Table de Matière

Liste des figures

Liste des tableaux

Introduction Générale

Contexte

La majorité des organisations actuellement se tournent vers des architectures à base de services Web pour le développement et l'intégration d'applications ou de systèmes d'information. L'importance des standards dans ce contexte a sans doute accentué le phénomène. Mais il arrive très fréquemment que plusieurs services répondent à un même ensemble de besoins fonctionnels, d'où la nécessité de la naissance du domaine de la sélection de services Web a base de qualité de service.

En recherche opérationnelle, et plus précisément dans le domaine de l'optimisation, de nombreuses méthodes sont inspirées par de telles études. Parmi ces domaines d'inspiration, la biologie est particulièrement prisée par les chercheurs en optimisation. En effet, la modélisation des systèmes intelligents rencontrés dans la nature peut servir à créer une intelligence artificielle, à même de résoudre des problèmes d'optimisation.

Parmi toutes les branches de la biologie, l'éthologie, étude du comportement des animaux, a récemment donné lieu à plusieurs avancées significatives dans le domaine de la recherche opérationnelle. La modélisation des comportements animaliers, principalement ceux des animaux évoluant en essaim, est exploitée majoritairement en robotique, en classification ou encore en optimisation. Un système en essaim est un système qui doit son intelligence à la coopération d'éléments qui, individuellement, sont de faible intelligence. La performance du système entier est supérieure à la somme des performances de ses parties. Dans le cadre de l'optimisation, cette approche a donné lieu à la naissance de nouvelles métaheuristiques. Les métaheuristiques forment une famille d'algorithmes stochastiques destinée à la résolution de problèmes d'optimisation. Ces

méthodes ont été conçues afin de résoudre un panel très large de problèmes, sans pour autant nécessiter de changements profonds dans les structures des algorithmes.

Parmi les méthodes inspirées de l'éthologie, la méthode d'Optimisation par Essaim Particulaire (OEP) a été proposée en 1995 par Kennedy et Eberhart pour la résolution de problèmes d'optimisation.

Problématique

Vu l'émergence des services web et l'explosion des services logiciels composés, il devient très important de développer des systèmes performants de sélection des services web à base de qualité de service.

Mais bien que l'ensemble de plans d'exécution possibles d'une requête client soit exponentiel, et la pauvreté des programmes linéaire à résoudre ces types de problèmes, l'utilisation des algorithmes d'optimisation devient très nécessaire.

L'objectif de ce travail est de sélectionner de façon optimale des services web à base de leurs qualités de services en utilisant un algorithme d'optimisation.

Contribution

Pour répondre à cet objectif nous préconisons l'utilisation des systèmes multiagents afin de bénéficier de leurs avantages pour permettre de sélectionner et invoquer des services web adaptés aux préférences des clients.

L'algorithme d'optimisation employé est celui de l'essaim particulaire. Dans notre mémoire nous allons utiliser l'algorithme d'OEP mono-objectif sans aucun paramètre de contrôle. Cet algorithme utilise une fonction objective synthétisée à partir d'un ensemble de sous fonctions objectives à l'aide des opérateurs d'agrégation (multiplication, somme, maximum, …). Cette fonction agrégée regroupe cinq (5) critères de qualité de service (coût, réputation, sûreté, disponibilité, latence).

Plan du mémoire

Ce manuscrit se divise en trois chapitres. Le premier chapitre est consacré au domaine des services web. Le deuxième chapitre est consacré à l'étude bibliographique de quelques méthodes d'optimisation. Enfin, le dernier chapitre correspond à l'application.

- **Chapitre1 :** nous allons introduire l'origine, quelques définitions ainsi que les caractéristiques des services web. Ensuite nous allons aborder à une présentation de l'architecture ainsi que les protocoles qui se coopèrent pour réaliser cette technologie. .

- **Chapitre2 :** ce chapitre est consacré pour présenter un état de l'art de quelques métaheuristiques d'optimisation. Une attention toute particulière est accordée bien sur à la méthode d'optimisation par Essaim Particulaire, principale méthode utilisée au cours de ce mémoire. Nous terminons ce chapitre par une présentation de l'intérêt des méthodes adaptatives ainsi que l'algorithme de TRIBES.

- **Chapitre3 :** Dans ce dernier chapitre, nous terminons notre travail par une description de la sélection des services web, la démarche suivi pour calculer la fonction objective ainsi que une explication détaillée de l'algorithme d'optimisation utilisé et nous finissons par une présentation de notre système, les résultats obtenus, les perspectives attendues, et finalement, une conclusion générale sur tout le travail.

Web Services

1. Introduction

Avec l'interconnexion des ordinateurs en réseau et en particulier à travers internet, il devient important de faire fonctionner des applications sur des machines distantes, l'intérêt est de répondre aux problématiques suivantes :

- Les données peuvent être présentes uniquement sur le serveur distant ;
- Le serveur distant peut disposer d'une puissance de calcul ou de capacités de stockage dont l'utilisateur local ne dispose pas ;
- L'application distante peut être utilisée simultanément par un grand nombre d'utilisateurs et sa mise à jour n'intervient qu'à un seul endroit ;
- La machine distante dispose de logiciels plus adaptés au traitement des données.

Faire interagir des programmes différents en réseau a toujours été une affaire complexe, notamment si on souhaite standardiser certains aspects de cette interaction, ces problématiques ont donné la naissance aux services Web.

2. L'origine des Services Web

Pour observer les débuts des Services Web il faut remonter à la fin des années 90s, une période où de nombreuses technologies fait leur apparition, tel que XML[1]. C'est aussi à cette période que l'on assiste à une véritable explosion du marché du web où les acteurs économiques annoncent que la majorité des consommateurs américains sera bientôt connectée à Internet.

A cette époque se naissent aussi des nouveaux consortiums dont l'objectif est de standardiser des outils de coopération commerciale sur Internet. C'est l'apparition des termes B2B[2] et B2C[3].

Plusieurs acteurs majeurs de l'Informatique (Microsoft, IBM, HP, Oracle, …) ont joint leurs efforts pour définir une certaine vision des Services Web. Cette approche aujourd'hui tient lieu de standard.

3. Services Web

3.1. Définitions

Le consortium W3C[4] définit un service Web comme étant un composant logiciel identifié par une URI[5] dont les interfaces publiques sont définies et appelées en XML. Sa définition peut être découverte par d'autres systèmes logiciels. Les Services

[1] **XML** (*eXtensible Markup Language)*

[2] **B2B** (*Business To Business)*: plateformes de gestion des échanges commerciaux entre entreprises par l'intermédiaire de l'Internet.

[3] **B2C** (*Business To Costumer)*: plateformes de gestion des échanges commerciaux d'une entreprise vers ses clients par l'intermédiaire de l'Internet.

[4]**W3C** (*World Wide Web Consortium*): créé en 1994, est l'un des principaux organismes internationaux de normalisation des technologies et protocoles basées sur Internet.

[5] **URI** (*Uniform Resource Identifier*)**:** courte chaîne de caractères qui identifie les ressources sur le Web: documents, images, fichiers, services, etc.

Web peuvent interagir entre eux d'une manière prescrite par leurs définitions, en utilisant des messages XML portés par des protocoles Internet [www1].

Nous trouvons une autre définition proposée par IBM: « Web service is a technology that allows applications to communicate with each other in a platform and programming language-independent manner. A Web service is a software interface that describes a collection of operations that can be accessed over the network through standardized XML messaging. »

D'après cette définition, un Service Web :

· Permet à des applications de communiquer entre elles indépendamment de la plateforme et du langage de programmation.

· Est une interface logicielle qui décrit un ensemble de méthodes accessibles à travers le réseau en utilisant des messages XML standardisés.

En résumé, un service Web est un composant [Kadima, 03]:

- implémenté dans n'importe quel langage ;
- déployé sur n'importe quelle plateforme ;
- enveloppé dans une couche de standards dérivés du XML ;
- assure l'interaction entre les applications, les ordinateurs et les processus métiers via les protocoles Internet et XML ;
- et doit pouvoir être découvert et invoqué dynamiquement par d'autres services.

3.2. Caractéristiques des services web

La technologie des Services Web possède plusieurs avantages, on peut citer :

Modularité : le modèle Service Web est modulaire, on peut facilement réutiliser et combiner des Services Web déjà existant pour former un nouveau Service Web.

Interopérabilité : Un Service Web est indépendant de la plateforme et du langage de programmation avec lesquels il est développé.

En plus de ces avantages, la technologie des services web est soutenu par des grands acteurs de l'industrie informatique tels que : IBM, HP, Microsoft et le consortium W3C.

3.3.Architecture et spécification des services web

La technologie des services Web est un moyen rapide de distribution de l'information entre clients, fournisseurs, partenaires commerciaux et leurs différentes plates formes. Les services Web sont basés sur l'architecture orientée services [Rampacek, 06].

Une architecture orientée services SOA [6] est une architecture logicielle s'appuyant sur un ensemble de services simples. L'objectif d'une telle architecture est de décomposer une fonctionnalité en un ensemble de fonctions basiques, appelées services, fournies par des composants, et de décrire finement le schéma d'interaction entre ces services. Elle repose sur un modèle à objet répartis. Ce modèle repose sur trois technologies [Amerdeilh, 07] :

- SOAP (*Simple Object Access Protocol*) est un protocole d'échange inter applications indépendant de toute plateforme, basé sur le langage XML. Un appel de service SOAP est un flux ASCII encadré dans des balises XML et transporté par un protocole Internet.

[6] **SOA** (*Service Oriented Architecture*)

- WSDL (*Web Services Description Language*) donne la description au format XML des services Web en précisant les méthodes pouvant être invoquées, leurs signatures et leur point d'accès (URL, port, etc..).

- UDDI (*Universal Description, Discovery and Integration*) normalise une solution d'annuaire distribué de services Web, permettant à la fois la publication et l'exploration (recherche) de services Web. UDDI se comporte lui-même comme un service Web dont les méthodes sont appelées via le protocole SOAP.

La figure (1.1) montre la contribution des composants de l'architecture d'un service web:

La publication du service : Tout d'abord le service doit s'inscrire auprès d'un référentiel UDDI en indiquant ça description avec WSDL (1).

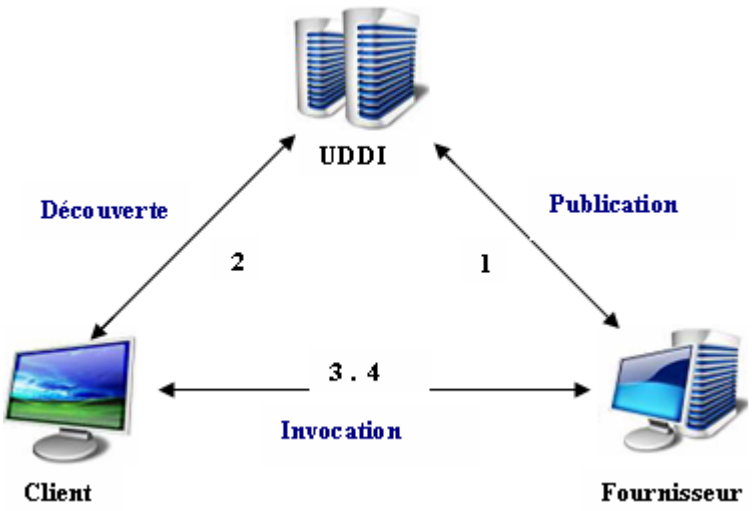


Figure 1.1 : L'architecture orientée services

La découverte du service : Le client cherche un service particulier, il consulte un référentiel UDDI qui va lui fournir les descriptions et les URL[7] des services demandés afin de lui permettre de les invoquer (2).

L'invocation du service : Le client consulte la description WSDL du service ce qui lui permet de connaître quel message SOAP est à envoyer et quel massage SOAP sera reçu après traitement, une fois que le client récupère l'URL et la description du service, il les utilise pour l'invoquer auprès du fournisseur de services (3). Le fournisseur traite alors le message du client, si aucune erreur ne se produit, une réponse est émise à l'application cliente (4).

3.3.1. Protocole de transport SOAP

SOAP (*Simple Object Access Protocol*) est un protocole de communication définit à l'origine par Microsoft, puis standardisé par le W3C, avec l'élaboration de IBM [Fierstone, 02].

Ce protocole repose entièrement sur le langage de description XML permettant de définir les mécanismes d'échanges d'information entre des clients et des fournisseurs de services Web [www2]. Il s'appuie sur n'importe quel protocole de communication (HTTP, SMTP, FTP …) pour transmettre les messages [Bieler, 05].

Un client peut accepter n'importe quelle structure de donnée et donner une réponse tout aussi complexe. Il est également envisageable qu'il n'y ait pas de réponse [Clari, 08].

SOAP définit une enveloppe contenant un "en-tête" et un "corps". L'en-tête fournit les instructions indiquant comment traiter le message. Le corps contient l'appel de la procédure distante dans un sens et la réponse du serveur dans l'autre.

[7] **URL** (*Uniform Resource Locator*)

a. XML

XML (*eXtensible Markup Language*) est un langage "extensible" contenant des "balises" comme HTML [8] . XML décrit la signification des données codées, indépendamment de la façon de les représenter. Contrairement à HTML dont les balises n'ont pour objet que de définir la présentation du document, XML permet d'inventer à volonté de nouvelles balises pour décrire, de façon arborescente, toutes les informations élémentaires expliquant la signification des données.

Son objectif initial est de faciliter l'échange automatisé de contenus entre systèmes d'informations hétérogènes.

XML contient aussi un aspect supplémentaire appelé "schéma", qui précise la structure des balises pour qu'un document soit valide. Il est ainsi possible de restreindre les données acceptées dans un document.

Un document XML peut contenir d'autres documents XML. Pour éviter toute confusion entre les balises utilisées, chaque document contient un "espace de noms", ce qui rend ainsi les balises uniques à l'intérieur d'un document.

b. HTTP

HTTP (*HyperText Transfer Protocol*) est un standard Internet proposé par l'IETF[9] utilisé sur le Web depuis 1990.

Le protocole HTTP utilise un jeu de requêtes/réponses entre un client, qui initie le dialogue, et un serveur. La communication peut être directe entre les deux acteurs mais elle peut également faire intervenir trois types d'intermédiaires, que voici :

- Un proxy, c'est-à-dire un agent qui transfère les messages vers le serveur après en avoir réécrit tout ou partie du contenu;

[8] **HTML** (*HyperText Markup Language*) : langage de description des pages web.

