles d'accès. En outre, de nombreuses observations font apparaître qu'une colonie peut rapidement déplacer son exploitation d'une source de nourriture à une autre.

Les abeilles sont capables d'ajuster très finement plusieurs composantes de leur comportement de récolte. Ainsi, en changeant périodiquement la richesse relative de deux sources de nourriture, vers lesquelles on a préalablement entraîné les récolteuses d'une ruche à venir s'approvisionner, ils ont pu mettre en découverte, que lors de leur retour au nid, le comportement de recrutement des abeilles était profondément affecté.

L'expérience de Seely, Cazmazine et Sneyd en 1991 à montrer que lorsqu'on donne le choix à un essaim d'abeilles, entre l'exploitation de deux sources de nourriture dont la concentration en sucre est très inégale (1M et 2.5M respectivement), et situées de manière diamétralement opposées par rapport à la ruche, l'une au Nord, et l'autre au Sud, la colonie va concentrer son effort de récolte sur la plus riche d'entre elles. Le nombre total de récolteuses présentes au niveau de chacune des deux sources, est directement proportionnel aux changements ayant affecté les taux de recrutement et d'abandon vers chacune des deux sources. C'est grâce à une modulation très précise de ces deux taux, que peut se produire, au niveau de la colonie, une exploitation sélective de la source la plus riche.

7.1.2. Mécanisme de communication

Chaque récolteuse, après s'être déchargée de sa récolte de nectar, entame une danse qui indique à ses congénères qui la suivent : La direction, la distance et la richesse de la source de nourriture qu'elle vient de visiter.

Elle va de cette manière inciter d'autres individus à se rendre dans cette zone pour y récolter à leur tour. Plus la source est riche, plus la vigueur avec laquelle l'abeille effectue cette danse augmente, de même que la cadence de ses visites. Parallèlement, l'intervalle de temps qui sépare son arrivée au nid du déchargement de sa récolte, diminue. Par la suite, lorsqu'on change expérimentalement la qualité de la source de nourriture, en offrant une source beaucoup moins riche, les récolteuses de retour au nid, n'effectuent presque plus de danse, et lorsque celles-ci se produisent, leur rythme s'affaiblit notablement.

Mais une question intéressante qui pourrait se poser est la suivante: Comment une abeille peut-elle estimer la richesse relative d'une source de nourriture qu'elle vient de visiter, par rapport à d'autres sources simultanément présents ?

Les recherches ont montré que cette estimation s'effectuait sans que l'abeille n'ait besoin de visiter plusieurs sources pour les comparer, ni qu'elles reçoivent cette information des abeilles chargées du stockage de la nourriture rapportée au nid. Expérimentalement, il est vérifié que les récolteuses ajustent l'intensité de leur danse, en intégrant de multiples facteurs,

comme l'abondance de la source, la distance de la source au nid, et la difficulté à puiser le nectar.

Néanmoins, le mystère et le secret de cette estimation et du traitement de l'information par le système nerveux centrale de l'abeille reste toujours posé !

7.2. La métaheuristique d'optimisation par essaim d'abeille (BSO) **[DRI06]**

La métaheuristique d'optimisation par essaim d'abeilles (notée BSO pour *Bees Swarm Optimization*), peut être appliquée aux problèmes d'optimisation combinatoire à espace d'état discret présentés comme suit :

- Un problème d'optimisation combinatoire est défini par un ensemble d'instances.
- A chaque instance du problème, sont associés un ensemble discret de solutions *S* et un sous-ensemble *X* de *S* représentant les solutions admissibles (réalisables), et une fonction objective *f* (ou fonction de coût) qui assigne à chaque solution s dans *X* une évaluation réelle (entière) *f(s).*

Résoudre un tel problème, consiste à trouver une solution s* dans l'ensemble *X*, optimisant la valeur de la fonction objective *f*. La solution s* s'appelle solution optimale ou optimum global. Ainsi, et selon la fonction objective *f*, on peut distinguer deux cas de figures :

- *f* est une fonction de maximisation, s* doit vérifier $f(s^*) \geq f(s)$ pour toute solution s dans *X*.
- *f* est une fonction de minimisation, s* doit vérifier $f(s^*) \leq f(s)$ pour toute solution s dans *X*.

Etant donné un problème d'optimisation combinatoire discret, la métaheuristique BSO est basée sur un essaim d'abeilles artificielles qui coopèrent pour le résoudre :

- D'abord, une des abeilles appelée *BeeInit* se charge de trouver un sommet présentant une bonne caractéristique appelée *Sref*. Ce sommet désigne la région d'exploitation qui s'appelle *SearchArea* dans laquelle les abeilles intensifieront leurs recherches ;
- La région d'exploration est constituée d'un ensemble de sommets. Chacun d'eux est obtenu à partir du sommet *Sref* et ce, en utilisant une certaine stratégie ;
- Après la définition de cette région, chaque abeille artificielle qu'on appelle *Bee* se voit assignée un sommet depuis lequel elle démarre sa recherche en transitant d'un sommet vers un autre sommet voisin jugé meilleur ;

- A l'issue de sa recherche, chaque abeille k mémorise le meilleur sommet qu'elle a visité à la $k^{ème}$ entrée qui lui est réservée dans une table *Dance*. Cette table permet aux abeilles virtuelles de communiquer leurs résultats à l'essaim, pour qu'à la fin de l'exploitation chaque abeille puisse prendre connaissance des résultats de ses congénères. Mais si au bout d'un certain nombre de fois l'essaim constate que la qualité ne s'améliore plus (optimum local) le critère de diversité sera appliqué pour s'éloigner des régions déjà exploitées ;
- Un des sommets de Dance deviendra le nouveau *Sref*, ainsi l'effort d'exploitation sera dirigé vers cette nouvelle zone prometteuse ;
- Pour éviter qu'un même sommet puisse être considéré deux fois comme référence, une liste taboue peut être utilisée pour éviter l'apparition des cycles.

L'algorithme général de la métaheuristique d'optimisation par essaim d'abeilles se présente comme suit :

Bees Swarm Optimisation

```
Début
    Sref ← Sommet retourné par BeeInit ;
    Tant que non critère d'arrêt faire
        Insérer Sref dans Tabou ;
        Déterminer la region searchArea référencée par Sref ;
        Affecter chaque abeille à un sommet de SearchArea ;
        Pour chaque abeille k faire
            Rechercher depuis le sommet Sk qui lui est associé ;
            Stocker le résultat dans la table danse ;
        Fait
        Choisir le nouveau sommet de référence Sref ;
    Fait
Fin.
```

8. Domaines d'application des essaims d'abeilles **[BAY07]**

La recherche de nourriture, l'apprentissage, la mémorisation et le partage d'information sont toutes des caractéristiques qui ont fait de la vie des abeilles un champ intéressant d'investigation et de recherche en intelligence en essaim. Les études sur les

abeilles butineuses ont tendance à s'accroitre en littérature ces quelques dernières années. Après un léger survol de la littérature, les travaux antérieurs sont catégorisés par la caractéristique comportementale des abeilles butineuses considérée, à savoir :

- La recherche de nourriture ;
- Le couplage
- La conception du royaume.

Le tableau ci-après résume l'ensemble des travaux antérieurs par catégories **[BAY07]**:

Type	***Publication***	***Algorithme***	***Application***
Recherche de nourriture	Yonezawa, Kikuchi (1996) Seeley, Buhrman (1999) Schmickl et al. (2005) Lemmens (2006)		*Simulation biologique.*
	Sato et Hagiwara (1997)	Bee System	Amélioration de l'algorithme génétique
	Karaboga (2005)	Artificial Bee Colony Algorithm (ABC)	Optimisation continue
	Yang (2005)	Virtual Bee Algorithm (VBA)	Optimisation continue
	Basturk, Karaboga (2006)	ABC	Optimisation continue
	Pham et al. (2006a)	Bees Algorithm (BA)	Optimisation continue
	Lucic , Teodorovic (2001)	Bee System (BS)	Problème du voyageur de commerce (TSP)
	Lucic (2002 Lucic , Teodorovic (2002)	BS BS	TSP et routage stochastique des véhicules TSP.
	Lucic, Teodorovic (2003a) Lucic , Teodorovic (2003b)	BS BS + fuzzy logic	TSP Routage stochastique de véhicules
	Teodorovic, Dell'Orco(2005)	Bee Colony Optimization (BCO) + Fuzzy Bee System (FBS)	*Ride-Matching Problem*
	Nakrani, Tovey (2003)	A Honey Bee Algorithm	Allocation dynamique des services internet
	Wedde et al. (2004)	BeeHive	Routage dans les réseaux de télécommunication.
	Bianco (2004)		Précision de la navigation à grande échelle.
	Chong et al. (2006)		Job Shop Scheduling
	Drias et al. (2005)	BSO	*Max-W-SAT*
	Pham et al. (2006b)	BA	*LVQ-Neural Network*
	Pham et al. (2006c)	BA	*MLP- Neural Network*
	Pham et al. (2006d)	BA	Neural Network
	Quijano and Passino (2007)		Dynamic Resource

	Markovic et al. (2007)	BCO Based	Max-Routing and Wavelength Assignment
Couplage	Abbass (2001a,b,c) Teo, Abbass (2001, 2003)	Marriage in Honey-Bees Optimization (MBO) Modified MBO	*3-SAT Problem* *3-SAT Problem*
	Bozorg Haddad Afshar(2004) Haddad et al. (2006) Chang (2006)	MBO Honey-Bees Mating Optimization –HBMO MBO Based	Water Resources Management Problems Nonlinear constrained and unconstrained optimization Stochastic Dynamic Programming
	Afshar et al. (2007)	Improved HBMO	Continuous Optimization
	Fathian et al. (2007)	HBMO Based	Data Mining -Clustering
	Koudil et al. (2007)	MBO Based	Integrated Partitioning/Scheduling
	Benatchba et al. (2005)	MBO Based	Data Mining
Conception du royaume	Sung (2003)	Queen-Bee Evolution Algorithm(QBE)	Amélioration de l'algorithme génétique
	Qin et al. (2004)	QBE Based	Economic Power Dispatch
	Kara (2004)	Bee Crossover	Amélioration de l'algorithme génétique
	Azeem, Saad (2004)	Modified QBE	Amélioration de l'algorithme génétique

Tableau 2 : Travaux antérieurs sur les essaims d'abeilles.

9. Conclusion

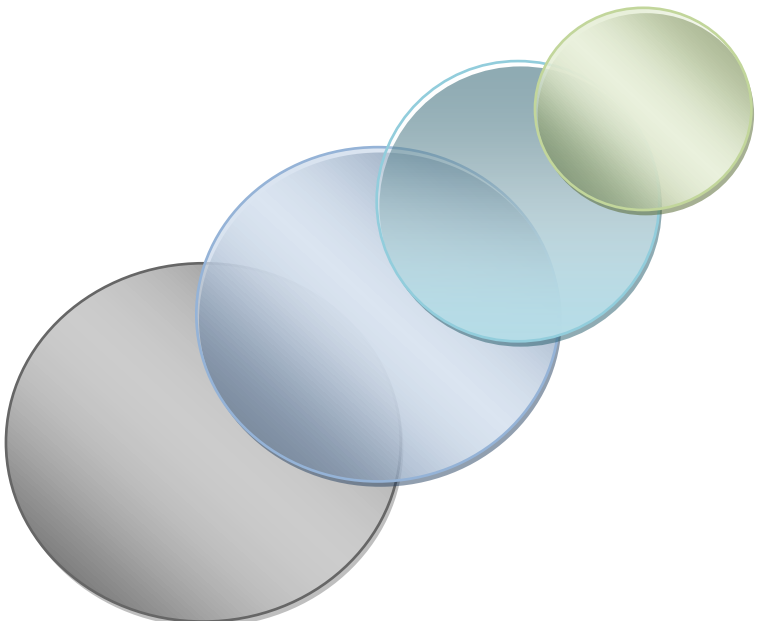
Face à l'impuissance des méthodes de résolution classiques des problèmes d'optimisation combinatoires, les métaheuristiques se sont illustrées comme une alternative intéressante. Les métaheuristiques représentent des techniques de recherche globale capables de résoudre une variété de problèmes complexes. Depuis quelques années, et en vue de définir de nouvelles métaheuristiques encore plus puissantes, la recherche dans le domaine s'est considérablement penchée sur la création de nouvelles méthodes en se basant sur des concepts et principes issus du domaine biologique tels que les algorithmes génétiques ou encore l'intelligence en essaims.

A travers leurs différents domaines d'applications et d'applicabilités, les métaheuristiques issues de l'intelligence en essaim ont montré leur efficacité ainsi que leur souplesse à s'adapter à plusieurs types de problèmes à la fois. Ces derniers nous enseignent également que des éléments simples et réactifs, correctement connectés dans un groupe sont capables de produire des résultats intelligents.

L'intelligence en essaim reste encore un domaine non entièrement exploité, ceci est principalement dû à la diversité des comportements d'un type d'insectes à un autre. Et donc de manière générale, il est possible de définir encore plus de métaheuristiques par l'étude et l'observation des phénomènes sociobiologiques.

Partie II :

Les essaims d'abeilles pour la recherche d'information à grande échelle



Chapitre III

Une approche basée sur les essaims d'abeilles pour la recherche d'information dans la collection de documents

1. Introduction

A travers le premier chapitre nous avons montré l'intérêt que présente la recherche d'information dans le domaine informatique ainsi que ces multiples principes, modèles et méthodologies de recherche. Le second chapitre quant à lui, englobe une brève synthèse des méthodes de résolution qui ont le plus servis au cours de cette dernière décennie et les méthodes qui ont récemment émergé grâce à l'inspiration des chercheurs des phénomènes et comportements sociobiologiques.

Le présent chapitre développe une première approche pour aborder la RI à grande échelle avec les essaims d'abeilles. L'espace des solutions possibles est constitué de la totalité des documents de la collection.

2. Problématique

La problématique consiste en la résolution du ***problème de recherche d'information à grande échelle***. L'approche proposée pour solutionner ce problème est issue de ***l'intelligence en essaim.*** Il s'agit de la métaheuristique 'BSO' (*Bees Swarm Optimization*).

2.1. Recherche d'information à grande échelle

Comme vu dans la première partie, le problème de la recherche d'information consiste à chercher dans une très large collection d'information structurée ou pas, un ensemble d'informations les plus pertinentes et qui correspondent à une requête exprimée sous forme d'un enchainement de mots-clés ou de connaissances bien spécifiques. La notion de « grande échelle » apparait de plus en plus cette dernière décennie, ceci est dû à la croissance considérable de l'information et de sa disponibilité. Pour avoir un léger aperçu de l'ampleur de cette croissance il suffit de remarquer la quantité de résultats fournis par n'importe quel moteur de recherche pour une requête, qui peut atteindre plusieurs milliers de documents. Ce passage à l'échelle dans la quantité d'information représentée dans les collections a rajouté encore plus de complexité au problème puisque maintenant hormis la qualité des solutions fournit aux utilisateurs nous devons également tenir compte du temps de recherche de l'information.

De manière générale, la recherche d'information étant un problème très vaste, sa résolution consiste dans la majeure partie des cas, à choisir un modèle de recherche bien spécifique et de proposer un mécanisme performant de recherche de solution. La plupart des méthodes de recherche qui existent actuellement se basent sur deux phases indispensables :

- L'indexation de la collection d'informations
- Le parcours de la collection grâce à cet index pour localiser les documents qui sont associés aux mots clés identifiées dans la requête utilisateur.

2.2. L'essaim d'abeille appliquée à la collection des documents

L'intelligence en essaim constitue une classe très puissante de métaheuristiques. Ces dernières sont inspirées de phénomènes sociobiologiques observés sur les insectes et leurs comportements face à leurs tâches quotidiennes. Elles sont à la base des méthodes de recherche qui essayent de construire un chemin minimal vers la solution optimale.

Sur les travaux de recherche publiés récemment sur les approches proposées pour la recherche d'informations, l'intelligence en essaim n'apparait nulle part. C'est ce qui a motivé notre choix pour cette approche. L'exploration de cette nouvelle direction nous semble prometteuse.

Pour notre part, le choix s'est porté sur la métaheuristique d'optimisation par essaim d'abeille plus connue sous l'appellation ***essaim d'abeilles***, grâce au fondement même de cette approche qui modélise n'importe quel problème en un processus de recherche de la nourriture principalement guidé par la concentration en nourriture des régions à exploiter. Par analogie au problème de recherche d'information l'essaim devra œuvrer sur un champ de document afin d'en extraire les plus répondants à la requête guidée dans ce vaste champ par la qualité en information des documents par rapport au besoin recherché.

En résumé, notre proposition consiste à développer un algorithme ou un mécanisme de recherche d'information à grande échelle basée sur l'intelligence en essaim.

Le modèle de RI adopté est le modèle vectoriel. Une solution au problème est donc un document représenté sous forme de vecteur de termes tf*idf.

Le modèle vectoriel présente deux variantes à savoir les vecteurs de termes binaires et les vecteurs de termes pondérés. Dans la première variante, le modèle traite de l'absence ou de la présence des termes dans un document sans considérer l'importance d'un terme par rapport à un autre contrairement à la seconde variante qui utilise une pondération associée aux termes pour faire ressortir d'importance d'un terme quelconque par rapport à un autre dans un même document ou dans une requête.

3. *Les essaims abeilles pour une recherche basée document*

Dans le but d'aboutir à une adaptation de l'essaim d'abeille pour une recherche d'information nous avons besoin de définir les points suivants :

- ***Le modèle de recherche d'information à adopter*** :

Le modèle de recherche d'information retenu correspond au modèle vectoriel ou un document est représenté par un vecteur de termes ou de mots clés. Ce modèle est très adapté pour une recherche basée « mot-clé ». De même que les documents, les requêtes aussi seront représentées par un vecteur de termes. Le vecteur binaire sera considéré pour la création de solutions alors que le vecteur pondéré sera utilisé pour le calcul de la similarité entre une requête et un document. De ce fait, un document est représenté par deux vecteurs, le premier binaire noté Db pour la construction des solutions et le second Dw pondéré selon la fréquence tf*idf, qui servira pour le calcul de la similarité.

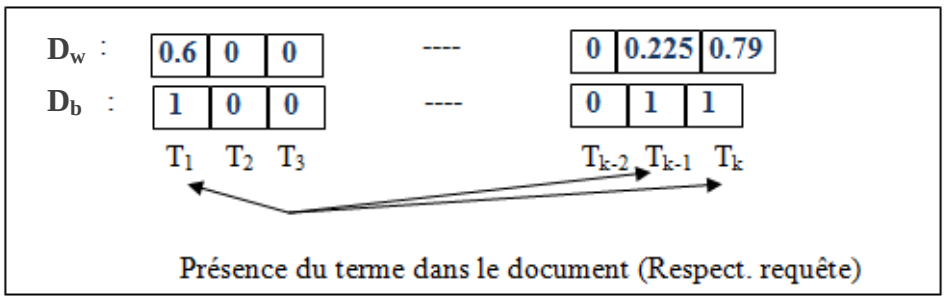


Figure 9 : Schéma de représentation des vecteurs documents et requêtes.

Sur la base de cette représentation nous pouvons considérer un document comme un vecteur (ou une suite) de K bits, chaque bit correspond au terme correspondant à sa position. Pour le calcul de la similarité, les vecteurs de termes pour les documents et les requêtes sous forme tf*idf seront considérés.

- ***Le corpus ou la collection de document :***

En tenant compte de la représentation adoptée pour les documents, le corpus ou la collection de documents, est un ensemble de vecteur ou une matrice (N*K) ; ou N correspond au nombre de document et K au nombre de termes considérés dans chaque document.