[9] **IETF** (*Internet Engineering Task Force*)

- Une passerelle, c'est-à-dire un agent qui agit comme une surcouche pour un serveur sous-jacent utilisant un autre protocole; cet agent se charge de traduire les messages pour permettre leur transfert vers ce serveur tiers;

- Un tunnel, c'est-à-dire un relais qui se charge de transmettre le message entre deux points de connexion sans modification du message (à travers un intermédiaire tel qu'un pare-feu).

La connexion établie par le protocole HTTP 1.0 est par défaut volatile, le client ouvre une connexion avec le serveur, envoie une requête et se met en attente de la réponse ; le serveur reçoit la requête, la traite, envoie la réponse et ferme la connexion. Pour garder la connexion ouverte au-delà du traitement de la requête/réponse courante, le client doit explicitement demander le maintien de la connexion (Keep-Alive).

c. Structure d'un message SOAP

Les messages SOAP sont échangés entre l'utilisateur de service et le fournisseur de service dans des enveloppes SOAP contenant une requête pour réaliser une action et le résultat de cette action. Les enveloppes SOAP sont formatées en XML et assez faciles à décoder [Belaid ,09].

Une enveloppe est constituée de deux parties: l'entête (*header*) et le corps du message (*body*):

En-tête : C'est un champ optionnel destiné à transmettre des informations quelconques telles que des informations sur l'authentification, le paiement, etc. Donc n'importe quel utilisateur du SOAP peut ajouter dans l'en-tête les données qu'il souhaite.

Corps : Le corps du message est un ensemble d'éléments (balises) XML. Il transporte les appels de procédures ou les rapports d'erreurs (Fault).Il n'existe pas de contraintes spécifiques sur ce contenu à part sur le format des données qui doivent respecter le modèle d'encodage précisé dans la déclaration de l'enveloppe.

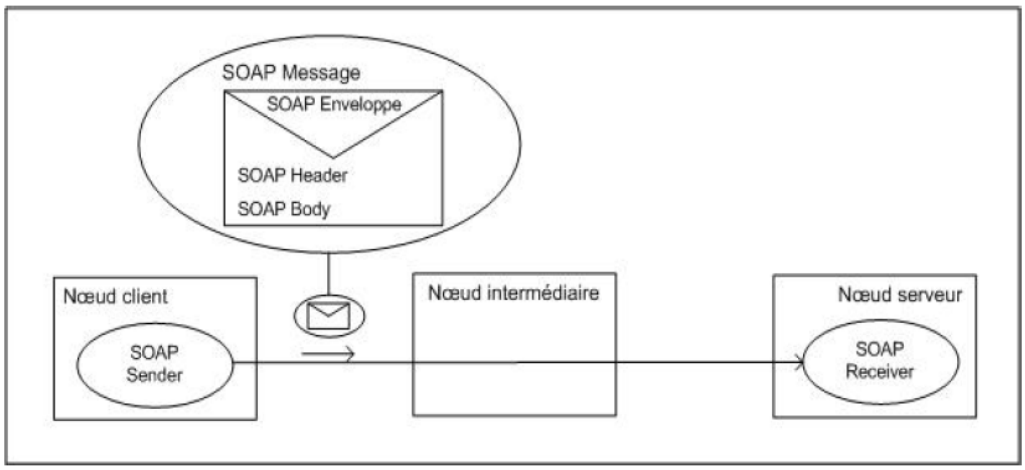


Figure 1.2 : Structure d'un document SOAP

3.3.2. Langage de description WSDL

a. Définition

WSDL (*Web Service Description Language*) est un langage de la famille XML permettant de décrire les types de données supportées et les fonctions offertes par un Service Web. L'objectif est de fournir la description, en XML, des services indépendamment de la plateforme et du langage utilisés et sous une forme que des personnes ou des programmes peuvent interpréter. Il est soutenu principalement par Ariba, Intel, IBM et Microsoft [Delacrétaz, 03]. Sa version 1.1 a été proposée en 2001 au W3C pour standardisation, mais elle n'a pas été approuvée contrairement à la version 2.0 qui est une recommandation officielle depuis 2007. [www3]

WSDL a été créé dans le but de fournir une description unifiée des services Web. Il décrit précisément quelles méthodes du service sont accessibles, et quels sont les types de leurs paramètres. Ce qui permet de décharger les utilisateurs des détails techniques de réalisation d'un appel.

Le langage WSDL s'appuie sur le format XML pour décrire des services réseau sous forme d'ensembles de noeuds de communication d'extrémités qui traitent des

messages contenant de l'information sur les données échangées ou sur la procédure par laquelle cet échange est fait.

b. Principaux concepts

Le document WSDL peut être divisé en deux groupes de sections. Le groupe du haut est constitué des définitions abstraites, tandis que le groupe du bas contient les descriptions concrètes. [Carlos, 01]

Définition abstraite : composée des éléments qui sont orientés vers la description des capacités du service Web. Ses éléments abstraits définissent les messages SOAP de façon indépendante de la plateforme et de la langue. Cela facilite la définition d'un ensemble de services pouvant être implémentés par différents sites Web. Les quatre éléments qui peuvent être définis dans un WSDL sont :

- **Types** décrit les types des données utilisées par le service Web indépendantes de la langue et de la machine en utilisant des XML Schéma. Il doit absolument se placer au début.
- **Message** décrit les noms et les types d'un ensemble de champs à transmettre. Il peut être comparé aux paramètres d'un appel de procédure.
- **portType** (*type de port)* décrit le service Web, les actions qu'il peut exécuter et les messages qui lui sont associés. Il peut être comparé aux prototypes des fonctions dans un langage de programmation traditionnel.
- **Opération** décrit les opérations invoquées de manière distante sur le service Web, chaque opération est décrite à l'aide de deux ou trois messages.
  - Le message reçu par le service Web lorsqu'il est sollicité par une requête ;
  - Le message émis en réponse par le service Web ;
  - L'éventuel message retourné par le service Web en cas de problèmes.

Descriptions concrètes : La description concrète est composée des éléments orientés vers le client pour le service physique. Les trois éléments concrets XML présents dans un WSDL sont :

- **binding** (*liaison*) définit le format des messages et le protocole utilisé par chaque type de port ;
- **Service** il s'agit de l'ensemble des ports exposés pour permettre l'accès aux services correspondants ;
- **Port** représente un point de terminaison identifié de manière unique par la combinaison d'une adresse Internet et d'une liaison.

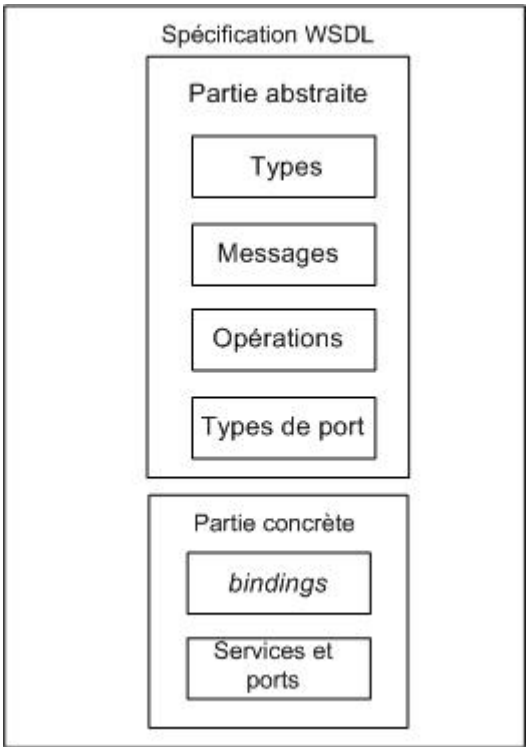


Figure 1.3 : Structure d'un document WSDL

3.3.3. Annuaire UDDI

Dans les sections précédentes ont a montré comment échanger des services entre différents processus, et la manière de les décrire et de les formaliser. La présente section va introduire, d'une manière brève, comment publier ces services et les rendre accessibles à une communauté de consommateurs.

En réponse à la montée en puissance du commerce électronique et plus spécialement de l'activité B2B sur Internet, un groupe de sociétés a lancé en 2000 le projet UDDI (*Universal, Description, Discovery and Integration*).

La spécification UDDI définit une architecture de communication et d'interopérabilité de services qui s'appuie sur des couches techniques déjà normalisées ou en voie de normalisation (HTTP, TCP/IP, SOAP, …).

a. Les structures de données

UDDI est un modèle d'information composé de structures de données persistantes appelées entités. Ces entités doivent être décrites en XML, et stockées dans les nœuds UDDI. Les différentes informations sont divisées en trois catégories : [Chauvet, 02]

- **pages blanches** (*White Pages*) : elles regroupent les informations sur les noms des entreprises publiant leurs services Web, les moyens de les contacter, etc.
- **pages jaunes** (*Yellow Pages*) : elles regroupent les informations à propos de la classification des entreprises.
- **pages vertes** (*Green Pages*) : elles contiennent des informations techniques comme les services offerts par une entreprise, leurs spécifications, etc.

Le modèle d'information UDDI (version 2) est composé de quatre types de structures de données. Cette répartition par type fournit des partitions simples pour faciliter la localisation rapide et la compréhension des différentes informations qui constituent un enregistrement. Les structures de données sont : [Rampacek, 06]

- **entité d'affaires** (*businessEntity*): cette entité décrit l'entreprise (également appelée fournisseur) qui permet d'accéder aux services Web qu'elle publie : cette entité permet de regrouper toutes les informations concernant le nom, les contacts de l'entreprise. Chaque enregistrement est associé à une clé unique appelée UUID (*Universal Unique Identifier*). Ce sont les éléments accessibles par l'annuaire pages blanches.
- **service d'affaires** (*businessService*): cette entité représente une classification des services. Elle est contenue dans l'entité décrite précédemment, et elle contient une clé unique identifiant un service particulier. Ce sont les éléments accessibles par l'annuaire pages jaunes.
- **modèle de rattachement** (*bindingTemplate*): cette entité est utilisée pour les détails techniques des services Web. Elle contient des informations sur le point d'accès du service (l'adresse du service). Ce sont les éléments accessibles par l'annuaire pages vertes.
- **index** (*tModel*)**:** cette entité permet de stocker les informations spécifiques aux services, comme le comportement, les conventions de typages et les types eux-mêmes utilisés par les services. Elle regroupe donc les informations contenues dans les fichiers WSDL.

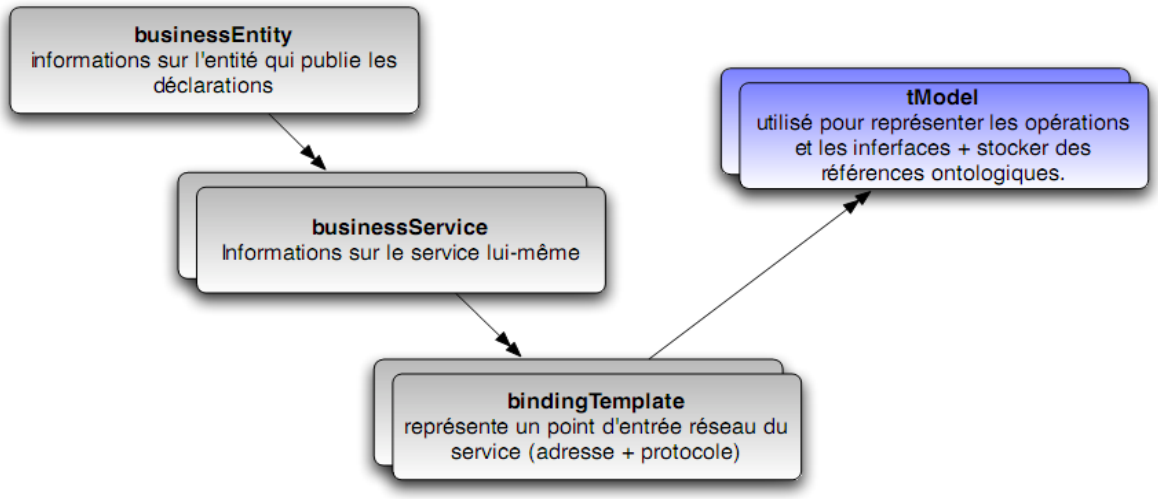


Figure 1.4 : Structure de données de l'annuaire UDDI **[Châtel, 07]**

b. L'accès à l'annuaire.

L'annuaire UDDI est accessible de deux manières [Mesano, 03]:

- via un navigateur Web qui dialogue avec une application Web dédiée, interface spécifique à l'annuaire accédé ;
- ou bien par programme, en utilisant l'API[10] définie par la spécification.

L'API comporte deux groupes de fonctions :

- les fonctions de recherche (Inquiry API) navigation, recherche et consultation des informations de l'annuaire;
- les fonctions de publication (Publishing API) publication, création, modification ou suppression des informations de l'annuaire.

Cette API de programmation est définie en WSDL et utilise le protocole SOAP pour interagir avec l'annuaire. En effet, elle est elle-même définie comme un Service Web.

Tous les appels de l'API sont synchrones. Le résultat d'une opération effectuée sur l'annuaire est immédiatement retourné.

L'accès à l'annuaire destiné à rechercher des informations est entièrement anonyme. Aucune identification n'est nécessaire pour ce type d'activité. En revanche, toute mise à jour des informations d'un annuaire requiert une phase d'identification et d'autorisation de la personne ou du processus qui se connecte. Toutes les fonctions de publication sont mises en œuvre via l'utilisation du protocole HTTPS.

[10] **API** (*Application Programming Interface*)

4. Conclusion

Les Services Web sont la dernière technologie pour l'intégration et l'interopérabilité des systèmes répartis. Basés sur le standard XML, ils sont caractérisés par leurs indépendances aux plates formes et aux systèmes d'exploitation, ce qui a impliqué leur adoption par les différentes organisations commerciales et industrielles offrant leurs services à travers le Web, et par conséquent l'augmentation du nombre de services offerts. On peut regretter aux Services Web le temps qu'ils prennent pour s'exécuter.