- ***Le mécanisme de recherche :***

Le mécanisme de recherche est la politique de parcours de la collection des documents dont le but d'en déterminer les plus pertinents par rapport à la requête. Dans notre cas, le mécanisme de recherche est une adaptation de l'algorithme d'essaim d'abeilles pour ce problème de recherche.

3.1. Génération de Benchmark (Collection)

Un Benchmark est un banc d'essai qui permet de tester les performances d'un système. C'est un fichier de données pour un problème bien spécifique. En plus de contenir les données, ce fichier comporte également les résultats obtenu pour ce problème par un moyen qui peut être automatique (un autre système ou algorithme), ou bien obtenu de façon manuelle. Ces résultats vont servir de référence pour l'évaluation d'un autre système pour le même problème.

Plusieurs éditeurs et chercheurs travaillent sur l'édition de Benchmark dans le but de faire avancer la recherche.

Dans le cas de la recherche d'information il existe plusieurs Benchmarks tels que : TREC, OHS, CACM, …etc. Une grande partie de ces Benchmarks reste la propriété exclusive de leurs éditeurs, d'autres sont payant telles que TREC et d'autres encore restent disponible gratuitement pour la communauté de chercheurs telles que les collections engendrées par le système SMART (CACM, ADI,…). La taille de ces collections qui est exprimé par le nombre de document qu'elles comportent reste plus ou moins réduit et ne correspond pas à des cas de recherche à grande échelle.

De ce fait, et pour les besoins de nos travaux nous avons eu recours à des collections générées de manière aléatoire. Le principe est le suivant : puisque pour le modèle retenu dans nos travaux les documents et les requêtes sont représentés sous forme de vecteurs de termes et que la collection est un ensemble de vecteurs, il est possible de générer ces vecteurs de manière aléatoire dans le but de construire une collection. De même que pour la construction des documents un certain nombre de requêtes sera également généré pour assurer la recherche. Une méthode de recherche exacte sera également appliquée à l'ensemble de ces requêtes dans le but de déterminer les résultats qui figureront dans le fichier Benchmark.

Il existe plusieurs moyens possibles pour générer des vecteurs de manière aléatoire tout en considérant qu'il ne peut pas exister deux documents ou deux requêtes qui possèdent la même représentation donc exclure les duplications et également construire des collections qui soient les plus diversifiés possible pour avoir une meilleure interprétation des résultats. Aussi bien les documents que les requêtes seront représentés sous forme de vecteurs pondérées en tf*idf, de ce fait la génération de la collection va se faire en deux phases :

- La première qui consiste à construire la représentation binaire associée à chaque document de la collection et d'attribuer de manière aléatoire les fréquences

d'apparition de chaque terme à '1' du vecteur binaire pour la construction du vecteur pondéré.

- La deuxième phase consiste à normaliser en tf*idf (selon la formule donnée en chapitre 1) les fréquences obtenues dans la phase précédente en vue d'obtenir la représentation pondérée de chaque document.

Pour obtenir les vecteurs binaires nécessaires pour la première phase nous avons retenu ces trois méthodes pour leur simplicité mais également pour leur facilité de mise en œuvre :

- **1er générateur :** Générer le vecteur comme étant une alternance aléatoire de '0' et de '1'. Ce qui veut dire, que pour construire un vecteur de taille K, il faut tirer K fois un élément dans l'ensemble {'0','1'}.

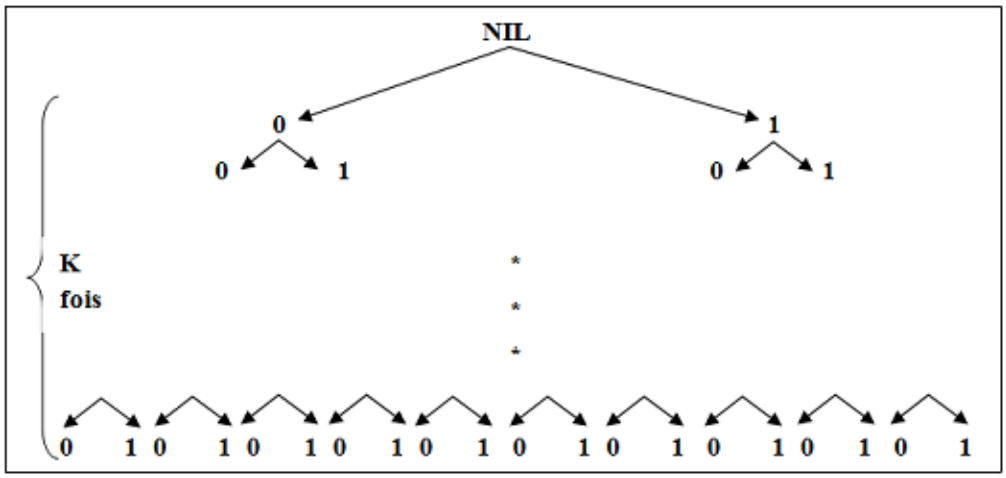


Figure 10 : Arbre de génération des vecteurs avec le 1er générateur.

- **2ème générateur :** La génération se fait en deux étapes : on commence d'abord par générer le nombre de termes à '1' dans un vecteur puis on leur attribut des positions de manière aléatoire dans les K cases du vecteurs en faisant attention à ce qu'une position ne soit attribuée qu'à au plus un élément à la fois.

Exemple : A titre d'exemple, on prend K = 5, nous allons générer un vecteur à 5 bits.

  - Le nombre de termes ou de bit à '1' : sera compris entre 1 et 5 ; soit 2 le nombre tiré pour ce vecteur.
  - Maintenant reste à placer ces deux '1' dans les 5 cases possibles du vecteur.

- **3ème générateur :** Un document ou une requête étant représenté par un vecteur binaire de longueur k, il est possible de l'obtenir en générant un nombre entier

entre 0 et 2^K-1, ensuite un transcodage de ce nombre en base binaire permettra d'aboutir au vecteur binaire de taille K.

Exemple : Considérons que K = 5 ; on tire un nombre entre 0 et 2^5-1 = 31, soit 25 ce nombre, en base binaire 25 s'écrit sous la forme suivante : 25 :11001. On obtient ainsi un vecteur binaire de taille K.

3.2.Recherche exacte (séquentielle)

Pour construire le Benchmark nous avons besoin des données (c.-à-d. documents et requêtes) mais aussi des résultats de recherche pour chaque requête figurant dans ce fichier. Pour obtenir ces résultats seule la méthode exacte que nous appelons ***naïve*** permet de les fournir parce qu'elle consiste en un parcours séquentiel de la collection de documents dans le but d'en déterminer les plus pertinents sur la base du calcul de la similarité.

L'algorithme général de la recherche naïve se présente comme suit : (un seul document)

```
Début
    Pour chaque requête Q
    Faire
        Max_Sim ← 0  // Variable qui va représentée la similarité maximal
        Result ← Premier document de la collection
        Pour chaque document D
        Faire
            S ← Sim (Qw, Dw)
            Si  ( S > Max_Sim )   alors Result ← D.
            Fsi
        Fait
    Fait
Fin
```

Sachant que Sim est la fonction de mesure de la similarité. Cet algorithme prend en considération une seule solution (Un seul document comme résultat); celui qui possède la similarité la plus élevée. Mais il est également possible de considérer une liste de solutions ou de documents qui soient les plus pertinents du point de vu de la similarité. Le principe de l'algorithme reste le même il suffit juste de considérer une liste pour tenir compte des résultats et de les ordonner en fonction de leur similarité. La taille de cette liste peut être fixe ou peut dépendre de la quantité de solutions trouvées.

Pour tenir compte d'une liste de document comme résultat l'algorithme devient comme suit :

```
Début
    Pour chaque requête Q
    Faire
        List_Result ← null
        Pour chaque document D
        Faire
            S ← Sim (Qw, Dw)
            Inserer (D, List_Result) Selon S;
        Fait
    Fait
Fin
```

Avec :

- List_Result est une liste triée dans l'ordre décroissant de la similarité. Cette liste est initialisée à null et est construite à fur et à mesure du parcours de la collection.
- Sim (Qw, Dw) c'est la fonction qui calcul la similarité d'un document D par rapport à une requête Q selon l'une des trois mesures vue précédemment en utilisant la représentation pondérée du document et de la requête.
- Inserer (D, List_Result) Selon S : permet d'insérer un document D dans la liste des résultats en respectant l'ordre de similarité.

3.3. Adaptation de l'essaim d'abeille pour la recherche basée document

Le principe de base de l'approche essaim d'abeille est d'initialiser une abeille pour lancer le processus de recherche de solutions. Cette recherche de solution consiste en une exploration et une exploitation de l'espace de recherche afin d'atteindre progressivement l'objectif recherché. L'abeille initiale va désigner les sommets (documents) qu'elle considère les plus prometteurs à partir de la position où elle se trouve, et les attribuer aux butineuses de la population initiale. Chaque abeille qui s'est vu attribuée un sommet va effectuer sa recherche au sein même de son voisinage et communiquer son résultat à l'ensemble de l'essaim. C'est en fonction de ces résultats que sera tracé le chemin vers la solution du problème.

L'adaptation de l'essaim d'abeilles au problème de recherche d'information consiste à concevoir un essaim d'abeilles pour explorer une collection de documents, à la recherche des

documents pertinents. L'essaim devra se déplacer de régions en régions dans le but d'exploiter celles qui sont prometteuses en tenant compte de la pertinence des documents.

Pour développer l'approche basée sur les essaims d'abeilles nous avons besoin de définir un certain nombre d'éléments clés à savoir :

3.3.1. L'Espace de recherche

L'espace de recherche ou le monde artificiel dans lequel opèrent les abeilles, est la collection de documents représentés par les vecteurs de termes. De ce fait l'espace de recherche est un ensemble de N vecteurs correspondant aux N documents de la collection. Chaque vecteur est une suite de K éléments qui caractérise la fréquence des K termes considérés dans la collection.

Cet espace de recherche va être exploré par l'essaim comme étant un graphe. Ce graphe va inclure toutes les solutions intermédiaires (document) exploitées lors de la recherche et les transitions effectuées par les abeilles de l'essaim dans le but d'atteindre le document le plus pertinent.

3.3.2. La qualité d'un document

La qualité d'un document D, correspond à la valeur de sa fonction objective. Nous rappelons que pour le problème de recherche d'information la fonction objective est la maximisation de la mesure de similarité. Donc la qualité d'un document c'est son taux de similarité avec la requête en question.

3.3.3. La distance d'un document

La distance d'un document est mesurée par le minimum des distances existant entre ce document D et les éléments de la liste Taboue. Sachant que la liste Taboue est utilisée pour éviter de ré-exploiter des régions par lesquelles la recherche est déjà passée. Donc la distance d'un document est le degré d'éloignement de ce document, des documents déjà exploités.

$$Distance\ (D) = \text{Min}\ \{d\ (D, T),\quad T \in \text{Tabou}\}$$

Soient deux documents D et T, la distance entre eux est donnée par la distance de *Hamming* définie comme suit :

$$d(D,T) = \sum_{i=1}^{n} Db_i \oplus Tb_i$$

Où :

- Db et Tb sont respectivement des vecteurs binaires obtenus de B et T en remplaçant chaque terme positif par '1'.
- Dbi, Tbi c'est l'évaluation du terme i respectivement pour le document D et T.

Et $: a \oplus b = \begin{cases} 0 \; si \; a = b \\ 1 \quad sinon \end{cases}$

L'algorithme général de l'approche par essaim d'abeille adaptée à la recherche d'information dans la collection de document est donné par :

Début
Construction de la solution initiale par BeeInit en fonction de la requête Q
Sref ← Document retourné par BeeInit ;
D* ← Sref ; //D* : le document le plus similaire par rapport à la requête.
R ← null ; //R : liste des meilleurs documents trouvés par l'essaim.
Tant que nombre maximal d'itération non atteint
Faire
Insérer Sref dans Tabou ;
Pop_bees = SearchArea (Sref) ;
Affecter à chaque abeille k un sommet de S_k SearchArea ;
Pour chaque abeille k de pop_Bees
Faire
Doc = Meilleur_Voisin (S_k) ;
Stocker le résultat dans la Kème entrée de la table Dance ;
Insérer Doc dans R en respectant l'ordre décroisant de la similarité.
Fait
D* ← Meilleure solution obtenue par les abeilles
Choisir le nouveau sommet de référence Sref ;
Fait
Fin.

3.3.4. La solution initiale :

Initialement, l'abeille *BeeInit* construit la solution de référence *Sref*. Cette solution peut être construite soit de manière aléatoire pour démarrer de n'importe quel document de la collection, ou bien obtenu via une recherche locale ou une heuristique dans le but de démarrer avec une solution de bonne qualité. Une heuristique possible serait de considérer le premier document qui possède au moins un terme en commun avec la requête.

3.3.5. La génération des régions d'exploitation

Une région d'exploitation *SearchArea*, est représentée par un ensemble de M documents (M étant le nombre d'abeille constituant l'essaim). Chacun de ces documents est calculé à partir de *Sref* en inversant un certain nombres de ses termes. Cette inversion est réalisée en tenant compte d'un paramètre Flip qui caractérise la fréquence d'inversion des termes. Cette génération et les inversions qu'elle engendre seront opérer sur la représentation binaire correspondant au document Sref.

Le choix du paramètre Flip est délicat, puisqu'il détermine le nombre de terme à inverser à partir de *Sref*. Une trop petite valeur de ce paramètre, implique que *Sref* est fort probablement l'optimum local de la nouvelle région d'exploitation, ce qui signifie donc que la probabilité d'une amélioration est très faible. En revanche, une valeur très importante permettrait à l'essaim de s'éloigner de la région contenant *Sref* au risque de perdre de bonnes solutions.

Pour procéder aux inversions et ainsi à la génération de la région d'exploitation, nous proposons deux stratégies assurant que les documents obtenus soient aussi distinct que possible. Si le nombre de documents générés s'avère insuffisant, nous aurons recours à une approche aléatoire pour compléter.

Première stratégie

Début
h ← 0 // nombre de document généré par cette stratégie
Tant que taille de *SearchArea* non atteinte et h < Flip
Faire
D ← *Sref*
K ← 0
Répéter
Inverser D [Flip*k+h]
k ← k + 1
Jusqu'à Flip*k+h ≥ n
SearchArea ← *SearchArea* ⋃ {D}
h ← h + 1
Fait
Fin.

Deuxième stratégie

Début

h ← 0

Tant que taille de *SearchArea* non atteinte et h < Flip

Faire

D ← *Sref*

K ← 0

Répéter

Inverser D [n/Flip*h+k]

k ← k + 1

Jusqu'à k ≥ n/Flip

SearchArea ← *SearchArea* ⋃ {D}

h ← h + 1

Fait

Fin.

Avec : n le nombre de termes dans le vecteur représentatif du document D.

Exemple : Supposons que n = 20 et Flip = 5. Sachant que les termes dans le vecteur sont indicés de 0 à 19.

Appliquer la stratégie 1, consiste à inverser les termes aux positions suivantes :

(0, 5, 10, 15), (1, 6, 11, 16), (2, 7, 12, 17), (8, 3, 13, 18), (4, 9, 14, 19).

La première stratégie permet d'inverser un nombre n/Flip distant deux à deux dans le vecteur de Flip termes.

Appliquer la stratégie 2, consiste à inverser les termes aux positions suivantes :

(0, 1, 2, 3), (4, 5, 6, 7), (8, 9, 10, 11), (12, 13, 14, 15), (16, 17, 18, 19).

La seconde stratégie quant à elle permet d'inverser un nombre n/Flip termes contigües.

La stratégie aléatoire est utilisée pour compléter la région dans le cas ou celles générées ne comblent pas toute la taille de la région de recherche, elle peut également être utilisée comme stratégie à part entière pour générer tous les documents de la région d'exploitation.

L'algorithme de la stratégie aléatoire se présente comme suit :

Stratégie aléatoire

Début

Tant que taille SearchArea non atteinte

Faire

D ← Sref

Inverser aléatoirement n/Flip termes de D

SearchArea ← SearchArea ⋃ {D}

Fait

Fin.

3.3.6. Exploitation d'une région

Une fois que chaque abeille de l'essaim se voie attribuer un document de la région d'exploitation, chacune entreprend sa recherche dans le but de déterminer le meilleur document dans son voisinage.

Le voisinage d'un document est constitué de l'ensemble des documents qui ne diffèrent que d'un seul terme avec le document en question. De ce fait, la construction du voisinage d'un document quelconque D se fait en inversant à chaque fois un seul terme dans la structure représentative du document.

Considérons un document D pour lequel nous nous apprêtons à construire le voisinage V. La construction du voisinage se fait au moyen de l'algorithme suivant :

Début

k ← 0

Tant que k < n

Faire

D' ← D

Inverser D' [k]

V ← V U {D'}

k ← k + 1

Fait.

Fin.

Exemple : Prenons un document D = '01101'

Le voisinage de D est défini par : {'11101','00101', '01001', '01111', '01100'}.

3.3.7. La recherche d'une abeille

La recherche d'une abeille est simulée par une recherche locale, dite recherche locale par étapes : c'est un processus itératif à deux phases :

- Une phase de recherche locale simple, qui consiste à évoluer dans le voisinage du document qui a été attribué à l'abeille dans le but de déterminer l'optimum locale dans sa région d'exploration.
- Une fois que l'optimum local est atteint, la deuxième phase consiste en l'amélioration de l'optimum local obtenu. Cette amélioration est effectuée en inversant le maximum de termes pour que le nouveau document soit de similarité supérieure ou égale à celle obtenu par l'optimum local de la première phase.

La recherche effectuée par chaque abeille est décrite par l'algorithme suivant :

```
Début
    D un document donné
    Dw ← Représentation pondérée du document D
    Db ← Représentation binaire du document D
      Répéter
        Sortie ← faux
        Tant que (non sortie)
        Faire
          Tant que (tous le voisinage n'a pas été explorer) et (non améliore)
          Faire
            D' ← voisin_courant (D)
            (Qw, Dw') ≥ Sim (Qw, Dw)
            Si Sim (Qw, Dw') > Sim (Qw, Dw)
                alors D ← D'
                améliore = vrai ;
            Fsi
          Fait
          Si améliore ← faux ;
          alors sortie ← vrai ;
          Fsi
        Fait
        Best ← Sim (Qw, Dw)
        Pour i ← 1 à n
        Faire
          Db_i ← 1 – Db_i /* Di étant le i^ème terme du document D*/
          Si ( Sim (Qw, Dw) < Best )  alors Db_i ← 1 – Db_i
          Fsi
        Fait
      Jusqu'à MaxIter
Fin
```

Avec MaxIter est le nombre d'itération maximal que peut effectuer une abeille.

3.3.8. Notion d'admissibilité

L'espace de recherche sur lequel l'essaim devra opérer est limité au périmètre de la collection. Lorsqu'une abeille se trouve devant une solution susceptible d'être la meilleure dans sa vision de l'environnement, celle-ci devra s'assurer que cette solution est admissible. En d'autres termes, elle devra s'assurer que cette solution ou document existe bel et bien dans la collection. Une solution est admissible si elle apparait dans la collection. Mais il se peut que l'abeille tombe sur une solution ou un document qui n'y figure pas. Dans ce cas, la solution est rejetée.

Vérifier qu'une solution est admissible ou déterminer la solution la plus ressemblante qui est admissible pourrait être une opération très couteuse en temps surtout avec la taille gigantesque que peut avoir la collection. Un mécanisme de hachage est nécessaire ici pour rendre cette tâche moins contraignante. Le principe est simple, il s'agit d'attribuer à chaque document une clé de hachage unique et la recherche de l'existence d'un document dans la collection se fera à présent à l'aide de la clé de hachage. La clé de hachage considérée dans notre cas est la conversion en entier de la représentation binaire du document. Si le document existe l'entrée de l'index correspondant à cette clé contient l'adresse du document de la collection associé. Dans le cas contraire cette entrée comporte 'nil' ou '-1'.