Algorithmes d'Optimisation

1. Introduction

D'une manière générale, l'efficacité des méthodes de résolution classiques est malheureusement dépendante d'un certain nombre d'hypothèses et de contraintes concernant le problème posé. Dans notre cas d'étude, ces contraintes sont liées principalement à la forme de la fonction de distribution des données à traiter.

Lorsqu'on veut proposer une nouvelle méthode de résolution de problèmes, il faut souvent une source d'inspiration. Celle-ci, peut être totalement imaginaire et n'est pas obligée de faire référence au monde réel, comme par exemple, les méthodes mathématiques abstraites, ou peut au contraire, être issue de la modélisation des systèmes naturels. Il s'agit dans ce dernier cas, d'imiter et d'adapter les concepts du monde des vivants pour la résolution de divers problèmes. Parmi ces derniers, se trouve le problème de l'organisation, du groupement et de la classification, qui se rencontre souvent chez les animaux et dans les systèmes biologiques.

En effet, la source d'inspiration que constitue la biologie a de plus en plus de succès dans une branche de l'intelligence artificielle, que l'on nomme *la biomimétique*.

2. Motivation

Pour de nombreux problèmes, il n'existe pas de solution déterministe qui donne le résultat en un temps raisonnable, et ceci malgré la création d'ordinateurs de plus en plus performants. Pour pallier à ce problème, on a recours à des méthodes dites heuristiques, c'est-à-dire des méthodes qui fournissent une solution approchée. Toutefois, il faut reproduire le processus sur plusieurs itérations pour tendre vers une solution acceptable.

On retrouve parmi ces heuristiques, certains algorithmes qui possèdent un principe générique adaptable et qui s'applique donc à plusieurs problèmes d'optimisation. On les appelle des métaheuristiques. La plus courante est la descente stochastique : on part d'une solution initiale, on la compare à tous ses voisins en conservant à chaque fois le meilleur résultat. L'optimisation par essaim particulaire, qui dérive de la descente stochastique, entre dans cette famille d'algorithmes. Elle s'inspire fortement des relations grégaires des oiseaux migrateurs qui doivent parcourir des longues distances et qui doivent donc optimiser leurs déplacements en termes d'énergie dépensée, comme par exemple la formation en V.

3. Problème d'optimisation

L'optimisation est une branche des mathématiques qui permet de résoudre des problèmes en déterminant le meilleur élément d'un ensemble selon certains critères prédéfinis. De ce fait, l'optimisation est omniprésente dans tous les domaines et évolue sans cesse depuis Euclide.

3.1. Définition

Un problème d'optimisation en général est défini par un espace de recherche *S* et une fonction objective *f*. Le but est de trouver la solution *s** € *S* de meilleure qualité *f*(*s**). Suivant le problème posé, on cherche soit le minimum soit le maximum de la fonction *f*. Dans la suite de ce mémoire, nous aborderons les problèmes d'optimisation essentiellement sous l'aspect minimisation, maximiser une fonction *f* étant équivalent à minimiser -*f*. L'équation (2.1) résume la définition précédente.

$$s^* = \min(f(s) \setminus s \in S) \qquad (2.1)$$

De plus, un problème d'optimisation peut présenter des contraintes d'égalité et/ou d'inégalité sur les solutions candidates $s \in S$, être multiobjectif si plusieurs fonctions objectifs doivent être optimisées ou encore dynamique, si la *topologie* de *f* change au cours du temps. Dans cette mémoire, nous n'étudierons que les problèmes mono-objectifs, le cas multiobjectif n'est pas abordé.

3.2. Quelques métaheuristiques pour l'optimisation

Les métaheuristiques sont une famille d'algorithmes stochastiques destinés à la résolution des problèmes d'optimisation. Leur particularité réside dans le fait que celles-ci sont adaptables à un grand nombre de problèmes sans changements majeurs dans leurs algorithmes, d'où le qualificatif *méta*. Leur capacité à optimiser un problème à partir d'un nombre minimal d'informations n'est contre balancée par le fait qu'elles n'offrent aucune garantie quant à l'optimalité de la meilleure solution trouvée. Seule une approximation de l'optimum global est donnée. Cependant, du point de vue de la recherche opérationnelle, ce constat n'est pas forcément un désavantage, étant donné que l'on préférera toujours une approximation de l'optimum global trouvée rapidement qu'une valeur exacte trouvée dans un temps rédhibitoire.

Les métaheuristiques sont des méthodes qui ont, en général, un comportement itératif, c'est-à-dire que le même schéma est reproduit un certain nombre de fois au cours de l'optimisation, et qui sont ''directes'', dans le sens où elles ne font pas appel au calcul du gradient de la fonction. Ces méthodes tirent leur efficacité du fait qu'elles sont moins facilement piégeables dans des optima locaux, car elles acceptent, au cours du traitement, des dégradations de la fonction objective et la recherche est souvent menée par une population de points et non un point unique. De plus, de par leur variété et leur capacité à résoudre des problèmes très divers, les métaheuristiques sont assez facilement sujettes à extensions.

Le monde des métaheuristiques est un monde en constante évolution. De nombreuses méthodes sont proposées chaque année pour tenter d'améliorer la résolution des problèmes les plus complexes. Du fait de cette activité permanente, un grand nombre de classes de métaheuristiques existe actuellement. Parmi celles-ci, on peut citer :

3.2.1 Recherche Tabou

L'algorithme de Recherche Tabou a été introduit par Glover en 1986 [Glover, 86]. Le but de cette méthode est d'inculquer aux méthodes de recherche locale un surcroît d'intelligence. L'idée ici est d'ajouter au processus de recherche une mémoire qui permette de mener une recherche plus ‘’intelligente’’ dans l'espace des solutions. La méthode de recherche tabou fonctionne avec une seule configuration courante, qui est actualisée au cours des itérations successives. La nouveauté ici est que, pour éviter le risque de retour à une configuration déjà visitée, on tient à jour un liste de mouvements interdits, appelée ‘’liste tabou’’. Cette liste contient m mouvements (t → s) qui sont les inverses des m derniers mouvements (s → t) effectués. L'algorithme modélise ainsi une forme primaire de mémoire à court terme. L'algorithme de recherche tabou peut être résumé par l'Algorithme 1.

Dans sa forme de base, l'algorithme de recherche tabou présente l'avantage de comporter moins de paramètres. Cependant, l'algorithme n'étant pas toujours performant, il est souvent approprié de lui ajouter des processus d'intensification et/ou de diversification, qui introduisent de nouveaux paramètres de contrôle [Glover & Laguna, 97].

– *Stratégie d'intensification :* On tente de faire émerger des propriétés communes aux meilleures solutions rencontrées afin de guider la recherche vers certaines régions intéressantes.

– *Stratégie de diversification :* C'est l'inverse de la stratégie d'intensification dans le sens ou l'on essaie de privilégier les régions pas encore parcourues.

Algorithme1 :

Déterminer une configuration aléatoire *s*
Initialiser une liste tabou vide
Tant que le critère d'arrêt n'est pas atteint **faire**
    Perturbation de *s* suivant *N* mouvements non tabous
    Evaluation des *N* voisins
    Sélection du meilleur voisin *t*
    Actualisation de la meilleure position connue *s**
    Insertion du mouvement t → s dans la liste tabou
    s = t
Fin Tant que

3.2.2 Colonies de fourmis

Cette approche, due à Dorigo et ses collaborateurs [Colorni,91], 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.

Les algorithmes de colonies de fourmis sont nés d'une constatation simple : les insectes sociaux, et en particulier les fourmis, résolvent naturellement des problèmes complexes. Un tel comportement est possible car les fourmis communiquent entre elles de manière indirecte par le dépôt de substances chimiques, appelées phéromones, sur le sol. Ce type de communication indirecte est appelé *stigmergie*.

La principale illustration de ce constat est donnée par la Figure (2.1). On voit sur cette figure que, si un obstacle est introduit sur le chemin des fourmis, les fourmis vont, après une phase de recherche, avoir tendance à toutes emprunter le plus court chemin entre le nid et l'obstacle [Goss,89]. Plus le taux de phéromone à un endroit donné est important, plus une fourmi va avoir tendance à être attirée par cette zone. Les fourmis qui sont arrivées le plus rapidement au nid en passant par la source de nourriture sont celles qui ont emprunté la branche la plus courte du trajet. Il en découle donc que la quantité de phéromones sur ce trajet est plus importante que sur le trajet plus long. De ce fait, le plus court chemin a une probabilité plus grande d'être emprunté par les fourmis que les autres chemins et sera donc, à terme, emprunté par toutes les fourmis.

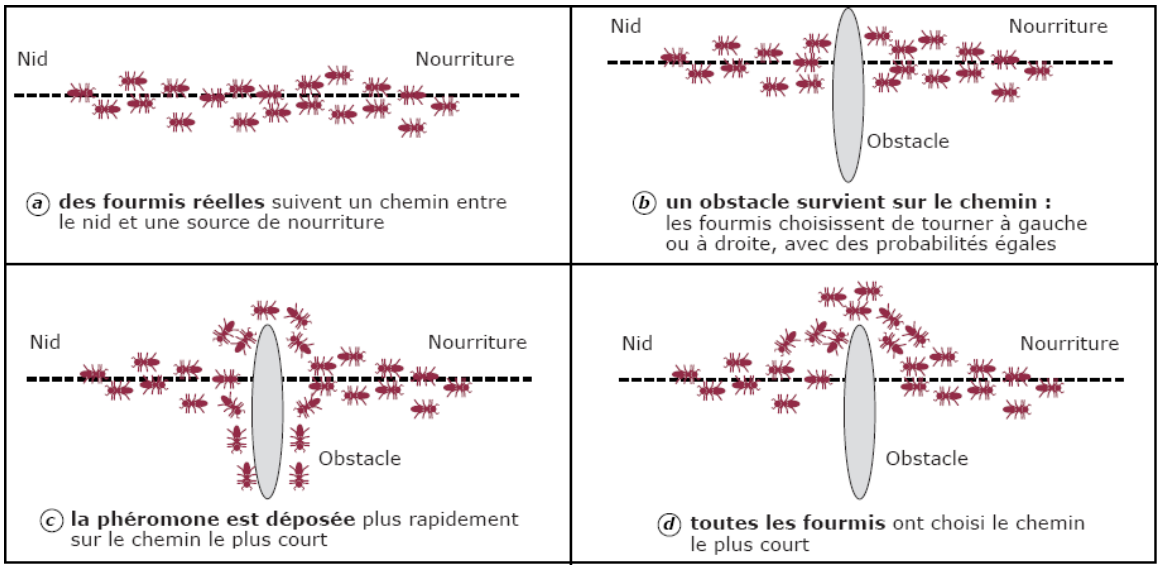


Figure 2. 1 : Détermination du plus court chemin par une Colonie de Fourmis

3.2.3. Optimisation par essaim particulaire

L'Optimisation par Essaim Particulaire est une nouvelle classe des métaheuristiques proposée en 1995 par Kennedy et Eberhart. [Kennedy & Eberhart, 95] Cet algorithme s'inspire du comportement social des animaux évoluant en essaim. Les différentes particules d'un essaim communiquent directement avec leurs voisins et construisent ainsi une solution à un problème, en s'appuyant sur leur expérience collective.

4. Algorithme a essaim particulaire

Dans le cas d'une formation d'oiseaux migrateurs, chaque oiseau essaie simplement de rester près de son voisin tout en évitant la collision avec lui. De plus, contrairement à ce que l'on pourrait penser, il n'y a pas de chef, chaque oiseau pouvant se trouver en tête de formation, au milieu ou à l'arrière de la formation. La forme caractéristique en V de telles populations n'est donc en rien calculée, elle résulte uniquement de l'émergence d'une forme stable étant donné des comportements individuels simples. Des comportements similaires se retrouvent dans le vol en groupe chez les insectes notamment les criquets à la recherche de la nourriture, ou encore dans la nage collective de certaines espèces de poissons. Par exemple, pour la chasse en

groupe, le banc de thons prend la forme d'un entonnoir, et pour échapper à un ennemi, le banc prend des mouvements d'explosion.

4.1. Etat de l'art

Plusieurs chercheurs ont créé des modèles interprétant le mouvement des vols d'oiseaux et des bancs de poissons. Plus particulièrement, Reynolds [Reynolds, 87] et Heppner et al. [Heppner, 90] ont présenté des simulations sur un vol d'oiseaux. Reynolds était intrigué par l'aspect esthétique du déplacement des oiseaux en groupe et Heppner, un zoologue, était intéressé à comprendre les règles permettant à un grand nombre d'oiseaux de voler en groupe : soit de voler sans se heurter, de changer soudainement de direction, de s'écarter et de se rapprocher de nouveau. Cette étude a grandement inspiré le développement de l'algorithme PSO.

En effet, lors de simulations des modèles mathématiques décrivant les vols d'oiseaux, Wilson [Wilson, 75] a suggéré que ces types de modèles pourraient très bien s'appliquer à la recherche de points caractéristiques dans un espace de recherche. Sa réflexion se base sur le fait que, lors de l'installation d'une mangeoire à oiseaux dans une cour, même si aucun oiseau ne l'a jamais visitée, après quelques heures de patience un grand nombre d'oiseaux viendront y manger. Lors des simulations de Wilson, la volée d'oiseaux cherchait une mangeoire dans un espace donné et finissait par découvrir son emplacement. En utilisant les algorithmes de modélisation de Heppner [Heppner, 90] et de Reynolds [Reynolds, 87], et en modifiant le modèle mathématique de Wilson [Wilson, 75], Kennedy et Eberhart [Kennedy & Eberhart, 95] ont transformé le tout en un vol d'oiseaux cherchant la « mangeoire » la plus grosse dans un lot de mangeoires contenues dans une région prédéterminée. L'algorithme d'optimisation PSO a ainsi vu le jour.

4.2. Modèle de Reynolds

Craig Reynolds a développé en 1986 un modèle d'animation pour coordonner le mouvement de groupes d'animaux de type vol d'oiseaux ou banc de poisson, nommés *Boids*. A partir de règles de comportement extrêmement simples, des animations complexes en 2D sont produites avec un réalisme étonnant.

a. Notion de Boid

Un boid est une entité sensée à modéliser un individu afin d'étudier les propriétés d'assemblage de plusieurs de ces individus. Chaque boid est implémenté individuellement et navigue selon sa propre perception de l'environnement dynamique. Le boid possède un angle de vision et une distance d'application des forces. Ce couple (angle, distance) définit la partie de l'espace qui l'influence.