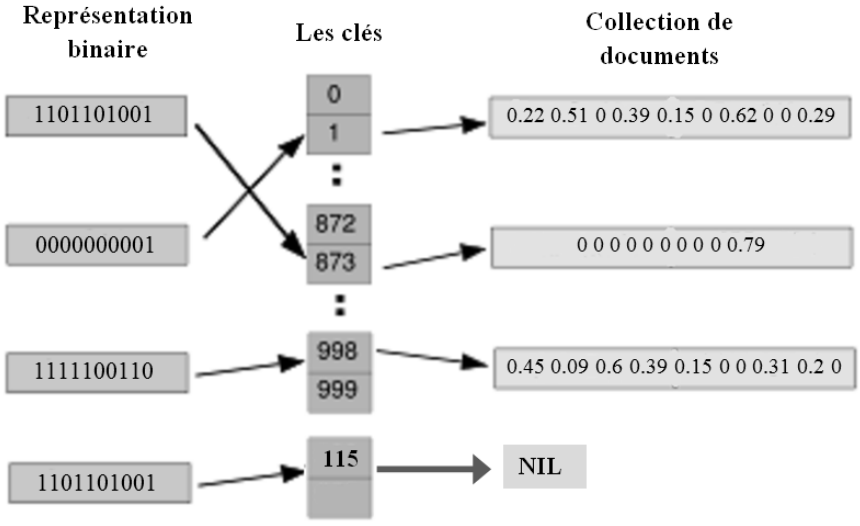


Figure 11 : Schéma générale du calcul d'admissibilité.

Si une clé existe pour cette solution ou document cela voudra dire que cette solution est admissible autrement elle ne l'est pas.

C’est qu’une fois que la solution admissible trouvée que l’abeille pourra communiquer son résultat à l’ensemble de l’essaim.

3.3.9. Mode de communication

Chaque abeille de l’essaim œuvre indépendamment des autres abeilles constituant l’essaim. Cette indépendance n’est pas absolue, à chaque fois qu’une abeille trouve l’optimum local de sa région elle devra le communiquer à l’ensemble de l’essaim. La communication entre les abeilles est assurée par une table *Danse* dans laquelle tous les résultats obtenus à cette étape par l’ensemble de l’essaim seront stockés. C’est aussi en fonction du contenu de cette table que sera construite la nouvelle solution de référence *Sref*, pour continuer la recherche.

3.3.10. Le choix de la solution de référence

La solution de référence choisie à l’itération t+1 dépend de la solution de référence à l’itération t notée Sref(t) et de la meilleure solution obtenue à l’itération t+1 notée Best. Ce choix est conditionné par la quantité suivante :

$$\Delta \text{ Sim} = \text{Sim (Q, Best)} - \text{Sim (Q, Sref(t))} ;$$

Cette quantité représente le taux d’amélioration de la solution lors de l’itération courante par rapport à la solution de référence par laquelle cette itération a débuté. Si Δ Sim est positive cela signifie que la solution a été améliorée au court de cette itération, par contre si cette quantité est négative cela se traduit par une altération de la qualité de la solution à l’itération courante par rapport à la solution de référence de cette itération.

- Si Δ Sim > 0 la solution de référence à retenir est la meilleure en qualité (qui possède la meilleure similarité et qui n’est pas en liste Taboue)
- Autrement, deux cas de figure se présentent :
  - Si le nombre de chance donné à l’essaim pour améliorer la solution n’a pas été atteint, la solution retenue sera la meilleure en qualité.
  - Mais si le nombre de chance a été atteint, on retient la solution meilleure en diversité dans le but d’explorer d’autres régions.
- Si deux solutions sont de même qualité, c’est celle qui possède le degré de diversité le plus élevé qui sera choisie. De même que si deux solutions présentent le même degré de diversité, c’est celle qui améliore la similarité qui sera retenue.

Il peut également arriver que toutes les solutions de *Danse* existent dans la liste Taboue, dans ce cas la solution de référence sera générée aléatoirement.

L'algorithme du choix de la solution de référence est donné par :

```
Début
    Si ( Δ Sim > 0 )
        alors Sref ← La meilleure solution en qualité ;
              Si NbChances < MaxChances
                  alors NbChances ← MaxChances ;
              Fsi ;
        Sinon NbChances ← NbChances – 1 ;
                  Si nbChances > 0
                      alors Sref ← La meilleure solution en qualité ;
                          Sinon Sref ← La meilleure solution en diversité ;
                                NbChances ← MaxChances
                  Fsi.
    Fsi
Fin.
```

3.3.11. Paramètres empiriques

Comme toutes métaheuristiques, l'approche basée sur les essaims d'abeilles est bâtie autour d'un certains nombre de paramètres réglables empiriquement. L'objectif de ces paramètres est d'adapter l'essaim aux caractéristiques du problème. Cette adaptation ne peut être faite que grâce à de nombreuses expérimentations. Ces paramètres sont les suivants :

- ***Flip :*** la fréquence d'inversion lors de la génération de la région d'exploitation.
- ***Nbees :*** Nombre d'abeilles de l'essaim.
- ***MaxIters :*** Nombre d'itérations de la recherche locale qu'effectue chaque abeille
- ***SearchIters :*** Nombre d'itérations qu'effectuera l'essaim lors de la recherche.
- ***MaxChances :*** Nombre de chances laissées à l'essaim pour améliorer la solution.

4. Conclusion

Dans ce chapitre, nous avons fourni une description sommaire de l'adaptation de la métaheuristique de l'optimisation par essaim d'abeilles pour la résolution du problème de la recherche d'information à grande échelle. Cette première adaptation consiste à faire évoluer tout un essaim d'abeilles sur une collection de document qui peut être de taille très importante. Cette adaptation consiste en un parcours de la collection selon le procédé de la métaheuristique. Les éléments essentiels de cette adaptation, et dont dépend la puissance et l'efficacité de l'algorithme sont le choix de *Sref*, la stratégie de génération de *SearchArea*.

Une seconde adaptation de l'essaim d'abeille pour ce problème, serait d'explorer le fichier inverse à la place de la collection complète des documents. Cette étude fait l'objet du prochain chapitre.

Chapitre IV

Une approche basée sur les essaims d'abeilles pour la recherche d'information dans le fichier inverse

3. Introduction

Le précédent chapitre présente une première adaptation de l'essaim d'abeille pour une recherche dans la collection de documents. L'approche dans ce cas considère l'ensemble de la collection de documents comme espace de recherche. Une autre approche possible serait de limiter l'espace de recherche au fichier inverse c'est-à-dire aux documents qui partagent au moins un terme en commun avec la requête. A priori, cette démarche semble plus efficace puisque l'exploration se fera dans un sous ensemble de l'espace considéré dans la première approche.

Ce présent chapitre, offre une amélioration en tenant compte des termes cette fois et non pas des documents. Cette amélioration a été inspirée du fonctionnement clé du système SMART qui fait recours à un index inverse caractérisant chaque terme et tous les documents qui les engendrent.

4. Approche basée sur les essaims d'abeilles pour la recherche dirigée par les termes

En s'inspirant du fonctionnement de SMART et particulièrement de la phase d'évaluation des requêtes nous pouvons améliorer l'algorithme de recherche dirigée par le « document » pour tenir compte du même principe d'évaluation basée sur les termes des documents. Le principe est tel que la recherche n'a plus besoin de tenir compte de toute la collection, mais de restreindre sa recherche à un sous ensemble qui contient uniquement les documents qui ont au moins un terme en commun avec la requête.

De même que pour l'approche précédente, l'objectif principal dans celle-ci reste de pouvoir retrouver une information particulière à partir d'une large collection de documents. La différence réside dans le fait que cette approche tente de réduit l'espace de recherche aux seuls documents qui ont un lien avec la requête.

Le modèle de recherche d'information reste le même adopté dans l'approche précédente. La principale différence réside dans l'utilisation du fichier inverse pour l'exploration de la collection.

4.1. Recherche d'information dans le fichier inverse

Lorsque la recherche d'information se basait sur le fichier index, la recherche consistait à parcourir l'ensemble de l'index (dans le cas exacte), ou de l'explorer selon le

procédé de recherche (dans le cas de l'approche basée sur l'essaim d'abeilles). Ce parcours à pour but d'évaluer la similarité de chacun des documents par rapport à la requête.

Une recherche à partir du fichier inverse consiste à démarrer de la requête d'en extraire les termes considérées (les termes ayant une fréquence supérieure à 0). Pour chacun de ces termes parcourir la liste des documents contenant ce terme à partir du fichier inverse.

La recherche d'information à partir du fichier inverse respect l'algorithme suivant :

```
Début
    Result ← null
    Pour chaque requête Q
    Faire
        Pour chaque terme de la requête Q
        Faire
            Si ( Qw[i] > 0 ) alors
                Pour chaque document D dans Inverse[i]
                Faire
                    S ← Sim (Q, D)
                    Inserer (D, Result) selon S
                Fait
            Fsi
        Fait
    Fait
Fin
```

Avec Qw[i] est le poids associé au terme i dans la requête Q, et Inverse[i] représente dans le fichier inverse l'entrée correspondant au terme i. En d'autres termes, représente la liste des documents de la collection contenant le terme i.

4.2. Espace de recherche

L'espace de recherche dans une recherche basée sur le fichier inverse est un sous ensemble de la collection de document noté C et qui comporte tous les documents ayant au moins un terme en commun avec la requête Q. C = $\bigcup_i Inverse[i]$, où i est un terme considéré dans la requête utilisateur Q.

C'est sur cette espace de recherche que devra se basée l'essaim d'abeilles en vu de déterminer les documents les plus pertinents comme réponse à l'utilisateur.

4.3. Construction du fichier inverse

La construction du fichier inverse se fait au même temps que l'indexation de la collection et la normalisation des fréquences sous tf*idf.

La construction se fait comme suit :

```
Début
    Pour chaque document D de la collection
    Faire
        Pour chaque terme du document D
        Faire
            Si ( Dw[i] > 0 )   Inserer ((D, Dw[i]), Inverse[i])
            Fsi
        Fait
    Fait
Fin.
```

4.4. Recherche dans le fichier inverse avec une approche basée sur les essaims d'abeilles

Le principe de fonctionnement de l'approche basée sur l'essaim d'abeilles reste inchangé, néanmoins, certaine amélioration doivent être apportée pour tenir compte du fichier inverse et pour procéder à une recherche dirigée par les termes.

En tenant compte du fichier inverse, et pour effectuer une recherche dirigée par les termes, le principe consiste à construire ce document terme par terme en tenant compte des termes présents dans la requête.

L'approche basée sur l'essaim d'abeilles pour une recherche dans le fichier inverse est indiqué par l'algorithme suivant :

```
Début
    Construction de la solution initiale par BeeInit en fonction de la requête Q
    Sref ← Document retourné par BeeInit ;
    D* ← Sref ; //D* : le document le plus similaire par rapport à la requête.
    R ← null ; //R : liste des meilleures documents trouvées par l'essaim.
    Pour chaque terme de la requête Q
    Faire
        Si ( Qw[i] > 0 )
            Tant que nombre maximal d'itération non atteint
            Faire
                Insérer Sref dans Tabou ;
                Pop_bees = SearchArea (Sref, i) ;
                Affecter à chaque abeille k un sommet de Sk SearchArea ;
                Pour chaque abeille k de pop_Bees
                Faire
                    Doc = Meilleur_Voisin (Sk,, i) ;
                    Stocker le résultat dans la Kème entrée de la table Dance ;
                    Inserer Doc dans R en respectant l'ordre décroisant de la similarité.
                Fait
                D* ← Meilleur solution obtenue par les abeilles
                Choisir le nouveau sommet de référence Sref ;
            Fait
        Fsi
    Fait
Fin.
```

4.4.1. Solution initiale

L'abeille BeeInit construit la solution de référence Sref à partir de l'espace de recherche de manière aléatoire qui signifie tirer au hasard un document qui à au moins un terme en commun avec la requête.

4.4.2. Génération des régions d'exploitation

La génération des régions d'exploitations reste basée sur la solution de référence et des multiples inversions effectuées en tenant compte du paramètre Flip et en tenant compte aussi du terme en cours. Les inversions ne devront pas influer sur le terme en cours, il restera figer dans toutes les régions d'exploitations. En d'autres termes, la recherche doit préserver ce terme dans les régions d'exploitations futures.

4.4.3. L'exploitation d'une région

L'exploitation d'une région se fait une fois que chaque abeille s'est vu attribuer un sommet. Cette exploitation se traduit par la construction du voisinage de chaque sommet attribué tout en maintenant le terme en cours. En d'autres termes, la construction du voisinage va tenir en compte le terme en cours eu vue de maintenir sa présence dans les solutions futurs.

4.4.4. La recherche d'une abeille

Chaque abeille effectue une recherche locale par étages en vu de déterminer la meilleure solution dans sa région d'exploitation. Cette recherche est effectuée de manière identique à la recherche d'une abeille dans l'approche de recherche dans la collection de document. La seule différence et que celle-ci lors de son processus de recherche tente de prolonger l'existence du terme en cours dans les solutions futurs.

4.4.5. Notion d'admissibilité

Pour que la recherche ne sorte pas en dehors du périmètre de l'espace de recherche, la notion d'admissibilité en plus de déterminer l'appartenance d'un document à la collection, elle devra également déterminer s'il appartient ou non à l'espace de recherche considéré. Si un document appartient à l'espace de recherche il sera jugé admissible.

4.4.6. Le choix de la solution de référence

Le choix de la solution de référence est basé sur le taux d'amélioration de la solution obtenue par rapport à la solution de référence par laquelle l'itération en cours à débuter.

La solution de référence à choisir peut faire l'objet de l'itération suivante ou bien du terme suivant. Dans les deux cas le choix se fera en respectant le procédé défini dans l'approche précédente.

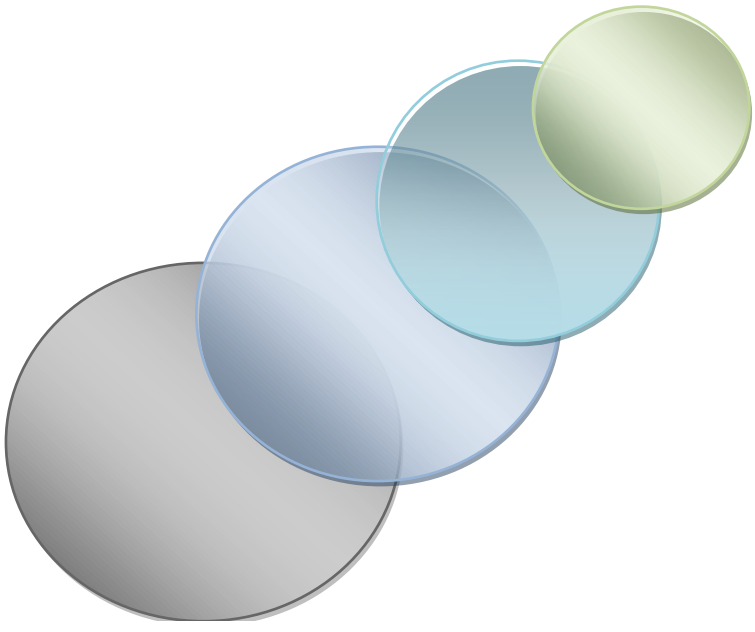
5. Conclusion

Le but de ce chapitre été d'aboutir à une amélioration qui puisse réduire un peu l'espace de recherche qui est considéré comme gigantesque en excluant tous les documents qui n'ont pas de relation avec les termes de la requêtes. En vue de comparer les performances des deux propositions nous avons fait recours à plusieurs tests dans le but de déterminer l'impacte de chaque contribution sur les résultats de la recherche.

La partie ci-après montre plus en détail tous les résultats obtenus avec chacune des deux méthodes.

Partie III:

Expérimentation



Chapitre V

Implémentation et résultats expérimentaux

4. Introduction

Après avoir présenté le problème de recherche d'information et les approches proposées pour aborder la RI à grande échelle avec les essaims d'abeilles, nous présentons dans ce chapitre le résumé des nombreux tests effectués sur les collections générées. L'analyse des résultats obtenus nous informera sur les performances des différents algorithmes.

5. Implémentation et environnement expérimental

Dans le but de tester les performances des algorithmes présentés dans la partie précédente nous avons eu besoin d'implémenter :

- Un générateur de Benchmark (pour chaque méthode de génération)
- Une recherche exacte séquentielle
- Une recherche exacte basée sur le fichier inverse
- Une recherche basée sur l'essaim d'abeilles à partir de la collection
- Une recherche basée sur l'essaim d'abeilles à partir du fichier inverse

Toutes ces implémentations ont été faites dans un environnement Visual Studio sous Windows Vista sur une machine dual Core 2.0 ayant 2 Go de RAM.

6. Résultats expérimentaux

6.1. Génération de Benchmark

Pour les raisons des tests que nous devrons effectuer sur ces Benchmarks nous avons généré les quatre Benchmarks suivants :

Benchmark	***Taille de la collection***	***Nombre de termes***	***Nombre de requêtes***
B1	100 000	20	20
B2	500 000	20	20
B3	1 000 000	25	20
B4	2 000 000	25	20

Tableau 3: Structure des Benchmarks

6.1.1. Choix de la méthode de génération

Le choix de la méthode de génération est conditionné par la diversification des collections qu'elle produit. Etant données des collections aléatoires, pour pouvoir donner un sens aux résultats obtenus en exploitant ces collections, ces dernières doivent être les plus diversifiées possibles.

Après plusieurs tests nous avons vu que les trois générateurs été plus ou moins équivalents et que le choix de l'un ou de l'autre des générateurs n'avait pas d'influence sur les résultats de la recherche.

6.1.2. Choix de la mesure de similarité

Dans le but de choisir la mesure de similarité à retenir pour la suite des tests à faire nous avons effectué les opérations suivantes :

- Générer les Benchmark B1, B2, B3 et B4.
- Evaluer les requêtes de chaque Benchmark à travers une recherche exacte sur la collection des documents.
- Comparer pour chaque collection les résultats obtenue par chacune des mesures
- Comparer par rapport aux quatre collections l'impact de la mesure utilisée sur les résultats obtenus.