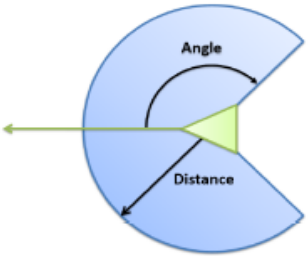


Figure 2.2 : L'espace d'influence d'un boid

b. Règles de Reynolds

Le comportement de chaque Boid est soumis à trois lois qui génèrent des forces s'appliquant sur lui et le font évoluer dans un monde peuplé d'autres Boids. Ces règles sont :

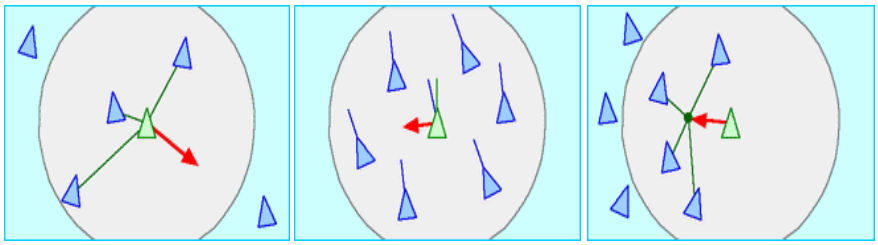
La séparation : stipule qu'un boid doit s'éloigner des autres boids trop proches de lui, deux boids ne peuvent pas se trouver au même endroit au même moment. Si deux boids sont trop proches, ils vont donc choisir de se déplacer selon des directions différentes afin de ne pas rentrer en collision;

La cohésion : stipule qu'un boid doit éviter de se trouver sur l'extérieur de la formation et ainsi se déplacer vers le centre perçu de la formation. Pour former un groupe, les boids se rapprochent les uns des autres. ;

L'alignement ou mimétisme: stipule qu'un boid doit se déplacer dans la même direction d'ensemble en moyennant les vitesses et les directions des autres boids. Pour rester groupés, les boids essayent de suivre une même direction. Chaque boid va

corriger sa direction, pour la faire correspondre à la direction moyenne des directions de ses voisins;

Ces trois règles de base, permettent l'attraction et la répulsion de chacun des individus et permettent la stabilité de l'ensemble.



(a) Séparation (b) Alignement (c) Cohésion

Figure 2.3 : Les règles de Reynolds

4.3. Modèle de Kennedy et Eberhart

L'Optimisation par Essaims Particulaires (OEP) est une métaheuristique inventée par Russel Eberhart et James Kennedy en 1995 [Kennedy & Eberhart, 95]. L'algorithme s'inspire du modèle développé par Craig Reynolds à la fin des années 1980 afin de simuler les mouvements d'un groupe d'oiseau. Cette méthode possède des similitudes avec l'algorithme des colonies de fourmis. Notamment car il fait intervenir des agents pouvant communiquer entre eux de manière très simple et permettant l'émergence de comportements complexes.

Kennedy et Eberhart [Kennedy & Eberhart, 95] ont cherché 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. Les particules sont les individus et elles se déplacent dans l'hyperespace de recherche. Le processus de recherche est basé sur les deux règles suivantes :

1. Chaque particule est dotée d'une mémoire qui lui permet de mémoriser le meilleur point par lequel elle est déjà passée et elle a tendance à retourner vers ce point.

2. Chaque particule est informée du meilleur point connu au sein de son voisinage et elle va tendre à aller vers ce point.

a. Notion de voisinage

Le voisinage constitue la structure du réseau social. Les particules à l'intérieur d'un voisinage communiquent entre-elles. Différents voisinages ont été étudiés [Kennedy, 99] et sont considérés en fonction des identificateurs des particules et non des informations topologiques comme les distances euclidiennes dans l'espace de recherche :

– Topologie en étoile (figure 2.4 (a)) : le réseau social est complet, chaque particule est attirée vers la meilleure particule notée gbest et communique avec les autres.

– Topologie en anneau (figure 2.4 (b)) : chaque particule communique avec n voisines immédiates. Chaque particule tend à se déplacer vers la meilleure dans son voisinage local notée lbest.

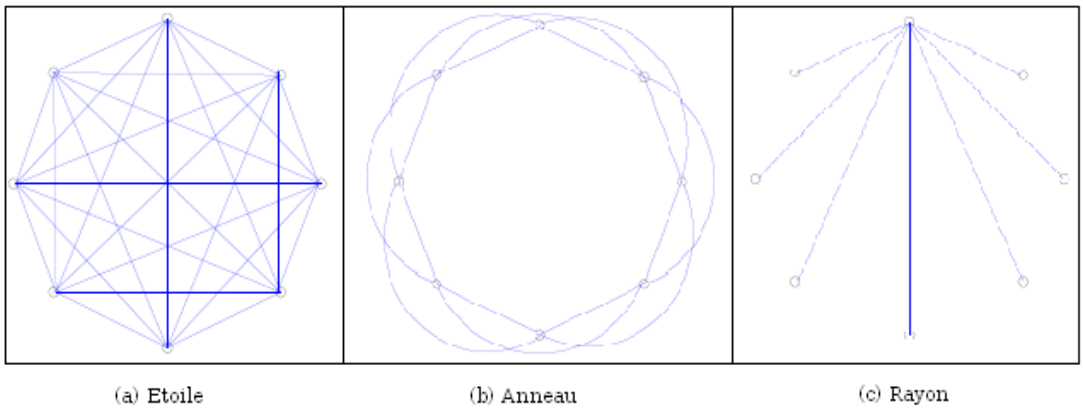


Figure 2.4 : Les topologies de voisinage

– Topologie en rayon (figure 2.4 (c)) : une particule "centrale" est connectée à tous les autres. Seule cette particule centrale ajuste sa position vers la meilleure, si cela provoque une amélioration l'information est propagée aux autres.

b. Algorithme d'OEP

A partir des quelques informations dont elle dispose, une particule doit décider de son prochain mouvement, c'est-à-dire décider de sa nouvelle vitesse. Pour ce faire, elle combine linéairement trois informations (figure 2.5) :

- sa vitesse actuelle
- sa meilleure performance
- la meilleure performance de ses voisines (ses informatrices).

À l'aide de trois paramètres parfois appelés *coefficients de confiance*, qui pondèrent trois tendances :

- tendance à suivre son propre voie
- tendance conservatrice (revenir sur ses pas)
- tendance « panurgienne » (suivre le meilleur voisin)

Les équations de base de l'algorithme sont les suivantes :

$$\mathbf{v}_i\,(t+1)= \omega\,\mathbf{v}_i\,(t)+ \varphi_p\, r_p\,(\mathbf{p}_i-\mathbf{x}_i) + \varphi_g\, r_g\,(\mathbf{g}-\mathbf{x}_i) \qquad (2.2)$$

$$\mathbf{x}_i\,(t+1)\leftarrow \mathbf{x}_i\,(t)+ \mathbf{v}_i\,(t+1) \qquad (2.3)$$

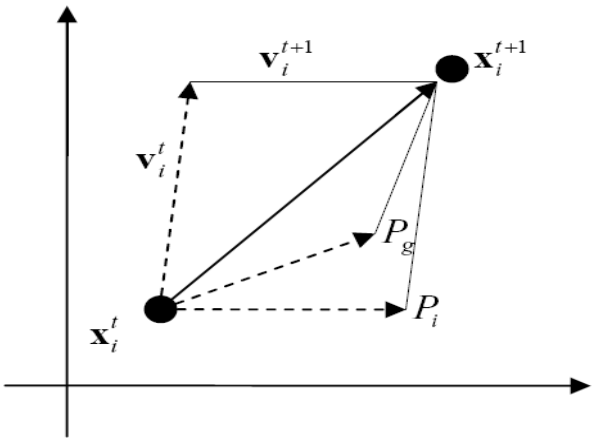


Figure 2.5 : Schéma de principe du déplacement d'une particule

Pseudo code d'OEP :

- Pour chaque particule $i = 1, ..., S$ faire:
  - Initialiser les positions des particules par des vecteurs uniformément distribués choisis aléatoirement. $\mathbf{x}_i \sim U(\mathbf{x}_{min}, \mathbf{x}_{max})$, d'où $\mathbf{x}_{min}$ et $\mathbf{x}_{max}$ sont la plus petit et la plus grand valeur de l'espace de recherche.
  - Initialiser les meilleures positions des particules par les positions initiales de ces derniers: $\mathbf{p}_i \leftarrow \mathbf{x}_i$.
  - Si $(f(\mathbf{p}_i) < f(\mathbf{g}))$ modifier la meilleure position du l'essaim: $\mathbf{g} \leftarrow \mathbf{p}_i$
  - Initialiser les vitesses des particules: $\mathbf{v}_i$
- Jusqu'à ce que les critères d'arrêt sont satisfait (exemple : nombre d'itérations acquis, ou la fitness adéquate est trouvé), répéter:
  - Pour chaque particule $i = 1, ..., S$ faire:
    - Choisir des nombres aléatoire: $r_p, r_g \sim U(0,1)$.
    - Modifier les vitesse des particules selon (2.2) .
    - Modifier les positions des particules par (2. 3).
    - Si $(f(\mathbf{x}_i) < f(\mathbf{p}_i))$ faire:
      - Modifier les meilleures positions des particules: $\mathbf{p}_i \leftarrow \mathbf{x}_i$.
      - Si $(f(\mathbf{p}_i) < f(\mathbf{g}))$ Modifier la meilleure position de l'essaim: $\mathbf{g} \leftarrow \mathbf{p}_i$.
- Maintenant **g** présente la meilleure solution trouvée.

c. OEP adaptative

Comme toutes les autres métaheuristiques, l'OEP possède de nombreux paramètres de contrôle. Le réglage de tous ces paramètres est un processus qui s'avère long et difficile. En effet, chaque paramètre ayant une forte influence sur le comportement de l'algorithme [Shi & Eberhart, 98], il est important de trouver un jeu de paramètres bien adapté au problème posé. De ce fait, chaque problème nécessiterait en théorie une étude qui permettrait de dégager le jeu de paramètres optimal pour le traiter. Or, un tel procédé est souvent long à réaliser et demande une bonne connaissance au préalable du comportement de l'algorithme. C'est pourquoi un nouvel axe de recherche, qui s'attache à réduire le nombre de paramètres "libres" des métaheuristiques, s'est développé.

Le but est de trouver des règles qui affranchissent l'utilisateur de définir les paramètres, en calculant ceux-ci au fur et à mesure du traitement, en fonction des résultats donnés par l'algorithme. Ainsi, l'algorithme ''adapte'' son comportement au problème posé, au lieu d'avoir un comportement figé, défini au préalable par l'utilisateur. Le but ultime serait de concevoir un algorithme qui n'aurait aucun paramètre de contrôle ; un tel algorithme agirait comme une ''boîte noire'' pour laquelle l'utilisateur n'aurait qu'à définir les problèmes et le critère d'arrêt. Évidemment, pour être valable, un tel algorithme se doit de présenter des résultats au moins égaux à ceux des algorithmes paramétriques.

En 2003, Clerc [Clerc, 03] a développé un algorithme adaptatif appelé TRIBES. Cette liste est non exhaustive, mais dresse un aperçu assez complet de tout ce qui peut être imaginé pour adapter un algorithme d'OEP.

d. TRIBES

TRIBES est un algorithme OEP créé par M. Clerc dans [Clerc, 03].Cet algorithme permet d'avoir un paramétrage automatique de l'OEP, l'utilisateur doit seulement définir la fonction objective et le critère d'arrêt. Cependant, il est à signaler que TRIBES ne peut pas résoudre tous les problèmes avec exactitude. De plus, ses

résultats sont probabilistes à cause de son caractère stochastique. Le but de TRIBES, d'après son auteur, est d'être efficace dans la plupart des cas et de permettre à ses utilisateurs de gagner du temps, en évitant l'étape de réglage de la métaheuristique.

Dans TRIBES, l'essaim est divisé en plusieurs sous-essaims appelés tribus. Le but est d'explorer simultanément plusieurs régions de l'espace de recherche, généralement des optima locaux, avant de prendre une décision globale. Dans le but de prendre une telle décision, les tribus échangent leurs résultats tout au long du traitement. Un tel type de structure est comparable aux structures multi-essaims utilisées dans d'autres algorithmes

Deux types de communications sont donc à définir : la communication intra-tribu et la communication inter-tribu. Les relations entre les particules à l'intérieur d'une tribu sont définies par une topologie complètement connectée. Chaque particule connait la meilleure et la plus mauvaise position jamais atteinte par la tribu, c'est-à-dire que chaque particule connait les positions pour lesquelles la tribu a trouvé la plus petite et la plus grande valeur de la fonction objective. Cette forme de communication est appelée communication intra-tribu.

Au fur et à mesure du traitement, chaque tribu va converger vers un optimum local. Il est donc nécessaire que celles-ci communiquent entre elles pour définir lequel de ces optima est à retenir par l'utilisateur. La communication entre les tribus est établie par l'intermédiaire des meilleures particules de chaque tribu. Ce type de communication est appelé communication inter-tribu.

En résumé, chaque particule est informée par elle-même (mémoire cognitive p), par tous les éléments de sa tribu (appelés informateurs internes), et, si cette particule est un chaman (e.g. la meilleure particule d'une tribu), alors elle est aussi informée par les chamans des autres tribus (appelés informateurs externes). Toutes ces positions sont appelées les informateurs de la particule. La mémoire sociale g de chaque particule est l'informateur pour lequel la valeur de la fonction objective est la plus petite. La Figure (2.6) illustre la topologie du graphe d'information de l'essaim Les traits gras symbolisent la communication inter-tribu alors que les traits fins matérialisent la communication intra-tribu.