Au terme de ces opérations nous avons obtenu les résultats résumés dans le tableau suivant, les résultats sont exprimés au moyen de la similarité du meilleure document fournit dans la réponse :

	Collection B1			***Collection B2***			***Collection B3***			***Collection B4***		
	Cos	***Dice***	***Jac***	***Cos***	***Dice***	***Jac***	***Cos***	***Dice***	***Jac***	***Cos***	***Dice***	***Jac***
Q1	***0.92***	0.46	0.30	***0.92***	0.45	0.29	***0.92***	0.35	0.21	***0.94***	0.37	0.22
Q2	***0.93***	0.49	0.32	***0.95***	0.64	0.47	***0.93***	0.35	0.21	***0.94***	0.44	0.28
Q3	***0.92***	0.47	0.31	***0.94***	0.46	0.30	***0.91***	0.28	0.16	***0.92***	0.36	0.22
Q4	***0.91***	0.39	0.24	***0.93***	0.44	0.28	***0.93***	0.25	0.14	***0.94***	0.31	0.18
Q5	***0.90***	0.40	0.25	***0.93***	0.41	0.26	***0.91***	0.31	0.18	***0.92***	0.39	0.16
Q6	***0.89***	0.50	0.33	***0.92***	0.32	0.19	***0.91***	0.23	0.13	***0.91***	0.29	0.17
Q7	***0.90***	0.48	0.31	***0.90***	0.69	0.52	***0.91***	0.35	0.21	***0.93***	0.32	0.19
Q8	***0.95***	0.38	0.23	***0.92***	0.53	0.36	***0.92***	0.45	0.29	***0.91***	0.34	0.20
Q9	***0.93***	0.42	0.27	***0.94***	0.36	0.22	***0.91***	0.43	0.27	***0.93***	0.38	0.23
Q10	***0.92***	0.39	0.24	***0.94***	0.38	0.24	***0.92***	0.42	0.26	***0.92***	0.57	0.40
Q11	***0.92***	0.38	0.23	***0.95***	0.42	0.26	***0.91***	0.34	0.20	***0.93***	0.28	0.16
Q12	***0.91***	0.43	0.28	***0.94***	0.40	0.25	***0.92***	0.34	0.20	***0.94***	0.43	0.27
Q13	***0.90***	0.46	0.29	***0.94***	0.47	0.31	***0.92***	0.40	0.25	***0.95***	0.75	0.60
Q14	***0.90***	0.31	0.18	***0.95***	0.38	0.23	***0.89***	0.36	0.22	***0.94***	0.38	0.24
Q15	***0.92***	0.32	0.19	***0.94***	0.33	0.20	***0.90***	0.45	0.29	***0.94***	0.38	0.23
Q16	***0.92***	0.45	0.29	***0.93***	0.47	0.31	***0.92***	0.25	0.14	***0.92***	0.35	0.21

Q17	***0.89***	0.38	0.23	***0.95***	0.53	0.36	***0.91***	0.40	0.25	***0.92***	0.35	0.21
Q18	***0.92***	0.59	0.42	***0.96***	0.37	0.23	***0.91***	0.34	0.21	***0.93***	0.32	0.19
Q19	***0.92***	0.66	0.49	***0.95***	0.31	0.18	***0.92***	0.30	0.18	***0.92***	0.46	0.29
Q20	***0.90***	0.32	0.19	***0.97***	0.55	0.38	***0.90***	0.27	0.16	***0.93***	0.40	0.25

Tableau 4: Impact de la mesure de similarité sur les résultats obtenus pour chaque collection.

A partir de ces résultats nous pouvons définir la similarité moyenne obtenue sur l'ensemble des documents figurants dans la réponse :

Mesures / collections	***Collection B1***	***Collection B2***	***Collection B3***	***Collection B4***
Cosinus	0.91	0.93	0.91	0.93
Dice	0.43	0.44	0.34	0.39
Jaccard	0.27	0.29	0.20	0.23

Tableau 5: Similarité moyenne pour chaque collection sur l'ensemble des requêtes.

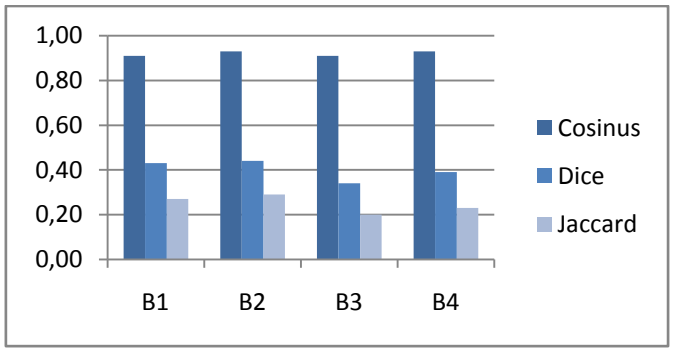


Figure 12 : Histogramme des mesures de similarités.

Les résultats obtenus montrent clairement que la formule du Cosinus est la plus adéquate pour les collections utilisées.

6.2.Recherche exacte à partir de la collection

Pour la recherche exacte à partir de la collection de documents, nous allons utiliser les Benchmarks B1, B2, B3 et B4. Pour le calcul de similarité nous allons utiliser la formule du Cosinus et évaluer le temps nécessaire pour la détermination d'un ensemble de documents noté R comportant les documents les plus pertinents. Cet ensemble est limité aux 20 premiers documents au sens de la similarité.

Requêtes	*Collection B1*		*Collection B2*		*Collection B3*		*Collection B4*	
	Sim	*tps(s)*	*Sim*	*tps (s)*	*Sim*	*tps (s)*	*Sim*	*tps (s)*
Q1	0.93	1.375	0.95	6.981	0.91	16.953	0.94	31.580
Q2	0.91	1.403	0.95	6.628	0.92	16.157	0.94	31.207
Q3	0.93	1.309	0.96	6.852	0.95	15.884	0.92	35.722
Q4	0.92	1.289	0.92	6.728	0.94	15.125	0.94	33.363
Q5	0.93	1.322	0.94	6.426	0.92	16.226	0.92	32.576
Q6	0.92	1.313	0.95	6.488	0.93	16.064	0.91	34.615
Q7	0.94	1.325	0.94	6.474	0.93	15.436	0.93	32.272
Q8	0.94	1.491	0.94	6.731	0.90	16.881	0.91	31.666
Q9	0.96	1.285	0.95	6.791	0.91	15.542	0.93	34.252
Q10	0.94	1.350	0.95	7.115	0.91	17.411	0.92	31.138
Q11	0.91	1.665	0.95	6.725	0.91	17.376	0.93	32.073
Q12	0.90	1.340	0.95	6.687	0.94	16.775	0.94	31.954
Q13	0.94	1.391	0.94	6.509	0.95	16.142	0.95	31.690
Q14	0.92	1.390	0.94	6.521	0.90	17.196	0.94	32.280
Q15	0.96	1.300	0.96	6.755	0.90	15.848	0.94	31.739
Q16	0.95	1.328	0.95	6.849	0.92	16.170	0.92	34.078
Q17	0.93	1.526	0.92	6.993	0.90	15.450	0.92	31.740
Q18	0.94	1.360	0.93	6.695	0.94	14.706	0.93	31.127
Q19	0.90	1.377	0.92	6.622	0.90	15.996	0.92	34.131
Q20	0.92	1.374	0.94	6.382	0.97	16.126	0.93	32.683

Tableau 6 : Résultats obtenus par la recherche exacte.

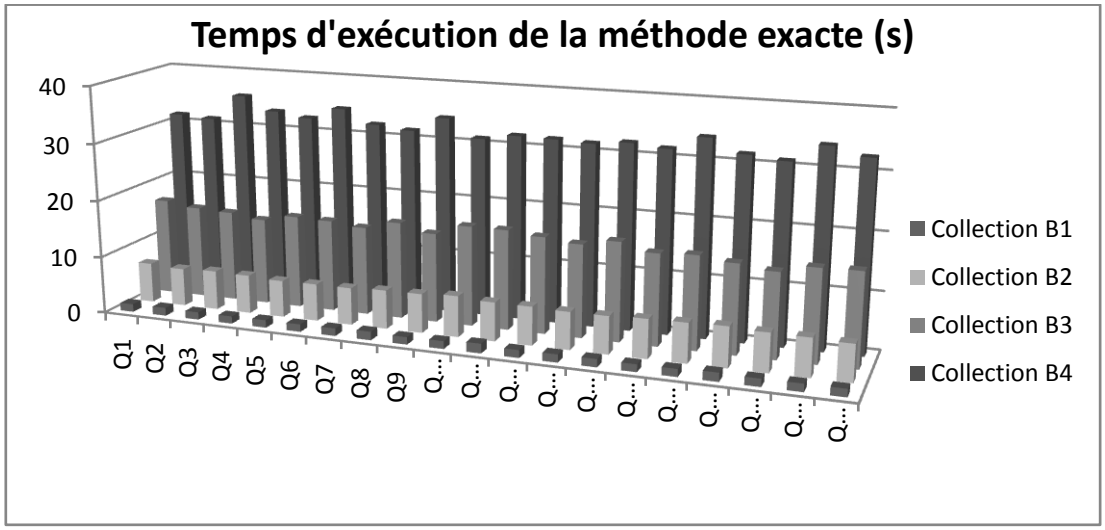


Figure 13 : Proportion des temps d'exécution des 20 requêtes pour chaque collection.

A partir des résultats obtenus nous pouvons calculer le temps d'exécution moyen proportionnel aux tailles des différentes collections.

	Collection B1	*Collection B2*	*Collection B3*	*Collection B4*
Taille	100 000	500 000	1 000 000	2 000 000
Temps d'exécution moyen (ms)	1 375.65	6 697.6	16 223.2	32 594.3

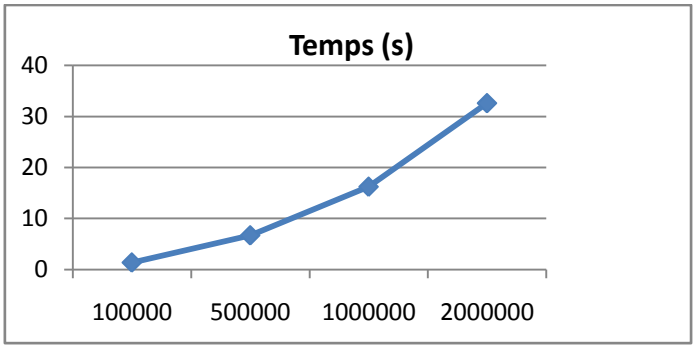


Figure 14 : Progression du temps d'exécution par rapport à la taille des collections.

Ce que nous pouvons remarquer de l'histogramme qui englobe les temps d'exécutions pour chaque requête de chaque collection, est que pour une même collection même si le principe de la recherche n'est autre qu'un parcours séquentiel de l'ensemble de la collection, le temps d'exécution peut varier d'une requête à une autre. Cette variation est due à la construction de l'ensemble des solutions qui doit être trié afin d'assurer que les solutions retournées soient les 20 meilleures du point de vu de leur similarité par rapport à la requête.

L'analyse des deux représentations graphiques qui précèdent montre que dans le cas d'une recherche séquentielle le temps d'exécution augmente de manière significative en fonction de la taille de la collection. Ces temps restent abordables puisque le principe de l'algorithme exacte sur la collection cherche à déterminer les meilleures solutions sans se soucier du temps d'exécution qui peut devenir de plus en plus important plus la taille des collections augmente.

Les résultats obtenus par la recherche exacte sur la collection de documents vont servir de référentiel pour la comparaison avec les résultats obtenus par l'approche basée sur l'essaim d'abeilles sur la collection de documents. Sur les 20 documents trouvés par la méthode exacte un jugement de pertinence sera appliqué pour en tenir compte lors du calcul des précisions et rappels pour les prochains tests en vu d'évaluer les performances de l'approche de recherche basée sur les essaims d'abeilles sur la collection de documents.

6.3. Recherche exacte à partir du fichier inverse

Pour la recherche exacte à partir du fichier inverse nous avons utilisé les mêmes Benchmarks que pour la recherche exacte à partir de la collection de documents. De même que dans la recherche dans la collection nous avons utilisé la formule du Cosinus pour la mesure de similarité. Nous avons construit un ensemble des 20 meilleurs documents (au sens de la similarité). Et pour chaque requête nous avons examiné le temps nécessaire pour obtenir ces documents.

Requêtes	*Collection B1*		*Collection B2*		*Collection B3*		*Collection B4*	
	Sim	*tps (ms)*	*Sim*	*tps (ms)*	*Sim*	*tps (ms)*	*Sim*	*tps (ms)*
Q1	0.93	250	0.95	1497	0.91	3806	0.94	3303
Q2	0.91	241	0.95	1289	0.92	**5066**	0.94	3414
Q3	0.93	234	0.96	1177	0.95	2466	0.92	7215
Q4	0.92	199	0.92	2131	0.94	1280	0.94	8620
Q5	0.93	245	0.94	**611**	0.92	4870	0.92	**8724**
Q6	0.92	258	0.95	1187	0.93	2879	0.91	6675
Q7	0.94	203	0.94	1400	0.93	1958	0.93	6578
Q8	0.94	**485**	0.94	996	0.90	3365	0.91	6366
Q9	0.96	148	0.95	**3017**	0.91	1517	0.93	4304
Q10	0.94	314	0.95	1547	0.91	3336	0.92	5074
Q11	0.91	315	0.95	1254	0.91	2584	0.93	6415
Q12	0.90	255	0.95	2230	0.94	3260	0.94	3911
Q13	0.94	474	0.94	1209	0.95	2546	0.95	**1045**
Q14	0.92	312	0.94	1466	0.90	4486	0.94	6857
Q15	0.96	174	0.96	1516	0.90	3873	0.94	5031
Q16	0.95	**144**	0.95	2692	0.92	3837	0.92	4573
Q17	0.93	405	0.92	2314	0.90	2348	0.92	6946
Q18	0.94	298	0.93	1932	0.94	1157	0.93	5246
Q19	0.90	480	0.92	1622	0.90	2950	0.92	3817
Q20	0.92	361	0.94	850	0.97	**1127**	0.93	4783

Tableau 7 : Résultats obtenus par la méthode exacte à partir du fichier inverse.

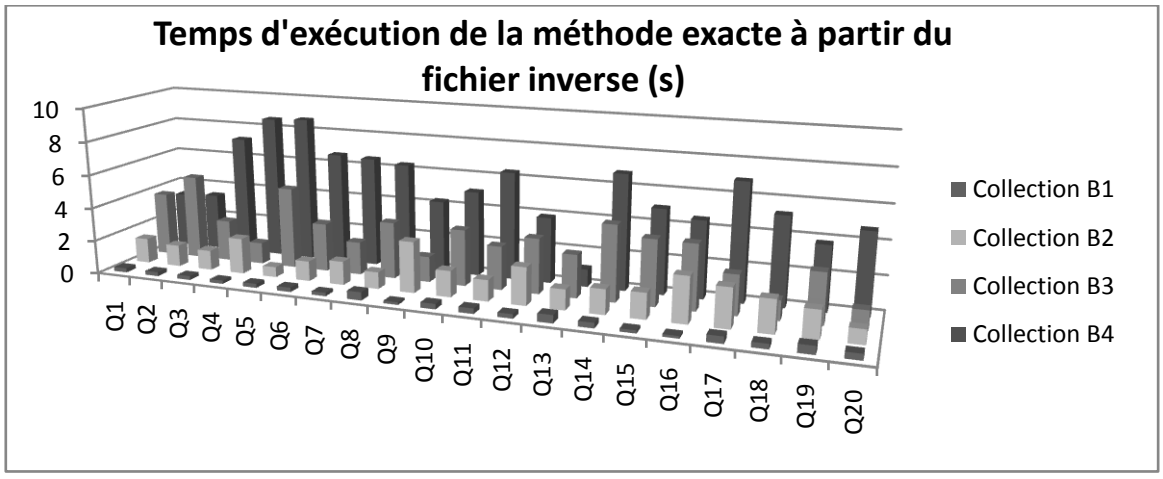


Figure 15 : Proportion des temps d'exécution des 20 requêtes pour chaque collection.

A partir des résultats obtenus nous pouvons calculer le temps d'exécution moyen proportionnel aux tailles des différentes collections.

	Collection B1	*Collection B2*	*Collection B3*	*Collection B4*
Taille	100 000	500 000	1 000 000	2 000 000
Temps d'exécution moyen (ms)	289.75	1596.85	2935.55	5444.85

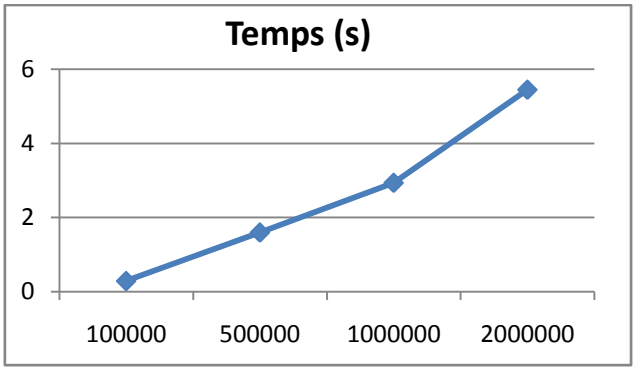


Figure 16 : Progression du temps d'exécution par rapport à la taille des collections.

Au cours de la recherche exacte à partir du fichier inverse le temps d'exécution varie d'une requête à une autre en fonction du nombre de termes que cette requête comporte vu que cette recherche est basée sur ces termes, plus la requête comporte de termes plus le nombre de documents évalués augmente, mais ce nombre de documents ne représente qu'un sous ensemble de la collection.

En tenant compte des deux représentations graphiques ci-avant, nous remarquons que le temps d'exécution augmente en fonction de la taille des collections sur lesquelles les tests ont été effectués. Ce temps d'exécution dépend également du nombre de termes que chaque requête comporte, vu que pour une même collection ce temps peut être le double ou même le triple pour des requêtes portant sur la même collection.

Une comparaison visuelle des représentations graphiques obtenues par la recherche à partir de la collection et la recherche à partir du fichier inverse permet de constater que les résultats obtenus en termes de similarité sont identiques vu que les deux recherches sont exactes et cherchent à déterminer les meilleures solutions qui soient et en travaillant sur les mêmes collections. En ce qui concerne le temps d'exécution, le simple fait de réduire l'ensemble des documents évalués réduit considérablement ce temps.

De même que les résultats de la recherche sur la collection de documents, les résultats obtenus par la recherche exacte sur le fichier inverse vont servir de référentiel pour la comparaison avec les résultats obtenus par l'approche basée sur l'essaim d'abeilles sur le fichier inverse en vu d'évaluer les performances de cette dernière.

6.4. Recherche basée sur les essaims d'abeilles à partir de la collection

Dans le but d'évaluer les performances de l'approche de recherche basée sur les essaims d'abeilles à partir de la collection de documents, nous avons effectué un certain nombre de tests en vu de déterminer les résultats que fournit cette approche de recherche pour chacune des collections précédemment utilisées. Mais avant d'effectuer ces tests nous avons ajusté les valeurs des paramètres empiriques.

Les paramètres empiriques ont une grande influence sur les performances des algorithmes. C'est la raison pour laquelle, nous avons consacré un temps, et des efforts considérables, pour déterminer les valeurs permettant d'obtenir les meilleurs résultats. Une valeur de paramètre est jugée bonne, si elle donne les meilleurs résultats en termes de qualité (précision et rappel) comme premier critère, et en termes de temps de calcul comme second critère. Les paramètres à considérer pour la métaheuristique BSO sont :

- L'heuristique par laquelle la solution initiale est générée.
- Le paramètre Flip
- Le nombre d'itération de la recherche locale effectuée par chaque abeille (MaxIters)
- Le nombre d'abeilles (NBees)
- Le nombre de chances accordées à une zone d'exploitation (MaxChances)
- Le nombre d'itérations de l'algorithme BSO (SearchIters).

6.4.1. L'Heuristique

La solution initiale déterminée par *BeeInit* peut soit être obtenue de manière aléatoire en commençant par un document pris au hasard de la collection, ou bien en effectuant une recherche locale afin de retenir un document qui a au moins un terme en commun avec la requêtes.

Dans le but de déterminer par quelle solution l'approche BSO va être initiée nous avons effectué des tests pour comparer les résultats obtenus par chacune des méthodes de détermination de la solution initiale. Ces tests ont été effectué en exploitation les 4 Benchmarks, et la comparaison entre les deux méthodes se fera sur la base de la précision de la méthode de recherche et du temps d'exécution.

Nous rappelons ici que la précision se calcule sur la base des documents retournés par l'approche et de ceux retournés par la recherche exacte associée. La précision est le nombre de documents pertinents retournés par l'approche basée sur l'essaim d'abeille par rapport à

l'ensemble des documents retournés par la même approche, le rappel est le nombre de documents pertinents retournés par l'approche basée sur l'essaim d'abeilles par rapport l'ensemble des documents jugés pertinents à partir des résultats de la recherche exacte.