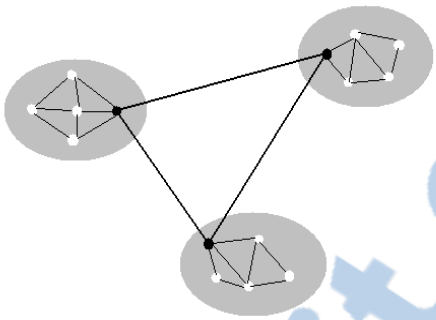


Figure 2.6 : Réseau d'information

5. Conclusion

Dans ce chapitre, nous nous sommes intéressés à une famille d'algorithmes stochastiques destinés à résoudre des problèmes d'optimisation. Utilisées dans de nombreux domaines, ces méthodes présentent l'avantage d'être généralement efficaces, sans pour autant que l'utilisateur ait à modifier la structure de base de l'algorithme qu'il utilise. Parmi celles-ci, l'Optimisation par Essaim Particulaire (OEP) est une nouvelle classe d'algorithmes proposée pour résoudre les problèmes d'optimisation. Les ''particules'' d'un même essaim communiquent de manière directe entre elles tout au long de la recherche pour construire une solution au problème posé, en s'appuyant sur leur expérience collective.

Le réglage des paramètres des algorithmes est un des défauts qui rebutent encore certains utilisateurs. Il est important, pour chaque problème posé, de trouver le jeu de paramètres qui conduise à des performances optimales de l'algorithme.

Conception et implémentation

1. Introduction

Notre application consiste à développer un système de sélection des services web à base de qualité de service dénommé WSelect.

Dans ce dernier chapitre, nous allons voir les différentes étapes suivies durant la réalisation de notre application, nous commençons par une présentation de la notion de la sélection où nous allons définir les attributs, les fonctions d'agrégation ainsi que la fonction objective puis, nous allons parler du processus de développement logiciel utilisé et nous finissons par présenter les différents diagrammes de conception ainsi que l'application et les résultats obtenus.

2. Sélection des services web

La migration des systèmes d'information des entreprises vers un schéma orienté Business to Business (i.e., des entreprises mettant leurs services à disposition d'autres entreprises sous forme de composants logiciels) a favorisé l'émergence des services Web depuis le début des années 2000 [Austin, 02]. Par exemple, une compagnie aérienne peut mettre en ligne à disposition des agences de voyage un service logiciel de réservation de vols. Pour fournir une valeur ajoutée à l'utilisateur final, une agence de voyage peut décider de composer plusieurs services Web afin de faciliter l'organisation d'un séjour. Par exemple, une agence peut proposer un pack ‘’déplacement‘’ comportant d'une part la réservation d'un billet pour un événement sportif, d'autre part une réservation de billet de train ou d'avion. On parle alors de composition de services Web [Dustdar, 05]

Il arrive très fréquemment que plusieurs services répondent à un même ensemble de besoins fonctionnels. Par exemple, pour réaliser la réservation d'une place au stade, un utilisateur peut mettre en concurrence plusieurs services de billetterie. Une sélection de services Web consiste alors à assigner un service Web à chaque classe. Cette affectation est appelée *plan d'exécution*. L'ensemble des plans d'exécution possibles est combinatoire : pour *n* classes avec un choix parmi *m* services pour chacune, il y a m^n compositions possibles. Afin de discriminer parmi ces plans d'exécution, on utilise la notion de qualité de service.

2.1. Qualité des services web

Puisque nous ne connaissons pas les services avant d'être lié, la seule ressource que nous avons pour évaluer les fournisseurs est leur "contrat", c'est-à-dire la spécification de service. La description de service est composée des éléments qui spécifient les propriétés fonctionnelles (propriétés liées à la fonction réalisée par le service) et aussi des propriétés non fonctionnelles (les propriétés supplémentaires).

Les propriétés non fonctionnelles peuvent être aussi appelées les qualités de service QoS [11] puisqu'elles contiennent de l'information qui peut aider les consommateurs à définir la qualité de service offert ainsi de choisir le fournisseur de service. Un consommateur acquiert les QoS de différents fournisseurs pour les comparer et choisir ce qui convient à ces besoins. Bien qu'il y ait plusieurs recherches pour grouper et modéliser les attributs QoS, il n'existe pas de modèle standard pour représenter les attributs.

Le consortium W3C recapitalise les attributs de QoS pour les Services Web et propose une définition standard de QoS. Au-dessus cette définition, [Wang, 07] propose un groupement des attributs QoS, composé de quatre sous-parties ; performance, sûreté de fonctionnement, sécurité et applications spécifiques.

Parallèlement, il existe différentes méthodes pour comparer et choisir les services selon les différentes caractéristiques et attributs qu'ils prennent en compte ; Pour qu'elle soit efficace, une méthode de sélection de fournisseur de service doit modéliser les critères non fonctionnels (les attributs QoS) d'une façon générique.

[11] **QoS** (*Quality of Service*)

2.2. Critères de qualité de services

Dans ce qui suit, nous présentons l'ensemble des attributs de QdS génériques. Nous distinguons ici les attributs mesurables et les attributs non mesurables.

Attributs mesurables :

Les attributs mesurables les plus communs sont décrits par les paramètres liés à la performance.

– *Le débit :* Le nombre de requêtes servies pendant un intervalle de temps.

– *Le temps de réponse :* Le temps requis pour compléter une requête du service web.

– *La fiabilité :* La capacité d'un service d'exécuter correctement ses fonctions.

– *La scalabilité :* La capacité du service de traiter le plus grand nombre d'opérations ou de transactions pendant une période donnée tout en gardant les mêmes performances.

– *La robustesse :* La probabilité qu'un service peut réagir proprement à des messages d'entrées invalides, incomplètes ou conflictuelles.

– *La disponibilité :* La probabilité d'accessibilité d'un service.

QdS non mesurables :

Il y a des attributs de QdS qui ne sont pas mesurables, mais qui ont de l'importance pour les services web comme :

– *Le prix d'exécution :* C'est le prix qu'un client du service doit payer pour bénéficier du service.

– *La réputation :* C'est une mesure de la crédibilité du service. Elle dépend principalement des expériences d'utilisateurs finals.

– *La sécurité :* C'est un regroupement d'un ensemble de qualités à savoir : la confidentialité, le cryptage des messages et le contrôle d'accès. [Ben Halima, 09]

Certains de ces attributs peuvent être évalués de manière quantitative, mais la plupart d'entre eux sont des attributs qualitatifs.

Due à la présence inévitable des erreurs et des défaillances, les systèmes ne sont jamais totalement disponibles, fiable et sur. Pour cela, ces attributs sont généralement exprimés relativement en probabilité. Pourtant, généralement tous les systèmes nécessitent de définir la disponibilité, l'intégrité et la maintenabilité.

L'ensemble des attributs de QoS peut être devisé en deux: attributs négatifs et attributs positifs.

Les valeurs des attributs négatifs ont besoin d'être minimisées, (exemple : temps de réponse, prix), les valeurs des attributs positifs ont besoin d'être maximisées (exemple : disponibilité).

Pour simplicité, nous considérons seulement les attributs négatifs car les attributs positifs peuvent être transformés en attributs négatifs en multipliant leurs valeurs par -1.

2.3. Calcul de la qualité du web service composite

Il y a beaucoup d'aspects de QdS importantes pour des services Web. Parmi ceux-ci nous avons choisis quelques une :

Latence : $Time_{process}(op) + Time_{results}(op)$ tel que :
$Time_{process}$ *:* temps d'exécution d'*op*.
$Time_{results}$ *:* temps de transmission et réception des résultats.

Sûreté : $N_{success}(op) / N_{invoked}(op)$ tel que :
$N_{success}$ *:* Nombre du temps où *op* a réussit dans son exécution.
$N_{invoked}$ *:* Nombre total du temps des invocation.

Disponibilité : *UpTime(op) / TotalTime(op)* tel que :
UpTime est le temps dont lequel *op* est accessible durant le temps total mesuré *TotalTime*.

Coût : le montant total pour exécuter une opération.

Réputation : $\sum_{u=1}^{n} Ranking_u(op)/n$, $1 \le Réputation \le 10$ tel que :
$Ranking_u$: c'est le rang du l'utilisateur *u* et *n* c'est le nombre total du temps où *op* est classé.

Nous avons besoin maintenant de calculer les paramètres de QoS pour la composition des services web où chaque service contient de multiples opérations, dans notre cas nous considérons seulement deux opérations dans chaque service. En se basant sur le sens de chaque QoS, nous définissons un ensemble de fonctions d'agrégation pour calculer les QoS pour la composition des services web. Les fonctions d'agrégation sont résumées dans tableau (3.1).

Paramètres de QoS	**Fonction d'agrégation**
Latence	$\sum_{i=1}^{n} lat(op_i)$
Sûreté	$\prod_{i=1}^{n} sur(op_i)$
Disponibilité	$\prod_{i=1}^{n} dis(op_i)$
Coût	$\sum_{i=1}^{n} cou(op_i)$
Réputation	$\frac{1}{n}\sum_{i=1}^{n} rep(op_i)$

Table 3.1 : Les fonctions d'agrégation de chaque QoS

La qualité du service web composite peut être caractérisé comme un vecteur de QoS :

Qualité (SC) = (lat(SC), sur(SC), dis(SC), cou(SC), rep(SC)).les intervalles de valeurs de ces attributs sont résumé dans le tableau (3.2).

Paramétres	Service web 1		Service web 2		Service web 3	
	Op1	Op2	Op3	Op4	Op5	Op6
lat (latence)	0-300(s)	0-300(s)	0-300(s)	0-300(s)	0-300(s)	0-300(s)
sur (sûreté)	0.5-1.0	0.5-1.0	0.5-1.0	0.5-1.0	0.5-1.0	0.5-1.0
dis (disponibilité)	0.7-1.0	0.7-1.0	0.7-1.0	0.7-1.0	0.7-1.0	0.7-1.0
cou (coût)	0-30(€)	0-30(€)	0-30(€)	0-30(€)	0-30(€)	0-30(€)
rep (réputation)	0-5	0-5	0-5	0-5	0-5	0-5

Table 3.2 : Les intervalles de chaque QoS

Avant d'entrer dans les détails, nous définissons quelques fonctions d'approximation qui vont être utilisé dans cette stratégie. Les deux fonctions d'agrégation (i.e., les fonction pour sûreté et disponibilité) présenté dans le tableau (3. 1) qui ne combinent pas les paramètres de QoS de multiples opérations des services par une méthode linéaire. Nous proposons deux fonctions linéaires pour approximer les fonctions originaux.

$$Sûreté = \sum_{i=1}^{n} \log(sur(op_i)) \quad , \quad Disponibilté = \sum_{i=1}^{n} \log(dis(op_i))$$

En résumé, Nous étendons notre model pour supporter la fonction d'agrégation suivante qui regroupe toutes les opérations de service composite :

$$Q_i(CS) = \sum_{j=1}^{n} Q_i(op_j)$$

2.4. Fonction objective

Dans l'ordre d'évaluer la qualité multidimensionnelle du web service composite, une fonction utilitaire est utilisée. Celle-ci peut faciliter la comparaison entre les qualités de chaque composition de web services.

Le calcul de la fonction objective nécessite le calcul de valeurs maximum et minimum des attributs de QoS. Par exemple la valeur maximum de l'attribut *Coût* pour n'importe quel composition des web services peut être calculée par la sommation du plus coûteux service dans chaque classe de service, et en parallèle pour calculer le plus coûteux service nous sommons les valeurs des opérations les plus coûteux. Formellement nous calculons la valeur agrégée minimum et maximum du i^{ème} attribut de QoS comme suit :

$$Q_i^{\min} = \sum_{j=1}^{n} Q^{\min}(op_j) \qquad Q_i^{\max} = \sum_{j=1}^{n} Q^{\max}(op_j)$$

Le présent processus est ensuite suivi par le processus de weighting pour représenter les priorités et les préférences des clients, puisque les utilisateurs peuvent avoir des préférences sur le résultat de leurs requêtes, ils peuvent spécifier l'importance relative à chaque paramètre de QoS. Nous attribuons des poids rangé entre 0 et 1 pour

chaque paramètre de QoS pour refléter le niveau de leur importance. Dans notre étude nous donnons à l'utilisateur de définir leurs poids, sinon il peut la laisser telle quelle est par défaut où les poids sont égaux. Nous utilisons le score de la fonction objective *F* pour évaluer la qualité de la composition des web services. Par l'utilisation de la fonction

objective, l'optimisation du QoS est de trouver la composition des web service avec un minimum score.

$$F(CS) = \sum_{Q_i \in Neg} W_i \frac{Q_i^{\max} - Q_i(CS)}{Q_i^{\max} - Q_i^{\min}} + \sum_{Q_i \in Pos} W_i \frac{Q_i(CS) - Q_i^{\min}}{Q_i^{\max} - Q_i^{\min}}$$

Tel que $W_i \in \Re_0^+$ et $\sum_{i=1}^{r} W_i = 1$ c'est les poids de chaque paramètres pour représenter les priorités de l'utilisateur.

2.5. Algorithme d'optimisation

Si nous regardons notre analogie à la vie quotidienne, nous choisissons continuellement des fournisseurs de services entre ceux disponibles : Regarder le chaîne de télé qui nous convient, écouter le radio dont la musique nous plait, choisir le fournisseur d'Internet le plus agréable,... Malgré que nos choix ne nous mènent pas toujours à une solution idéale, nous essayons de trouver la plus convenable par rapport à nos besoins.

Nous avons choisis dans notre étude comme il est cité précédemment dans le 2ème chapitre, l'algorithme d'essaim particulaire comme algorithme d'optimisation, dans ce qu'il suit nous détaillons la configuration et le paramétrage utilisé.

2.5.1. La base de donnée

Nous avons créé un schéma de service qui contient trois (3) classes de services web. Pour simplicité chaque classe se compose de deux (2) opérations .Le nombre d'instances dans chaque classe est quarante (40) fournisseurs de services web. Chaque

fournisseur se caractérise par des qualités de service. Nous utilisons cinq (5) paramètres pour évaluer les opérations des services : latence, sûreté, disponibilité, coût et réputation. Les valeurs de ces paramètres sont générées en se basant sur une distribution uniforme. Un échantillon de notre base est illustré dans le tableau (3. 3) qui représente le fournisseur i de la classe 1.