Aux termes des tests effectués nous avons obtenus les résultats suivants :

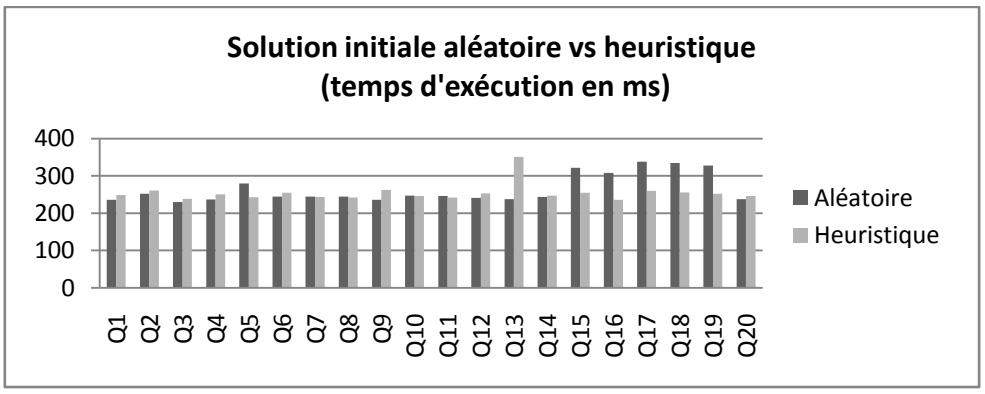


Figure 17 : Solution initiale aléatoire vs heuristique (temps d'exécution).

Du point de vue du temps d'exécution, la solution initiale obtenue par une approche heuristique donne de meilleurs résultats qu'une solution aléatoire sur la globalité des requêtes utilisées.

Mais comme en RI nous avons souvent besoin que l'ensemble des solutions retournées soit bon et non pas la meilleure solution uniquement, il a fallu utiliser les deux mesures d'évaluation des systèmes de recherche d'information à savoir le rappel et la précision.

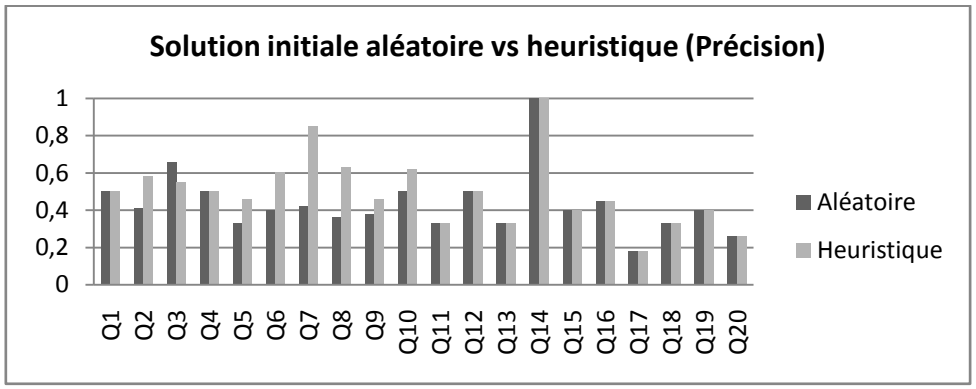


Figure 18 : Solution initiale vs heuristique (Précision).

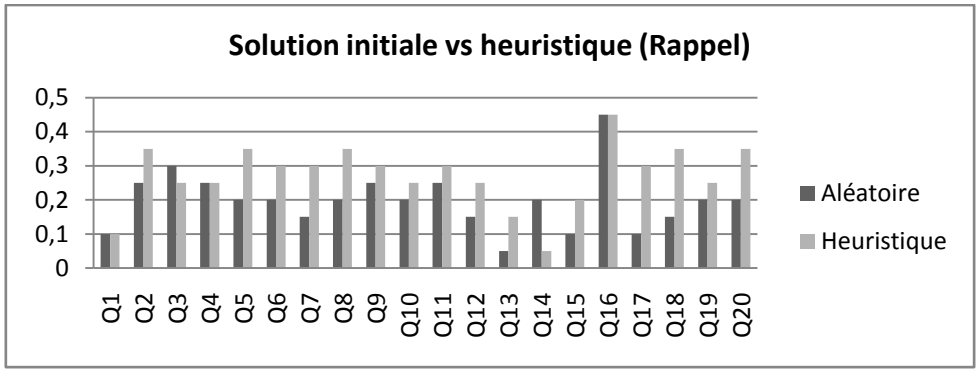


Figure 19 : solution initiale vs heuristique (Rappel).

Compte tenu des résultats obtenue par rapport à la précision et au rappel, nous pouvons conclure que pour de meilleurs résultats dans la recherche nous avons besoin de démarrer par une solution (document) qui soit de bonne qualité ce qui implique de retenir la méthode heuristique pour la détermination de la solution initiale.

De ce fait nous adopterons la méthode heuristique pour la détermination de la solution initiale dans tout ce qui suit.

6.4.2. MaxIters

Le nombre d'itération de la recherche par étapes qu'effectue chacune des abeilles de l'essaim est un facteur très important pour déterminer la qualité des résultats finaux.

Nous avons effectué de nombreux tests sur ce paramètre en fixant les autres paramètres aux valeurs suivantes :

- SearchIter = 20, Flip = 5, NBees = 20, MaxChances = 3.

Aux termes de ces tests nous avons obtenue les résultats suivant :

Cet histogramme montre clairement qu'avec le paramètre MaxIter = 20 et MaxIter = 25, nous obtenons les meilleures similarités, et que pour ces deux valeurs du paramètre nous obtenons des solutions équivalentes. C'est le temps d'exécution qui va nous permettre de départager entre ces deux valeurs.

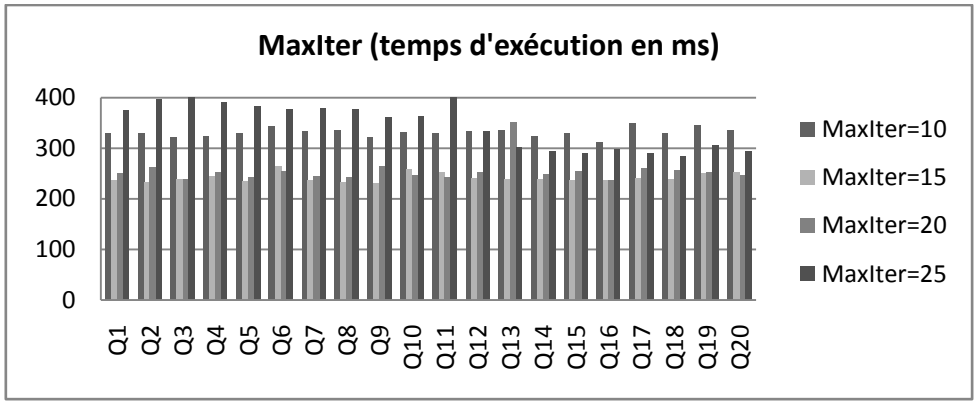


Figure 20 : Impact du paramètre MaxIter sur le temps d'exécution.

La comparaison entre les temps d'exécutions respectives aux deux valeurs pour le paramètre MaxIter permet de voir clairement que même si du point de vue de la similarité ces deux valeurs sont équivalentes, le temps d'exécution pour la valeur 20 est inférieur d'un peu plus que 100 ms que pour la seconde valeur.

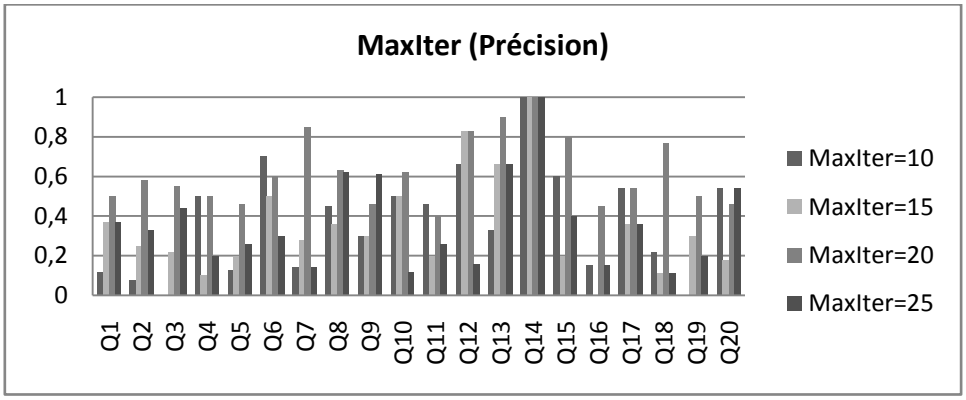


Figure 21 : Impact du paramètre MaxIter sur la précision.

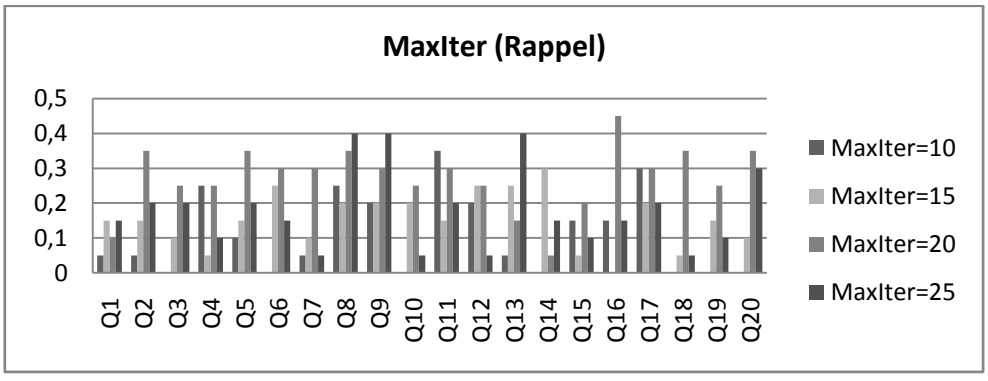


Figure 22 : Impact du paramètre MaxIter sur le rappel.

Du point de vue de la précision et du rappel MaxIter = 20 donne les meilleurs résultats.

Donc, en tenant compte de l'ensemble des résultats obtenus pour les différentes valeurs du paramètre nous pouvons conclure que la valeur 20 donne les meilleurs résultats globaux. De ce fait nous adopterons celle-ci pour les prochains tests.

6.4.3. SearchIter

L'objectif de ces tests est d'estimer la meilleure valeur du nombre d'itérations qu'effectuera l'essaim lors de sa recherche.

Aux termes des nombreux tests effectués nous avons obtenus les résultats suivants :

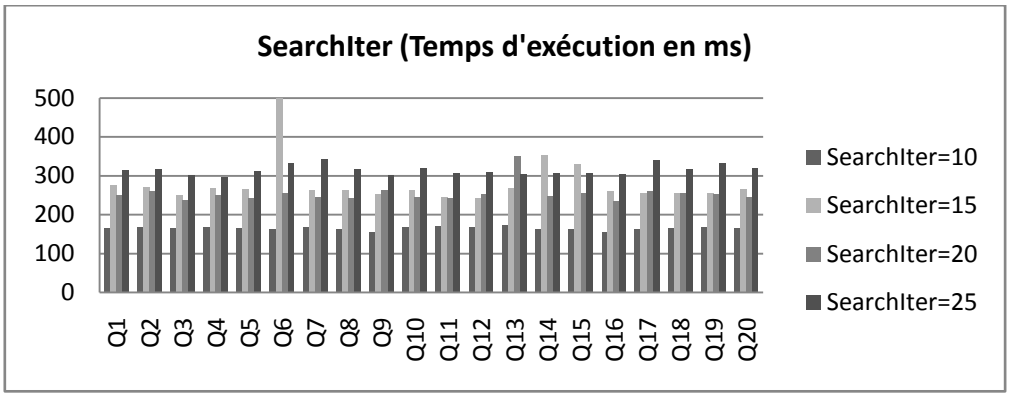


Figure 23 : Impact du paramètre SearchIter sur le temps d'exécution.

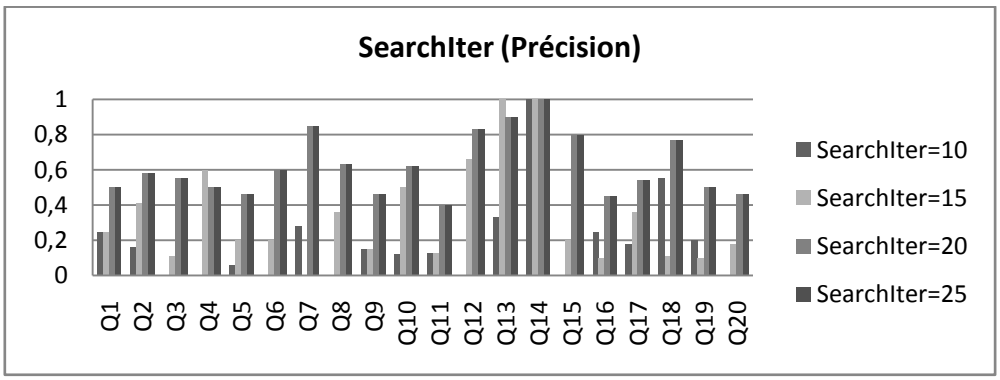


Figure 24 : Impact du paramètre SearchIter sur la précision.

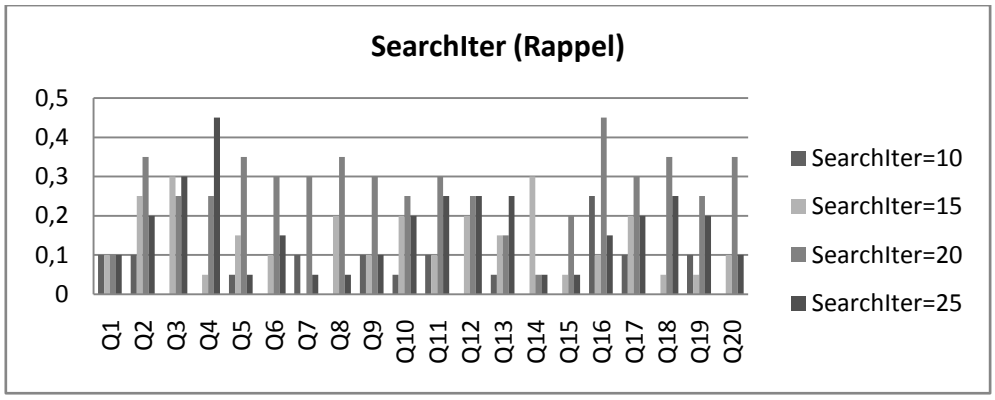


Figure 25: Impact du paramètre SearchIter sur le rappel.

En analysant ces histogrammes, nous remarquons que les meilleurs résultats ont été obtenus en fixant le nombre d'itérations à 20. En effet, pour cette valeur nous obtenons les meilleurs résultats globaux sur l'ensemble des requêtes utilisées et avec les temps plus raisonnables tout en préservant un meilleur taux de précision et de rappel. Donc, nous choisirons comme nombre d'itération de la BSO 20.

6.4.4. Flip

Le paramètre flip est le paramètre crucial de la méthode puisque de lui dépend la détermination de la zone de recherche. Nous rappelons que flip indique le nombre de termes à inverser à partir de Sref.

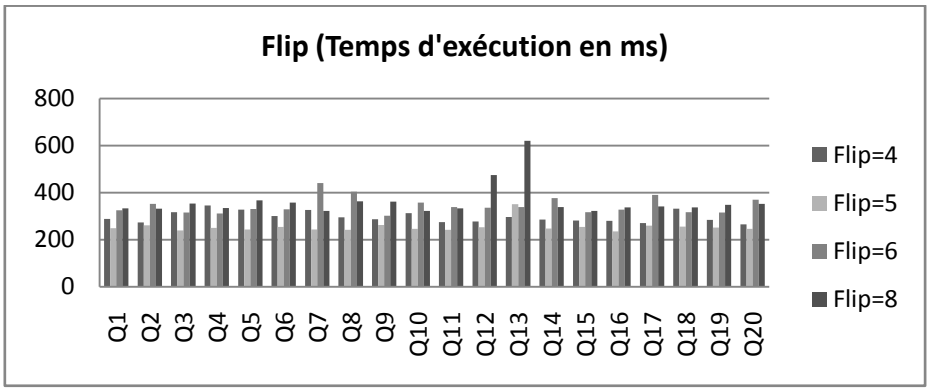


Figure 26 : Impact du paramètre flip sur le temps d'exécution.

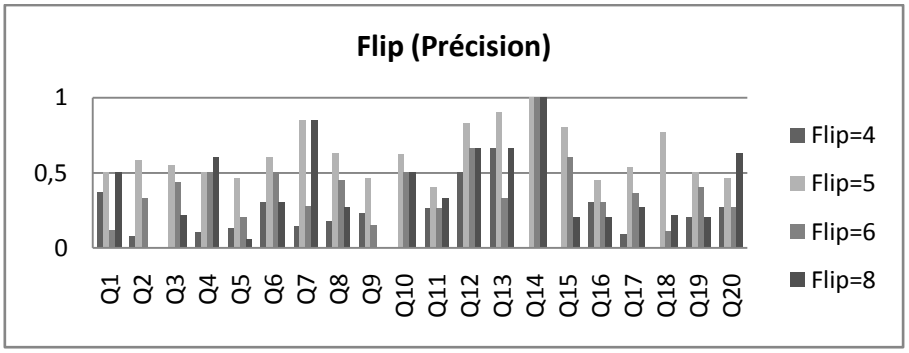


Figure 27 : Impact du paramètre flip sur la précision

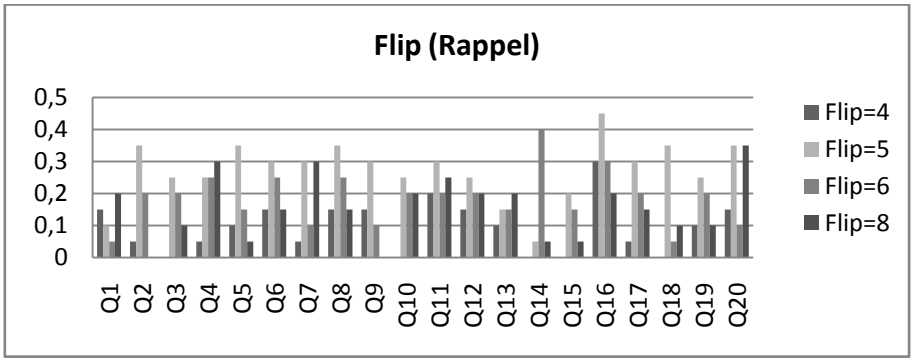


Figure 28 : Impact du paramètre flip sur le rappel.

L'analyse de l'ensemble des résultats obtenus avec les différentes valeurs du paramètre Flip nous mène à dire qu'une valeur trop grande de Flip (10) nous emprisonne dans un optimum local ce qui n'améliore pas la solution aux files des itérations, alors qu'une valeur trop petite (4) nous éloigne d'une zone prometteuse qui pourrait fournir de bons résultats. Les meilleurs résultats globaux du point de vu du temps d'exécution, de la précision et du rappel sont atteints avec le paramètre flip = 5.

6.4.5. NBees

NBees est le paramètre qui désigne le nombre d'abeilles chargées de l'exploitation d'une zone donnée. Afin de déterminer la valeur optimale pour ce paramètre nous avons effectué des tests qui nous ont fournit les résultats suivants :

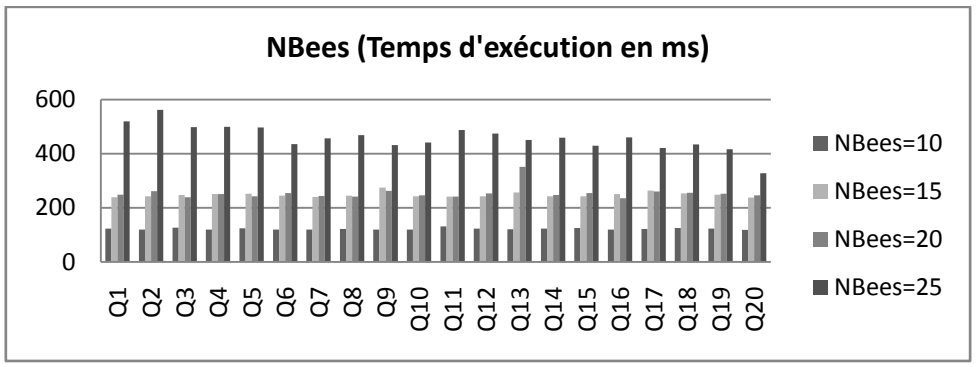


Figure 29 : Impact du paramètre NBees sur le temps d'exécution.