Les attributs	**Opération 1**	**Opération 2**
Latence	239	49
Sûreté	0,96	0,50
Disponibilité	0,72	0,79
Coût	13	28
Réputation	4	1

Table 3.3 : Un exemple de base de données

2.5.2. La configuration de l'essaim

Dans TRIBES, l'essaim est divisé en plusieurs sous-essaims appelés tribus. Le but est d'explorer simultanément plusieurs régions de l'espace de recherche, généralement des optima locaux, avant de prendre une décision globale. Dans le but de prendre une telle décision, les tribus échangent leurs résultats tout au long du traitement. Un tel type de structure est comparable aux structures multi-essaims utilisées dans d'autres algorithmes

Nous avons utilisé en global trente (30) agents, représentant le nombre total de particules dans l'essaim. Notre essaim est divisé en six (6) sous-essaims ou groupes, Le but est d'explorer simultanément plusieurs régions de l'espace de recherche afin de trouver les optima locaux, avant de prendre une décision globale. Dans le but de prendre une telle décision, les tribus échangent leurs résultats tout au long du traitement.

Dans notre modèle, la particule est modélisée par un simple vecteur de 3 champs, Ces trois champs prennent leurs valeurs d'indices des fournisseurs. Par exemple le vecteur (12,39,1) qui a ces valeurs représente le fournisseur numéro 12 du 1ère classe, numéro 39 du 2ème classe et le fournisseur numéro 1 du 3ème classe.

2.5.3. Les étapes de l'algorithme

Etape1 : Initialisation des particules

Nous initialisons les vecteurs *Positions* des particules par des valeurs aléatoirement choisis de la base. Ces derniers sont affectés aux vecteurs *Best-positions*.

Etape 2 : Initialisation des groupes

En calculant la valeur du fitness de chaque particule qui est initialisé dans l'étape précédente, nous initialisons les vecteurs *Best-gr-Positions* qui représentent la meilleure position de chaque tribu c'est-à-dire les positions pour lesquelles la tribu a trouvé la plus petite valeur de la fonction objective.

Etape 3 : Application de l'algorithme

Afin d'appliquer l'algorithme, nous utilisons une boucle, Cette dernière a comme critère d'arrêt une variable entier *MaxIteration* initialisé à 10.

- Pour changer les valeurs du *Best*-Positions par des valeurs mieux que selon initialisé, nous comparons les valeurs de fitness.

Si (fitness (*Positions*) < fitness (*Best-positions*)) faire changer les valeurs de *Best-Positions* par les valeurs de *Positions*.

- Pour changer les valeurs du *Best-gr-Positions* par des valeurs mieux que selon initialisé précédemment, nous comparons les valeurs du fitness.

Si (fitness (*Best-positions*) < fitness (*Best-gr-Positions*)) faire changer les valeurs de *Best-gr-Positions* par les valeurs de *Best-positions.*

- Pour changer les valeurs de *Positions* nous utilisons la mutation probabiliste qui se divise en deux étapes, la contribution individuelle et la contribution du voisinage. Prenons un exemple simple du vecteur de *Positions* qui se compose du plan d'exécution (S_x , S_y , S_z).

La contribution individuelle :

Nous choisissons l'un des trois services web de la solution actuelle S_x ou S_y ou S_z et nous le remplaçons par une valeur choisie de façon aléatoire de la base, et bien sur qui appartient à la même classe. Supposons qu'on a choisis S_x , la valeur qui sera remplacé doit être appartient au $1^{ère}$ classe.

La contribution du voisinage :

Nous choisissons l'un des deux valeurs du vecteur du plan d'exécution qui reste et nous le remplaçons par son équivalent dans le vecteur *Best-gr-Positions*. Supposons qu'on a choisis S_z , la valeur qui sera remplacé cette dernière est choisis depuis le vecteur *Best-gr-Positions* tel que le fournisseur z appartient a ce groupe. Ce type de communication est appelé communication inter-tribu.

Si nous atteignons la condition d'arrêt alors on s'arrête.

Sinon nous retournons pour reboucler.

Au fur et à mesure du traitement, chaque tribu va converger vers un optimum local. Il est donc nécessaire que celles-ci communiquent entre elles pour définir lequel de ces optima est l'optimum global. La communication entre les tribus est établie par l'intermédiaire des meilleures particules de chaque tribu. Afin de les trouver nous Comparons les fitness des *Best-gr-Positions*, et nous retournons le minimum.

3. Conception de l'application

Nous allons présenter notre méthode de travail et le processus logiciel qu'on a suivi, à savoir le Processus Unifié (UP).

Mais tous d'abord voyons c'est quoi un processus logiciel.

3.1. Processus de développement logiciel

Un processus définit une séquence d'étapes, en partie ordonnées, qui concourent à l'obtention d'un système logiciel ou à l'évolution d'un système existant. [Rocques, 07]

En d'autres termes, c'est les différentes opérations réalisées afin d'élaborer le produit logiciel. Dans notre cas nous a opté pour un processus unifié, ce choix est justifiés par les principes sur les quels se base ce processus.

3.2. Processus unifié (Unified Process)

Un processus unifié est un processus de développement logiciel construit sur UML ; il est itératif et incrémental, centré sur l'architecture, conduit par les cas d'utilisation et piloté par les risques. [Rocques, 07]

Itératif et incrémental, au sens que la réalisation du produit se fait en plusieurs itérations où chaque itération aboutit à livrer une partie du produit (un module) et l'ajoute aux différentes parties déjà construit (incrémentation) en gardant un produit homogène.

Centré sur l'architecture, car il impose le respect des décisions d'architecture à chaque étape de construction du modèle.

Piloté par les risques, dans le cadre, où les différentes sources d'échec du produit doivent être écartées. Une source importante d'échec d'un produit logiciel est son inadéquation aux attentes de l'utilisateur.

Conduit par les cas d'utilisation, ce qui veut dire que la conception est réalisé conformément aux attentes des utilisateurs du produit.

La gestion du processus est organisée suivant les quatre phases suivantes : initialisation, élaboration, construction et transition.

3.3. Modélisation avec UML

UML (*Unified Modeling Language*) : se définit comme un langage de modélisation graphique et textuel destiné à comprendre et décrire des besoins, spécifier, concevoir des solutions et communiquer des points de vue.

UML unifie à la fois les notations et les concepts orientés objet. Il ne s'agit pas d'une simple notation, mais les concepts transmis par un diagramme ont une sémantique

précise et sont porteurs de sens au même titre que les mots d'un langage. UML permet de modéliser de manière claire et précise la structure et le comportement d'un système indépendamment de toute méthode ou de tout langage de programmation. UML est à présent un standard défini par l'Object Management Group (OMG). [Rocques, 07]

3.3.1. Diagramme de cas d'utilisation

Un diagramme de cas d'utilisation permet de structurer les besoins des utilisateurs et les objectifs correspondants d'un système. Il permet aussi d'identifier les possibilités d'interactions entre le système et les acteurs (intervenants extérieurs au système). Il part du principe que les objectifs du système sont tous motivés. La figure (3.1) représente le diagramme de cas d'utilisation de notre application.

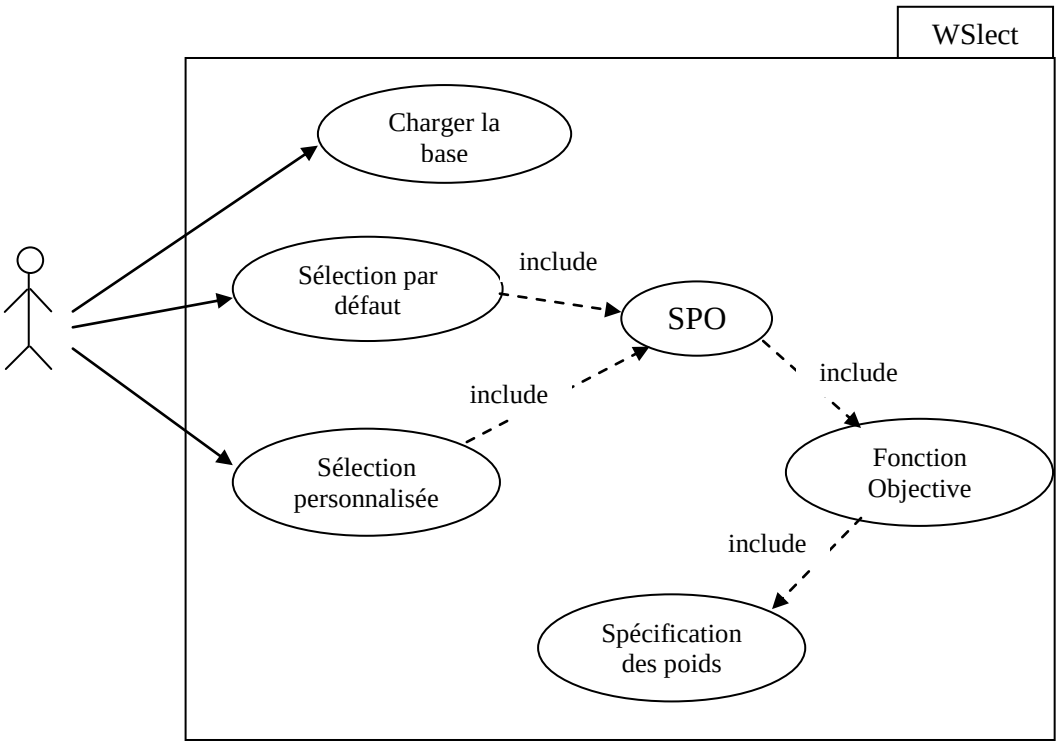


Figure 3.1 : Diagramme de cas d'utilisation

3.3.2. Diagramme de séquence

Un diagramme de séquence permet de représenter des collaborations entre objets selon un point de vue temporel. Les objets communiquent en échangeant des messages

représentés sous forme de flèches. Il peut servir à illustrer un cas d'utilisation. Un exemple de diagramme de séquence de notre application est représenté dans la figure 3.2.

(1) : l'utilisateur choisit la sélection personnalisée, le système lui répond avec une boite de dialogue pour spécifier ses préférences (2). Après la spécification des poids, ces derniers sont envoyés à la classe **FctObjective** (3) où se fera l'affectation de ces valeurs au vecteur **W** des poids.

L'utilisation de l'algorithme **SPO** nécessite l'appel à la classe **FctObjective** durant tout le déroulement de l'algorithme.

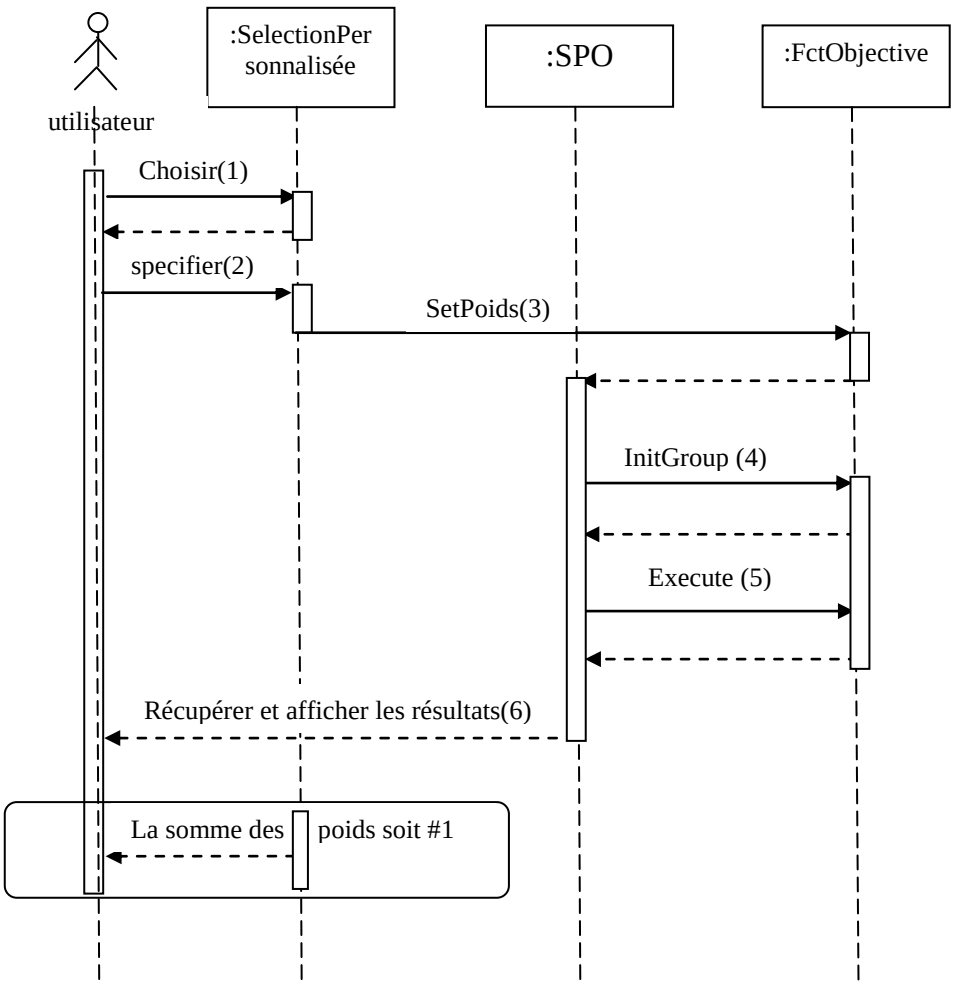


Figure 3.2 : Diagramme de séquence (exemple de sélection personnalisé)

(4) : c'est la méthode de l'initialisation des groupes

(5) : c'est la méthode spécialisé à l'algorithme d'optimisation. Cette dernière retourne un vecteur de trois (3) valeurs, chaque valeur représente le fournisseur de la classe correspondant.

L'exception mentionnée se génère lorsque l'utilisateur a mal choisis les poids, c'est dans le cas où la somme des poids soit supérieur à 1.

3.3.3. Diagramme de classes

Un diagramme de classes permet de représenter les classes intervenant dans le système. Il constitue un élément très important de la modélisation et permettre de définir quelles seront les composantes du système final. Il permet de structurer le travail de développement de manière très efficace. La figure 3.3 représente le diagramme de classes de notre application.