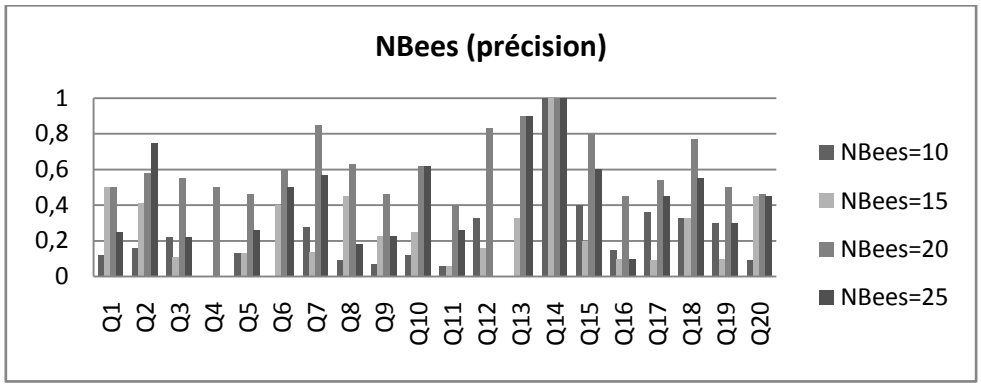


Figure 30 : Impact du paramètre NBees sur la précision.

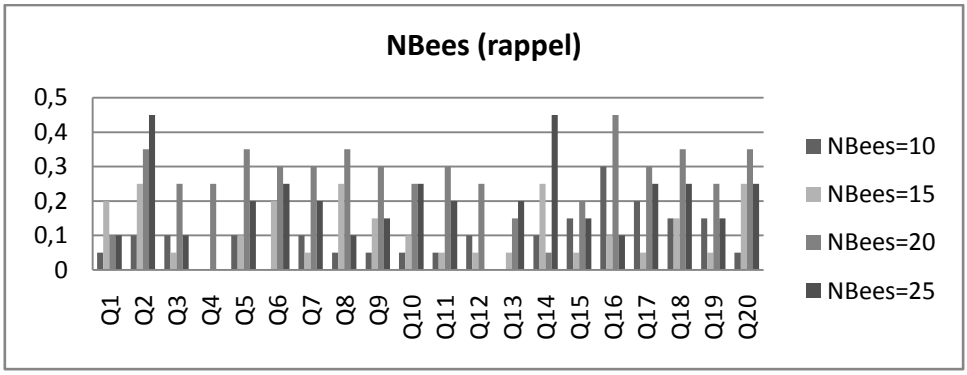


Figure 31 : Impact du paramètre NBees sur le rappel.

Il apparait clairement d'après les résultats et les histogrammes les illustrant, que la valeur pour le paramètre NBees qui permet d'obtenir les résultats optimaux est 20.

6.4.6. MaxChances

Le paramètre MaxChances désigne le nombre de chance accordé à une zone d'exploitation. Une fois ce nombre atteint l'essaim devra exploitation une autre région vue que la recherche dans celle-ci n'a pas été fructueuse. En vu de fixer ce paramètre nous avons obtenu les résultats suivants :

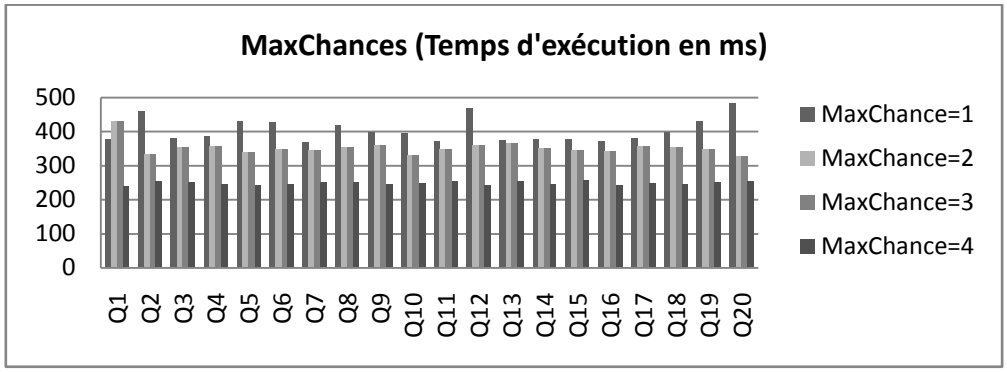


Figure 32 : Impact du paramètre MaxChances sur le temps d'exécution.

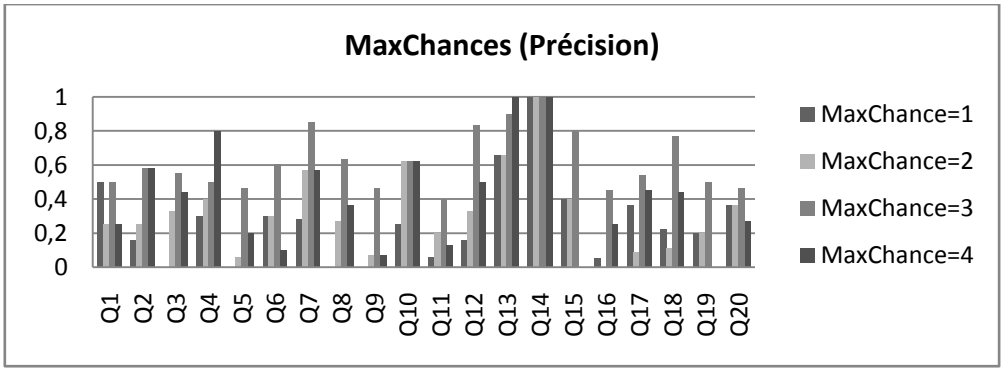


Figure 33 : Impact du paramètre MaxChances sur la précision.

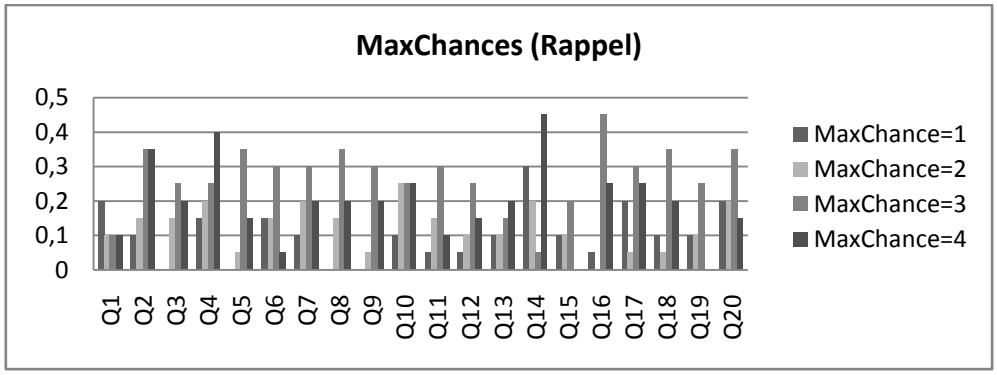


Figure 34 : Impact du paramètre MaxChances sur le rappel.

Une évaluation à 1 du paramètre MaxChances signifie qu'on accord qu'une seule chance à une région d'exploitation, ce qui signifie également que cela pourrait mener l'essaim à quitter trop vite une région pouvant éventuellement contenir de bonnes solutions. De même que pour la valeur 2, alors que le fait d'accorder beaucoup de chance à une région pourrait contraindre l'essaim à intensifier les recherches dans une zone qui apparemment ne peut plus mener à une solution de meilleure qualité que la solution courante. Donc la valeur qui permet d'atteindre les meilleures solutions en un temps raisonnable correspond à 3.

Une fois que tous les paramètres expérimentaux de la méthode fixés, l'approche basée sur les essaims d'abeilles pour la recherche dans la collection de documents à donner les résultats suivants sur l'ensemble des quatre collections prises en compte dans ce travail.

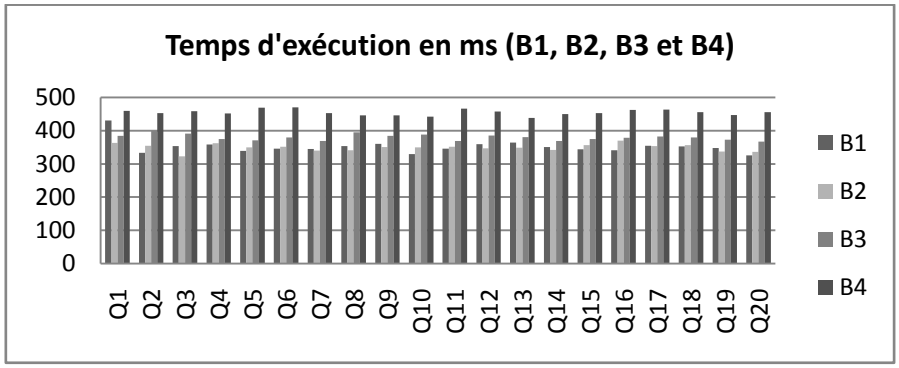


Figure 35: Temps d'exécution obtenu pour les quatre collections.

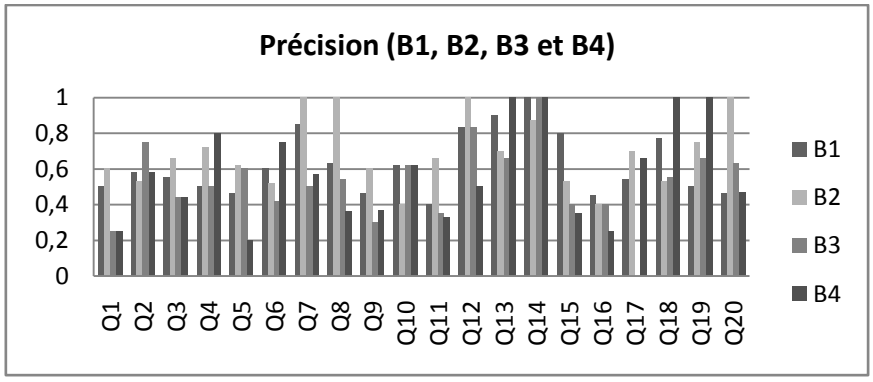


Figure 36 : Précision obtenue pour les quatre collections.

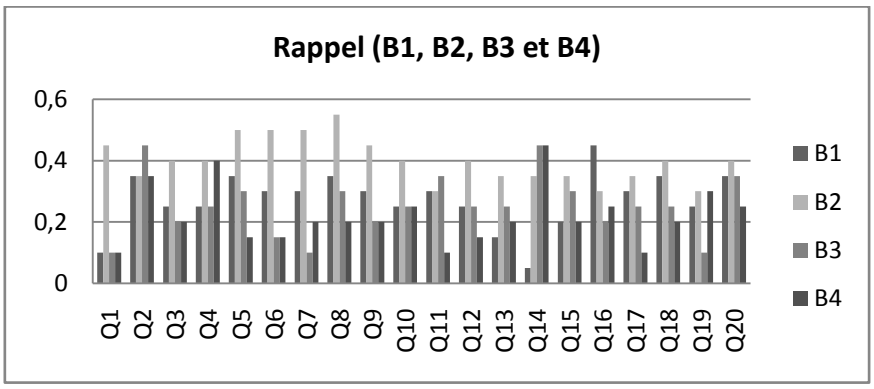


Figure 37 : Rappel obtenue pour les quatre collections.

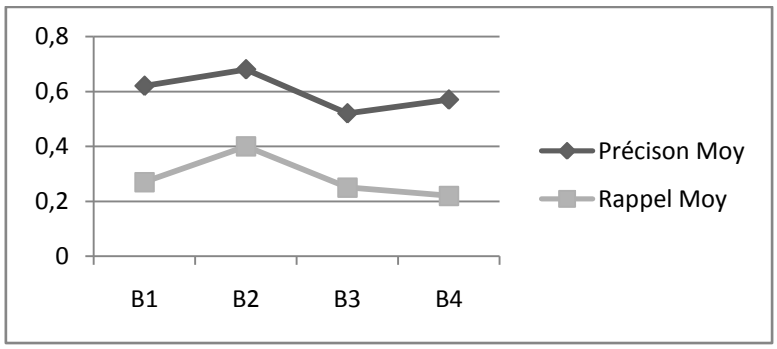


Figure 38 : Récapitulatif des résultats de la méthode pour les quatre collections.

Sur l'ensemble des quatre collections utilisées lors de nos tests nous avons obtenues un temps d'exécution qui varie entre (350 ms et 450 ms), et une précision supérieure à 57% et un rappel aussi supérieure à 20%.

Par ces résultats nous pouvons en juger que l'adaptation des essaims d'abeilles pour la recherche d'information dans la collection des documents nous a permis d'atteindre des résultats satisfaisants tant pour le temps d'exécution que pour le rappel et la précision compte tenu des tailles respectives des collections exploitées.

6.5.Recherche basée sur les essaims d'abeille à partir du fichier inverse

Dans le but de comparer les performances de la recherche basée sur les essaims d'abeilles à partir de la collection de documents avec les performances atteints par la recherche basée sur les essaims d'abeilles dans le fichier inverse, nous devons d'aborder régler les paramètres empiriques de cette dernière comme nous l'avons précédemment fait pour la BSO basée sur la collection de documents.

6.5.1. Solution initiale

Contrairement à la recherche basée sur la collection de documents, dans la recherche basée sur le fichier inverse et puisque son but est de réduire l'espace de recherche qui devra être exploité par l'essaim d'abeille, la méthode n'a pas d'autre choix que d'initier le processus de recherche par une solution de bonne qualité et en tenant compte des termes de la requête. Ce qui signifie que la solution initiale dans cette deuxième approche devra également être obtenu de manière heuristique.

6.5.2. MaxIter

Dans le but de fixer ce paramètre de nombreux tests ont été effectués et ont fournis les résultats suivants :

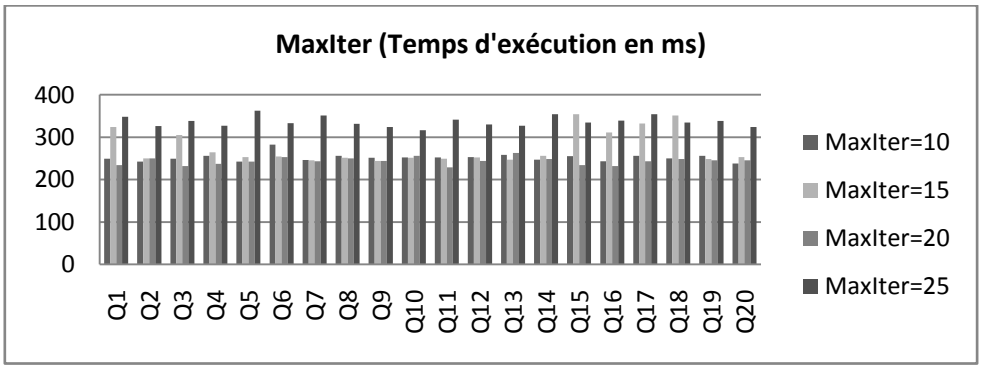


Figure 39 : Impact du paramètre MaxIter sur le temps d'exécution.

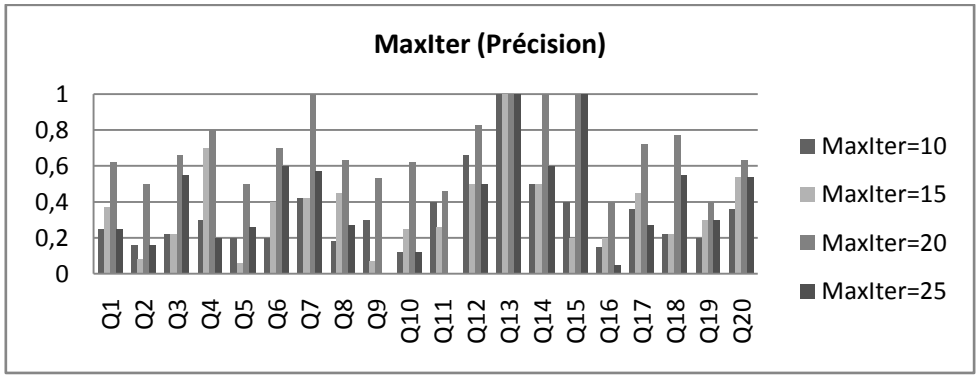


Figure 40 : Impact du paramètre MaxIter sur la précision.

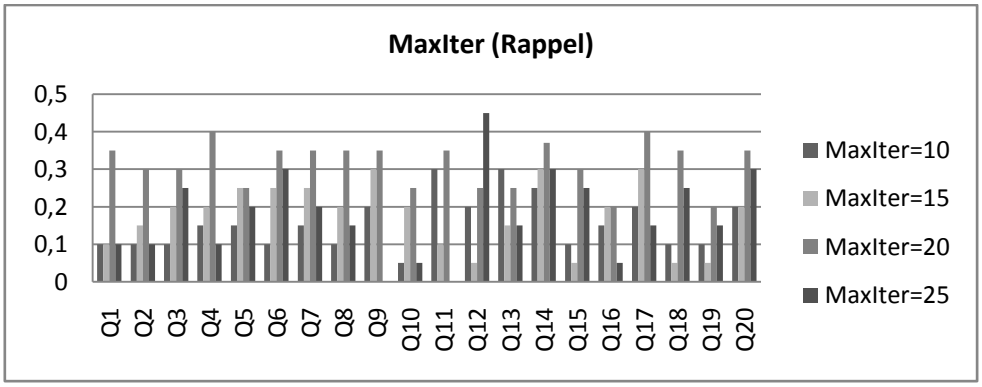


Figure 41 : Impact du paramètre MaxIter sur le rappel.

MaxIter représente le nombre d'itération de la recherche par étage que devra effectuer chaque abeille de l'essaim. Sur la base des résultats obtenus pour le réglage du paramètre MaxIter, nous pouvons fixer la valeur de MaxIter à 20 puisque en tenant compte des histogrammes ci-avant, cette valeur permet d'atteindre les résultats les plus optimaux pour les mesures utilisées.

6.5.3. SearchIter

Dans le but de fixer ce paramètre de nombreux tests ont été effectués et ont fournis les résultats suivants :

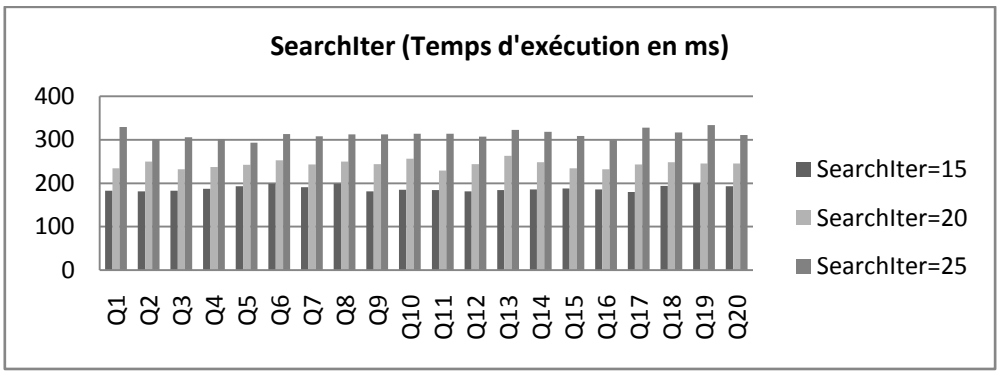


Figure 42 : Impact du paramètre SearchIter sur le temps d'exécution.