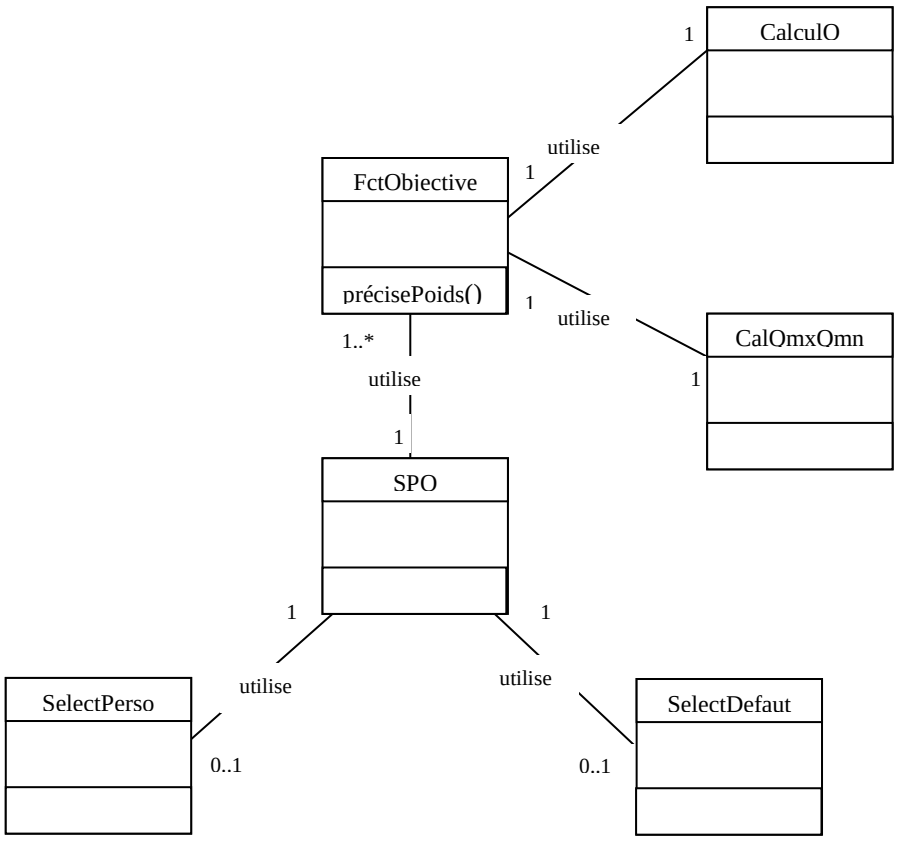


Figure 3.3 : Diagramme de classes

4. Outils et environnement de développement

Avant de commencer l'implémentation de notre application, nous allons tout d'abord spécifier les outils utilisés qui nous ont semblé être un bon choix vu les avantages qu'ils offrent.

4.1. Langage JAVA

Notre choix du langage java a été guidé par les avantages qu'offre la programmation orientée objet. Java utilise des processus qui augmentent les performances des entrées/sorties, facilitent l'internationalisation. Il examine le programme au fil de l'exécution et libère automatiquement la mémoire. Cette fonctionnalité diminue les risques de panne du programme et ne laisse pas la possibilité de mal utiliser la mémoire.

4.2. Netbeans

Notre logiciel est écrit en Netbeans version 6.8 sous Windows. Le choix de la programmation sous Windows a été pris à cause de l'interface graphique qu'offre cet environnement ; Le choix de Netbeans était fondamental puisqu'il est un logiciel permettant principalement le développement en java. Il fournit un environnement standard de développement pour créer des interfaces très puissants.

4.3. Excel

Nous avons préféré d'utiliser Excel pour éditer notre base de donnée. C'est un simple tableur ; autrement dit, il se présente sous forme de tableaux structurés en lignes et colonnes dans des onglets séparés où pour chaque cellule qui compose chaque feuille, des caractéristiques particulières pour les calculs et plein d'autre fonctionnalité caractérisant ce dernier.

5. Présentation du prototype

5.1. Présentation de l'IHM

L'interface homme/machine représente l'élément clé dans l'utilisation de tout système informatique. Les interfaces de notre système de recherche sont conçues de manière à être simples, naturelles, compréhensible et d'utilisation faciles.

1. Fenetre principale :

La fenetre principale se compose de deux panneaux, un paneau droit destiné pour afficher la base de données après le chargement de celle-ci, et un paneau gauche qui contient les fonctions à exécuter. Ce dernier lui-même se compose de deux partie, la premiere destinée pour le chargement de la base et l'afficher dans le panneau droit, la deuxieme destiné pour la selection des services web, dont l'utilisateur a le choix de choisir la sélection qui lui convient.

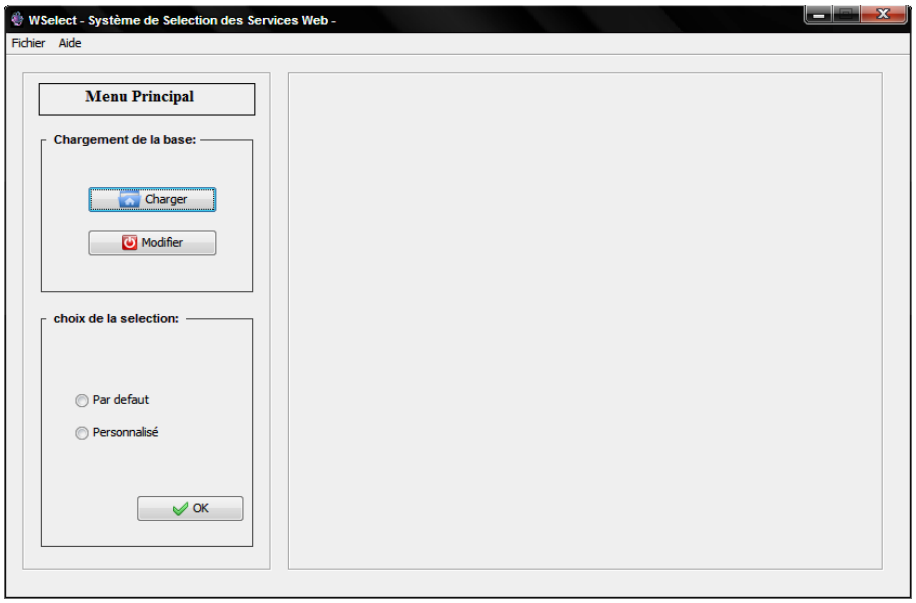
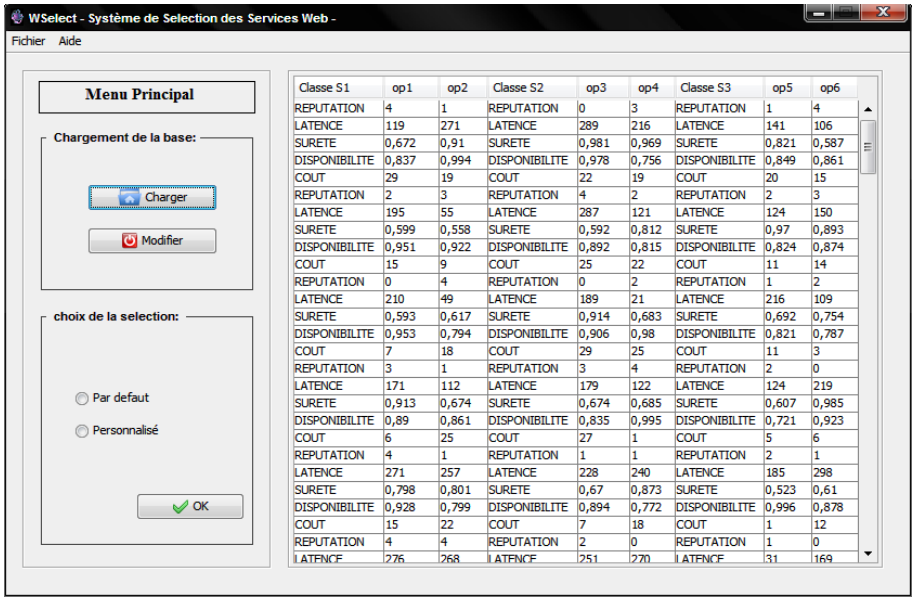


Figure 3.4 : Fenêtre principale de W-Select

2. Chargement de la base de données



Classe S1	op1	op2	Classe S2	op3	op4	Classe S3	op5	op6
REPUTATION	4	1	REPUTATION	0	3	REPUTATION	1	4
LATENCE	119	271	LATENCE	289	216	LATENCE	141	106
SURETE	0,672	0,91	SURETE	0,981	0,969	SURETE	0,821	0,587
DISPONIBILITE	0,837	0,994	DISPONIBILITE	0,978	0,756	DISPONIBILITE	0,849	0,861
COUT	29	19	COUT	22	19	COUT	20	15
REPUTATION	2	3	REPUTATION	4	2	REPUTATION	2	3
LATENCE	195	55	LATENCE	287	121	LATENCE	124	150
SURETE	0,599	0,558	SURETE	0,592	0,812	SURETE	0,97	0,893
DISPONIBILITE	0,951	0,922	DISPONIBILITE	0,892	0,815	DISPONIBILITE	0,824	0,874
COUT	15	9	COUT	25	22	COUT	11	14
REPUTATION	0	4	REPUTATION	0	2	REPUTATION	1	2
LATENCE	210	49	LATENCE	189	21	LATENCE	216	109
SURETE	0,593	0,617	SURETE	0,914	0,683	SURETE	0,692	0,754
DISPONIBILITE	0,953	0,794	DISPONIBILITE	0,906	0,98	DISPONIBILITE	0,821	0,787
COUT	7	18	COUT	29	25	COUT	11	3
REPUTATION	3	1	REPUTATION	3	4	REPUTATION	2	0
LATENCE	171	112	LATENCE	179	122	LATENCE	124	219
SURETE	0,913	0,674	SURETE	0,674	0,685	SURETE	0,607	0,985
DISPONIBILITE	0,89	0,861	DISPONIBILITE	0,835	0,995	DISPONIBILITE	0,721	0,923
COUT	6	25	COUT	27	1	COUT	5	6
REPUTATION	4	1	REPUTATION	1	1	REPUTATION	2	1
LATENCE	271	257	LATENCE	228	240	LATENCE	185	298
SURETE	0,798	0,801	SURETE	0,67	0,873	SURETE	0,523	0,61
DISPONIBILITE	0,928	0,799	DISPONIBILITE	0,894	0,772	DISPONIBILITE	0,996	0,878
COUT	15	22	COUT	7	18	COUT	1	12
REPUTATION	4	4	REPUTATION	2	0	REPUTATION	1	0
LATENCE	276	268	LATENCE	251	270	LATENCE	31	169

Figure 3.5 : Chargement de la base de données

3. la sélection par défaut

Dans cette partie, nous avons initilisé les poids par des valeurs fixé *W*[i]=0.2 pour tout l'ensemble des attributs.

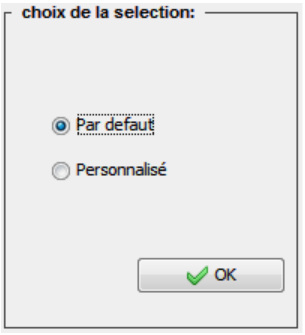


Figure 3.6 : La sélection par défaut

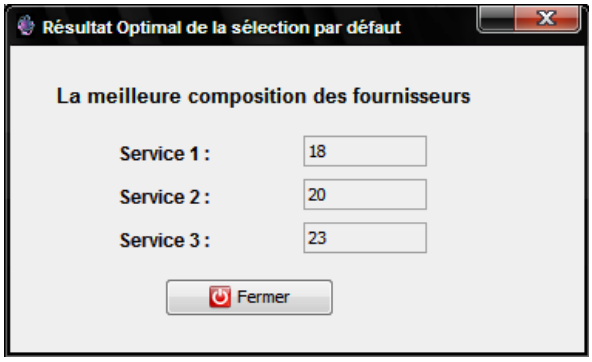


Figure 3.7 : Résultats de la sélection par défaut

4. la sélection personnalisée

Dans cette branche nous donnons la main aux utilisateurs de spécifier leurs preferences et importances, mais bien sur la somme des poids doit égal à 1.

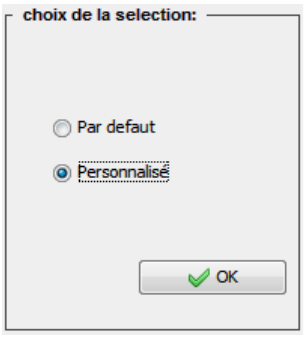


Figure 3.8 : La sélection personnalisée

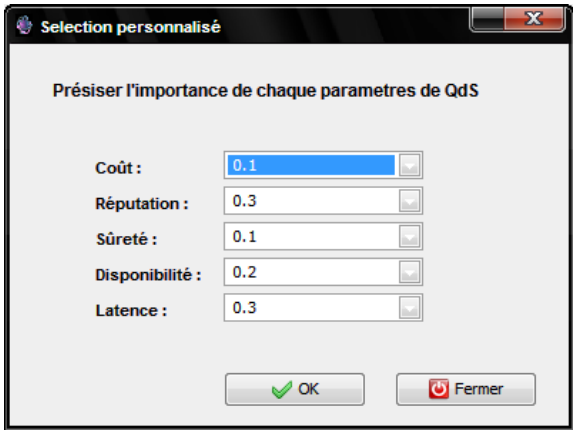


Figure 3.9 : Le choix des poids selon l'utilisateur

Si l'utilisateur a trempé dans le choix des poids, et il a choisis des valeurs où la somme est supérieure à 1, nous le donnons la possibilité de les réinitialiser une autre fois, le message d'erreur est illustré dans la figure 3.10.

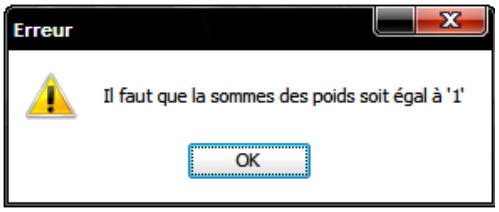


Figure 3.10 : Le message d'erreur

Une fois l'utilisateur spécifie des poids juste, c'est-à-dire en les sommant nous trouvons 1, après l'exécution une boite du dialogue s'affiche. La figure 3.11 montre le résultat de la sélection personnalisée.