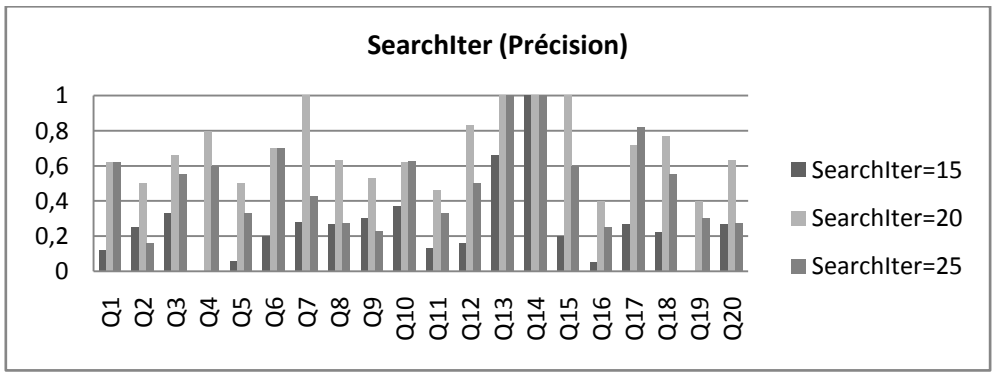


Figure 43 : Impact de SearchIter sur la précision.

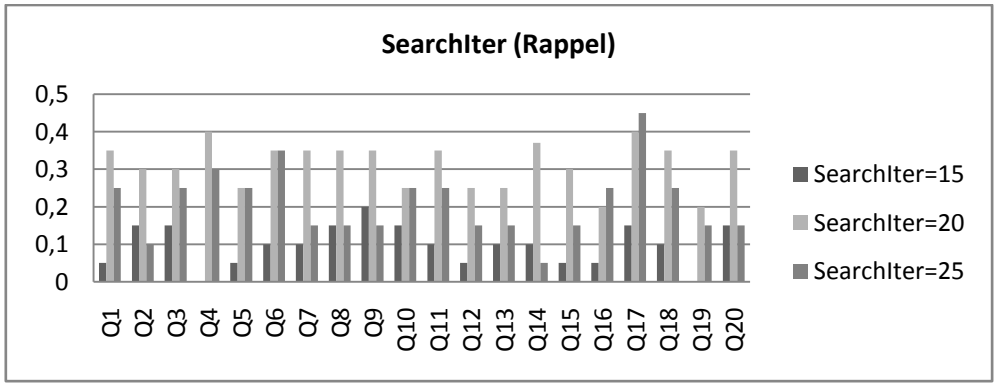


Figure 44 : Impact de SearchIter sur le rappel.

En Tenant en compte des histogrammes représentatifs des trois mesures à savoir le temps d'exécution, la précision et le rappel nous pouvons clairement remarquer que pour atteindre les meilleurs résultats par rapport à ces mesures il suffit de retenir pour le paramètre SearchIter la valeur 20.

6.5.4. Flip

Dans le but de fixer ce paramètre de nombreux tests ont été effectués et ont fournis les résultats suivants :

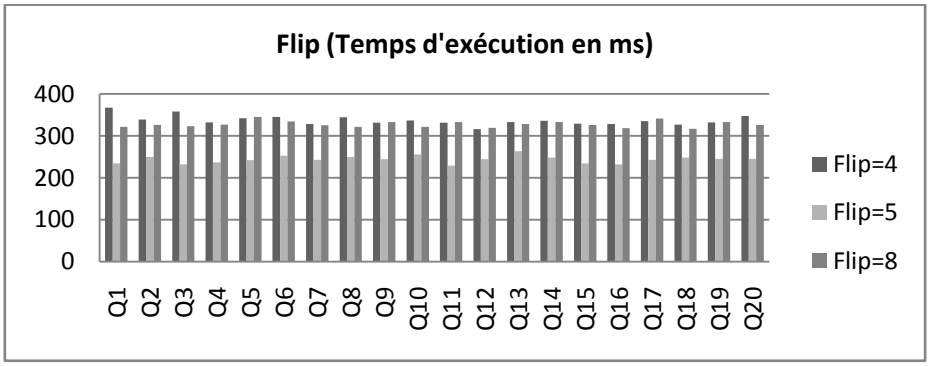


Figure 45 : Impact de Flip sur le temps d'exécution.

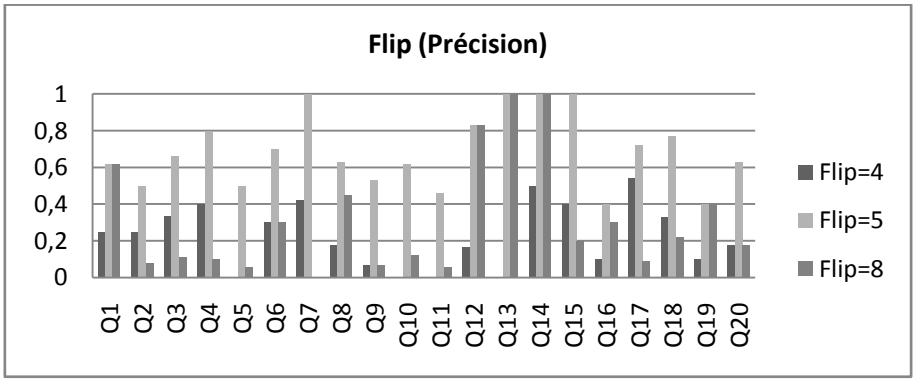


Figure 46 : Impact de Flip sur la précision.

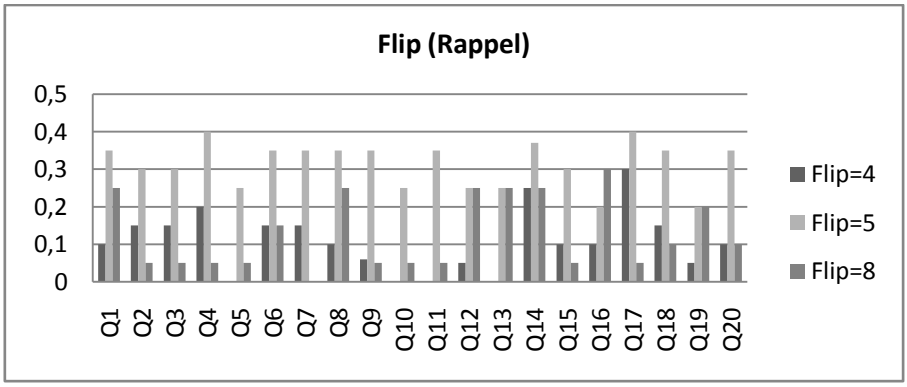


Figure 47 : Impact de Flip sur le rappel.

L'importance du paramètre Flip réside dans le fait qu'il est à la base de la génération des régions d'exploitation. Une valeur trop petite mènerait l'essaim à évoluer dans des régions rapprochée voir même chevauchée et une trop grande valeur mènerait l'essaim à s'éloigner de régions qui peuvent être prometteuses. Sur la base des résultats obtenus la valeur du paramètre à retenir est de 5.

6.5.5. NBees

Dans le but de fixer ce paramètre de nombreux tests ont été effectués et ont fournis les résultats suivants :

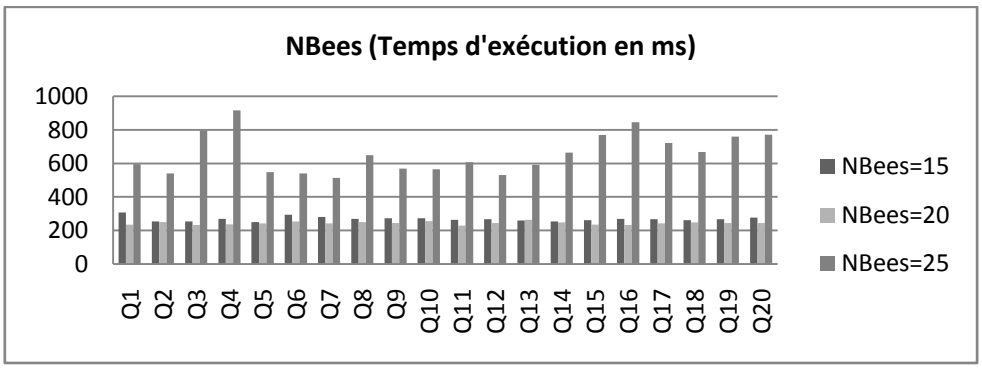


Figure 48 : Impact de NBees sur le temps d'exécution.

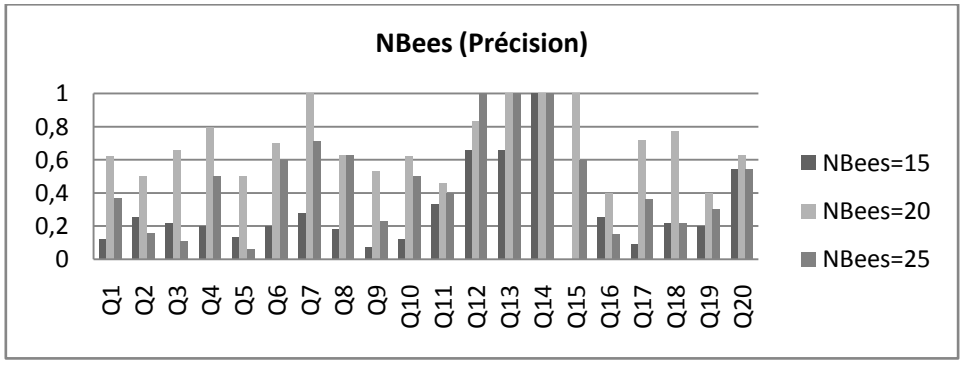


Figure 49 : Impact de NBees sur la précision.

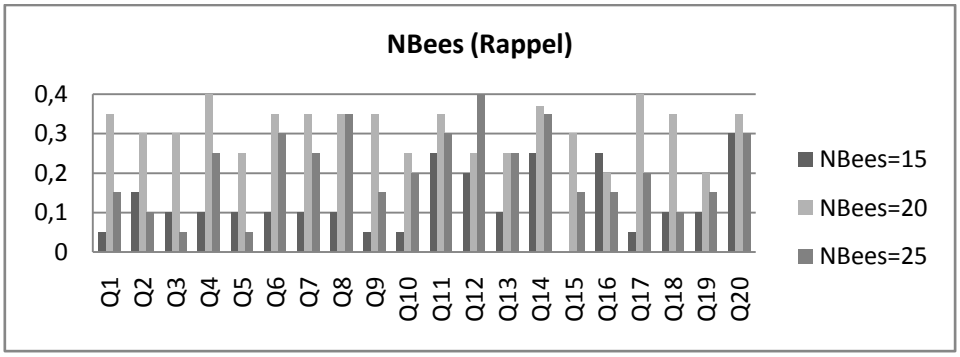


Figure 50 : Impact de NBees sur le rappel.

Sur la base des résultats illustrés ci-avant nous pouvons conclure que la valeur à retenir pour le paramètre NBees est de 20.

6.5.6. MaxChances

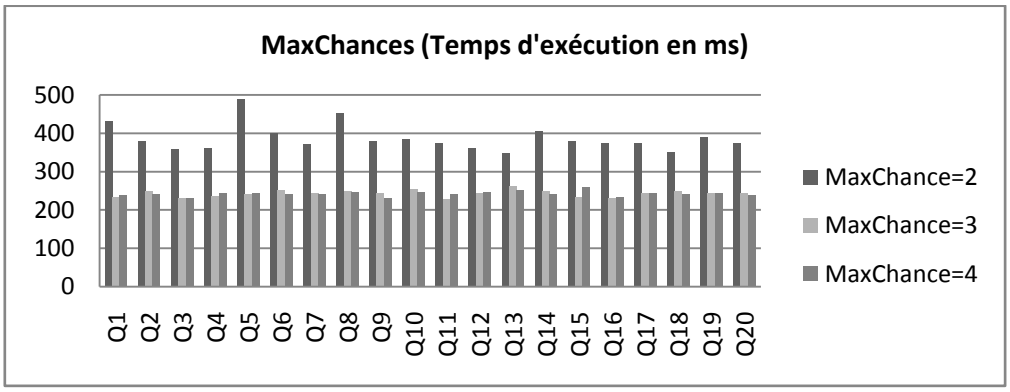


Figure 51 : Impact de MaxChances sur le temps d'exécution.

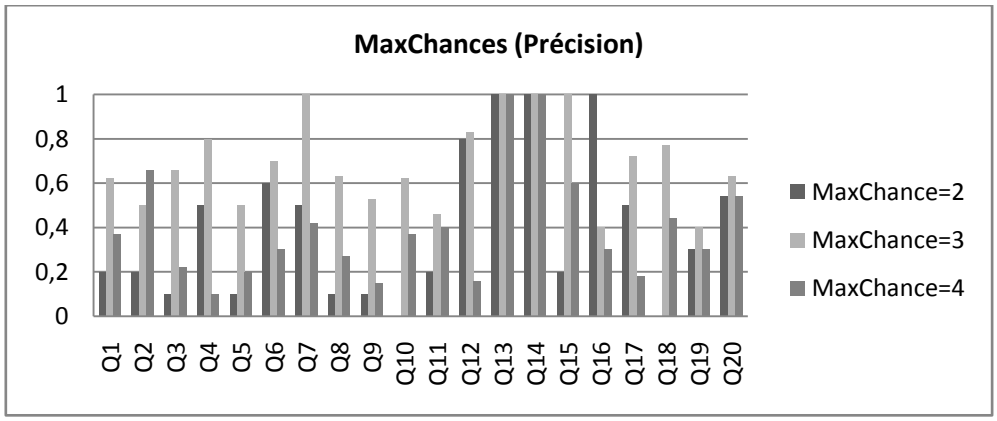


Figure 52 : Impact de MaxChances sur la précision.

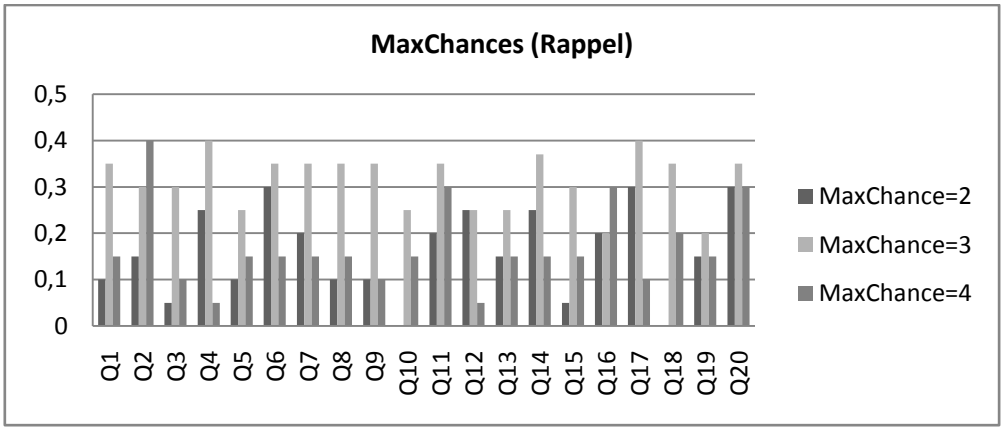


Figure 53 : Impact de MaxChances sur le rappel.

MaxChances est le paramètre qui assure la diversification de la recherche dans le cas ou celle-ci peine à améliorer la solution par rapport à Sref au bout d'un certain nombre d'itération. Le nombre d'itération pour lequel on autorise à l'essaim d'essayer d'améliorer la solution dans la région où il se trouve. La valeur à retenir pour ce paramètre est une valeur médiane qui permet d'un coté à l'essaim d'intensifier sa recherche dans une région en vu d'atteindre de meilleures solutions, et d'un autre coté permet à l'essaim de ne pas trop s'attarder sur une région, si au bout de quelques itérations aucune amélioration n'a été perçu. Aux termes des tests effectués MaxChances a été fixé à 3.

Une fois que tous les paramètres ont été fixés, la méthode basée sur les essaims d'abeilles pour une recherche dans le fichier inverse à donner les résultats suivants sur l'ensemble des quatre collections :

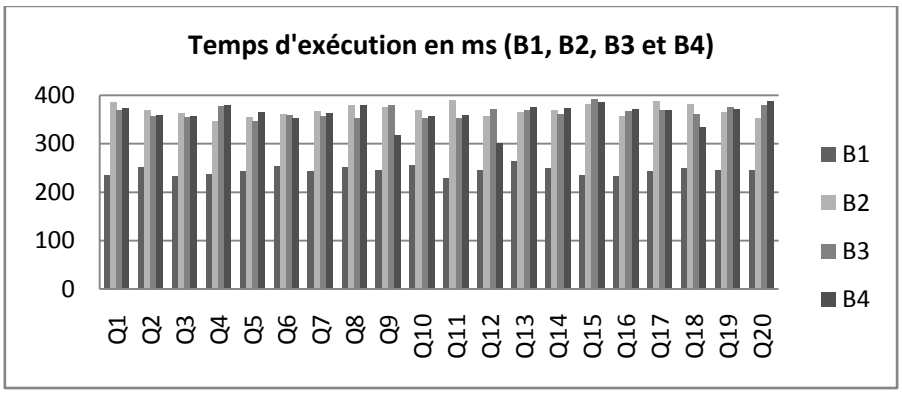


Figure 54 : Temps d'exécution nécessaire pour les quatre collections.

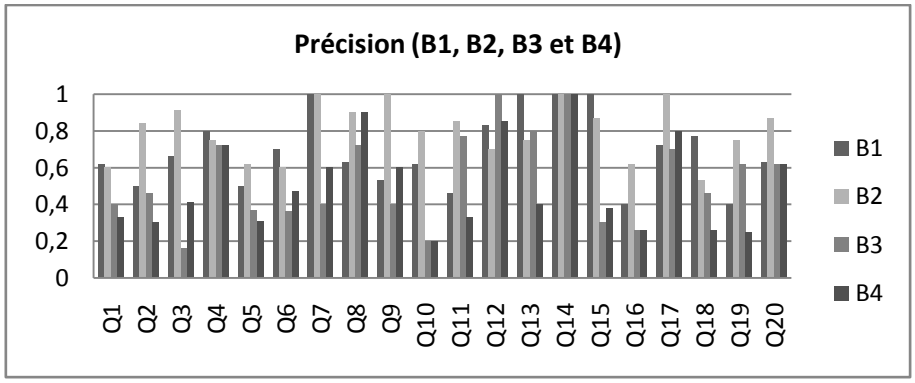


Figure 55 : Précision obtenue pour les quatre collections.

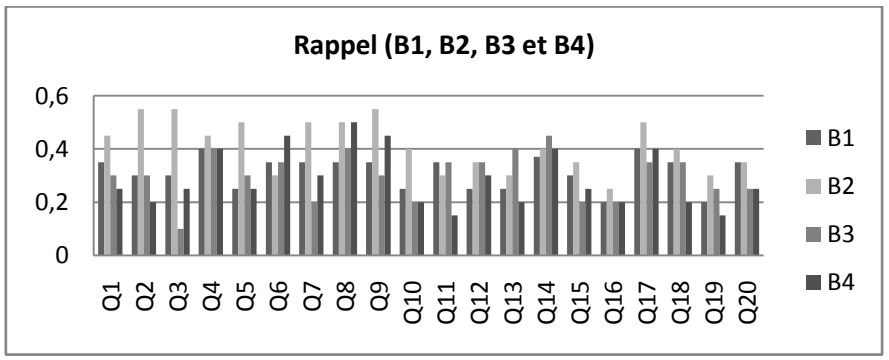


Figure 56 : Rappel obtenue pour les quatre collections.