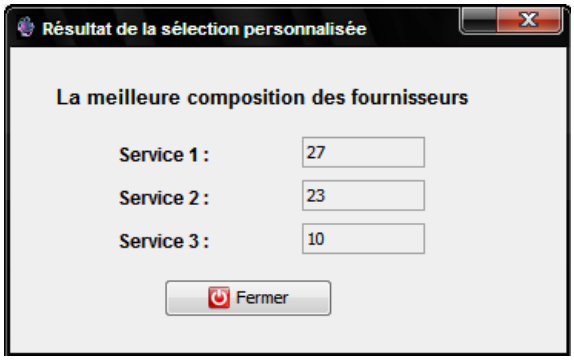


Figure 3.11 : Résultats de la sélection personnalisée

5.2. Les expérimentations

Nous présentons maintenant un échantillon des résultats obtenus au cours de nos expérimentations.

Pour la sélection par défaut :

Groupe	Itération1	Itération 2	Itération3
0	(9, 34, 25)	(17, 34, 25)	(17, 37,25)
1	(27, 19, 5)	(27, 17, 5)	(27, 17, 5)
2	(20, 26, 10)	(20, 26, 10)	(0, 26, 15)
3	(20, 10, 8)	(20, 10, 8)	(20, 10, 8)
4	(10, 23, 8)	(10, 23, 8)	(10, 23, 8)
5	(25, 1, 18)	(25, 13, 8)	(25, 23, 8)

meilleurs positions de chaque itération	(10, 23, 8)	(10, 23, 8)	(25, 23,8)

Table 3.4: Un exemple de sélection par défaut
(Cas de 3 itérations)

Groupe	Itération1	Itération 2	Itérartion3	Itérartion4	Itérartion5
0	(29, 22, 10)	(29, 22, 10)	(12, 13, 8)	(12, 13, 8)	(12, 13, 8)
1	(12, 20, 24)	(12, 20, 33)	(12, 20, 33)	(12, 20, 33)	(33, 20, 33)
2	(29, 26, 36)	(29, 26, 36)	(29, 26, 25)	(29, 28, 25)	(29, 28, 25)
3	(33, 23, 14)	(33, 23, 14)	(26, 23, 14)	(23, 23, 14)	(23, 23, 14)
4	(5, 20, 36)	(12, 20, 31)	(6, 20, 33)	(17, 20, 33)	(17, 20, 33)
5	(9, 2, 23)	(16, 10, 23)	(1, 10, 23)	(1, 10, 23)	(1, 10, 23)

(33, 23, 14)	(12, 20, 33)	(1, 10, 23)	(17, 20, 33)	(33, 20, 33)

Table 3.5: Un exemple de sélection par défaut
(Cas de 5 itérations)

Pour la sélection personnalisée : les tableaux qui se suivent montrent un exemple d'exécution de la sélection personnalisée avec ces poids (0.3, 0.1, 0.1, 0.2, 0.3), pour une expérimentation de trois itérations puis pour cinq. Les résultats finaux sont respectivement (9, 23, 8), (10, 13, 33).

Groupe	**Itération1**	**Itération 2**	**Itérartion3**
0	(1, 10, 8)	(9, 10, 8)	(9, 23, 8)
1	(12, 14, 4)	(9, 14, 15)	(9, 14, 36)
2	(24, 25, 18)	(24, 23, 18)	(2', 14, 36)
3	(33, 11, 36)	(6, 11, 36)	(6, 11, 36)
4	(18, 1, 29)	(18, 1, 29)	(18, 1, 25)
5	(10, 4, 9)	(10, 22, 31)	(10, 22, 21)

meilleurs positions de chaque itération	(24, 25, 18)	(9, 10, 8)	(9, 23, 8)

Table 3.6: Un exemple de sélection personnalisée
(Cas de 3 itérations)

Groupe	**Itération1**	**Itération 2**	**Itérartion3**	**Itérartion4**	**Itérartion5**
0	(4, 20, 0)	(4, 20, 0)	(4, 20, 0)	(4, 20, 15)	(4, 20, 15)
1	(37, 39, 18)	(11, 3, 18)	(21, 30, 18)	(21, 30, 18)	(0, 37, 18)
2	(10, 3, 33)	(10, 3, 33)	(10, 3, 33)	(10, 13, 33)	(10, 13, 33)
3	(12, 17 ,25)	(12, 17 ,25)	(12, 28 ,25)	(11, 28 ,25)	(11, 10, 25)
4	(23, 20, 25)	(23, 20, 25)	(23, 20, 25)	(23, 20, 25)	(23, 20, 25)
5	(9, 10, 5)	(9, 10, 5)	(9, 10, 8)	(9, 10, 8)	(18, 10, 36)

(23, 20, 25)	(23, 20, 25)	(23, 20, 25)	(10, 13, 33)	(10, 13, 33)

Table 3.7: Un exemple de sélection personnalisée
(Cas de 5 itérations)

6. Conclusion

Nous avons présenté dans ce chapitre les grandes parties du système, la plus importante étape était l'application de l'algorithme d'optimisation, en effet ce dernier a demandé un grand effort dans sa conception.

Les résultats obtenus sont testés en premier lieu par une petite base composée de quelques fournisseurs, le moment où nous sommes sûr des résultats nous avons la généralisé par la grande base.

Conclusion Générale

Dans ce mémoire, nous avons présenté un travail basé sur un algorithme d'Optimisation par Essaim Particulaire. L'optimisation par essaim particulaire est une méta-heuristique destinée à la résolution de problèmes à variables continues, inspirée du comportement en essaim de certains animaux. Un essaim de particules présente un comportement auto-organisé qui conduit, par des interactions au niveau local, à l'émergence d'un comportement global complexe. Cet algorithme est appliqué dans notre étude à la sélection des services web.

Notre projet de fin d'étude nous a permis d'une part d'approcher le domaine des services web, d'autre part, nous avons aussi acquis une base assez important sur les méthodes métaheuristiques et plus particulièrement la méthode d'optimisation à essaim particulaire. Le travail réalisé peut être résumé comme suit :

- L'adoption d'une version d'essaim particulaire basé sur les groupes et la mutation probabiliste.

- l'adoption d'une fonction objective agrégée à partir des différents critères de qualité de services.

Tout travail est amené à être amélioré, en ce sens, notre système peut encore évoluer et se voir améliorer. Une voie de recherche pour l'amélioration de cette méthode est d'appliquer d'autres algorithmes d'optimisations telles que les colonies de fourmis, colonies d'abeilles, …

L'étape de mise à jour des liens d'information à l'intérieur des groupes est un enjeu important, car une bonne gestion de la structure de l'essaim pourrait permettre de maintenir la diversité nécessaire au sein de l'essaim. La question est ici de savoir quelle méthode utiliser pour modifier de manière optimale la structure de l'essaim. Il serait

intéressant de proposer une autre méthode pour réaliser cette mise à jour.

Une autre proposition de donner la main à l'utilisateur de spécifier leurs attributs de qualité de services ou encore de préciser des conditions sur les intervalles de ces derniers. De même, il serait intéressant d'utiliser d'autres types de fonctions afin de pouvoir bénéficier de ses avantages.

Références bibliographiques

Bibliographie :

[Amardeilh, 07] : Florence Amardeilh, Web Sémantique et Informatique Linguistique : propositions méthodologiques et réalisation d'une plateforme logicielle, Thèse de doctorat, Université Paris Nanterre, 2007.

[Austin, 02] Austin, D., and Barbir, A., Ferris, C. and Garg, S. : Web Services Architecture Requirements. W3C Working Draft, http://www.w3c.org/TR/2002/WD, 2002

[Belaid, 09] : Youssef Belaid, Service Web – SOAP, Centre d'enseignements de Grenoble, 2009

[Bieler, 05] : Batiste Bieler, Etude de faisabilité d'une application SOAP avec un système embarqué, école HE-ARC ingénierie informatique, 2005.

[Ben Halima, 09] : Riadh BEN HALIMA, Conception, implantation et expérimentation d'une architecture en bus pour l'auto-réparation des applications distribuées à base de services web,l'Université Toulouse III - Paul Sabatier et la Faculté des Sciences Économiques et de Gestion – Sfax (Mai 14,2009)

[Carlos, 01] : Carlos C. Tapang, Description du langage WSDL, Infotects, 2001.
[Châtel, 07] : Pierre Châtel, Une architecture pour la découverte et l'orchestration de services Web sémantiques, Thales Communications, Laboratoire d'Informatique de Paris VI , 2007

[Chauvet, 02] : Chauvet J M., Services Web avec SOAP, WSDL, UDDI, ebXML, Eyrolles, 2002.

[Clari, 08]: Fabrice Clari, Web Services, Université Nice-Sophia Antipolis, 2008

[Clerc, 03] : Clerc M. , TRIBES - Un exemple d'optimisation par essaim particulaire sans paramètres de contrôle, Conférence OEP'03, Paris, France, 2 Octobre, 2003.

[Colorni, 91] : Colorni A. , Dorigo M. et MANIEZZO V. , Distributed Optimization by Ant Colonies , Proceedings of the European Conference on Artificial Life ECAL.91, pages 134-142. Elsevier Publishing, 1991.

[Delacrétaz, 03]: Delacrétaz Jean-Marc, Web Services, laboratoire Génie Documentiel, 2003.

[Dustdar, 05]: Dustdar, S. and Schreiner, W. : A survey on web services composition. International Journal of Web and Grid Services, Vol. 1, No. 1, pages 1-30 ,2005

[Fierstone, 02]: Jeremy FIERSTONE, Les services Web, ESSI, 2002.

[Glover, 86]: Glover F. , Future paths for integer programming and links to artificial intelligence, Computers and Operations Research, Vol. 13, pages 533-549, 1986

[Glover & Laguna, 97] : Glover F. , Laguna M. , Tabu Search, Kluwer Academic Publishers, 1997

[Goss, 89] : Goss S. , Aron S., Deneubourg J.L. , Pasteels J.M. , Self-Organized Shortcuts in the Argentine Ant, Naturwissenchaften, Vol. 76, pages 579-581, 1989.

[Heppner, 90]: Heppner, F. and Grenander, U. , A stochastic nonlinear model for coordinated bird flocks. In S. Krasner, Ed., The Ubiquity of Chaos. AAAS Publications, Washington, DC, pages 233-238, 1990

[Kadima & Montfort, 03] : Kadima H. & Montfort V. , Les services Web : Techniques, démarches et outils, édition Dunod, 2003

[Kennedy, 99]: Kennedy J. , Small Worlds and Mega-Minds : Effects of Neighborhood Topology on Particle Swarm Performance. In IEEE Congress on Evolutionary Computation, volume III, pages 1932–1938, 1999

[Kennedy & Eberhart, 95]: Kennedy J. , Eberhart R.C. ,Particle Swarm Optimization, Proceedings of the IEEE International Conference On Neural Networks, pages 1942-1948, IEEE Press, 1995

[Maesano, 03] : Maesano L., Bernard C. et Galles X. « Services web avec J2EE et.NET » ÉDITIONS EYROLLES, septembre 2003

[Rampacek, 06]: Sylvain RAMPACEK, Sémantique, interactions et langages de description des services web complexes, thèse de Doctorat, Université de Reims Champagne-Ardenne, 2006.

[Reynolds, 87]: Reynolds, C.W. , Flocks, herbs and schools : a distributed behavioral model, Computer Graphics, pages 25-34, 1987

[Rocques, 07] : Rocques P. et Vallées F., « UML 2 en Action, de l'analyse des besoins à la conception », Eyrolles , 2007

[Shi & Eberhart, 98]: Shi Y. , Eberhart R. , Parameter Selection in Particle Swarm Optimization, Proceedings of the 7th Annual Conference on Evolutionary Programming, pages 591-600, LNCS 1447, Springer, 1998; F. Van den Bergh , An analysis of Particle Swarm Optimizer, PhD thesis, University of Pretoria, South Africa, 2002

[Wang, 07]: Wang Y. and Vassileva, J. Toward trust and reputation based web service selection: A survey. Proc. International Transactions on Systems Science and Applications (ITSSA) Journal, 3, 2, 2007

[Wilson, 75]: Wilson E.O., Sociobiology: The new synthesis, Cambridge, MA: Belknap Press, 1975

Webographie :

[www1] :http://www.w3.org/2002/ws/ Dernière consultation 18/09/2011.

[www2] : http://www.w3.org/TR/SOAP/ Dernière consultation 18/09/2011.

[www3] : http://www.w3.org/TR/2007/REC-wsdl20-20070626/ Dernière consultation 18/09/2011.

Résumé :

Le domaine de services web est une nouvelle technologie qui permet aux applications de communiquer entre eux. Ces applications se diffèrent les unes des autres par leurs QdS et vu que le résultat d'une invocation de services donne un ensemble exponentiel de plans d'exécutions d'où la nécessité de choisir des méthodes heuristiques. Dans ce travail nous décrivons la conception et le développement d'un système de sélection de service web à base de qualité de service.
Pour atteindre cet objectif, nous avons utilisé l'algorithme d'optimisation à essaim particulaire mono- objective dans sa version TRIBES.

Mots Clefs :
Service web, algorithme d'optimisation, fonction objective, méta heuristique, plan d'exécution, essaim particulaire.

Abstract:

The domain of web services is a novel technology that allows applications to communicate with each others. These applications are differentiated according to their quality of service. Usually we remark that the majority of requests can accept an exponential number of execution plans. Therefore we are obliged to use heuristics to filter this set. In this work we highlight a system that permits to select the best execution plan based on the QoS criteria.
To reach this goal, we have used the mono objective particular swarm optimization algorithm and particularly the TRIBES version.

Keywords:
Web services, optimization algorithm, objective function, métaheuristic, execution plan, particular swarm.

العربية :
إن مجال خدمات الواب تقنية جديدة تسمح للبرامج من الاتصال فيما بينها. هذه البرامج تختلف بعضها عن بعض بجودة الخدمات، وبما أن البحث عن خدمة من الخدمات يمكن أن يعطي عدد خيالي من النتائج كان لابد من اختيار طرق مساعدة على الكشف. في هذا العمل نشرح كيفية تصوير و تنمية نظام لانتقاء خدمات الواب على أساس جودة الخدمات.
لتحقيق هذا الغرض، استعملنا خوارزمية سرب الجسيمات للتحسين أحادية الهدف في نسختها TRIBES.

الكلمات المفتاحية:
خدمات الواب، خوارزمية التحسين، دالة هدفية، ميتا كشفية، خطة التنفيذ، سرب الجسيمات.