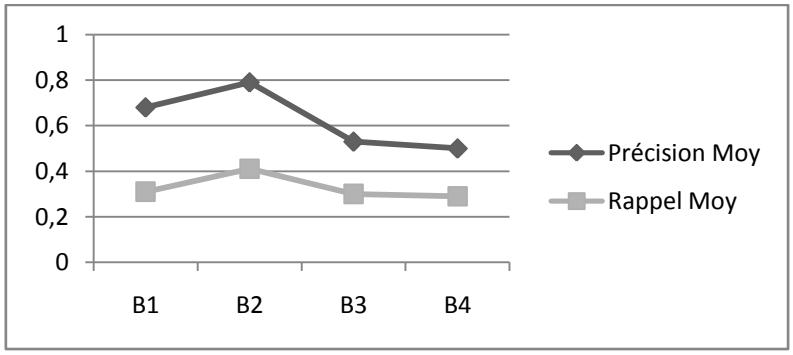


Figure 57 : Récapitulatif des résultats obtenus pour chaque collection.

Sur l'ensemble des quatre collections utilisées lors de nos tests nous avons obtenu un temps d'exécution qui varie entre (200 ms et 400 ms), et une précision qui peut atteindre les 80% et un rappel aussi supérieure à 20%.

7. Etude comparative

L'étude comparative à pour but de montrer le gain important en temps de recherche des deux approches basées sur les essaims d'abeilles par rapport aux approches classiques, elle a pour but également de comparer les deux approches basées sur les essaims d'abeilles au moyen de la courbe de précision-rappel.

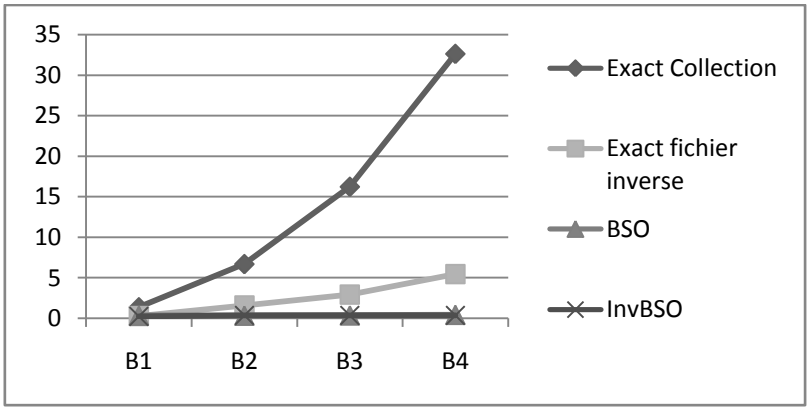


Figure 58 : Evolution du temps d'exécution en seconde par rapport aux tailles des collections pour chaque approche de recherche.

Ce graphe englobe l'évolution du temps d'exécution pour chaque méthode présentée en fonction des tailles respectives des différentes collections utilisées.

Ces courbes montrent que pour la recherche exacte le temps d'exécution évolue de manière presque linéaire par rapport aux différentes tailles, mais évolue plus rapidement quand il s'agit de la recherche à partir de la collection de documents. Les deux approches BSO évoluent quant à elles de manière presque constante. Ceci s'explique par la construction même des deux approches BSO pour lesquelles le processus de recherche n'est plus basé sur le parcours de la totalité de la collection ou bien d'un sous ensemble dans le cas de l'utilisation du fichier inverse mais plutôt sur une exploitation intelligente de l'espace de recherche sans tenir compte de la taille des collections.

Pour une meilleure comparaison entre les deux approches BSO dans la collection de document et dans le fichier inverse nous avons fait appel à la courbe de précision-rappel :

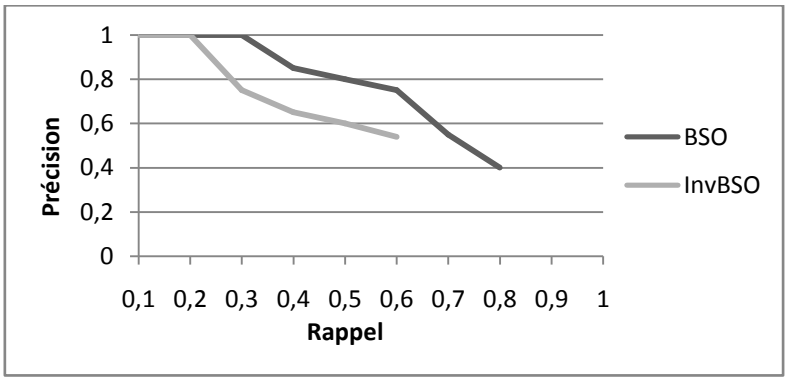


Figure 59 : Courbe de rappel-précision pour les deux approches basées sur BSO.

A partir de la courbe de Rappel-Précision nous pouvons conclure que l'approche BSO à partir de la collection de document s'adapte aux domaines où l'on préfèrerait atteindre un

meilleurs taux de précisions contrairement à l'adaptation des essaims d'abeilles pour la recherche à partir du fichier inverse qui conviendrait plutôt aux domaines où l'on cherche à atteindre un taux élevé de rappel.

8. Conclusion

D'après les résultats numériques obtenus à l'issue des séries de tests effectuées, nous avons constaté que l'adaptation de l'approche des essaims d'abeilles pour la recherche d'information a fourni des résultats plutôt satisfaisants. Une comparaison avec les résultats obtenus par les méthodes exactes a permis d'atteindre un bon niveau de précision et de rappel par des temps considérablement réduis ce qui nous mène à juger de l'efficacité de la méthode de recherche par essaims d'abeilles pour la recherche d'information.

Conclusion générale

Avec l'expansion de l'information et de sa manipulation, la recherche d'information est devenue un moyen incontournable pour accéder à l'information de nos jours. Que ce soit pour trouver des articles ou des publications pour des travaux de recherche, une simple recherche bibliographique dans le campus de l'université, ou encore la recherche d'une agence de voyage qui répond aux exigences d'un client, ou même d'identifier le profil pour lui autoriser l'accès ou non à une ressource sécurisée. Ce besoin conséquent en information rend le problème de recherche de plus en plus complexe et le contraint d'être plus rapide et plus efficace.

Le problème de recherche d'information par ces différents modèles fait partie de la classe des problèmes fondamentaux en informatique, qu'il faudrait résoudre de manière efficiente. Au fil des années ce problème a bénéficié des avancées technologiques en termes d'infrastructure de réseaux, d'Internet qui a encore engendré plus de complexité au problème.

Plusieurs travaux ont été entrepris pour proposer des solutions à des problèmes pratiques de recherche d'informations. Ces travaux se basent principalement sur l'indexation de l'espace de recherche.

En parallèle, ces dernières années, d'innombrables travaux de recherche ont été réalisés dans le but de mettre au point de nouvelles méthodes de résolution issues de sources d'inspirations diverses et variées : les processus physiques, les phénomènes biologiques et récemment bio- sociologiques à savoir l'intelligence en essaim.

Notre apport dans le domaine consistait principalement à proposer une solution à l'un des modèles de recherche d'information par une métaheuristique issue de l'intelligence en essaim afin que ce problème puisse bénéficier des avantages que ces méthodes ont apporté dans le domaine de la recherche en général.

Une première contribution été de faire évoluer les essaims d'abeilles sur un très vaste champ de documents de la collection. Cette première approche à donner des résultats satisfaisants qui ont permis de juger l'efficacité de cette méthode pour ce problème. Une seconde contribution consistait à réduire le vaste champ de recherche aux documents qui ont au moins un lien en commun avec le besoin d'information de l'utilisateur. Ceci n'est possible qu'à travers l'utilisation du fichier inverse associé à la collection de document. Cette approche

s’est également montrée efficiente pour le problème posé et a permis d’avoir des réponses précises et concises au besoin d’information.

Perspectives :

Comme perspectives pour nous travaux futurs, nous envisageons :

- Etendre le travail déjà accompli à d’autres modèles de recherche d’information tels que : le modèle probabilité et le modèle d’anthologie.
- Faire fonctionner notre travail sur de réels benchmarks publics.
- Travailler sur la réalisation d’un benchmark réel pour avoir une meilleure interprétation des résultats.
- Réduire encore plus le temps de calcul grâce à la parallèlisation du fonctionnement de l’essaim d’abeilles.
- Explorer d’autres métaheuristiques issues de l’intelligence en essaims sur la RI telles que les essaims de particules.

Références bibliographiques

Références bibliographiques

[BAY07]: A. Baykasoglu1, L. Özbakır and P. Tapkan «***Artificial Bee Colony Algorithm and Its Application to Generalized Assignment Problem*** » Itech education and publishing. Page 532. Décembre 2007.

[BER01]: V. D. Bergh, « ***An Analysis of Particle Swarm Optimizers*** ». PhD thesis, Department of Computer Science, University of Pretoria, November 2001

[BDT99]: E. Bonabeau, M. Dorigo and G. Theraulaz, **« *Swarm Intelligence: From Natural to Artificial Systems* »**, Oxford University Press, 1999

[BLA85]: Blair, D.C., Maron, M.E., ***An evaluation of retrieval effectiveness for a full-text document-retrieval system***. Commun. of the *ACM*, 28, 1985, pp. 289-299.

[BON01]: E. Bonabeau, C. Meyer « ***Swarm Intelligence: A whole new way to think about business*** » Harvard Business Review. Mai 2001.

[BOT94]: E. Bonabeau, G. Théraulaz « ***Intelligence collective*** » Editions Hermès. 1994.

[BRA02]: L. Braun « ***Information Retrieval from Dutch Historical Corpora***» Maastricht 25 Novembre 2002.

[CHE07]: H. Chen «***The Expected Metric Principle for probabilistic Information Retrieval***» Magister. Massachusetts Institute Of Technology. February 2007

[CLE67]: Cleverdon, C.W, ***The Cranfield tests on index language devices***. *Aslib Proceedings* 19(6), 173-193, 1967.

[COL92]: A. Colorni, M. Dorigo et V. Maniezzo « ***Distributed Optimization by Ant Colonies*** » Proc. Of ECAL'91 – First European Conference on Artificial Life, édité par F.Varela et al. pages 134-142, Elsevier Publishing, Paris, France 1992.

[DOR99]: M. Dorigo, G. Di Caro et L. M. Gambardella « ***Ant Algorithms for Discrete Optimization*** » Artificial life, 5(2), 1999.

[DRE04] : J. Dréo «***Adaptation de la méthode des colonies de fourmis pour l'optimisation en variables continues. Application en génie biomédical*** » thèse de doctorat Décembre 2004.

[DRE03] : J. Dréo, A. Pétrowski, P. Siarry et E. Taillard « ***Métaheuristiques pour l'optimisation difficile*** ». Eyrolles, 2003.

[DRI00]: H. Drias, N. Alliche, S. Dahmani « ***Contribution à la résolution du problème Max-W-SAT par la méta-heuristique GLS*** » projet de fin d'étude. USTHB 2000.

[DRI04]: H. Drias, S. Sadeg, S. Yahi, « ***Conception d'une méta-heuristique optimisation par essaim d'abeilles et son adaptation au problème MAX-W-SAT*** » projet de fin d'étude USTHB 2004.

[DRI05]: H. Drias, S. Sadeg, S. Yahi, «***Cooperative Bees Swarm for solving the Maximum Weighted Satisfiability Problem***» IWAAN International Work Conference on Artificial and Natural Neural Networks, Barcelona, Spain, 318-325,2005.

[DRI06]: H. Drias, Z. Bouzid, M. W. Benabderrahmane « ***l'approche essaim d'abeilles pour la résolution du problème d'affectation tridimensionnelle (3DA)*** » projet de fin d'études, INI 2006.

[DRK01]: H. Drias et M. Khabzaoui « ***Scatter search With random Walk Strategy for Sat and Max-W-sat Problems*** » Springer Verlang Berlin Heidelberg 2001.

[DUT02] : A. Dutot et D. Olivier, « ***Optimisation par essaim de particules Application au problème des n-Reines*** ». Laboratoire informatique du Havre. Université du Havre 2002.

[FAR01]: O. Faroe, D. Pisinger and M. Zachariasen«***Local search for final Placement in VLSI Design*** » Dept. of Computer science University of Copenhagen. Danemark. In International Conference on Computer-Aided Design, pages 565--572. IEEE, 2001.

[GEN02] : M. Gendreau, « ***an introduction to the tabou search*** », Centre de recherche sur les transports et département d´informatique et de recherche opérationnelle. Université de Montréal. Juillet 2002

[GLO86]: F. Glover « ***Future Paths for Integer Programming and Links to Artificial Intelligence*** », Comput. & Ops. Res.Vol. 13, No.5, pp. 533-549, 1986.

[HAN97]: M. P. Hansen « ***Tabou search for multiple objective combinatorial optimization: MOTS.*** » The 13th International Conference On Multiple Criteria Decision Making (MCDM), University of Cap Town 1997.

[HAR92]: Harman, D. K. (ed.) « ***The First Text REtrieval Conference (TREC-1)*** », NIST Special Publication 500-207. 1992.

[JYN06a]: J-Y. Nie, « ***Le domaine de recherche d'information – Un survol d'une longue histoire »,*** Département d'informatique et recherche opérationnelle Université de Montréal. 2006.

[JYN06b]: J-Y Nie, « ***Introduction à la RI»,*** Département d'informatique et recherche opérationnelle Université de Montréal 2006.

[JYN06c]: J-Y Nie, « ***Les modèles de recherche d'information»,*** Département d'informatique et recherche opérationnelle Université de Montréal 2006.

[LAN68]: Lancaster, F.W., ***Evaluation of the MEDLARS Demand Search Service,*** National Library of Medicine, Bethesda, Maryland, 1968.

[MAB01]: M. H Mabed « ***métaheuristiques pour l'optimisation multicritère : Etude des problèmes d'ordonnancement*** » Thèse de Magister, USTHB 2001

[MAG00]: G. Magyar, M. Johnson and O. Nevarlaimen « ***An Adaptative Hybrid Genetic Algorithm for the Three-Matching Problem*** », IEEE Transactions On Evolutionnary Computation, vol 4,n°2, Juillet 2000.

[MAN08]: C. D. Manning, Prabhakar Raghavan Hinrich Schutze « ***An introduction to information Retrieval*** » Combridge University Press. Combridge England 2008

[MOO48]: Mooers, C.N., « ***Application of Random Codes to the Gathering of Statistical Information*** », MIT Master's Thesis, 1948.

[NIE08] : J-Y. Nie***,*** « ***Le domaine de recherche d'information – Un survol d'une longue histoire*** »***,*** Département d'informatique et recherche opérationnelle Université de Montréal. 2008.

[PAV05]: A. Pavé « ***Biodiversité, science et gouvernance: le point de vue d'un biométricien*** » Natures Sciences Sociétés 13, pages 440-446. 2005.

[RAJ98]: M. Rajman, L. Lebart « ***Similarités pour données textuelles*** », 4th International Conference on Statistical Analysis. 1998.

[REI07]: H. A. Reijers, M. H. Jansen-Vullers, M. z. Muehlen, and W. Appl «***Workflow Management Systems + Swarm Intelligence = Dynamic Task Assignment for Emergency Management Applications*** » Springer-Verlag Berlin Heidelberg 2007.

[SAL71]: G. Salton, « ***The SMART Retrieval System*** ». Prentice Hall, Englewood Cliffs, NJ, 1971.

[SAM05]: O. Sammoud, C. Solnon, and K. Ghédira « ***Ant Algorithm for the graph matching problem*** ». Lecture notes in computer Science. Volume 3448. Page 213-223 2005.

[SEL07]: S. B. Selvadurai «***Implementing a metasearch Framework with Content-directed Result Merging*** » Master Thesis. North California State University. 2007

[SON06]: M. Song, Il-Y. Song, X. Hu, R. B. Allen « ***Integration of association Rules and Onlology for Semantic-based Query Expansion***» Drexel E-Repository and Archive. 2006.

[TRA03] : I. C. Trelea, « ***L'essaim de particules vu comme un système dynamique: convergence et choix des paramètres*** », séminaire : « L'optimisation par essaim de particules (OEP) », Paris, Carré des Sciences, octobre 2003

[VAL05]: D. Vallet, M. Fernandez, P. Castells « ***An Ontology Based Information Retrieval Model*** » Publication universitaire, Université de Madrid 2005.

[VDH07]: M. Ven Der Haegen « ***Analyse des méthodes bio-inspirées pour l'auto-organisation dans les SANETs*** » Juin 2007.

[VOU95]:C. Voudouris and E. Tsang « ***Guided Local Search*** » Technical Report CSM-247 1995.

[VOU97]:C. Voudouris « ***Guided Local Search for Combinatorial Optimisation Problems*** » phD, Thesis, Dept. Computer Science, University of Assex, Colchester, UK, July 1997.

[VOU99]:C. Voudouris, E. Tsang « ***Guided Local Search and its application to the travelling salesman problem*** » European Journal of Operational Research 113 page 469-499, 1999.

[WAL79]: Waller, W. G. and Kraft, D. H. « ***A mathematical model for a weighted Boolean retrieval system*** »**.** Information Processing & Management, 15: 235-245. 1979

[ZEI03]: D. Zeinalipour-Yazti, V. Kalogeraki, D. Gunopulos «***Information Retrieval in Peer-to-Peer Networks*** »; Publication universitaire, Department of Computer Science and Engineering University of California – Riverside 2003.

[ZIV05]: B-Y. Ziv, G. Ido «***Information retrieval: Algorithms for large Data Sets***». Spring 2005.

[ZHA05]:W. Zang and M. Looks «**A *novel Local Search algorithm for the Travelling Salesman Problem that exploits Backbones*** » Dept. of Computer Science and Engeneering Washington University in Saint Louis. *Proc. 19th Intern. Joint Conf. on Artificial Intelligence (IJCAI-05)*. 2005.

Résumé

Le problème de recherche d'information par ces différents modèles fait partie de la classe des problèmes fondamentaux en informatique, qu'il faudrait résoudre de manière efficiente. Au fil des années ce problème a bénéficié des avancées technologiques en termes d'infrastructure de réseaux, d'Internet qui a encore engendré plus de complexité au problème.

Plusieurs travaux ont été entrepris pour proposer des solutions à des problèmes pratiques de recherche d'informations. Ces travaux se basent principalement sur l'indexation de l'espace de recherche.

En parallèle, ces dernières années, d'innombrables travaux de recherche ont été réalisés dans le but de mettre au point de nouvelles méthodes de résolution issues de sources d'inspiration diverses et variées : les processus physiques, les phénomènes biologiques et récemment bio- sociologiques à savoir l'intelligence en essaim.

Notre apport dans le domaine consistait principalement à proposer une solution à l'un des modèles de recherche d'information par une métaheuristique issue de l'intelligence en essaim afin que ce problème puisse bénéficier des avantages que ces méthodes ont apporté dans le domaine de la recherche en général.

Une première contribution consiste à faire évoluer les essaims d'abeilles sur un très vaste champ de documents de la collection. Cette première approche à donner des résultats satisfaisants qui ont permis de juger l'efficacité de cette méthode pour ce problème. Une seconde contribution à pour but de réduire le vaste champ de recherche aux documents qui ont au moins un lien en commun avec le besoin d'information de l'utilisateur. Ceci n'est possible qu'à travers l'utilisation du fichier inverse associé à la collection de document. Cette approche également s'est montrée efficiente pour le problème posé et a permis d'avoir des réponses précises et concises au besoin d'information.